Manuel MySQL 4.1.x − Version Française

http://www.nexen.net

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

1 Informations générales

Le logiciel `MySQL (TM)` est un serveur de base de données `SQL` très rapide, multi−threadé, multi−utilisateur et robuste. Le serveur `MySQL` est destiné aux missions stratégiques et aux systèmes de production à forte charge, ainsi qu'à l'intégration dans des logiciels déployés à grande échelle. `MySQL` est une marque déposée de `MySQL AB` .

Le logiciel `MySQL` dispose de `deux licenses` . Les utilisateurs peuvent choisir entre utiliser `MySQL` comme un logiciel `Open Source` / `Logiciel libre` , sous les termes de la licence `GNU General Public License` (http://www.gnu.org/licenses/) ou bien, ils peuvent acheter une licence commerciale auprès de `MySQL AB` . Support et licences MySQL .

Le site web de `MySQL` (http://www.mysql.com/) fournit les dernières informations sur le serveur `MySQL` .

La liste suivante décrit les sections particulières de ce manuel :

- Pour plus d'informations sur la société derrière le `serveur de base de données MySQL` , voyez Qu'est ce que MySQL AB ? .
- Pour une présentation des capacités de `serveur de base de données MySQL` , voyez Principales fonctionnalités de MySQL .
- Pour les instructions d'installation, voyez Installation de MySQL .
- Pour des conseils sur le port du `serveur de base de données MySQL` sur de nouvelles architectures ou systèmes d'exploitation, voyez Port vers d'autres systèmes .
- Pour des informations sur la mise à jour vers la version 4.0, voyez Mise à jour de Version 4.0 vers 4.1 .
- Pour des informations sur la mise à jour vers la version 3.23, voyez Mise à jour de Version 3.23 vers 4.0 .
- Pour des informations sur la mise à jour vers la version 3.22, voyez Mise à jour de Version 3.22 vers 3.23 .
- Pour une introduction au `serveur de base de données MySQL` , voyez Tutoriel MySQL .
- Pour des exemples de `SQL` et des tests de performances, voyez le dossier de tests (`sql-bench` de la distribution.
- Pour connaître l'historique des fonctionnalités et bogues, voyez Historique de MySQL .
- Pour une liste des bogues connus et des limitations, voyez Erreurs connues et manques de MySQL .
- Pour les plans de développement, voyez Liste de tâches MySQL .
- Pour une liste de tous les contributeurs à ce projet, voyez Crédits .

Important :

Les rapports d'erreurs (aussi appelés bogues), ainsi que les questions et commentaires, doivent être envoyés à la liste de diffusion générale. Les listes de diffusion MySQL . Comment rapporter des bogues ou problèmes .

Le script `mysqlbug` doit être utilisé pour générer le rapport de bogues. (Les distributions Windows contiennent un fichier `mysqlbug.txt` dans le dossier racine qui peut être utilisé comme formulaire pour un rapport de bug).

Pour les distributions sources, le script `mysqlbug` est accessible dans le dossier `scripts` . Pour les distributions binaires, `mysqlbug` est installé dans le dossier `bin` (`/usr/bin` pour le paquet `RPM` du `serveur MySQL`).

Si vous avez trouvé un problème de sécurité critique dans le code du `serveur MySQL` , vous devez envoyez un email à security@mysql.com .

1.1 A propos du manuel

Ceci est le manuel de référence de `MySQL` ; il documente `MySQL` jusqu'à la version 4.0.5a. Les évolutions fonctionnelles sont toujours indiquées avec une référence à la version d'évolution, de manière à ce que ce manuel soit toujours valable, même si vous utilisez une ancienne version de `MySQL` . Etant un manuel de référence, il ne fournit aucune description générale sur le langage `SQL` ou les concepts de bases de données relationnelles.

Comme le `logiciel de base de données MySQL` est en développement constant, ce manuel es mis à jour fréquemment. La version la plus récente est disponibles à http://dev.mysql.com/doc/ en différents formats, incluant HTML, PDF et Windows HLP.

L'original du document est un fichier au format Texinfo. La version HTML est produite automatiquement avec une version modifiée de `texi2html` . La version en texte plein et version Info sont produites par `makeinfo` . La version PostScript est produite avec `texi2dvi` et `dvips` . La version PDF est produite avec `pdftex` .

Si vous avez du mal à trouver des informations dans ce manuel, vous pouvez essayer notre version avec moteur de recherche, sur notre site web : http://www.mysql.com/doc/ .

Si vous avez des suggestions concernant des ajouts ou des corrections à ce manuel, vous pouvez les envoyez à l'équipe de documentation à docs@mysql.com .

Ce manuel a été écrit initialement par David Axmark et Michael (Monty) Widenius. Il est actuellement entretenu par Michael (Monty) Widenius, Arjen Lentz et Paul DuBois. Pour les autres contributeurs, voyez les Crédits .

La traduction de ce manuel a été faite sous la direction de Damien Séguy. Mehdi Achour, Patrick Haond, David Manusset, Sylvain Maugiron, Guillaume Plessis et Yannick Torres ont contribué largement à cette traduction et son entretien.

Le copyright (2002) de ce manuel est la propriété de la société suédoise `MySQL AB` . Copyrights et licenses utilisées par MySQL .

1.1.1 Conventions utilisées dans ce manuel

Ce manuel utilise certaines conventions typographiques :

`constant`
: La police à largeur fixe est utilisée pour les noms de commandes et les options, les requêtes SQL, les noms de bases de données, de tables et de colonnes, le code C et Perl, les variables d'environnement. Par exemple, ``Pour voir comment `mysqladmin` fonctionne, exécutez−le avec l'option `--help` ."

`filename`
: La police à largeur fixe avec des guillemets d'encadrement indique des noms de fichiers et de chemins de dossiers. Par exemple : ``La distribution est installée dans le dossier `/usr/local/` ."

`'c'`
: La police à largeur fixe avec des guillemets d'encadrement est aussi utilisée pour indiquer des séquences de caractères. Par exemple : ``Pour spécifier un caractère joker, utilisez le caractère `'%'` ."

italique
: Les polices en italique sont utilisées pour attirer l'attention, **comme ceci** .

gras

Le gras est utilisé pour les entêtes de tables, et aussi pour **attirer fortement votre attention** .

Lorsque les commandes qui sont affichées sont destinées à être exécutées par un programme particulier, le nom du programme est indiqué dans l'invite de la commande. Par exemple, `shell>` indique une commande que vous exécutez depuis votre console Shell, et `mysql>` indique une commande que vous exécutez depuis le client `mysql` :

```
 shell> tapez une commande shell ici
mysql> tapez une requête SQL ici
```

Le ``Shell'' est votre interpréteur de ligne de commande. Sous Unix, c'est typiquement un programme comme `sh` ou `csh` . Sous Windows, le programme équivalent est `command.com` ou `cmd.exe` , typiquement utilisée en console.

Lorsque vous saisissez une commande dans un exemple, omettez simplement de saisir l'invite de commande affichée.

Souvent, les noms de bases de données, tables ou colonnes doivent être remplacés dans les commandes. Pour indiquer qu'une telle substitution est nécessaire, ce manuel utilise les noms de `nom_de_base` , `nom_de_table` et `nom_colonne` . Par exemple, vous pourriez avoir une requête comme ceci :

```
mysql> SELECT nom_colonne FROM nom_de_base.nom_de_table;
```

Cela signifie que si vous devez saisir une requête semblable, vous devriez utiliser votre propre nom de colonne, table et base de données, ce qui pourrait se traduire par ceci :

```
mysql> SELECT author_name FROM biblio_db.author_list;
```

Les mot réservés SQL ne sont pas sensibles à la casse, et peuvent être écrits en majuscules ou minuscules. Ce manuel utilise les majuscules.

Dans les illustrations de syntaxe, les crochets (`'['` et `']'`) sont utilisés pour indiquer des clauses ou mots optionnels. Par exemple, dans la requête suivante, `IF EXISTS` est optionnel :

```
DROP TABLE [IF EXISTS] nom_de_table
```

Lorsqu'un élément de syntaxe est constitué d'un certain nombre d'alternatives, les alternatives sont séparées par des barres verticales (`'|'`). Lorsqu'un membre d'un tel jeu de possibilités **peut** être choisi, les alternatives sont listées entre crochets (`'['` et `']'`):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

Lorsqu'un élément d'un jeu de possibilités **doit** être choisi, les alternatives sont placées entre accolades (`'{'` et `''` }):

```
&#123;DESCRIBE | DESC&#125; nom_de_table &#123;nom_colonne | wild&#125;
```

Des crochets peuvent aussi indiquer que l'élément syntaxique précédent peut être répété. Dans l'exemple suivant, plusieurs valeurs `reset_option` peuvent être donnés, séparées par des virgules :

```
RESET reset_option [,reset_option] ...
```

Les commandes d'assignation des variables de Shell sont présentées avec la syntaxe Bourne Shell. Par exemple, la syntaxe suivante modifie une variable d'environnement :

```
shell> VARNAME=value some_command
```

Si vous utilisez `csh` ou `tcsh` , vous devez utiliser une syntaxe légèrement différente. Il faut écrire :

```
 shell> setenv VARNAME value
shell> some_command
```

1.2 Qu'est ce que MySQL?

`MySQL` , le plus populaire des serveurs de bases de données SQL `Open Source` , est développé, distribué et supporté par `MySQL AB` . `MySQL AB` est une société commerciale, fondée par les développeurs de MySQL, qui développent leur activité en fournissant des services autour de MySQL . Qu'est ce que MySQL AB ? .

Le site web de MySQL (http://www.mysql.com/) fournit les toutes dernières actualités sur le logiciel MySQL et sur la société MySQL AB .

MySQL *est un système de gestion de bases de données.*
: Une base de données est un ensemble organisé de données. Cela peut aller d'une simple liste de courses au supermarché à une galerie de photos, ou encore les grands systèmes d'informations des multi−nationales. Pour ajouter, lire et traiter des données dans une base de données, vous avez besoin d'un système de gestion de bases de données tel que le serveur MySQL . Comme les ordinateurs sont très bons à manipuler de grandes quantités de données, le système de gestion de bases de données joue un rôle central en informatique, aussi bien en tant qu'application à part entière, qu'intégré dans d'autres logiciels.

MySQL *est un serveur de bases de données relationnelles.*
: Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête. Le SQL dans `` MySQL '' signifie `` Structured Query Language '' : le langage standard pour les traitements de bases de données.

MySQL *est* Open Source *.*
: Open Source (Standard Ouvert) signifie qu'il est possible à chacun d'utiliser et de modifier le logiciel. Tout le monde peut télécharger MySQL sur Internet, et l'utiliser sans payer aucun droit. Toute personne en ayant la volonté peut étudier et modifier le code source pour l'adapter à ses besoins propres. Le logiciel MySQL utilise la licence GPL (GNU General Public License), http://www.gnu.org/licenses/ , pour définir ce que vous pouvez et ne pouvez pas faire avec ce logiciel, dans différentes situations. Si vous ne vous sentez pas confortable avec la licence GPL ou bien que vous devez intégrer MySQL dans une application commerciale, vous pouvez acheter une licence commercial auprès de MySQL AB. Licences MySQL .

Le serveur de bases de données MySQL *est très rapide, fiable*
: et facile à utiliser

  Si c'est ce que vous recherchez, vous devriez faire un essai. Le serveur de bases de données MySQL dispose aussi de fonctionnalités pratiques, développées en coopération avec nos utilisateurs. Vous pouvez trouver une comparaison des performances du serveur MySQL avec d'autres systèmes de bases de données dans nos pages de tests de performances. La suite de tests comparatifs de MySQL .

  Le serveur MySQL a été développé à l'origine pour gérer de grandes bases de données plus rapidement que les solutions existantes, et a été utilisé avec succès dans des environnements de production très contraints et très exigeants, depuis plusieurs années. Bien que toujours en développement, le Le serveur MySQL offre des fonctions nombreuses et puissantes. Ses possibilités de connexions, sa rapidité et sa sécurité font du serveur MySQL une serveur hautement adapté à Internet.

MySQL Server fonctionne en mode client/serveur ou en système embarqué.
: Le serveur MySQL est un système client / serveur qui est constitué d'un serveur SQL multi−threadé qui supporte différentes interfaces, clients, bibliothèques et outils d'administration, ainsi qu'une large gamme de pilotes pour différents langages (API).Nous proposons aussi le serveur MySQL comme une bibliothèque embarquée, que vous pouvez intégrer dans vos applications pour en faire des produits plus petits, plus rapides et plus simples à utiliser.

Il existe un grand nombre de contributions à `MySQL` *.*

Il est très probable que vous pourrez trouver votre éditeur préféré ou que votre environnement de programmation supporte déjà le `serveur de base de données MySQL` .

La prononciation officielle de `MySQL` est `` `My Ess Que Ell` '' (en anglais), ce qui donne `` `Maille Esse Cu Elle` '' en phonétique française. Evitez d'utiliser la prononciation ``my sequel'', mais nous ne nous formaliserons pas que vous utilisiez `` `my sequel` '' (ma séquelle, en français) ou une autre prononciation adaptée.

1.2.1 Histoire de MySQL

Nous avons débuté avec l'intention d'utiliser `mSQL` pour se connecter à nos tables en utilisant nos propres routines bas niveau ISAM. Cependant, après quelques tests, nous sommes arrivés à la conclusion que `mSQL` n'était pas assez rapide et flexible pour nos besoins. Cela nous a conduit à créer une nouvelle interface SQL pour notre base de données, mais en gardant la même API que `mSQL` . Cette API a été choisie pour la facilité de port des programmes de tiers.

Les liens avec le nom `MySQL` ne sont pas parfaitement établis. Notre dossier de base et un grand nombre de bibliothèques et outils étaient préfixés par ``my'' depuis plus de 10 ans. Mais la fille de Monty, plus jeune que lui, était aussi appelée My. Lequel des deux a conduit au nom de `MySQL` est toujours un mystère, même pour nous.

Le nom du dauphin MySQL (notre logo) est `Sakila` , qui a été choisi par les fondateurs de MySQL AB à partir d'une grande liste de noms suggérés par les utilisateurs dans le concours "Name the Dolphin" ("Nommez le dauphin"). Le nom a été suggéré par Ambrose Twebaze, un développeur de softwares open source de Swaziland, Afrique. D'après Ambrose, le nom Sakila puise ses origines du SiSwati, la langue locale du Swaziland. Sakila est aussi le nom d'une ville dans Arusha, Tanzanie, près du pays d'origine d'Ambrose, Uganda.

1.2.2 Les fonctionnalités principales de MySQL

La liste suivante décrit les caractéristiques principales du `logiciel de bases de données MySQL` . Voyez la Plans de développements MySQL pour plus d'informations sur les fonctionnalités courantes et à venir.

Interne et portabilité

Ecrit en C et C++.
Testé sur un large éventail de compilateurs différents.
Fonctionne sur de nombreuses plates−formes. Plate−formes supportées par MySQL .
Utilise GNU Automake, Autoconf et Libtool pour une meilleure portabilité.
Dispose d'API pour C, C++, Eiffel, Java, Perl, PHP, Python, Ruby et Tcl. Interfaces MySQL .
Complètement multi−threadé, grâce aux threads du noyau. Cela signifie que vous pouvez l'utiliser facilement sur un serveur avec plusieurs processeurs.
Fournit des moteurs de tables transactionnels et non−transactionnels.
Index `B-tree` très rapide, avec compression d'index.
Facilité relative à ajouter un nouveau moteur de table. C'est utile si vous voulez ajouter une interface SQL à votre base de donnée maison.
Système l'allocation mémoire très rapide, exploitant les threads.
Jointures très rapides, exploitant un système de jointures multiples en une seule passe optimisé.
Tables en mémoire, pour réaliser des tables temporaires.

Les fonctions SQL sont implémentées grâce à une bibliothèque de classes optimisées, qui sont aussi rapides que possible! Généralement, il n'y a aucune allocation mémoire une fois que la requête a été initialisée.
Le code de `MySQL` est vérifié avec `Purify` (un utilitaire de détection des fuites mémoires commercial) ainsi qu'avec Valgrind, un outil GPL (http://developer.kde.org/~sewardj/).

Types de colonnes

Nombreux types de colonnes : entiers signés ou non, de 1, 2, 3, 4, et 8 octets, `FLOAT` , `DOUBLE` , `CHAR` , `VARCHAR` , `TEXT` , `BLOB` , `DATE` , `TIME` , `DATETIME` , `TIMESTAMP` , `YEAR` , `SET` et `ENUM` . Types de colonnes .
Enregistrements de taille fixe ou variable.
Toutes les colonnes ont des valeurs par défaut. Vous pouvez utiliser la commande `INSERT` pour insérer un sous ensemble de colonnes : les colonnes qui ne sont pas explicitement cités prennent alors leur valeur par défaut.

Commandes et fonctions

Support complet des opérateurs et fonctions dans la commande `SELECT` et la clause `WHERE` . Par exemple :

```
mysql> SELECT CONCAT(first_name, " ", last_name)
    -> FROM tbl_name
    -> WHERE income/dependents > 10000 AND age > 30;
```

Support complet des clauses SQL `GROUP BY` et `ORDER BY` . Support des fonctions de groupages (`COUNT()` , `COUNT(DISTINCT ...)` , `AVG()` , `STD()` , `SUM()` , `MAX()` et `MIN()`).
Support des clauses `LEFT OUTER JOIN` et `RIGHT OUTER JOIN` avec les syntaxes ANSI SQL et ODBC.
Les alias de tables et colonnes sont compatibles avec le standard SQL92.
`DELETE` , `INSERT` , `REPLACE` et `UPDATE` retourne le nombre de lignes affectées. Il est possible d'obtenir le nombre de lignes trouvées en modifiant une option lors de la connexion au serveur.
La commande spécifique à `MySQL SHOW` est utilisée pour obtenir des informations sur les bases, tables et index. La commande `EXPLAIN` sert à optimiser les requêtes.
Les noms de fonctions ne sont jamais en conflit avec les noms de tables ou colonnes. Par exemple, `ABS` est un nom de colonne valide. La seule restriction est que, lors d'un appel de fonction, aucun espace n'est toléré entre le nom de la fonction et la parenthèse ouvrante '`(`' suivante. Est−ce que MySQL est sensible aux mots réservés ? .
Vous pouvez utiliser simultanément des tables de différentes bases (depuis la version 3.22).

Sécurité

Un système de droits et de mots de passe très souple et sécuritaire, qui vérifie aussi les hôtes se connectant. Les mots de passe sont bien protégés, car tout les échanges de mot de passe sont chiffrés, même lors des connexions.

Charges supportées et limites

Gère les très grandes bases de données. Nous utilisons le `serveur MySQL` avec des bases qui contiennent 50 millions de lignes et nous connaissons des utilisateurs qui utilisent le `serveur MySQL` avec plus de 60 000 tables et 5 000 000 000 (milliards) de lignes.
Jusqu'à 32 index sont permis par table. Chaque index est constitué de 1 à 16 colonnes ou parties de colonnes. La taille maximale d'un index est de 500 octets (ce qui peut être configuré à la compilation du `serveur MySQL` . Un index peut utiliser un préfixe issu d'un champs `CHAR` ou `VARCHAR` .

Connexions

Les clients peuvent se connecter au serveur `MySQL` en utilisant les sockets TCP/IP, les sockets Unix ou les pipes nommés sous NT.

Support de `ODBC` (`Open−DataBase−Connectivity`) pour Windows 32 bits (avec les sources). Toutes les fonctions ODBC 2.5 et de nombreuses autres. Par exemple, vous pouvez utiliser MS Access pour vous connecter au serveur `MySQL` . Interface ODBC .

L'interface `Connector/JDBC` fournit le support pour les clients Java qui utilisent JDBC. Ces clients peuvent être utilisés sur Windows et Unix. Les sources de Connector/JDBC sont libres. Connexion Java (JDBC) .

Traductions

Le serveur fournit des messages d'erreurs au client dans de nombreuses langues, y compris le français. Messages d'erreurs non−anglophones .

Support complet de plusieurs jeux de caractères, comprenant `ISO−8859−1` (Latin1), `german` , `big5` , `ujis` , etc. Par exemple, les caractères nordiques `'Â'` , `'ä'` et `'ö'` sont autorisés dans les noms de tables et colonnes.

Toutes les données sont sauvées dans le jeu de caractères choisi. Les comparaisons normales de chaînes sont insensibles à la casse.

Le tri est fait en fonction du jeu de caractères choisi (par défaut, le jeu suédois). Il est possible de le changer lorsque le serveur `MySQL` est démarré. Pour voir un exemple très avancé de tri, voyez le code de tri pour le Tchèque. Le `serveur MySQL` supporte de nombreux jeux de caractères qui peuvent être spécifié à la compilation et durant l'exécution.

Clients et utilitaires

Inclut `myisamchk` , un utilitaire rapide pour vérifier les tables, les optimiser et les réparer. Toutes les fonctionnalités de `myisamchk` sont aussi disponibles via l'interface SQL. Administration de la base .

Tous les programmes `MySQL` peuvent être appelés avec l'option `−−help` ou `−?` pour obtenir de l'aide en ligne.

1.2.3 Jusqu'à quel point MySQL est il stable ?

Cette section répond aux questions `` **Jusqu'à quel point MySQL est il stable ?** '' et `` **Puis−je faire confiance à MySQL pour mon projet ?** '' Nous allons tenter d'apporter des réponses claires à ces questions importantes qui concernent tous les utilisateurs potentiels. Les informations de cette section sont fournies par les listes de diffusions, qui sont très actives et promptes à identifier les problèmes et les rapporter.

Le code original date du début des années 80 et fournit une base de code stable, tout en assurant une compatibilité ascendante avec le format `ISAM` . A TcX, le prédécesseur de `MySQL AB` , le code de `MySQL` a fonctionné sur des projets depuis la mi 1996, sans aucun problème. Lorsque le `Serveur MySQL` a été livré à un public plus large, nous avons réalisé qu'il contenait du code ``jamais testé'' qui a été rapidement identifié par les utilisateurs, qui effectuait des requêtes différentes des nôtres. Chaque nouvelle version avait moins de problèmes de portabilité, même si chaque nouvelle version avait de nombreuses nouvelles fonctionnalités.

Chaque version du `Serveur MySQL` était parfaitement fonctionnelle. Les seuls problèmes étaient rencontrés par les utilisateurs de code de ces ``zone d'ombres''. Naturellement, les nouveaux utilisateurs ne connaissent pas ces zones : cette section tente de les présenter, dans la mesure de nos connaissances. Les descriptions correspondent surtout aux versions 3.23 du `Serveur MySQL` . Tous les bogues connus et rapportés ont été corrigés dans la dernière version, à l'exception de ceux qui sont listés dans la section Bugs, qui sont des problèmes de conception. Erreurs connues et manques de MySQL .

La conception du `serveur MySQL` est faite en plusieurs couches, avec des modules indépendants. Certains des modules les plus récents sont listés ici, avec leur niveau de test :

Réplication −− Gamma
De grands serveurs en grappe utilisant la réplication sont en production, avec de bons résultats. L'amélioration de la réplication continue avec MySQL 4.x.

Tables InnoDB *−− Stable (en 3.23 depuis 3.23.49)*
Le gestionnaire transactionnel de tables InnoDB a été déclaré stable en MySQL version 3.23, à partir de la version 3.23.49. InnoDB est utilisé dans de grands systèmes complexes, avec forte charge.

Tables BDB *−− Gamma*
Le code de Berkeley DB est très stable, mais nous sommes encore en train d'améliorer l'interface du gestionnaire transactionnel de table BDB du serveur MySQL . Cela demande encore du temps pour qu'il soit aussi bien testé que les autres types de tables.

FULLTEXT *−− Beta*
La recherche en texte plein fonctionne mais n'est pas encore largement adoptée. Des améliorations importantes sont prévues pour MySQL 4.0.

Connector/ODBC *3.51 (Stable)*
Connector/ODBC 3.51 utilise le SDK ODBC SDK 3.51 et est en production. Certains problèmes qui ont surgi sont liée aux applications, et indépendant du pilote ODBC ou le serveur sous−jacent.

Tables à restauration automatique MyISAM *−− Gamma*
Ce statut ne concerne que le nouveau code du gestionnaire de tables MyISAM qui vérifie si la table a été correctement fermée lors de l'ouverture, et qui exécute automatiquement la vérification et réparation éventuelles de la table.

MySQL AB fournit un support de première qualité pour les clients payant, mais les listes de diffusions de MySQL sont généralement rapides à donner des réponses aux questions les plus communes. Les bogues sont généralement corrigés aussitôt avec un patch. Pour les bogues sérieux, il y a presque toujours une nouvelle version.

1.2.4 Quelles tailles peuvent atteindre les tables MySQL

MySQL version 3.22 a une limite de 4Go par table. Avec le nouveau format de table MyISAM , disponible avec MySQL version 3.23, la taille maximale des tables a été poussée à 8 millions de teraoctets (2 ^ 63 octets).

Notez, toutefois, que les systèmes d'exploitation ont leur propres limites. Voici quelques exemples :

Système d'exploitation	Limite
Linux-Intel 32 bit	2Go, 4Go ou plus, suivant la version de Linux
Linux-Alpha	8To (?)
Solaris 2.5.1	2Go (4Go possibles avec un patch)
Solaris 2.6	4Go (peut être modifié avec une option)
Solaris 2.7 Intel	4Go
Solaris 2.7 UltraSPARC	512Go
NetWare avec/NSS	8TB

En Linux 2.2, vous pouvez avoir des tables plus grandes que 2Go en utilisant le patch LFS pour les systèmes de fichiers ext2. En Linux 2.4, le patche existe aussi pour ReiserFS. La plupart des distribution Linux courantes sont basées sur un noyau 2.4, et supporte déjà tous les patchs pour les grands fichiers (LFS). Cependant, la taille maximale de fichier dépend de nombreux facteurs, notamment le système de fichier utilisé pour stocker les pages MySQL.

Pour une introduction détaillée à LFS sur Linux, voyez la page d' Andreas Jaeger Large File Support in Linux à http://www.suse.de/~aj/linux_lfs.html .

Par défaut, les tables MySQL peuvent atteindre une taille de 4Go. Vous pouvez vérifier la taille des tables avec la commande SHOW TABLE STATUS ou la commande en ligne myisamchk −dv nom_de_table . Syntaxe de

SET et SHOW .

Si vous avez besoin de tables plus grandes que 4Go (et que votre système d'exploitation le supporte, modifiez les paramètres AVG_ROW_LENGTH et MAX_ROWS lorsque vous créez votre table. Syntaxe de CREATE TABLE . Vous pouvez aussi modifier ces valeurs avec la commande ALTER TABLE . Syntaxe de ALTER TABLE .

D'autres méthodes pour contourner les limitations des systèmes de fichiers avec les tables MyISAM :

- Si votre table est en lecture seule, utilisez myisampack pour la compresser. myisampack compresse une table à 50%, ce qui double environs la taille des tables. myisampack peut aussi combiner plusieurs tables en une seule. myisampack , le générateur de tables MySQL compressées .
- Une autre méthode pour contourner les limites du système de fichiers pour les tables MyISAM est d'utiliser les options RAID . Syntaxe de CREATE TABLE .
- MySQL inclut une bibliothèque MERGE qui permet de gérer plusieurs tables identiques comme une seule. Tables MERGE .

1.2.5 Compatibilité an 2000

Le serveur MySQL lui même n'a aucun problème de compatibilité avec l'an 2000 (Y2K) :

- Le serveur MySQL utilise les fonctions de date Unix, et n'a aucun problème avec les dates jusqu'en 2069 ; toutes les années écrites en deux chiffres sont supposées faire partie de l'intervalle allant de 1970 à 2069 , ce qui signifie que si vous stockez la date 01 dans une colonne de type year , le serveur MySQL la traitera comme 2001 .
- Toutes les fonctions de dates de MySQL sont stockées dans un fichier sql/time.cc , et sont codées très soigneusement pour être compatibles avec l'an 2000.
- En MySQL version 3.22 et plus récent, le type de colonne YEAR peut stocker les valeurs 0 et de 1901 à 2155 sur un seul octet, tout en affichant 2 ou 4 chiffres.

Vous pouvez rencontrer des problèmes avec les applications qui utilisent le serveur MySQL sans être compatible avec l'an 2000. Par exemple, les vieilles applications utilisent des valeurs d'années sur deux chiffres (ce qui est ambigu), plutôt qu'avec 4 chiffres. Ce problème peut être complété par des applications qui utilisent des valeurs telles que 00 ou 99 comme indicateur de données ``manquante''.

Malheureusement, ces problèmes peuvent se révéler difficiles à corriger car différentes applications peuvent être écrites par différents programmeurs, et chacun utilise un jeu différent de conventions et de fonctions de gestion des dates.

Voici une illustration simple qui montre que le serveur MySQL n'a aucun problème avec les dates jusqu'en 2030 :

```
 mysql> DROP TABLE IF EXISTS y2k;
Query OK, 0 rows affected (0.01 sec)

 mysql> CREATE TABLE y2k (date DATE,
    ->                   date_time DATETIME,
    ->                   time_stamp TIMESTAMP);
Query OK, 0 rows affected (0.00 sec)

 mysql> INSERT INTO y2k VALUES
    -> ("1998-12-31","1998-12-31 23:59:59",19981231235959),
    -> ("1999-01-01","1999-01-01 00:00:00",19990101000000),
    -> ("1999-09-09","1999-09-09 23:59:59",19990909235959),
    -> ("2000-01-01","2000-01-01 00:00:00",20000101000000),
    -> ("2000-02-28","2000-02-28 00:00:00",20000228000000),
    -> ("2000-02-29","2000-02-29 00:00:00",20000229000000),
    -> ("2000-03-01","2000-03-01 00:00:00",20000301000000),
    -> ("2000-12-31","2000-12-31 23:59:59",20001231235959),
    -> ("2001-01-01","2001-01-01 00:00:00",20010101000000),
    -> ("2004-12-31","2004-12-31 23:59:59",20041231235959),
```

```
    -> ("2005-01-01","2005-01-01 00:00:00",20050101000000),
    -> ("2030-01-01","2030-01-01 00:00:00",20300101000000),
    -> ("2050-01-01","2050-01-01 00:00:00",20500101000000);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0

 mysql> SELECT * FROM y2k;
+------------+---------------------+----------------+
| date       | date_time           | time_stamp     |
+------------+---------------------+----------------+
| 1998-12-31 | 1998-12-31 23:59:59 | 19981231235959 |
| 1999-01-01 | 1999-01-01 00:00:00 | 19990101000000 |
| 1999-09-09 | 1999-09-09 23:59:59 | 19990909235959 |
| 2000-01-01 | 2000-01-01 00:00:00 | 20000101000000 |
| 2000-02-28 | 2000-02-28 00:00:00 | 20000228000000 |
| 2000-02-29 | 2000-02-29 00:00:00 | 20000229000000 |
| 2000-03-01 | 2000-03-01 00:00:00 | 20000301000000 |
| 2000-12-31 | 2000-12-31 23:59:59 | 20001231235959 |
| 2001-01-01 | 2001-01-01 00:00:00 | 20010101000000 |
| 2004-12-31 | 2004-12-31 23:59:59 | 20041231235959 |
| 2005-01-01 | 2005-01-01 00:00:00 | 20050101000000 |
| 2030-01-01 | 2030-01-01 00:00:00 | 20300101000000 |
| 2050-01-01 | 2050-01-01 00:00:00 | 00000000000000 |
+------------+---------------------+----------------+
13 rows in set (0.00 sec)
```

Cet exemple montre que les types `DATE` et `DATETIME` ne poseront aucun problème avec les dates futures (ils gèrent les dates jusqu'en 9999).Le type `TIMESTAMP` , qui est utilisé pour stocker la date courante, est valide jusqu'en `2030-01-01` . `TIMESTAMP` va de `1970` en `2030` sur les machines 32 bits (valeur signée). Sur les machines 64 bits, il gère les dates jusqu'en `2106` (valeur non signée).

Même si le `serveur MySQL` est compatible an 2000, il est de votre responsabilité de fournir des données non ambiguës. Voyez Problèmes de l'an 2000 pour les règles du `serveur MySQL` pour traiter les dates ambiguës (les données contenant des années exprimées sur deux chiffres).

1.3 Qui est MySQL AB ?

`MySQL AB` est l'entreprise des fondateurs de `MySQL` et les principaux développeurs. A l'origine, `MySQL AB` a été établie en Suède, par David Axmark, Allan Larsson et Michael `Monty` Widenius.

Tous les développeurs du serveur `MySQL` sont employés par l'entreprise. Nous sommes une organisation virtuelle, avec des employés répartis dans une douzaine de pays à travers le monde. Nous communiquons intensivement entre nous sur l'Internet tous les jours, et avec nos utilisateurs, amateurs et partenaires.

Nous nous consacrons au développement du logiciel `MySQL` et à la diffusion de notre base de données auprès des nouveaux utilisateurs. `MySQL AB` est propriétaire des droits du code source de `MySQL` , du logo `MySQL` et de la marque de commerce, ainsi que du manuel. Qu'est ce que MySQL ? .

Les valeurs fondamentales de `MySQL` témoignent de notre implication auprès de `MySQL` et des `Logiciels libres` .

Nous souhaitons que `la base de données MySQL` soit :

- Le meilleur serveur de bases de données dans le monde et le plus répandu.
- Le rendre accessible à tous.
- Facile à utiliser.
- En amélioration constante, tout en restant rapide et sécuritaire.
- Plaisant à utiliser et améliorer.
- Sans aucun bogue.

`MySQL AB` et les collaborateurs de `MySQL AB` :

- Font la promotion des `logiciels libres` et supportent la communauté des `Logiciels libres` .

- Sont de bons citoyens.
- Préfèrent les partenaires qui partagent nos valeurs et notre état d'esprit.
- Répondent aux emails et fournissent de l'aide.
- Sont une entreprise virtuelle, qui travaille en réseau avec les autres.
- Militent contre les brevets logiciels.

Le site web de `MySQL` (http://www.mysql.com/) fournit les dernières informations à propos de `MySQL` et `MySQL AB` .

1.3.1 Les services et le modèle d'affaire de MySQL AB

Une des questions les plus fréquentes que nous rencontrons est : `` **Comment arrivez−vous à vivre en développant un produit gratuit ?** '' Voici comment.

- `MySQL AB` fait des affaires en vendant du support, des services, des licences commerciales et en percevant des royalties.
- Nous utilisons ces fonds pour le développement des produits et des affaires de `MySQL` .

La compagnie est profitable depuis sa conception. En octobre 2001, nous avons accepté un financement de la part d'un groupe d'investisseurs scandinaves importants et de quelques business angels. Cet investissement est utilisé pour consolider notre modèle d'affaires et assurer les bases de notre croissance à long terme.

1.3.1.1 Support

`MySQL AB` est dirigé par ses propriétaires, qui sont les fondateurs et les principaux développeurs de la base de données `MySQL` . Les développeurs se consacrent au support des utilisateurs et des autres utilisateurs, afin de rester au courant de leurs besoins et leurs problèmes. Tout notre support est donné par des développeurs qualifiés. Les questions vraiment épineuses sont étudiées par Michael `Monty` Widenius, auteur principal du code du `serveur MySQL` .

Les clients payant reçoive une attention particulière du support de MySQL AB. MySQL AB fournit aussi des listes de diffusions MySQL comme ressource pour la communauté, où tout le monde peut poser des questions.

Pour plus d'informations et pour commander un support de différents niveaux, voyez la section Support et licences MySQL .

1.3.1.2 Formation et certification

`MySQL AB` organise des formations `MySQL` à travers le monde entier. Nous offrons des cours inter− et intra−entreprise, adaptés aux besoins spécifiques de chaque société. La `formation MySQL` est aussi disponible auprès de nos partenaires, les `centres de formation certifiés MySQL` .

Nos documents de formation utilisent les mêmes exemples de bases de données que notre documentation et nos applications d'exemple, et ils sont toujours mis à jour pour prendre en compte les dernières versions de `MySQL` . Nos formateurs sont épaulés par notre équipe de développement pour garantir la qualité de la formation et le développement continu des documents de cours. Cela vous assure aussi qu'il n'y aura pas de questions laissées ouvertes durant les cours.

Suivre nos formations vous permettra d'atteindre tous vos buts avec votre application `MySQL` . Vous allez aussi :

- Gagner du temps.
- Améliorer les performances de vos applications.
- Réduire ou éliminer les besoins en matériels supplémentaires, ce qui réduira vos coûts.
- Améliorer la sécurité.
- Améliorer la satisfaction de vos clients et collaborateurs.
- Vous préparer à la certification MySQL .

Si vous êtes intéressé par nos formations, en tant que participant potentiel, ou comme partenaire de formation, visitez la section de formation à http://www.mysql.com/training/ ou contactez−nous à : training@mysql.com .

Le programme de certification MySQL est publié dans le second semestre 2002. Pour plus de détails, voyez http://www.mysql.com/certification/ .

1.3.1.3 Conseil

MySQL AB et ses partenaires accrédités offrent des services de conseil aux utilisateurs du serveur MySQL et à ceux qui intègrent le serveur MySQL dans leurs logiciels, à travers le monde.

Nos consultants peuvent vous aider à concevoir et paramétrer vos bases, construire des requêtes efficaces, optimiser votre plate−forme, résoudre les problèmes de migration, installer la réplication, bâtir des applications transactionnelles robustes et bien plus encore. Nous aidons aussi les clients à intégrer le serveur MySQL dans leurs produits et applications, pour un déploiement d'envergure.

Nos consultants travaillent en collaboration étroite avec notre équipe de développement pour assurer la qualité technique de nos services professionnels. Les missions de conseil peuvent aller de sessions de démarrage de deux jours à des projets de plusieurs semaines ou mois. Notre expertise couvre non seulement le serveur MySQL , mais s'étend aussi aux langages de programmation tels que PHP, Perl et d'autres encore.

Si vous êtes intéressé par nos services de conseil ou si vous souhaitez devenir un partenaire conseil, visitez la section conseil de notre site web à http://www.mysql.com/consulting/ ou contactez notre équipe de conseil à consulting@mysql.com .

1.3.1.4 Licences commerciales

La base de données MySQL est publiée sous la licence GNU General Public License (GPL). Cela signifie que le logiciel MySQL peut être utilisé gratuitement, en acceptant les termes de la licence GPL . Si vous ne voulez pas être lié par les termes de la licence GPL (comme le fait que votre application aussi doit être GPL), vous pouvez acheter une licence du même produit auprès de MySQL AB . Voyez http://www.mysql.com/support/arrangements/price.html . Comme MySQL AB est propriétaire du copyright du code source de MySQL , nous pouvons utiliser une double licence , qui fait que le même produit est disponible sous la licence GPL et sous une licence commerciale. Cela ne change en rien l'implication de MySQL AB dans le mouvement des logiciels libres . Pour plus de détails sur quand une licence commerciale est nécessaire, voyez Licences MySQL .

Nous vendons aussi des licences commerciales aux logiciels Open Source GPL qui ajoutent à la valeur du serveur serveur MySQL . Un bon exemple est le gestionnaire de table transactionnel InnoDB qui offre le support ACID , le verrouillage de ligne, la restauration après crash, le multi−versionnage, le support des clés étrangères, etc. Le moteur de tables InnoDB .

1.3.1.5 Partenariats

MySQL AB a un programme de partenariat mondial qui couvre la formation, le conseil et le support, les publications, la revente et la distribution des produits MySQL . Les partenaires MySQL AB gagnent en visibilité grâce au site http://www.mysql.com/ et le droit d'utiliser certaines versions spéciales des marques de commerce MySQL pour identifier leurs produits et promouvoir leur entreprise.

Si vous êtes intéressé à devenir un partenaire MySQL AB , envoyez un email à partner@mysql.com .

Le mot MySQL et le logo MySQL avec le dauphin sont des marques commerciales de MySQL AB . Logos MySQL AB et marques déposées . Ces marques représentent un investissement capital que les fondateurs de MySQL ont placé depuis plusieurs années.

Le site web MySQL (http://www.mysql.com/) est très populaire auprès des développeurs et utilisateurs. En octobre 2001, nous avons servi 10 millions de pages. Nos visiteurs représente un groupe capable de prendre des décisions d'achat, et de faire des recommandations de logiciels et de matériel. 12% de nos visiteurs autorise les décisions d'achat, et seulement 9% ne sont pas du tout impliqué dans les décisions d'achats. Plus de 65% ont réalisés des achats professionnels en ligne durant les derniers 6 mois, et 70% prévoit de le faire dans les 6 prochains mois.

1.3.2 Contacts

Le site web de MySQL (http://www.mysql.com/) fournit les dernières informations à propos de MySQL et MySQL AB .

Pour les contacts presse et les questions qui ne sont pas couvertes par les annonces officielles (http://www.mysql.com/news/), envoyez un email à press@mysql.com .

Si vous avez un contrat de support valide avec MySQL AB , vous obtiendrez des réponses rapides et précises de notre équipe technique sur le logiciel MySQL . Pour plus d'informations, voyez Support offert par MySQL AB . Sur notre site web, voyez http://www.mysql.com/support/ , ou envoyez un email à sales@mysql.com .

Pour des informations sur les formations MySQL , visitez la section formation à http://www.mysql.com/training/ . Si vous avez un accès restreint à Internet, contactez l'équipe de formation de MySQL AB à l'adresse training@mysql.com . Formations et certifications .

Pour des informations sur le programme de certification MySQL , voyez la section http://www.mysql.com/certification/ . Formations et certifications .

Si vous êtes intéressé par du conseil, visitez la section conseil à http://www.mysql.com/consulting/ . Si vous avez un accès restreint à Internet, contactez l'équipe de conseil de MySQL AB à consulting@mysql.com . Conseil .

Les licences commerciales peuvent être commandées en ligne à http://order.mysql.com/ . Vous trouverez aussi des informations sur les commandes par fax à MySQL AB . Plus d'informations sur les licences sont disponibles à http://www.mysql.com/products/pricing.html . Si vous avez des questions qui concernent les licences ou que vous souhaitez un devis pour un achat de nombreuses licences, remplissez le formulaire de contact disponible sur le site web (http://www.mysql.com/) ou envoyez un email à licensing@mysql.com (pour les questions de licence) ou à sales@mysql.com (pour les devis). Licences MySQL .

Si vous représentez une entreprise qui est intéressée par un partenariat avec MySQL AB , envoyez un email à partner@mysql.com . Partenariats .

Pour plus d'informations sur la politique de marque de commerce de MySQL , reportez−vous à http://www.mysql.com/company/trademark.html ou envoyez un email à trademark@mysql.com . Logos MySQL AB et marques déposées .

Si vous êtes intéressé par un des emplois à MySQL AB présentés dans notre section job (http://www.mysql.com/company/jobs/), envoyez un email à jobs@mysql.com . N'envoyez pas votre CV en pièce jointe, mais collez−le dans le corps du texte à la fin de votre message.

Pour des discussions générales entre utilisateurs, dirigez vos questions sur la liste de diffusion appropriée. Liste de diffusion MySQL .

Les rapports d'erreurs (aussi appelés bogues), ainsi que les questions et suggestions doivent être envoyés à les liste de diffusion. Les listes de diffusion MySQL . Si vous avez trouvé un trou de sécurité critique dans le serveur MySQL , envoyez votre courriel à security@mysql.com . Comment rapporter des bogues ou problèmes .

Si vous avez un résultat de tests que nous pouvons publier, contactez−nous à benchmarks@mysql.com .

Si vous avez des suggestions concernant les améliorations ou les corrections de ce manuel, envoyez−les à l'équipe du manuel docs@mysql.com . Pour les remarques spécifiques à la version française, vous pouvez aussi contacter directement Damien Seguy à damien.seguy@nexen.net .

Pour les questions ou commentaires à propos du fonctionnement ou du contenu du site web MySQL (http://www.mysql.com/), envoyez un email à webmaster@mysql.com .

MySQL AB a une politique de protection des données privées qui est présentée à http://www.mysql.com/company/privacy.html . Pour toutes les questions concernant cette politique, envoyez un email à privacy@mysql.com .

Pour toutes les autres questions, envoyez un email à info@mysql.com .

1.4 Support MySQL et licences

Cette section décrit le support MySQL et les accords de licence.

1.4.1 Support proposé par MySQL AB

Le support technique de MySQL AB est la réponse individualisée à vos problèmes particuliers, en direct, de l'équipe d'ingénieurs qui programme la base de données MySQL .

Nous tâchons d'avoir un support technique exhaustif et global. Presque tous les problèmes qui impliquent MySQL sont importants pour nous, s'ils sont importants pour vous. Typiquement, les clients qui recherchent de l'aide sur les différentes commandes, qui souhaitent résoudre des problèmes de performance, réparer des systèmes corrompus, comprendre les impacts des systèmes d'exploitation ou des réseaux sur les performances de MySQL , mettre en place des bonnes pratiques pour la sauvegarde et l'entretien, utiliser les API s, etc. Notre support couvre uniquement le serveur MySQL et nos propres utilitaires, pas les produits tiers d'entreprises qui accèdent au serveur MySQL , même si nous pouvons parfois aider.

Les informations détaillées sur les différents niveaux de support sont disponibles à http://www.mysql.com/support/ , et les contrats de support peuvent aussi être commandés en ligne. Si vous avez un accès restreint à Internet, vous pouvez contacter notre équipe commerciale à sales@mysql.com .

Le support technique est identique à l'assurance vie. Vous pouvez vivre très heureux sans pendant plusieurs années, mais lorsque vous rencontrerez une catastrophe, il sera trop tard pour l'acheter. Si vous utilisez `MySQL` pour des applications importantes et que vous rencontrez des problèmes, cela peut vous prendre très longtemps pour comprendre par vous−même. Vous pourriez alors avoir besoin de communiquer avec les techniciens les plus expérimentés de `MySQL` , ceux qui sont employés par `MySQL AB` .

1.4.2 Copyrights et licences utilisées par MySQL

`MySQL AB` est propriétaire du copyright du code source de `MySQL` , des logos `MySQL` , de la marque de commerce et de ce manuel. Qu'est ce que MySQL AB ? . Plusieurs licences distinctes sont disponibles pour la diffusion de `MySQL` :

- Toutes les sources spécifiques de `MySQL` pour le serveur, le client `mysqlclient` et la bibliothèque, ainsi que la bibliothèque `GNU readline` sont couverts par la licence `GNU General Public License` . Licence GNU . Le texte de cette licence est aussi disponible dans le fichier intitulé `COPYING` dans les distributions.
- La bibliothèque `GNU getopt` est sous la licence `GNU Lesser General Public License` . Voyez http://www.fsf.org/licenses/ .
- Certaines parties du code source (la bibliothèque `regexp`) sont placées sous un copyright de type Berkeley.
- Les anciennes versions de `MySQL` (3.22 et plus récentes) sont sujettes à des licences plus strictes (http://www.mysql.com/support/arrangements/mypl.html). Voyez la documentation spécifique de chaque version pour plus de détails.
- La manuel n'est actuellement **pas** distribué sous la licence `GPL` . L'utilisation du manuel est sujette à ces conditions :
    - Les conversions en d'autres formats sont autorisées, mais le contenu ne doit pas être modifié ou édité de quelque manière que ce soit.
    - Vous pouvez créer une version imprimée pour votre utilisation personnelle.
    - Pour toutes les autres utilisations, telles que la vente de versions imprimées, ou l'utilisation (en partie ou en totalité) du manuel dans d'autres publications, un accord doit être conclu au préalable avec `MySQL AB` .

  Communiquez par e−mail avec docs@mysql.com pour plus d'informations, ou si vous êtes intéressé par la traduction du manuel.

Pour plus de détails sur comment les licences `MySQL` fonctionnent, voyez Licences MySQL . Voyez aussi Logos MySQL AB et marques déposées .

1.4.3 Licences MySQL

Le logiciel `MySQL` est publié sous la licence `GNU General Public License` (`GPL`), qui est probablement mieux connue sous le nom de `Open Source` . Les termes exacts de la licence `GPL` sont disponibles sur le site de http://www.fsf.org/licenses/ . Voyez aussi http://www.gnu.org/licenses/gpl−faq.html et http://www.gnu.org/philosophy/enforcing−gpl.html .

Notre licence GPL recèle une exception qui permet à de nombreux projets Free/Libre et Open Source Software (``FLOSS'') d'inclure les bibliothèques clientes GPL de MySQL, malgré l'absence de compatibilité avec de nombreuses licences FLOSS. Pour plus de détails, voyez http://www.mysql.com/products/licensing/foss−exception.html .

Comme le logiciel MySQL est publié sous la licence GPL , il est souvent utilisé gratuitement, mais pour certains usages vous souhaiterez peut−être acheter une licence commerciale auprès de MySQL AB à http://order.mysql.com/ . Voyez http://www.mysql.com/support/arrangements.html pour plus d'informations.

Les anciennes versions de MySQL (3.22 et plus anciennes) sont sujettes à une licence plus stricte (http://www.mysql.com/support/arrangements/mypl.html). Voyez la documentation spécifique de chaque version pour information.

Notez bien que l'utilisation du logiciel MySQL sous une licence commerciale. GPL , ou toute autre ancienne licence MySQL ne vous donne pas automatiquement le droit d'utiliser les marques commerciales de MySQL AB . Logos MySQL AB et marques déposées .

1.4.3.1 Utiliser MySQL avec la licence commerciale

La licence GPL est contagieuse, dans le sens où lorsqu'un programme est lié à la licence GPL , toutes les sources de toutes les parties du produit final doivent aussi être publiée sous la licence GPL . Sinon, vous violez la licence, et annulez vos droits d'utiliser le programme GPL .

Vous avez besoin d'une licence commerciale dans un de ces cas :

- Lorsque vous associez un programme à un logiciel GPL issu de MySQL et que vous ne voulez pas que le produit final soit publié sous la licence GPL , peut être parce que vous souhaitez publier un produit commercial ou conserver du code non− GPL pour d'autres raisons. Lorsque vous achetez une licence commerciale vous n'utilisez plus MySQL sous la licence GPL , même si c'est le même code.
- Lorsque vous distribuez une application non− GPL qui fonctionne **uniquement** avec MySQL , et que vous distribuez cette application avec MySQL . Ce type de solution est considéré comme un lien, même si c'est fait via le réseau.
- Lorsque vous distribuez des copies de MySQL sans fournir le code source original, comme requis par la licence GPL .
- Lorsque vous voulez supporter le développement de MySQL même si vous n'avez pas besoin formellement de la licence commerciale. Acheter du support auprès de MySQL AB est une autre bonne solution pour contribuer au développement de MySQL , avec des avantages directs pour vous. Support offert par MySQL AB .

Notre licence GPL recèle une exception qui permet à de nombreux projets Free/Libre et Open Source Software (`` FLOSS '') d'inclure les bibliothèques clientes GPL de MySQL, malgré l'absence de compatibilité avec de nombreuses licences FLOSS . Pour plus de détails, voyez http://www.mysql.com/products/licensing/foss−exception.html .

Si vous avez besoin d'une licence, vous en aurez besoin d'une pour chaque installation de MySQL . Cela est valable quelque soit le nombre de processeurs de la machine, et il n'y a pas de limite artificielle de nombre de connexion simultanées.

Pour les licences commerciales, voyez notre site web http://www.mysql.com/support/arrangements/price.html . Pour les contrats de support, voyez http://www.mysql.com/support/ . Si vous avez des besoin spéciaux, ou que vous avez un accès restreint à internet, contactez notre équipe de vente à sales@mysql.com .

1.4.3.2 Utiliser MySQL sous licence GPL libre

Vous pouvez utiliser le logiciel MySQL sous la licence gratuite GPL si vous acceptez les termes et conditions de la licence GPL . Pour une liste exhaustive des questions courantes à propos de la licence GPL , voyez la FAQ générale de la Free Software Foundation à http://www.gnu.org/licenses/gpl−faq.html .

Notre licence GPL est recèle une exception qui permet à de nombreux projets Free/Libre et Open Source Software (`` FLOSS '') d'inclure les bibliothèques clientes GPL de MySQL, malgré l'absence de compatibilité avec de nombreuses licences FLOSS . Pour plus de détails, voyez http://www.mysql.com/products/licensing/foss−exception.html .

Quelques cas courants :

- Lorsque vous distribuez votre application et le code source MySQL , ensemble avec la licence GPL .
- Lorsque vous distribuez le code source MySQL avec d'autres programmes qui ne sont pas liés ou dépendant de MySQL pour leur fonctionnalités, même si vous vendez la distribution commercialement. Ceci s'appelle une agrégation simple en termes GPL .
- Si vous ne distribuez **aucune** partie de MySQL , vous pouvez l'utiliser gratuitement.
- Lorsque vous êtes un fournisseur de services Internet (ISP), qui offre un hébergement web avec le MySQL . Cependant, nous vous encourageons à faire affaires avec un ISP qui dispose du support MySQL, car cela vous donne l'assurance que si votre hébergeur a des problèmes avec l'installation de MySQL , votre hébergeur aura les ressources pour résoudre votre problème. Notez que si votre ISP n'a pas de licence commerciale pour le le MySQL , il devrait au moins fournir l'accès en lecture aux sources aux clients pour qu'ils vérifient que l'installation de MySQL est correctement corrigé.
- Lorsque vous utilisez le serveur de bases de données MySQL en conjonction avec un serveur web, vous n'avez pas besoin de licence commerciale (aussi longtemps que ce n'est pas un produit que vous distribuez). Ceci est même vrai si vous avec un serveur web qui utilise le serveur MySQL , car vous ne distribuez pas de partie de MySQL . Cependant, dans ce cas, nous vous demandons d'acheter du support MySQL car MySQL aide votre entreprise

Si vous utilisez le serveur de bases de données MySQL sans avoir besoin de licence commerciale, nous vous encourageons à acheter le support auprès de MySQL AB malgré tout. De cette façon, vous contribuez au développement de MySQL et en obtenez des avantages immédiats vous−même. Support offert par MySQL AB .

Si vous utilisez MySQL dans un contexte commercial tel que vous en tirez profit, nous vous demandons de participer au développement de MySQL en achetant du support. Nous pensons que comme MySQL aide votre entreprise, il est raisonnable de vous demander d'aider à votre tour MySQL AB . (Sinon, lorsque vous nous posez des questions de support, non seulement vous utilisez gratuitement un système dans lequel nous avons investi beaucoup de temps, mais en plus, vous nous demandez du support gratuit, en plus !)

1.4.4 Logos MySQL AB et marque déposée

Beaucoup d'utilisateurs de MySQL souhaitent afficher le logo du dauphin MySQL AB sur leur sites web, leur livres ou leurs produits. Nous encourageons ces actes, tant qu'on part du principe que le mot MySQL et le logo du dauphin MySQL sont des marques déposées par MySQL AB et ne doivent être utilisés que dans les conditions décrites à la page suivante : http://www.mysql.com/company/trademark.html .

1.4.4.1 Le logo original de MySQL

Le dauphin MySQL a été conçu par l'agence de publicité finlandaise Priority, en 2001. Le dauphin a été choisi en tant que symbole représentatif de MySQL , car c'est un animal intelligent, rapide et élancé, qui parcourt sans effort l'océan des données. Nous aimons aussi les dauphins.

Le logo original de MySQL ne peut être utilisé que par les représentants de MySQL AB et par ceux qui ont un accord signé leur permettant de le faire.

1.4.4.2 Logos MySQL qui peuvent être utilisés dans autorisation préalable

Nous avons conçu un jeu de logos qui peuvent être utilisés **sous conditions** , et qui peuvent être téléchargés depuis notre site web à http://www.mysql.com/press/logos.html et utilisés sur les sites web tiers sans autorisation écrite de MySQL AB . L'utilisation de ces logos n'est pas inconditionnelle, mais, comme leur nom l'implique, sujette à notre politique de marque de commerce qui est aussi disponible sur notre site web. Il est recommandé de lire ce document avant d'utiliser les logos sur votre site web. En bref, les pre−requis sont :

- Utilisez le logo que vous souhaitez tel que présenté sur le site web http://www.mysql.com/ . Vous pouvez l'étendre ou le réduire suivant vos besoins, mais ne modifier ni les couleurs, ni le design, ni le graphisme en aucune façon.
- Faîtes en sorte qu'il soit évident que vous, et non pas MySQL AB , êtes le créateur de ce site, qui arbore les couleurs de MySQL .
- N'utilisez pas le logo d'une façon qui porte préjudice à MySQL AB ou aux valeurs de MySQL AB . Nous nous réservons le droit de retirer notre autorisation d'utiliser le logo de MySQL AB .
- Si vous utilisez le logo sur un site web, rendez−le cliquable, avec un lien sur le site http://www.mysql.com/ .
- Si vous utilisez le serveur web MySQL sous licence GPL dans votre application, votre application doit porter la mention Open Source et être capable de se connecter à un serveur MySQL .

Contactez−nous à trademark@mysql.com pour voir avec nous tous les arrangements spéciaux qui vous conviendraient.

1.4.4.3 Quand avez vous besoin d'autorisation pour utiliser le logo MySQL?

Dans les cas suivants vous devez obtenir une permission écrite de MySQL AB avant d'utiliser les logos MySQL :

- Lorsque vous affichez un des logos MySQL AB n'importe où, en dehors de votre site web.
- Lorsque vous affichez l'un des logos MySQL AB hormis les logos à **utilisation conditionnée** mentionnés auparavant sur le site web, ou ailleurs

En dehors des raisons commerciales et légales, nous nous devons de suivre l'utilisation du logo MySQL sur les produits, livres, etc. Nous demanderons une compensation pour l'affichage des logos MySQL AB sur les produits, car nous pensons qu'il est raisonnable qu'une partie des revenus ainsi générés servent à poursuivre le développement de la base de données MySQL .

1.4.4.4 Logos de partenariat MySQL AB

Le logo partenariat MySQL ne doit être utilisé que par les compagnies et les personnes ayant un partenariat écrit avec MySQL AB Le partenariat inclut une certification en tant que consultant ou professeur MySQL .

Merci de visiter Partenariats .

1.4.4.5 Utiliser le nom MySQL sur des documents imprimés ou des présentations

MySQL AB apprécie les références à la base MySQL . Il faut toutefois garder à l'esprit que MySQL est une marque commerciale, propriété de MySQL AB . A cause de cela, il faut ajouter au logo le symbole de marque de commerce (™) lors de la première ou de la plus visible utilisation du mot MySQL dans le texte et, là où approprié, établir clairement que MySQL est une marque commerciale de MySQL AB . Reportez−vous à notre politique de marque commerciale à http://www.mysql.com/company/trademark.html pour plus de détails.

1.4.4.6 Utilisation du nom MySQL dans un nom de société ou de produit

L'utilisation du mot MySQL dans le nom d'une compagnie, d'un produit ou d'un nom de domaine Internet est interdite sans une permission écrite de MySQL AB .

1.5 MySQL Plan de développement

Cette section donne un aperçu du plan de développement de MySQL, incluant les futures fonctionnalités prévues pour MySQL 4.0, 4.1, 5.0 et 5.1. Les sections suivantes donnent plus de détails sur chaque version.

La série de production est MySQL 4.0, qui a été déclarée stable pour un environnement de production depuis la version 4.0.12, publiée en Mars 2003. Cela signifie que les développements futurs de la série des 4.0 est limitée aux corrections de bugs. Pour les anciennes version 3.23, seuls les bugs critiques seront corrigés.

L'effort de développement MySQL a lieu actuellement dans les versions MySQL 4.1 et 5.0. Cela signifie que les nouvelles fonctionnalités sont ajoutées aux versions 4.1 et 5.0. Les versions 4.1 et 5.0 sont disponibles en version alpha.

Avant de mettre à jour une version vers une autre, lisez les notes de la section Mise à jour ou rétrogradage de MySQL .

Les plans de certains fonctionnalités sont résumés dans cette table.

Fonctionnalité	**version MySQL**
Unions	4.0
Sous−requêtes	4.1
R−trees	4.1 (pour les tables MyISAM)
Procédures stockées	5.0
Vues	5.0 ou 5.1
Curseurs	5.0
Clés étrangères	5.1 (déjà implémentés en 3.23 par InnoDB)
Triggers	5.1
Jointures externes	5.1
Contraintes	5.1

1.5.1 MySQL 4.0 en bref

Promise depuis longtemps par MySQL AB et attendue avec impatience par nos utilisateur, le serveur

MySQL 4.0 est disponible en version de production.

MySQL 4.0 est disponible au téléchargement depuis http://www.mysql.com/ et nos miroirs. MySQL 4.0 a été testée par un grand nombre d'utilisateurs et il est en production sur de très grands sites.

Les fonctionnalités principales de MySQL serveur 4.0 sont destinées à nos utilisateurs professionnels et communautaire : elles améliorent le capacités de MySQL pour gérer les missions critiques et les systèmes fortement chargés. D'autres fonctionnalités sont destinées aux utilisateurs de solutions intégrées.

1.5.1.1 Fonctionnalités disponibles en MySQL 4.0

Amélioration des performances

MySQL 4.0 dispose d'un cache de requêtes qui peut vous accélérer grandement vos applications qui utilisent souvent les mêmes requêtes. Le cache de requêtes MySQL .

La version 4.0 accélère la vitesse du serveur MySQL dans de nombreux domaines, notamment les `INSERT` de masse, la recherche sur les index compressés, la création d'index `FULLTEXT` ainsi que les comptes `COUNT(DISTINCT)` .

Serveur MySQL embarqué

La nouvelle bibliothèque `Embedded Server` (au lieu de client/serveur) peut être facilement utilisée pour créer des applications indépendantes ou intégrées. Bibliothèque MySQL embarquée .

Le moteur `InnoDB` en standard

Le moteur de tables `InnoDB` est désormais livré en standard avec le serveur MySQL, apportant le support complet des transactions ACID, les clés étrangères avec modifications et effacement en cascade, ainsi que le verrouillage de ligne. Le moteur de tables `InnoDB` .

Nouvelles fonctionnalités

Les nouvelles possibilités de recherche en `FULLTEXT` de MySQL Serveur 4.0 permettent l'utilisation d'index `FULLTEXT` sur de grandes quantités de texte, avec des logiques binaires ou en langage naturel. Les utilisateurs peuvent paramétrer la taille minimum des mots, et définir leur propre liste de mots interdits, dans n'importe quel langue. Cela ouvre la possibilité de nombreuses applications avec MySQL Serveur. Fonctions de recherche en texte plein .

Respect des standards, portabilité et migration

Simplification de la migration depuis d'autres bases de données vers MySQL Serveur, et notamment `TRUNCATE TABLE` (comme sous Oracle) et `IDENTITY` comme synonyme pour les clés automatiquement incrémentées (comme sous Sybase).

De nombreux utilisateurs seront heureux de savoir que le serveur MySQL supporte aussi les requêtes `UNION` , une fonctionnalité SQL attendue avec impatience.

MySQL peut s'exécuter nativement sur les plates−formes NetWare 6.0. Installer MySQL sous NetWare .

Internationalisation

Nos utilisateurs allemands, autrichiens et suisses remarqueront que nous avons un nouveau jeu de caractères, `latin1_de` , qui corrige les problèmes de **tri des valeurs allemandes** , en plaçant les umlauts allemands dans le même ordre que dans l'annuaire d'Allemagne.

Amélioration de l'ergonomie

Durant la mise en place de fonctionnalités pour de nouveaux utilisateurs, nous n'avons pas oublié notre communauté de loyaux utilisateurs.

Une fonctionnalité pratique pour les administrateurs de base de données est que la plupart des paramètres de démarrage de `mysqld` peuvent être modifiées sans redémarrer le serveur. Syntaxe de `SET` .
Les commandes `DELETE` et `UPDATE` peuvent désormais fonctionner sur plusieurs tables.
En ajoutant le support des `liens symboliques` à `MyISAM` au niveau des tables (et non plus au niveau des bases, comme auparavant), et en autorisant les liens symboliques sur Windows, nous espérons que nous avons pris au sérieux vos demandes d'amélioration.
Des fonctions comme `SQL_CALC_FOUND_ROWS` et `FOUND_ROWS()` rendent possible le comptages de lignes sans utiliser la clause `LIMIT` .

La section sur les nouveautés du manuel rassemble toutes les nouveautés. Changements de la version 4.0.x (Production) .

1.5.1.2 MySQL Server intégré (embedded)

`libmysqld` rend le serveur MySQL disponible pour toute une gamme d'applications très vaste. En utilisant la bibliothèque du serveur MySQL intégré, vous pouvez utiliser MySQL dans différentes applications et appareillages, où l'utilisateur final n'aura même pas idée de sa présence. Le serveur MySQL intégré est idéal pour équiper les bornes internet, les kiosques publics, les paquets matériel/ logiciels clé en main, les serveurs MySQL haute performances, et les bases de données autonomes sur CDrom.

De nombreux utilisateurs de `libmysqld` profiteront de la **double licence** . Pour ceux qui ne souhaitent pas être liés par la licence GPL, la bibliothèque est aussi disponible avec une licence commerciale. La bibliothèque MySQL intégrée utilise la même interface que la bibliothèque cliente classique, ce qui la rend pratique à utiliser. libmysqld, la bibliothèque intégrée MySQL .

1.5.2 MySQL 4.1 en bref

MySQL 4.0 a posé les fondations pour de nouvelles fonctionnalités telles que les sous−requêtes imbriquées et l'Unicode qui sont d'ores et déjà implémentées en version 4.1, ainsi que les procédures stockées SQL−99, qui seront disponibles pour la version 5.0. Ils représentent les fonctionnalités les plus demandées par de nombreux clients.

Avec ces améliorations, les critiques du serveur de base de données MySQL devront être plus imaginatifs que jamais pour identifier des manques dans le serveur MySQL. Déjà connu depuis longtemps pour sa stabilité, sa rapidité et sa facilité d'emploi, le serveur MySQL va désormais satisfaire la liste de tous les voeux des clients les plus exigeants.

1.5.2.1 Fonctionnalités disponibles en MySQL 4.1

Les fonctionnalités ci−dessous sont implémentées en MySQL 4.1. Quelques autres fonctionnalités sont prévues pour MySQL 4.1, mais très peu. Voyez Ce qui est prévu pour la version 4.1 .

Les plus récentes fonctionnalités en cours de réalisation, comme par exemple les procédures stockées, seront disponibles en MySQL 5.0. Ce qui est prévu pour la version 5.0 .

Support des sous−requêtes et tables dérivées
Une sous−requête est une commande `SELECT` imbriquée dans une autre

requête. Une table dérivée (une vue anonyme) est une sous−requête dans une clause `FROM` d'une autre commande. Syntaxe des sous−requêtes .

Accélération

Protocole binaire plus rapide, avec préparation des commandes et paramétrage. Commandes préparées en C .
Indexation `BTREE` pour les tables `HEAP` , ce qui améliore significativement le temps de réponse pour les recherches non exactes.

Nouvelle fonctionnalité

`CREATE TABLE table_name2 LIKE table_name1` vous permet de créer, avec une seule commande, une nouvelle table, avec une structure identique à celle d'une autre table existante.
Support pour les types géométriques OpenGIS (données géométriques). Extensions spatiale de MySQL .
La réplication peut être faite sur connexions SSL.

Compatibilité avec les standards, portabilité et migration

Le nouveau protocole client−serveur apporte la possibilité de faire passer plusieurs alertes au client, plutôt qu'une seule. Cela améliore grandement la gestion des erreurs lors des manipulations de masse.
`SHOW WARNINGS` affiche les erreurs de la dernière commande. Syntaxe de `SHOW WARNINGS` .

Internationalisation

Pour supporter notre base d'utilisateurs en pleine croissance, et leur configurations locales, MySQL exploite désormais l'Unicode (UTF8).
Les jeux de caractères peuvent désormais être définis par colonnes, tables et bases. Cela permet d'améliorer la souplesse dans la conception des applications, en particuliers pour les sites multi−langues.
Pour la documentation sur l'amélioration du support des jeux de caractères, voyez Jeux de caractères .

Améliorations d'ergonomie

En réponse à la demande populaire, nous avons ajouté une commande `HELP command` coté serveur, qui peut être utilisée en ligne de commande du client `mysql` et d'autres clients, pour obtenir de l'aide sur les commandes SQL. Avec ces informations sur le serveur, elles seront parfaitement adaptées à la version et configuration du serveur.
Avec le nouveau protocole client/serveur, les requêtes multiples sont désormais activées. Cela vous permet d'émettre plusieurs requêtes en une seule commande, puis de lire tous les résultats en une seule fois. Gestion des requêtes multiples en C .
Le nouveau protocole client/serveur supporte aussi les jeux de résultats multiples. Cela peut arriver après une commande multiple, par exemple. Voir le point précédent.
Nous avons implémenté une syntaxe pratique `INSERT ... ON DUPLICATE KEY UPDATE ...` . Elle vous permet de modifier une ligne avec `UPDATE` , si l'insertion `INSERT` avait généré un double dans la colonne `PRIMARY` ou `UNIQUE` . Syntaxe de `INSERT` .
Nous avons ajouté une fonction d'agrégation, `GROUP_CONCAT()` , qui permet de concaténer des colonnes dans une seule chaîne de résultat. Fonctions à utiliser dans les définitions `GROUP BY` .

La section sur les nouveautés du manuel rassemble toutes les nouveautés. Changements de la version 4.1.x (Alpha) .

1.5.2.2 Phases de publication

De nouvelles fonctionnalités sont ajoutées à MySQL 4.1, qui est déjà disponible dans notre système BitKeeper. Installer à partir des sources de l'arbre de développement .

Le jeu de fonctionnalités destiné à la version 4.1 est globalement définitif. Les nouveautés iront essentiellement à la nouvelle version de développement 5.0. MySQL 4.1 va suivre son développement normal avec une phase **Alpha** (durant laquelle de nouvelles fonctionnalités peuvent être ajoutées ou modifiées), puis **Beta** (où les fonctionnalités sont fixées, et seules des corrections sont faîtes), puis **Gamma** (qui indique qu'une version de production verra le jour dans quelques semaines), avant que la version 4.1 ne deviennent la nouvelle version officielle.

1.5.2.3 Utilisation immédiate en production

MySQL 4.1 est actuellement en phase alpha, et les paquets binaires sont disponibles en téléchargement sur http://www.mysql.com/downloads/mysql−4.1.html . Tous les paquets passent nos tests exhaustifs sans erreur, sur les plates−formes que nous utilisons. Changements de la version 4.1.x (Alpha) .

Pour ceux qui souhaitent utiliser les plus récents développement de MySQL 4.1, nous avons rendu notre serveur BitKeeper public. Installer à partir des sources de l'arbre de développement .

1.5.3 MySQL 5.0, les prochains développement

Les nouveaux développements de MySQL sont désormais concentrés sur la version 5.0. Les procédures stockées et d'autres fonctionnalités seront en vedette. Ce qui est prévu pour la version 5.0 .

Pour ceux qui veulent jeter un oeil aux tout derniers développements de MySQL, nous avons rendu notre serveur BitKeeper disponible au public pour MySQL version 5.0. Installer à partir des sources de l'arbre de développement . Depuis décembre 2003, des paquets binaires de MySQL version 5.0 sont aussi disponible.s

1.6 Les évolutions de MySQL (la liste des tâches)

Cette section liste les fonctionnalités que nous prévoyons d'ajouter à MySQL. Les listes sont réparties par version, et les actions sont réparties approximativement par ordre de priorité.

Note : si vous êtes un utilisateur professionnel, avec un besoin urgent pour une fonctionnalité, contactez sales@mysql.com pour discuter de possibilités de sponsor. En allouant des finances à certaines fonctionnalités, nous pouvons nous concentrer sur certains objectifs à court terme. Un des exemples de fonctionnalité sponsorisé est la réplication.

1.6.1 Nouvelles fonctionnalités prévues pour la version 4.1

Les fonctionnalités suivantes ne sont pas encore en MySQL 4.1, mais il est prévu de les implémenter avant que MySQL 4.1 ne passe en version beta. Pour une liste des fonctionnalités déjà disponibles en MySQL 4.1, voyez Fonctionnalités disponibles depuis MySQL 4.1 .

- Support OpenSSL stable (MySQL 4.0 le supporte déjà de manière rudimentaire, mais ce n'est pas un support testé à 100%).
- Plus de tests sur les requêtes préparées et les jeux de caractères multiples.
- Options de groupe `ROLLUP` OLAP (Online Analytical Processing) pour les applications d'entrepôt de données.

Le développement des nouvelles fonctionnalités est passé en version 5.0.

1.6.2 Nouvelles fonctionnalités prévues pour la version 5.0

Les fonctionnalités suivantes sont prévues pour la version 5.0. Notez que comme nous avons de nombreux développeurs qui travaillent sur différents projets, il peut aussi y avoir des ajouts de fonctionnalités. Il y a aussi des chances que ces fonctionnalités soient ajoutées en MySQL 4.1. Pour une liste des fonctionnalités déjà disponibles en MySQL 4.1, voyez Fonctionnalités disponibles depuis MySQL 4.1 .

Pour ceux qui souhaitent étudier en détail les tous derniers développements de MySQL, vous pouvez accéder au serveur BitKeeper public de MySQL 5.0. Installer à partir des sources de l'arbre de développement . Depuis décembre 2003, des versions binaires pour les versions 5.0 sont aussi disponibles.

Procédures stockées
: Les procédures stockées sont actuellement développées avec notre version 5.0. Procédures stockées et fonctions .Nous allons aussi ajouter un environnement pour permettre l'utilisation de langages externes et assurer (lorsque c'est possible) une compatibility avec PL/SQL et T−SQL.

Nouvelles fonctionnalités
: Support élémentaire des curseurs. Curseurs .
: La capacité de spécifier explicitement pour les tables `MyISAM` qu'un index est de type `RTREE` . En version 4.1, les index `RTREE` sont généralement utilisé pour les données géométriques (types GIS), mais ne peuvent pas être créés à la demande.
: Les tables `HEAP` à taille de ligne dynamique.

Compatibilité avec les standards, portabilité et migration.
: Ajout d'un véritable type `VARCHAR` (ce support est déjà disponible avec le format de table `MyISAM`).

Amélioration de la vitesse
: `SHOW COLUMNS FROM table_name` (utilisé par le client `mysql` pour permettre la recherche de nom de colonnes) ne devrait pas ouvrir la table, mais seulement le fichier de définition. Cela prendra moins de mémoire, et sera bien plus rapide.
: Permettre à la commande `DELETE` sur les tables `MyISAM` l'utilisation des caches de lignes. Pour cela, nous devons modifier le thread de cache de ligne lorsque nous modifions le fichier `.MYD` .
: Amélioration des tables en mémoire `HEAP` :
  - Lignes de taille dynamique.
  - Gestion plus rapide des lignes (moins de copies).

Amélioration de l'ergonomie
: Résolution du problème avec `RENAME TABLE` , utilisé sur une table active dans un `MERGE` , qui conduit à la corruption de la table.

Le manuel inclut le changelog de la version, avec plus de détails sur les nouvelles fonctionnalités. Changements de la version 5.0.0 (Development) .

1.6.3 Nouvelles fonctionnalités prévues pour 5.1

Nouvelle fonctionnalité

`FOREIGN KEY` supporte tous les types de tables.
Contraintes de niveau colonne.
Réplication protégé contre les crash.
Sauvegarde à chaud, avec peu d'impact sur les performances. La sauvegarde à chaud permettra de simplifier l'ajout d'esclaves de réplication sans avoir à éteindre le maître.

Amélioration de la vitesse

Nouveau format de définition des tables en format texte (fichiers `.frm`) et un cache de table pour les définitions de tables. Cela permettra d'accélérer les requêtes de structure de tables, et de consolider le support des clés étrangères.
Optimisation du type `BIT` pour qu'il prenne 1 bit (actuellement, `BIT` prend 1 caractère).

Amélioration de l'ergonomie

Ajout Ajout d'option au protocole client/serveur pour obtenir des indications de progression pour les requêtes longues.
Implémentation de `RENAME DATABASE` . Pour que cela soit sécuritaire pour tous les pilotes de stockage, cela doit fonctionner comme ceci :

Crée une nouvelle base de données.
Pour chaque table, faire un renommage d'une base à l'autre, comme nous le faisons avec `RENAME` .
Effacer l'ancienne base.

Nouvelle interface interne pour les fichiers. Cela rendra la gestion des fichiers bien plus générale, et simplifiera l'ajout d'extensions comme `RAID` . (l'implémentation actuelle est un bidouillage).

1.6.4 Ce qui doit être fait dans un futur proche

Nouvelle fonctionnalité

Commande `CONNECT BY PRIOR ...` , inspirée d'Oracle, pour traiter les structures de type arbre (hiérarchisée).
Tous les types manquants de ANSI92 et ODBC 3.0.
Ajout de `SUM(DISTINCT)` .
`INSERT SQL_CONCURRENT` et `mysqld --concurrent-insert` pour faire des insertions concurrents à la fin du fichier, même si il est verrouillé en lecture.
Permettre la modification de variables dans une commande `UPDATE` . Par exemple : `UPDATE TABLE foo SET @a=a+b,a=@a, b=@a+c` .
Changer le moment de modifications des variables, pour que l'on puisse les utiliser avec les commandes de groupement `GROUP BY` , comme ceci : `SELECT id, @a:=COUNT(*), SUM(sum_col)/@a FROM table_name GROUP BY id` .
Ajout de l'option `IMAGE` à la commande `LOAD DATA INFILE` pour ne pas modifier les champs `TIMESTAMP` et `AUTO_INCREMENT` .
Ajout de la syntaxe `LOAD DATE INFILE ... UPDATE` , qui fonctionne comme ceci :

Pour les tables ayant des clés primaires, si les données contiennent une clé primaire, les données qui existent et correspondent à cette clé, sont modifiées. Cependant, les colonnes **omises** sont ignorées.
Pour les tables ayant une clé primaire dont il manque une partie dans les données entrantes, ou qui n'ont pas de clé primaire, le flux est traité comme `LOAD DATA INFILE ... REPLACE INTO` .

Rendre la syntaxe `LOAD DATA INFILE` utilisable comme ceci :

```
LOAD DATA INFILE 'file_name.txt' INTO TABLE tbl_name
     TEXT_FIELDS (text_field1, text_field2, text_field3)
     SET table_field1=CONCAT(text_field1, text_field2),
         table_field3=23
     IGNORE text_field3
```

Cela peut être utilisé pour ignorer des colonnes supplémentaires dans le fichier texte, ou pour modifier des colonnes en fonction de données dans les valeurs entrantes.

De nouvelles fonctions pour travailler avec le type de colonne `SET` :

```
ADD_TO_SET(value,set)
REMOVE_FROM_SET(value,set)
```

Si vous interrompez `mysql` au beau milieu d'une requête, vous pouvez ouvrir une autre connexion pour tuer la requête en cours. Ou bien, une tentative doit être faite pour détecter cela sur le serveur.

Ajout d'une interface au moteur de sauvegarde pour que vous puissiez l'utiliser comme table système. Cela sera un peu lent si vous demandez toutes les informations de toutes les tables, mais très souple. `SHOW INFO FROM tbl_name` doit être implémenté pour connaître les informations de bases des tables.

Permettre `SELECT a FROM crash_me LEFT JOIN crash_me2 USING (a)` ; dans ce cas, `a` est supposé provenir de la table `crash_me` .

Les options `DELETE` et `REPLACE` dans les commandes `UPDATE` (cela va effacer les lignes lors d'un doublon dans la requête).

Modification du format de `DATETIME` pour stocker les fractions de secondes.

Rendre possible l'utilisation de la bibliothèque GNU `regexp` au lieu de la bibliothèque actuelle (la bibliothèque GNU devrait être plus rapide que l'ancienne).

Compatibilité avec les standards, migration et portabilité.

Ne pas utiliser de valeur par défaut (`DEFAULT`) pour les colonnes. Indiquer une erreur lors des commandes `INSERT` , si elle contient une colonne qui n'a pas de valeur par défaut.

Ajout des fonctions de groupement `ANY()` , `EVERY()` et `SOME()` . En ANSI SQL, elles ne fonctionnent que sur les colonnes de type booléens, mais nous pouvons étendre leur champ d'action aux colonnes et expression, en appliquant la règle suivante : value == 0 −> FALSE et value <> 0 −> TRUE.

Correction du type pour que `MAX(column)` ait le même type de colonne :

```
 mysql> CREATE TABLE t1 (a DATE);
mysql> INSERT INTO t1 VALUES (NOW());
mysql> CREATE TABLE t2 SELECT MAX(a) FROM t1;
mysql> SHOW COLUMNS FROM t2;
```

Amélioration des performances

Ne permettre qu'à un nombre défini de threads d'exécuter des réparations `MyISAM` simultanément.

Changer `INSERT ... SELECT` pour utiliser optionnellement des insertions simultanées.

Ajouter une option pour écrire sur le disque les pages de clés pour les tables à clés retardées, qui n'ont pas été sauvées dernièrement.

Permettre les jointures sur des parties de clés (problème d'optimisation).

Un analyseur de logs, qui permettrait d'extraire des informations sur les tables les plus souvent utilisées, les jointures, etc... Cela permettra aux utilisateurs d'identifier les tables qui peuvent être optimisées.

Internationalisation

Ergonomie

Retourner le type de champs original lors des requêtes `SELECT MIN(column) ... GROUP BY` .

Rendre possible la spécification de `long_query_time` avec une granularité de l'ordre de la microseconde.

Intégrer le code de `myisampack` dans le serveur, activer les commandes `PACK` or `COMPRESS` depuis le serveur.

Ajouter un cache de buffer de clé temporaire durant les commandes `INSERT/DELETE/UPDATE` pour que nous puissions nous rétablir si jamais le fichier d'index est plein.
Si vous effectuez la commande `ALTER TABLE` sur une table qui est un lien symbolique sur un autre disque, les tables temporaires seront aussi faites sur ce disque.
Implémenter un type `DATE/DATETIME` qui gère les fuseaux horaire correctement, pour résoudre le problème du décalage horaire.
Corriger la configuration pour que l'on puisse compiler toute les bibliothèques (comme `MyISAM`) sans les threads.
Permettre toutes les variables SQL dans la clause `LIMIT` , comme par exemple `LIMIT @a,@b` .
Affichage automatique de `mysql` vers un navigateur web.
`LOCK DATABASES` (avec différentes options).
De nombreuses nouvelles variables dans `SHOW STATUS` . Les lignes lues et modifiées. Les sélections sur tables uniques et les jointures. Le nombre moyen de tables dans une sélection. Le nombre de requêtes `ORDER BY` et `GROUP BY` .
`mysqladmin copy database new-database` ; a besoin de la commande `COPY` dans `mysqld` .
La liste des processus doit afficher le nombre de requêtes et de threads.
`SHOW HOSTS` affichera les informations concernant le cache de noms d'hôtes.
Changer le nom des tables de chaîne vide à `NULL` pour les colonnes calculées.
Ne pas utiliser la fonction `Item_copy_string` sur les valeurs numériques, pour éviter les conversions nombre −> chaîne −> nombre en cas de : `SELECT COUNT(*)*(id+0) FROM table_name GROUP BY id`
Changer `ALTER TABLE` pour qu'il n'interrompe pas le client qui exécute des `INSERT DELAYED` .
Lorsque les colonnes sont spécifiées dans `UPDATE` , elles contiennent les anciennes valeurs d'avant la modification.

Nouveaux systèmes d'exploitation.
Port des clients MySQL vers LynxOS.

1.6.5 Ce qui est prévu à moyen terme

- Implémenter la fonction : `get_changed_tables(timeout,table1,table2,...)` .
- Remplacer la lecture dans les tables par une zone mémoire aussi souvent que possible. Actuellement, seules les tables compressées utilisent des memmap.
- Rendre le code des timestamp automatiques bien plus pratique. Ajouter les timestamps dans le log avec `SET TIMESTAMP=#;` .
- Utiliser un mutex de lecture/écriture pour gagner de la vitesse.
- Vues simples (implémentation progressive jusqu'à support total). Vues .
- Fermer automatiquement des tables si une table, une table temporaire ou un fichier temporaire reçoit une erreur 23 (plus assez de fichiers ouverts).
- Meilleure propagation des constantes. Lorsqu'une occurrence de `col_name=n` est trouvée dans une expression, pour une `n` , remplacer les autres occurrences de `col_name` de l'expression avec `n` . Actuellement, cela n'arrive que pour les cas les plus simples.
- Changer toutes les expressions constantes par des expressions calculées, si possible.
- Optimiser les comparaisons `key` = `expression` . Actuellement, seules les relations `key` = `column` or `key` = `constant` sont optimisées.
- Fusionner les fonctions de copie pour améliorer le code.
- Changez `sql_yacc.yy` pour le remplacer par un analyseur de ligne de commande plus petit, et qui gère mieux les messages.
- Changer l'analyseur pour utiliser uniquement une règle pour tous les nombres d'arguments possibles dans une fonction.
- Utiliser les calculs de noms complets dans la clause ORDER (pour ACCESS97).

- MINUS , INTERSECT et FULL OUTER JOIN . (actuellement, UNION [en 4.0] et LEFT OUTER JOIN fonctionnent).
- SQL_OPTION MAX_SELECT_TIME=# pour donner une limite de temps à une requête.
- Diriger le log de modification vers une base.
- Améliorer LIMIT pour permettre la lecture de données à la fin du résultat.
- Alertes lors des connexions/écritures/lectures du client.
- Notez ces modifications de safe_mysqld : selon la FSSTND (que Debian essaie de suivre) les fichiers PID devraient être placés dans /var/run/<progname>.pid et les fichiers de logs dans /var/log . Il serait bien si vous pouviez mettre le "DATADIR" dans la première déclaration de "pidfile" et "log", de façon à ce que l'emplacement de ces fichiers puisse être modifié en une seule ligne.
- Permettre au client de commander le log des actions.
- Ajouter l'utilisation de zlib() pour les fichiers gzip , avec la commande LOAD DATA INFILE .
- Corriger le tri et le groupage avec les colonnes BLOB (en partie résolu).
- Utiliser des sémaphores pour compter les threads. Il faut commencer par implémenter des sémaphores pour MIT-pthreads .
- Ajouter le support complet pour les JOIN avec parenthèses.
- Comme alternative à la relation un thread, une connexion, gérer un groupe de threads pour répondre aux requêtes.
- Permettre la pose de plusieurs verrous avec GET_LOCK . Lors de ces verrous multiples, gérer le cas des blocages par verrous qui pourrait être introduit.

Le temps est indiqué en temps de travail et non pas en temps normal.

1.6.6 Ce qui n'est pas prévu

Rien. Nous nous dirigeons vers la compatibilité complète avec ANSI 92/ANSI 99.

1.7 Sources d'informations MySQL

1.7.1 Listes de diffusion MySQL

Cette section vous présente les listes de diffusions MySQL, et donne des conseils quand à leur utilisation. En vous inscrivant à une des listes de diffusion, vous recevrez les messages que les autres auront envoyé, et vous pourrez envoyer vos propres questions et réponses.

1.7.1.1 Les listes de diffusion de MySQL

Pour vous inscrire ou vous désinscrire à la liste de diffusion principale de MySQL, visitez le site http://lists.mysql.com/ . **N'envoyez pas** de messages pour vous inscrire ou vous désinscrire sur la liste, car ces messages seront transmis automatiquement à des milliers d'utilisateurs.

Votre site local peut avoir beaucoup d'inscrits à une liste de diffusion. Si c'est le cas, vous pouvez avoir une liste de diffusion locale, de façon à ce que les messages envoyés par lists.mysql.com à votre site local soit propagés par votre serveur local. Dans ce cas, contactez votre administrateur local pour être ajouté ou retiré de la liste.

Si vous voulez que le trafic de cette liste soit envoyé à une autre boîte aux lettres de votre client mail, installez un filtre basé sur les entêtes du message. Vous pouvez utiliser notamment les entêtes List-ID: et Delivered-To: pour identifier les messages de la liste. Les listes de diffusion MySQL suivantes existent :

`announce`
: Ceci est la liste de diffusion d'annonces des versions de MySQL et des programmes compagnons. C'est une liste à faible volume, et tout utilisateur doit y être inscrit.

`mysql`
: La liste de diffusion principale pour les discussions générales sur MySQL. Notez que certains sujets sont à diriger sur les listes spécialisées. Si vous postez sur la mauvaise liste, vous pourriez ne pas avoir avoir de réponse.

`mysql-digest`
: La liste `mysql` en format journalier. Cela signifie que vous recevrez tous les messages de la journée en un seul gros email.

`bugs`
: Sur cette liste, vous ne devriez envoyez que des bogues complets, reproductibles ainsi que le rapport qui va avec, en utilisant le script `mysqlbug` (si vous utilisez Windows, il faut aussi inclure la description du système d'exploitation et la version de MySQL). Comment rapporter des bogues ou problèmes .

`bugs-digest`
: La liste `bugs` en format journalier.

`internals`
: Une liste pour ceux qui travaillent sur le code MySQL. Sur cette liste, vous pouvez discuter du développement de MySQL et envoyer des correctifs.

`internals-digest`
: La liste `internals` en format journalier.

`mysqldoc`
: La liste des personnes qui travaillent sur la documentation MySQL : des employés de MySQL AB, des traducteurs et d'autres membres de la communauté.

`mysqldoc-digest`
: La liste `mysqldoc` en format journalier.

`benchmarks`
: Cette liste est pour tous ceux qui sont intéressé par les performances. Les discussions se concentrent sur les performances (mais pas seulement avec MySQL), mais abordent aussi des problèmes de noyau, le système de fichiers, les disques, etc.

`benchmarks-digest`
: La liste `benchmarks` en format journalier.

`packagers`
: Cette liste se concentre sur les paquets et les distributions MySQL. C'est l'un des forums utilisé par les responsables pour échanger des idées sur les paquets MySQL, pour s'assurer que MySQL est le même sur toutes les plates−formes.

`packagers-digest`
: La liste `packagers` en format journalier.

`java`
: Une liste pour ceux qui utilisent MySQL et java. Elle concerne majoritairement les pilotes JDBC.

`java-digest`
: La liste `java` en format journalier.

`win32`
: Une liste pour ceux qui utilisent MySQL sur les systèmes d'exploitation de Microsoft, tels que Windows 9x/Me/NT/2000/XP.

`win32-digest`
: La liste `win32` en format journalier.

`myodbc`
: Une liste pour tout ce qui concerne la connexion à MySQL avec le pilote ODBC.

`myodbc-digest`
: La liste `myodbc` en format journalier.

`mysqlcc`
: Une liste pour tout ce qui concerne le client graphique `MySQL Control Center` .

`mysqlcc-digest`
: La liste `mysqlcc` en format journalier.

`plusplus`
: Une liste pour tout ce qui concerne la programmation avec les API C++ de MySQL.

`plusplus-digest`
: La liste `plusplus` en format journalier.

`msql-mysql-modules`

Une liste pour tout ce qui concerne Perl et le support du module msql / mysql.

`msql-mysql-modules-digest`
: La liste `msql-mysql-modules` en format journalier.

Vous pouvez vous inscrire ou vous désinscrire de toutes les listes en même temps de la même façon que nous l'avons décrit au début. Dans votre message d'inscription, utilisez simplement le nom de liste approprié. Par exemple, pour vous inscrire à la liste `myodbc` .

Si vous ne pouvez pas obtenir d'informations sur la liste de diffusion, une de vos options est de prendre un contrat de support auprès de MySQL AB, qui vous donnera un contact direct avec les développeurs MySQL. Support offert par MySQL AB .

Le tableau suivant présente diverses autres listes de diffusions consacrée à MySQL, dans d'autres langues que l'anglais. Notez que ces ressources ne sont pas gérées par MySQL AB, ce qui fait que nous ne pouvons pas garantir leur qualité.

mysql−france−subscribe@yahoogroups.com
: Une liste de diffusion française

list@tinc.net
: Une liste de diffusion coréenne Envoyez un message à `subscribe mysql your@e-mail.address` .

mysql−de−request@lists.4t2.com
: Une liste de diffusion allemande Envoyez un message à `subscribe mysql-de your@e-mail.address` . Vous aurez plus d'informations sur cette liste à http://www.4t2.com/mysql/ .

mysql−br−request@listas.linkway.com.br
: Une liste de diffusion portugaise Envoyez un message à `subscribe mysql-br your@e-mail.address` .

mysql−alta@elistas.net
: Une liste de diffusion espagnole Envoyez un message à `subscribe mysql your@e-mail.address` .

1.7.1.2 Poser des questions ou rapporter un bogue

Avant de soumettre un rapport de bogue ou une question, commencez par ces étapes simples :

- Etudiez le manuel MySQL et faites y une recherche à :

  http://www.mysql.com/doc/

  Nous nous efforçons de mettre à jour le manuel fréquemment, en y ajoutant les solutions aux nouveaux problèmes. L'historique de modification (http://www.mysql.com/doc/en/News.html) est particulièrement pratique car il est possible qu'une nouvelle version de MySQL propose déjà la solution à votre problème.
- Cherchez dans la base de données des bogues sur http://bugs.mysql.com/ pour voir si le bogue a déjà été reporté/résolu.
- Recherchez dans les archives des listes de diffusion de MySQL :

  http://lists.mysql.com/
- Vous pouvez aussi utiliser l'URL http://www.mysql.com/search/ pour rechercher dans toutes les pages web (y compris le manuel) sur le site web de MySQL.

Si vous n'arrivez pas à trouver une réponse à votre question dans le manuel ou dans les archives, vérifiez auprès de votre expert MySQL local. Si vous ne trouvez toujours pas la réponse, vous pouvez lire la section suivante.

1.7.1.3 Comment rapporter un bogue ou un problème

Notre base de données de bogues est publique, et peut être lue par tous sur le site http://bugs.mysql.com/ . Si vous vous identifiez sur le système vous serez aussi capable d'envoyer des rapports.

Ecrire un bon rapport de bogue requiert de la patience, et le faire dès le début épargnera votre temps et le notre. Un bon rapport de bogue qui contient un cas de test complet améliorera vos chances de voir le bogue corrigé à la prochaine version. Cette section vous aidera à écrire correctement un rapport de bogue, de manière à ce que vous ne gaspillez pas votre temps à faire des textes qui ne nous aideront que peu ou pas.

Nous vous recommandons d'utiliser le script `mysqlbug` pour générer un rapport de bogue (ou rapporter un problème), dans la mesure du possible. `mysqlbug` est situé dans le dossier `scripts` de la distribution, ou, pour les distributions binaire, dans le dossier `bin` du dossier d'installation de MySQL. Si vous êtes dans l'incapacité d'utiliser `mysqlbug` , vous devez tout de même inclure toutes les informations nécessaires listées dans cette section.

Le script `mysqlbug` vous aide à générer un rapport en déterminant automatiquement les informations suivantes, mais si quelque chose d'important lui échappe, ajoutez le dans votre message! Lisez cette section avec attention, et assurez vous que toutes les informations décrites ici sont présentes dans votre message.

De préférence, vous devriez tester le problème avec la dernière version de production ou de développement de MySQL. Il doit être facile de reproduire le test avec simplement la commande ' `mysql test < script` ', appliquée au cas de test, ou en exécutant le script Shell ou Perl inclut dans le rapport.

Tous les bogues postés sur le site de rapports de bogues http://bugs.mysql.com/ seront corrigés ou documentés dans la prochaine version de MySQL. Si seuls, de petits changements sont nécessaires, nous publierons aussi un patch.

Si vous avez découvert un problème de sécurité critiques avec MySQL, il faut envoyer un email à security@mysql.com .

Si vous avez un rapport de bogue reproductible, envoyez un rapport sur le site http://bugs.mysql.com/ . Notez que même dans ce cas, il est bon d'utiliser le script `mysqlbug` pour rassembler des informations sur votre système. Tous les bogues que nous pourrons reproduire auront de bonnes chances d'être corrigés lors de la prochaine version de MySQL.

Pour signaler d'autres problèmes, utilisez une des listes de diffusion MySQL.

Sachez qu'il est toujours possible de répondre à un message qui contient trop d'informations, alors qu'il est impossible de répondre à un message qui contient trop peu d'informations. Souvent, il est facile d'omettre des faits parce que vous pensez connaître la cause du problème et supposez que ces détails ne sont pas importants. Un bon principe à suivre est : si vous avez un doute à propos de quelque chose, faites nous en part. Il est bien plus rapide et bien moins frustrant d'écrire quelques lignes de plus dans un rapport plutôt que d'être obligé de demander une nouvelle fois et d'attendre une réponse parce que vous avez oublié une partie des informations la première fois.

L'erreur la plus commune est de ne pas indiquer le numéro de la version de MySQL qui est utilisé, ou de ne pas indiquer le système d'exploitation que vous utilisez (y compris le numéro de version de ce système d'exploitation). Ce sont des informations de première importance, et dans 99% des cas, le rapport de bogue est inutilisable sans ces informations. Souvent, nous recevons des questions telles que ``Pourquoi est ce que cela ne fonctionne pas pour moi?''. Puis nous nous apercevons que la fonctionnalités en question n'est même pas programmée dans la version de MySQL utilisée, ou que le bogue décrit est déjà corrigé dans une nouvelle version de MySQL. Parfois aussi, les erreurs sont dépendantes des plates−formes. Dans ce cas, il est presque impossible de les corriger sans savoir quel système d'exploitation et quelle version exacte est utilisée.

Pensez aussi à fournir des informations concernant votre compilateur, si c'est pertinent. Souvent, les développeurs trouvent des bogues dans les compilateurs, et pensent que c'est liés à MySQL. La plupart des compilateurs sont en constant développement, et s'améliorent de version en version. Pour déterminer si votre problème dépend de votre compilateur, nous avons besoin de savoir quel compilateur est utilisé. Notez que les problèmes de compilations sont des bogues, et doivent être traités avec un rapport de bogues.

Il est particulièrement utile de fournir une bonne description du bogue dans le rapport de bogue. Cela peut être un exemple de ce que vous avez fait qui a conduit au problème, ou une description précise. Les meilleurs rapports sont ceux qui incluent un exemple complet permettant de reproduire le bogue. Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .

Si un programme produit un message d'erreur, il est très important d'inclure ce message dans votre rapport. Il est préférable que le message soit le message exact, car il est alors possible de le retrouver en utilisant les archives : même la casse doit être respectée. N'essayez jamais de vous rappeler d'un message d'erreur, mais faites plutôt un copier/coller du message complet dans votre rapport.

Si vous avez un problème avec `MyODBC` , essayez de générer un fichier de trace `MyODBC` . Reporter les problèmes rencontrés avec MyODBC .

Pensez aussi que de nombreux personnes qui liront votre rapport utilisent un formatage de 80 colonnes. Lorsque vous générez votre rapport et vos exemples avec l'outil de ligne de commande, utilisez une largeur de 80 colonnes. Utilisez l'option `--vertical` (ou la fin de commande `\G`) pour les affichages qui excèdent une telle largeur (par exemple, avec la commande `EXPLAIN SELECT` ; voyez l'exemple un peu plus tard dans cette section.

Voici un pense−bête des informations à fournir dans votre rapport :

* Le numéro de version de la distribution de MySQL que vous utilisez (par exemple MySQL Version 3.22.22). Vous pouvez connaître cette version en exécutant la commande `mysqladmin version` . `mysqladmin` est situé dans le dossier `bin` de votre distribution MySQL.
* Le fabricant et le modèle de votre serveur.
* Le système d'exploitation et la version que vous utilisez. Pour la plupart des systèmes d'exploitation, vous pouvez obtenir cette information en utilisant la commande Unix `uname -a` .
* Parfois, la quantité de mémoire (physique et virtuelle) est important. Si vous hésitez, ajoutez la.
* Si vous utilisez une version de MySQL sous forme de source, le nom et le numéro de version du compilateur sont nécessaires. Si vous utilisez une version exécutable, le nom de la distribution est important.
* Si le problème intervient lors de la compilation, incluez le message d'erreur exact et les quelques lignes de contexte autour du code en question dans le fichier où il est situé.
* Si `mysqld` s'est arrêté, il est recommandé d'inclure la requête qui a mené à cet arrêt de `mysqld` . Vous pouvez généralement la trouver en exécutant `mysqld` en ayant activé les logs. Utilisation des fichiers de log pour trouver d'où viennent les erreurs de mysqld .
* Si une table ou une base sont liés au problème ajoutez le résultat de la commande `mysqldump --no-data db_name tbl_name1 tbl_name2 ...` . C'est très simple à faire, et c'est un moyen efficace d'obtenir

un descriptif de table, qui nous permettra de recréer une situation comparable à la votre.
- Pour les problèmes liés à la vitesse, ou des problèmes liés à la commande `SELECT` , pensez à inclure le résultat de la commande `EXPLAIN SELECT ...` , et au moins le nombre de ligne que la commande `SELECT` doit produire. Vous devriez aussi inclure le résultat de la commande `SHOW CREATE TABLE table_name` pour chaque table impliquée. Plus vous nous fournirez d'informations, plus nous aurons de chance de vous aider efficacement. Par exemple, voici un excellent rapport de bogue (posté avec le script `mysqlbug` , et effectivement rédigé en anglais) :

  Exemple réalisé avec `mysql` en ligne de commande (notez l'utilisation de la fin de commande `\G` , pour les résultats qui pourraient dépasser les 80 colonnes de large) :

```
 mysql> SHOW VARIABLES;
mysql> SHOW COLUMNS FROM ...\G
       <output from SHOW COLUMNS>
mysql> EXPLAIN SELECT ...\G
       <output from EXPLAIN>
mysql> FLUSH STATUS;
mysql> SELECT ...;
       <A short version of the output from SELECT,
       including the time taken to run the query>
mysql> SHOW STATUS;
       <output from SHOW STATUS>
```
- Si un bogue ou un problème survient lors de l'exécution de `mysqld` , essayez de fournir un script qui reproduit l'anomalie. Ce script doit inclure tous les fichiers sources nécessaires. Plus votre script reproduira fidèlement votre situation, mieux ce sera. Si vous pouvez réaliser un cas de test postez−le sur le site http://bugs.mysql.com/ pour un traitement prioritaire!

  Si vous ne pouvez pas fournir de script, fournissez tout au moins le résultat de la commande `mysqladmin variables extended-status processlist` dans votre mail pour fournir des informations sur les performances de votre système.
- Si vous ne pouvez pas reproduire votre situation en quelques lignes, ou si une table de test est trop grosse à être envoyée par mail (plus de 10 lignes), exportez vos tables sous forme de fichier avec la commande `mysqldump` et créez un fichier `README` qui décrit votre problème.

  Créez une archive compressée de votre fichier en utilisant `tar` et `gzip` ou `zip` , et placez le via `ftp` sur le site de ftp://support.mysql.com/pub/mysql/secret/ . Puis, entrez la description de l'anomalie sur le site http://bugs.mysql.com/ .
- Si vous pensez que le serveur MySQL fournit des résultats étranges pour une requête, incluez non seulement le résultat, mais aussi votre propre explication sur ce que le résultat devrait être, et un diagnostic de la situation.
- Lorsque vous donnez un exemple du problème, il est mieux d'utiliser des noms de variables et de tables qui existent dans votre situation, plutôt que d'inventer de nouveaux noms. Le problème peut être lié au noms des variables ou tables que vous utilisez! Ces cas sont rares, mais il vaut mieux éviter les ambiguïtés. Après tout, il est plus facile pour vous de fournir un exemple qui utilise votre situation réelle, et c'est bien mieux pour nous aussi. Si vous avez des données que vous ne souhaitez pas divulguer, vous pouvez utiliser le site `ftp` pour les transférer dans le dossier secret ftp://support.mysql.com/pub/mysql/secret/ . Si les données sont vraiment ultra secrètes et que vous ne souhaitez même pas nous les montrer, alors utilisez d'autres noms et données pour votre rapport, mais considérez cela comme un dernier recours.
- Incluez toutes les options utilisées, si possible. Par exemple, indiquez les options que vous utilisez lors du démarrage de `mysqld` , et celle que vous utilisez avec les programmes comme `mysqld` et `mysql` , et le script `configure` , qui sont souvent primordiaux et pertinents. Ce n'est jamais une mauvaise idée que de les inclure. Si vous utilisez des modules, comme Perl ou PHP, incluez aussi les versions de ces logiciels.
- Si votre question porte sur le système de droits, incluez le résultat de l'utilitaire `mysqlaccess` , celui de `mysqladmin reload` et tous les messages d'erreurs que vous obtenez lors de la connexion. Lorsque vous testez votre système de droits, il faut commencer par utiliser la commande

`mysqladmin reload version` et de vous connecter avec le programme qui vous pose problème. `mysqlaccess` est situé dans le dossier `bin` de votre installation de MySQL.

- Si vous avez un patch pour un bogue, c'est une excellente chose. Mais ne supposez pas que nous n'avons besoin que du patch, ou même que nous allons l'utiliser, si vous ne fournissez pas les informations nécessaires pour le tester. Nous pourrions trouver des problèmes générés par votre patch, ou bien nous pourrions ne pas le comprendre du tout. Si tel est le cas, nous ne l'utiliserons pas.

  Si nous ne pouvons pas vérifier exactement ce pourquoi est fait le patch, nous ne l'utiliserons pas. Les cas de tests seront utiles ici. Montrez nous que votre patch va générer toutes les situations qui pourraient arriver. Si nous trouvons un cas limite dans lequel votre patche ne fonctionne pas, même si il est rare, il risque d'être inutile.
- Les diagnostics sur la nature du bogue, la raison de son déclenchement ou les effets de bords sont généralement faux. Même l'équipe MySQL ne peut diagnostiquer sans commencer par utiliser un débogueur pour déterminer la cause véritable.
- Indiquez dans votre mail que vous avez vérifié le manuel de référence et les archives de courrier, de façon à avoir épuiser les solutions que d'autres avant vous auraient pu trouver.
- Si vous obtenez un message `parse error` , vérifiez votre syntaxe avec attention. Si vous ne pouvez rien y trouvez à redire, il est très probable que votre version de MySQL ne supporte pas encore cette fonctionnalité que vous essayez d'utiliser. Si vous utilisez la version courante de mySQL et que le manuel http://www.mysql.com/doc/ ne couvre pas la syntaxe que vous utilisez, c'est que MySQL ne supporte pas votre syntaxe. Dans ce cas, vos seules options sont d'implémenter vous même la syntaxe ou d'envoyez un message à licensing@mysql.com pour proposer de l'implémenter.

  Si le manuel présente la syntaxe que vous utilisez, mais que vous avez une ancienne version du serveur MySQL, il est recommandé de vérifier l'historique d'évolution de MySQL pour savoir quand la syntaxe a été supportée. Dans ce cas, vous avez l'option de mettre à jour votre MySQL avec une version plus récente. Historique de MySQL .
- Si vous avez un problème tel que vos données semblent corrompues, ou que vous recevez constamment des erreurs lors d'accès à une table, vous devriez commencer par essayer de réparer votre table avec l'utilitaire de ligne de commande `myisamchk` ou les syntaxes SQL `CHECK TABLE` et `REPAIR TABLE` . Administration de la base .
- Si vous avez des tables qui se corrompent facilement, il vous faut essayer de trouver quand et pourquoi cela arrive. Dans ce cas, le fichier `mysql-data-directory/'hostname'.err` peut contenir des informations pertinentes qu'il est bon d'inclure dans votre rapport de bogues. Le log d'erreurs . Normalement, `mysqld` **ne doit jamais** corrompre une table si il a été interrompu au milieu d'une mise à jour. Si vous pouvez trouvez la cause de l'arrêt de `mysqld` , il est bien plus facile pour nous de fournir un correctif. Comment déterminer ce qui cause problème .
- Si possible, téléchargez et installez la version la plus récente du serveur MySQL, et vérifiez si cela résout votre problème. Toutes les versions de MySQL sont testées à fond, et doivent fonctionner sans problème. Nous croyons à la compatibilité ascendante, et vous devriez pouvoir passer d'une version à l'autre facilement. Choisir quelle distribution installer .

Si vous disposez de l'accès au support client, contactez aussi le support client à mysql−support@mysql.com , en plus de la liste de rapport de bogues, pour un traitement prioritaire.

Pour des informations sur les rapports de bogues avec `MyODBC` , voyez Comment reporter les problèmes rencontrés avec MySQL ODBC .

Pour des solutions aux problèmes les plus courants, voyez Problèmes et erreurs communes .

Lorsque des solutions vous sont envoyées individuellement et non pas à la liste, il est considéré comme bien vu de rassembler ces réponses et d'en envoyer un résumé sur le liste, de manière à ce que les autres en profitent aussi.

1.7.1.4 Conseils pour répondre sur la liste de diffusion

Si vous pensez que votre réponse peut avoir un intérêt général, vous pouvez envisager de l'envoyer sur la liste de diffusion, plutôt que de faire une réponse personnelle aux demandeurs. Essayez de rendre votre réponse aussi générale que possible, pour que suffisamment d'autres personnes puissent en profiter. Lorsque vous envoyez une réponse sur la liste, assurez vous qu'elle ne représente pas un doublon d'une réponse précédente.

Essayez de résumer l'essentiel de la question dans votre réponse. Ne vous croyez pas obligé de citer tout le message original.

Attention : n'envoyez pas de message avec le mode HTML activé ! De nombreux utilisateurs ne lisent pas leurs emails avec un navigateur.

1.7.2 Support de la communauté MySQL sur IRC (Internet Relay Chat)

En plus des différentes listes de diffusion MySQL, vous pouvez rencontrer des utilisateurs expérimentés sur `IRC` (`Internet Relay Chat`). Voici les meilleurs canaux, à notre connaissance :

- **freenode** (voyez http://www.freenode.net/ pour les serveurs)
    - `#mysql` Principalement, des questions sur MySQL, mais les autres bases de données et le langage SQL sont aussi acceptés.
- **EFnet** (voyez http://www.efnet.org/ pour les serveurs)
    - `#mysql` Questions sur MySQL.

Si vous recherchez un client IRC pour vous connecter à un réseau IRC, voyez donc `X-Chat` (http://www.xchat.org/). X−Chat est disponible sous Unix et sous Windows.

1.8 Quels standards respecte MySQL?

Cette section présente comment MySQL interprète les standards SQL ANSI. Le serveur MySQL dispose de nombreuses extensions au standard ANSI et vous trouverez ici comment les exploiter. Vous trouverez aussi des informations sur les fonctionnalités manquantes de MySQL et comment y trouver des palliatifs.

Notre but n'est pas, sans une bonne raison, de restreindre les capacités de MySQL à un usage unique. Même si nous n'avons pas les ressources de développement à consacrer à toutes les opportunités, nous sommes toujours intéressés et prêts à aider ceux qui utilisent MySQL dans de nouveaux domaines.

Un de nos objectifs avec ce produit est de tendre à la compatibilité ANSI 99, mais sans sacrifier la vitesse ou la robustesse. Nous ne reculons pas devant l'ajout de nouvelle fonctionnalités au langage SQL, ou le support de fonctionnalités hors SQL, qui améliorent le confort d'utilisation de MySQL. La nouvelle interface de gestionnaires `HANDLER` de MySQL 4.0 est un exemple de cette stratégie. Syntaxe de `HANDLER` .)

Nous continuons de supporter les bases transactionnelles et non transactionnelles pour combler les besoins des sites web ou des applications à fort besoin d'archivage, ainsi que les applications critiques à très haute disponibilité.

Le serveur MySQL a été conçu pour travailler avec des bases de taille moyenne (de 10 à 100 millions de lignes, ou des tables de 100 Mo) sur des systèmes de petite taille. Nous continuons d'améliorer MySQL pour qu'il fonctionne avec des bases gigantesques (tera−octets), tout en conservant la possibilité de compiler une version réduite de MySQL pour qu'il fonctionne sur des appareils embarqués ou nomades. L'architecture compacte de MySQL rend possible le support de ces applications si différentes, sans aucun conflit dans les sources.

Nous n'étudions pas le support du temps réel ou des bases de données en grappe (même si vous pouvez dores et déjà réaliser de nombreuses applications avec les services de réplication).

Nous ne croyons pas au support natif du XML en base, mais nous allons faire en sorte d'ajouter le support XML que réclame nos clients du coté client. Nous pensons qu'il est préférable de conserver le serveur central aussi ``simple et efficace'' que possible, et développer les bibliothèques qui gèrent la complexité du coté client. Cela fait partie de la stratégie que nous avons mentionné plus tôt, pour ne sacrifier ni la vitesse, ni la robustesse du serveur.

1.8.1 Quels standards suit MySQL ?

Nous nous dirigeons vers le support complet du standard ANSI SQL, mais sans aucune concession sur la vitesse ou la qualité du code.

ODBC niveau 0−{}3.51.

1.8.2 Sélectionner les modes SQL

Le serveur MySQL peut opérer avec différents modes SQL, et peut appliquer des modes différents pour chaque client. Cela permet aux applications d'adapter le comportement du serveur à ses attentes.

Le mode définit quelle syntaxe SQL MySQL doit supporter, et quel type de validations il doit effectuer sur les données. Cela facilite l'utilisation de MySQL dans différents environnements, et avec d'autres bases de données.

Vous pouvez configurer le mode SQL par défaut en lançant le serveur `mysqld` avec l'option `--sql-mode="modes"` . Depuis MySQL 4.1, vous pouvez aussi changer le mode après le lancement, en changeant la variable `sql_mode` avec la commande `SET [SESSION|GLOBAL] sql_mode='modes'` .

Pour plus d'informations sur les modes serveurs, voyez Le mode SQL du serveur .

1.8.3 Exécuter MySQL en mode ANSI

Vous pouvez lancer `mysqld` en mode ANSI avec l'option de démarrage `--ansi` . Options de ligne de commande `mysqld` .

Le mode ANSI revient à lancer le serveur avec les options suivantes (spécifiez la valeur de `--sql_mode` sur une seule ligne) :

```
 --transaction-isolation=SERIALIZABLE
--sql-mode=REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,
IGNORE_SPACE,ONLY_FULL_GROUP_BY
```

En MySQL version 4.1, vous pouvez arriver à la même configuration avec ces deux options (spécifiez la valeur de `--sql_mode` sur une seule ligne) :

```
 SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET GLOBAL sql_mode = 'REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,
IGNORE_SPACE,ONLY_FULL_GROUP_BY';
```

Choisir son mode SQL .

En MySQL version 4.1.1, les options `sql_mode` présentée ci−dessus peuvent être configurée avec :

```
SET GLOBAL sql_mode='ansi';
```

Dans ce cas, la valeur de la variable `sql_mode` prendre toute les options du mode ANSI. Vous pouvez vérifier le résultat comme ceci :

```
 mysql> SET GLOBAL sql_mode='ansi';
mysql> SELECT &#64;&#64;global.sql_mode;
        -> 'REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,
            IGNORE_SPACE,ONLY_FULL_GROUP_BY,ANSI';
```

1.8.4 Extensions MySQL au standard SQL−92

Le serveur MySQL inclut des extensions que vous ne trouverez probablement pas dans les autres bases de données. Soyez prévenus que si vous les utilisez, votre code ne sera probablement pas portable sur d'autres serveurs SQL. Dans certains cas, vous pouvez écrire du code qui inclut des spécificités de MySQL, mais qui restent portables, en les incluant dans des commentaires de la forme `/*! ... */` . Dans ce cas, le serveur MySQL va analyser la chaîne et exécuter le code à l'intérieur de ces commentaires comme une commande normale, mais d'autres serveurs ignoreront ces commentaires. Par exemple :

```
SELECT /*! STRAIGHT_JOIN */ col_name FROM table1,table2 WHERE ...
```

Si vous ajoutez le numéro de version après le point d'exclamation `'!'` , la syntaxe sera exécutée uniquement si la version du serveur MySQL est égale ou plus récente que le numéro de version utilisé.

```
CREATE /*!32302 TEMPORARY */ TABLE t (a int);
```

Cela signifie que si vous avez la version 3.23.02 ou plus récente, le serveur MySQL va utiliser le mot réservé `TEMPORARY` .

Voici une liste des apports spécifiques de MySQL :

Organisation des données sur le disque

MySQL fait correspondre à chaque base un dossier dans le dossier de données MySQL, et à chaque table des fichiers portant le même nom. Ceci a plusieurs implications :

- Les noms des bases de données et des tables sont sensibles à la casse sur les systèmes d'exploitation qui ont des systèmes de fichiers sensibles à la casse (comme la plupart des systèmes Unix). Identifier Case Sensitivity .
- Vous pouvez utiliser les commandes systèmes standard pour sauver, renommer, déplacer, effacer et copier des tables. Par exemple, pour renommer une table, il suffit de renommer les fichiers `.MYD` , `.MYI` et `.frm` et de leur donner un nouveau nom.

Les noms de bases, tables, index, colonnes ou alias peuvent commencer par des chiffres, mais ne peuvent pas être constitués uniquement de noms.

Syntaxe générale du langage

- Les chaînes de caractères peuvent être soit délimitées par `'"'` , soit par `'''` . Pas seulement par `'''` .
- L'utilisation du caractère de protection `'\'` .
- Dans une requête SQL, vous pouvez accéder à des tables situées dans

différentes bases de données, avec la syntaxe `db_name.tbl_name` . Certains serveurs SQL fournissent la même fonctionnalité, mais l'appellent un `User space` . Le serveur MySQL ne supporte par les espaces de nom de tables, comme dans : `create table ralph.my_table...IN my_tablespace` .

Syntaxe de commande SQL

Les commandes `ANALYZE TABLE` , `CHECK TABLE` , `OPTIMIZE TABLE` et `REPAIR TABLE` .
Les commandes `CREATE DATABASE` et `DROP DATABASE` . Syntaxe de `CREATE DATABASE` .
La commande `DO` .
La commande `EXPLAIN SELECT` pour avoir le détail des jointures de tables.
Les commandes `FLUSH` et `RESET` .
La commande `SET` . Syntaxe de `SET` .
La commande `SHOW` . Syntaxe de `SET` et `SHOW` .
L'utilisation de la commande `LOAD DATA INFILE` . Dans de nombreuses situations, cette syntaxe est compatible avec la commande d'Oracle `LOAD DATA INFILE` . Syntaxe de `LOAD DATA INFILE` .
L'utilisation de `RENAME TABLE` . Syntaxe de `RENAME TABLE` .
L'utilisation de `REPLACE` au lieu de `DELETE` + `INSERT` . Syntaxe de `REPLACE` .
L'utilisation de `CHANGE col_name` , `DROP col_name` ou `DROP INDEX` , `IGNORE` ou `RENAME` dans une commande `ALTER TABLE` . Syntaxe de `ALTER TABLE` .
L'utilisation de noms d'index, de préfixes d'index, et l'utilisation des mots−clé `INDEX` or `KEY` dans une commande de création de table `CREATE TABLE` . Syntaxe de `CREATE TABLE` .
L'utilisation des clauses `TEMPORARY` et `IF NOT EXISTS` avec `CREATE TABLE` .
L'utilisation de `DROP TABLE` avec les mots−clé `IF EXISTS` .
Vous pouvez effacer plusieurs tables avec une seule commande `DROP TABLE` .
La clause `LIMIT` de la commande `DELETE` .
La syntaxe `INSERT INTO ... SET col_name = ...` .
La clause `DELAYED` des commandes `INSERT` et `REPLACE` .
La clause `LOW_PRIORITY` des commandes `INSERT` , `REPLACE` , `DELETE` et `UPDATE` .
L'utilisation de `INTO OUTFILE` et `STRAIGHT_JOIN` dans les requêtes `SELECT` . Syntaxe de `SELECT` .
L'option `SQL_SMALL_RESULT` de la commande `SELECT` .
Vous n'êtes pas obligé de nommer toutes les colonnes que vous sélectionnez dans la clause `GROUP BY` . Cela donne de meilleures performances pour certaines situations spécifiques, mais classiques. Fonctions à utiliser dans les définitions `GROUP BY` .
Vous pouvez spécifier `ASC` ou `DESC` dans la clause `GROUP BY` .
La possibilité de modifier les variables dans les commandes avec l'opérateur `:=` :

```
 SELECT &#64;a:=SUM(total),&#64;b=COUNT(*),&#64;a/&#64;b AS avg FROM test_table;
SELECT &#64;t1:=(&#64;t2:=1)+&#64;t3:=4,&#64;t1,&#64;t2,&#64;t3;
```

Types de colonnes

Les types de colonnes `MEDIUMINT` , `SET` , `ENUM` et les types `BLOB` et `TEXT` .
Les attributs de champs `AUTO_INCREMENT` , `BINARY` , `NULL` , `UNSIGNED` et `ZEROFILL` .

Fonctions et opérateurs

Pour aider les utilisateurs qui viennent d'autres environnements SQL, le serveur MySQL supporte des alias de nombreuses fonctions. Par exemple, toutes les fonctions de chaînes de caractères supportent simultanément les syntaxes ANSI SQL et ODBC.
Le serveur MySQL comprend les opérateurs `||` et `comme opérateurs logiques OR et AND , comme en langage C. Pour le serveur MySQL, les opérateurs || et OR sont synonymes, ainsi que et AND . En conséquence, MySQL ne supporte pas l'opérateur de concaténation de chaînes ANSI SQL || . Utilisez plutôt la fonction CONCAT() . Comme CONCAT() prend un nombre illimité d'arguments, il est facile de convertir des expressions utilisant || , pour qu'elles fonctionnent sur le serveur MySQL.`
L'utilisation de `COUNT(DISTINCT list)` où `list` contient plus d'un élément.
Toutes les comparaisons de chaînes sont insensibles à la casse par défaut, et l'ordre de tri est déterminé par le jeu de caractères courant (`ISO-8859-1 Latin1`

par défaut). Si vous en souhaitez un autre, il faut déclarer les colonnes avec l'attribut `BINARY` ou utiliser l'opérateur `BINARY` pour forcer les comparaisons à prendre en compte la casse, en fonction du jeu de caractères utilisé sur l'hôte du serveur MySQL.
L'opérateur `%` est synonyme de `MOD()` . C'est à dire que `N % M` est équivalent à `MOD(N,M)` . `%` est supporté pour les programmeurs C, et pour la compatibilité avec `PostgreSQL` .
Les opérateurs `=` , `<>` , `<=` , `<` , `>=` , `>` , `<<` , `>>` , `<=>` , `AND` , `OR` ou `LIKE` peuvent être utilisés pour les comparaisons de colonnes à gauche de la clause `FROM` dans les commandes `SELECT` . Par exemple :

```
mysql> SELECT col1=1 AND col2=2 FROM tbl_name;
```

La fonction `LAST_INSERT_ID()` , qui retourne la plus récente valeur de colonne `AUTO_INCREMENT` . Fonctions d'informations .
`LIKE` est possible avec des colonnes numériques.
Les opérateurs d'expressions régulières étendus `REGEXP` et `NOT REGEXP` .
`CONCAT()` et `CHAR()` avec un argument ou plus de deux arguments. Avec le serveur MySQL, ces fonctions peuvent prendre n'importe quel nombre d'arguments.
Les fonctions `BIT_COUNT()` , `CASE` , `ELT()` , `FROM_DAYS()` , `FORMAT()` , `IF()` , `PASSWORD()` , `ENCRYPT()` , `MD5()` , `ENCODE()` , `DECODE()` , `PERIOD_ADD()` , `PERIOD_DIFF()` , `TO_DAYS()` et `WEEKDAY()` .
L'utilisation de la fonction `TRIM()` pour réduire les chaînes. L'ANSI SQL ne supporte que les suppressions de caractères uniques.
Les fonctions de groupe de la clause `GROUP BY STD()` , `BIT_OR()` , `BIT_AND()` , `BIT_XOR()` et `GROUP_CONCAT()` . Fonctions à utiliser dans les définitions `GROUP BY` .

Pour une liste hiérarchisée des nouvelles extensions qui seront ajoutées à MySQL, vous pouvez consulter la liste de tâche en ligne sur http://dev.mysql.com/doc/mysql/en/TODO.html . C'est la dernière liste qui est utilisée dans ce formulaire. Liste de tâches MySQL .

1.8.5 Différences entre MySQL et le standard SQL−92

Nous tâchons de rendre le serveur MySQL compatible avec le standard ANSI SQL, et le standard ODBC SQL, mais dans certains cas, MySQL se comporte différemment.

- Pour les colonnes de type `VARCHAR` , les espaces terminaux sont supprimés lors du stockage de la valeur. Erreurs connues et manques de MySQL .
- Dans certains cas, les colonnes `CHAR` sont transformées automatiquement en colonnes `VARCHAR` . Changement de type de colonne automatique .
- Les droits d'un utilisateur sur une table ne sont pas supprimés si la table est détruite. Vous devez explicitement utiliser la commande `REVOKE` pour supprimer les droits d'un utilisateur sur une table. Syntaxe de `GRANT` et `REVOKE` .

1.8.5.1 Sous−requêtes

MySQL version 4.1 supporte les sous−requêtes et les tables dérivées (vues anonymes). Une ``sous−requête'' est une commande `SELECT` imbriquée dans une autre commande. Une ``table dérivée'' (vues anonymes) est une sous−requête placée dans une clause `FROM` d'une autre commande. Syntaxe des sous−requêtes .

Pour les versions de MySQL antérieure à la 4.1, la plupart des sous−requêtes peuvent être réécrites avec des jointures et d'autres méthodes. Voyez Se passer des sous−requêtes pour des exemples d'illustration.

1.8.5.2 SELECT INTO TABLE

Le serveur MySQL ne supporte pas encore l'extension Oracle SQL : `SELECT ... INTO TABLE ...` . A la place, le serveur MySQL supporte la syntaxe ANSI SQL `INSERT INTO ... SELECT ...` , qui revient au même. Syntaxe de `INSERT ... SELECT` .

```
INSERT INTO tblTemp2 (fldID)
      SELECT tblTemp1.fldOrder_ID
      FROM tblTemp1 WHERE tblTemp1.fldOrder_ID > 100;
```

Alternativement, vous pouvez utiliser `SELECT INTO OUTFILE...` et `CREATE TABLE ... SELECT` .Depuis la version 5.0, MySQL supporte `SELECT ... INTO` avec les variables serveur. La même syntaxe peut aussi être utilisée dans les procédures stockées, en utilisant les curseurs et les variables locales. Commande `SELECT ... INTO` .

1.8.5.3 Transactions et opérations atomiques

Le serveur MySQL (version 3.23 `MySQL-max` et toutes les versions 4.0 et plus récent) supporte les transactions avec les gestionnaires de tables `InnoDB` et `BDB` . `InnoDB` dispose aussi de la compatibilité `ACID` **totale** . Types de tables MySQL .

Toutefois, les tables non transactionnelles de MySQL telles que `MyISAM` exploitent un autre concept pour assurer l'intégrité des données, appelé `` `opérations atomiques` ''. Les opérations atomiques disposent d'une bien meilleure protection des données pour des performances également accrues. Comme MySQL supporte les deux méthodes, l'utilisateur est capable de choisir celle qui correspond à ses besoins, suivant qu'il a besoin de vitesse ou de sécurité. Ce choix peut être fait table par table.

Comment exploiter les capacités de MySQL pour protéger l'intégrité des données, et comment ces fonctionnalités se comparent elles avec les méthodes transactionnelles ?

- En mode transactionnel, si votre application a été écrite en dépendant de l'appel de `ROLLBACK` au lieu de `COMMIT` dans les situations critiques, les transactions sont plus pratiques. Les transactions s'assurent que les modifications non achevées ou les activités corrosives ne sont pas archivées dans la base. Le serveur a l'opportunité d'annuler automatiquement l'opération, et votre base de données est sauve.

  Le serveur MySQL, dans la plupart des cas, vous permet de résoudre les problèmes potentiels en incluant de simples vérifications avant les modifications, et en exécutant des scripts simples pour vérifier l'intégrité de vos bases de données, ainsi que les incohérences, et pour réparer automatiquement les problèmes, ou encore vous alerter si une erreur est identifiée. Notez qu'en utilisant simplement le log de MySQL, ou en utilisant un log supplémentaire, vous pouvez normalement réparer à la perfection toutes les tables, sans aucune perte de données.
- Souvent, les modifications de données transactionnelles fatales peuvent être réécrites de manière atomique. En général, tous les problèmes d'intégrité que les transactions résolvent peuvent être corrigés avec la commande `LOCK TABLES` ou des modifications atomiques, qui assurent que vous n'aurez jamais d'annulation automatique de la base, ce qui est un problème commun des bases transactionnelles.
- Même un système transactionnel peut perdre des données si le serveur s'arrête. La différence entre les systèmes repose alors dans ce petit laps de temps où ils peuvent perdre des données. Aucun système n'est sécurisé à 100%, mais simplement ``suffisamment sécurisé''. Même Oracle, réputé pour être la plus sûre des bases de données transactionnelles, est montré du doigt pour perdre des données dans ces situations.Pour être tranquille avec MySQL, que vous utilisiez les tables transactionnelles ou pas, vous n'avez besoin que de sauvegardes et de logs de modifications. Avec ces deux outils, vous

pourrez vous protéger de toutes les situations que vous pourriez rencontrer avec d'autres bases de données transactionnelles. De toute manière, il est bon d'avoir des sauvegardes, indépendamment de la base que vous utilisez.

La méthode transactionnelle a ses avantages et ses inconvénients. De nombreux utilisateurs et développeurs d'applications dépendent de la facilité de pallier un problème lorsqu'une annulation semble nécessaire ou presque. Cependant, même si vous êtes néophyte des opérations atomiques, ou plus familier avec les transactions, prenez en considération le gain de vitesse que les tables non transactionnelles offrent. Ces gains vont de 3 a 5 fois la vitesse des tables transactionnelles les plus rapides et les mieux optimisées.

Dans des situations où l'intégrité est de la plus grande importance, le serveur MySQL assure une intégrité du niveau des transactions, ou encore mieux avec les tables non transactionnelles. Si vous verrouillez les tables avec `LOCK TABLES` , toutes les modifications seront bloquées jusqu'à ce que la vérification d'intégrité soit faite (à comparer avec un verrou en écriture), les lectures et insertions sont toujours possibles. Les nouvelles lignes ne seront pas accessibles en lecture tant que le verrou n'aura pas été levé. Avec `INSERT DELAYED` , vous pouvez faire attendre les insertions dans une pile, jusqu'à ce que les verrous soit levés, sans que le client n'attende cette levée de verrou. Syntaxe de `INSERT DELAYED` .

``Atomique'', avec le sens que nous lui donnons, n'a rien de magique. Ce terme signifie simplement que vous pouvez être certain que lorsque vous modifiez des données dans une table, aucun autre utilisateur ne peut interférer avec votre opération, et qu'il n'y aura pas d'annulation automatique (ce qui pourrait arriver avec des tables transactionnelles si nous ne sommes pas trop soigneux). Le serveur MySQL garantit aussi qu'il n'y aura pas de lectures erronées.

Voici quelques techniques pour travailler avec des tables non transactionnelles :

- Les boucles qui requièrent les transactions peuvent normalement être implémentées avec la commande `LOCK TABLES` , et vous n'avez nul besoin de curseur lorsque vous modifiez des lignes à la volée.
- Pour éviter d'utiliser l'annulation `ROLLBACK` , vous pouvez adopter la stratégie suivante :
    - Utilisez la commande `LOCK TABLES ...` pour verrouiller toutes les tables que vous voulez utiliser.
    - Testez vos conditions.
    - Modifiez si tout est correct.
    - Utilisez `UNLOCK TABLES` pour libérer vos tables.

  Ceci est probablement une méthode bien plus rapide que ne le proposent les transactions, avec des annulations `ROLLBACK` possibles mais pas certaines. La seule situation que ce cas ne prend pas en compte est l'interruption du processus au milieu d'une mise à jour. Dans ce cas, tous les verrous seront levés, mais certaines modifications peuvent ne pas avoir été exécutées.
- Vous pouvez aussi utiliser des fonctions pour modifier des lignes en une seule opération. Vous pouvez créer une application très efficace en utilisant cette technique :
    - Modifiez les champs par rapport à leur valeur actuelle.
    - Modifiez uniquement les champs que vous avez réellement changé.

  Par exemple, lorsque nous modifions les données d'un client, nous ne modifions que les données du client qui ont changé et nous vérifions uniquement si les données modifiées ou les données qui en dépendent ont changé comparativement aux données originales. Les tests sur les données modifiées sont faits avec la clause `WHERE` dans la commande `UPDATE` . Si la ligne a été modifiée, nous indiquons au client : `"Some of the data you have changed has been changed by another user"` . En français : "certaines données que vous voulez modifier ont été modifiées par un autre utilisateur". Puis nous affichons l'ancienne ligne et la nouvelle ligne, pour laisser l'utilisateur décider quelle version il veut utiliser.

Cela nous conduit à un résultat proche du verrouillage de ligne, mais en fait, c'est bien mieux, car nous ne modifions que les colonnes qui en ont besoin, en utilisant des valeurs relatives. Cela signifie qu'une commande `UPDATE` typique ressemble à ceci :

```
UPDATE tablename SET pay_back=pay_back+'relative change';

UPDATE customer
 SET
   customer_date='current_date',
   address='new address',
   phone='new phone',
   dette=dette+'emprunt'
 WHERE
   customer_id=id AND address='old address' AND phone='old phone';
```

Comme vous pouvez le voir, c'est très efficace, et fonctionne même si un autre client a modifié la valeur `pay_back` ou `dette` .

- Dans de nombreuses situations, les utilisateurs ont souhaité les commandes `ROLLBACK` et/ou `LOCK TABLES` afin de gérer des identifiant uniques pour certaines tables. Ils peuvent être gérés bien plus efficacement en utilisant une colonne de type `AUTO_INCREMENT` , en corrélation avec la fonction `LAST_INSERT_ID()` ou la fonction C `mysql_insert_id()` . Fonctions d'informations . `mysql_insert_id()` .

  Vous pouvez éviter le verrouillage de ligne. Certaines situations le requièrent vraiment, mais elles sont rares. Les tables `InnoDB` supportent le verrouillage de ligne. Avec les tables `MyISAM` , vous pouvez utiliser une colonne de type sémaphore, et faire ceci :

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID;
```

  MySQL retournera 1 pour le nombre de lignes affectées si la ligne a été trouvée, car `row_flag` ne vaut pas déjà 1 dans la ligne originale.

  Vous pouvez comprendre la requête ci−dessus comme si le serveur MySQL avait utilisé la commande suivante :

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID AND row_flag <> 1;
```

1.8.5.4 Procédures stockées et triggers

Les procédures stockées sont implémentées en version 5.0. Procédures stockées et fonctions .

Un trigger est une procédure stockée qui est activée lorsqu'un événement particulier survient. Par exemple, vous pouvez installer une procédure stockée qui est déclenchée dès qu'une ligne est effacée dans une table d'achat, pour que le client soit automatiquement effacé si tous ses achats sont effacés.

1.8.5.5 Les clés étrangères

En MySQL version 3.23.44 et plus récentes, les tables `InnoDB` supportent les vérifications d'intégrité référentielles. Le moteur de tables `InnoDB` . Pour les autres types de tables, le serveur mySQL accepte la syntaxe `FOREIGN KEY` dans la commande `CREATE TABLE` , mais ne la prend pas en compte.

Pour les autres moteurs de stockage que `InnoDB` , MySQL analyse la clause `FOREIGN KEY` de la commande `CREATE TABLE` , mais ne l'utilise pas et ne la stocke pas. Dans le futur, l'implémentation va stocker cette information dans le fichier de spécifications de tables, pour qu'elle puisse être lue par `mysqldump` et ODBC. Ultérieurement, les contraintes de clé étrangères seront incluses dans les tables `MyISAM` .

Voici des avantages aux contraintes de clés étrangères :

- En supposant que les relations soient proprement conçues, les clés étrangères rendent plus difficile pour un programmeur d'insérer des valeurs incohérentes dans la base.
- La vérification centralisée de contraintes par le serveur de base de données rend inutiles l'application de ces vérifications du coté de l'application. Cela élimine la possibilité que d'autres applications ne fassent pas les vérifications de la même façon que les autres.
- L'utilisation des modifications et effacement en cascade simplifie le code du client.
- Les règles de clés étrangères proprement conçues aident à la documentation des relations entre les tables.

Gardez bien en tête que ces avantages ont un coût supérieur pour le serveur de bases, qui doit effectuer les tests. Les vérifications supplémentaires affectent les performances, ce qui est parfois suffisamment rebutant pour des applications qui les éviteront. Certaines applications commerciales ont placé la logique de vérification dans l'application, pour cette raison.

MySQL donne aux développeurs de bases de données le choix de leur approche. Si vous n'avez pas besoin des clés étrangères, et que vous voulez éviter leur surcoût, vous pouvez choisir un autre type de table, comme `MyISAM` . Par exemple, les tables `MyISAM` sont extrêmement rapides pour les applications qui font essentiellement des opérations `INSERT` et `SELECT` , car elles peuvent être utilisées simultanément. Problèmes avec le verrouillage de tables .

Si vous décidez de ne pas tirer avantage des contraintes d'intégrité, vous devez garder en tête ces conseils :

- En l'absence de vérification du coté du serveur, l'application doit se charger de ces vérifications. Par exemple, elle doit s'assurer que les lignes sont insérées dans le bon ordre, et que les lignes ne sont pas orphelines. Il faut aussi pouvoir rattraper une erreur au milieu d'une opération multiple.
- Si la clause `ON DELETE` est la seule fonctionnalité nécessaire, notez que depuis MySQL version 4.0, vous pouvez utiliser des commandes `DELETE` multi−tables pour effacer les lignes dans plusieurs tables en une seule commande. Syntaxe de `DELETE` .
- Un palliatif au manque de `ON DELETE` est d'ajouter la commande `DELETE` appropriée lorsque vous effacez des lignes dans une table qui dispose d'une clé étrangère,. EN pratique, c'est souvent plus rapide que d'utiliser les clés étrangères, et c'est plus portable.

Soyez conscient que l'utilisation des clés étrangères dans certaines circonstances peuvent conduire à des problèmes :

- Les clés étrangères règlent des problèmes de cohérence, mais il est nécessaire de concevoir les contraintes correctement, pour éviter les contraintes circulaires, ou des cascades d'effacements incorrects.
- Il n'est pas exceptionnel pour un administrateur de créer une topologie de relations qui rende difficile la restauration de bases à partir d'une sauvegarde. MySQL résout ce problème en vous permettant de désactiver temporairement les contraintes. Contraintes `FOREIGN KEY` . Depuis MySQL 4.1.1, `mysqldump` génère un fichier d'export qui exploite cette possibilité de désactivation automatique à l'import.

Notez que les clés étrangères SQL sont utilisées pour assurer la cohérence des données, et non pas pour joindre des tables. Si vous voulez obtenir des résultats de tables multiples dans une commande `SELECT` , vous devez le faire avec une jointure :

```
SELECT * FROM t1, t2 WHERE t1.id = t2.id;
```

Syntaxe de `JOIN` . Utiliser les clefs étrangères .La syntaxe `FOREIGN KEY` sans `ON DELETE ...` est souvent utilisée par les applications ODBC pour produire automatiquement des clauses `WHERE` .

1.8.5.4 Procédures stockées et triggers

1.8.5.6 Les vues

Il est prévu d'implémenter les vues dans la version 5.0 ou 5.1 du serveur MySQL.

Historiquement, MySQL a été utilisé dans des applications et sur les systèmes Web, où l'auteur de l'application a un contrôle complet sur le système de base de données. L'utilisation a évolué au cours du temps, et de nombreux utilisateurs pensent maintenant que c'est important.

Les vues anonymes (**tables dérivées** , une sous−requête de la clause `FROM` de la commande `SELECT`) sont déjà disponibles en version 4.1.

Les vues sont la plupart du temps utiles pour donner accès aux utilisateurs à un ensemble de relations représentées par une table (en mode inaltérable). Beaucoup de bases de données SQL ne permettent pas de mettre à jour les lignes dans une vue, vous devez alors faire les mises à jour dans les tables séparées. Sécurité générale et accès à MySQL .

De nombreuses bases de permettent pas les modifications de vues, mais permettent les mises à jour dans des tables individuelles. Lors de la conception de notre système de vues, nous envisageons le support complet de la règle ``numéro 6 de Codd'' (autant que possible en SQL) : toutes les vues qui sont théoriquement modifiables, et doivent l'être en pratique.

1.8.5.7 '−−' comme début de commentaire

Certaines bases de données SQL utilisent `'--'` comme début de commentaire. Le serveur MySQL utilise `'#'` . Vous pouvez aussi utiliser la syntaxe du langage C `/* ceci est un commentaire */` . Syntaxe des commentaires .

MySQL version 3.23.3 et plus récent supporte les commentaires de type `'--'` , si ce commentaire est suivi d'un espace. Ceci est dû au fait que ce type de commentaire a causé beaucoup de problèmes avec les requêtes générées automatiquement, qui contiennent du code tel que celui−ci, où nous insérons automatiquement la valeur du paiement dans la table `!payment!` :

```
UPDATE tbl_name SET credit=credit-!payment!
```

Pensez à ce qui se passe lorsque la valeur de `payment` est négative, comme `-1` :

```
UPDATE account SET credit=credit--1
```

`credit--1` est une expression légale en SQL mais `--` est interprété comme un commentaire et la fin de l'expression est ignorée. Le résultat est que la commande prend une signification complètement différente :

```
UPDATE account SET credit=credit
```

La commande ne produit aucun changement! Cela montre que l'utilisation des commentaires de type `'--'` peuvent avoir des conséquences graves.

En utilisant notre implémentation des commentaires avec le serveur MySQL version 3.23.3 et plus récent, `1-- ceci est un commentaire` ne pose pas ce type de problème.

Une autre fonctionnalité supplémentaire est que le client en ligne de commande `mysql` supprime toutes les lignes qui commencent par `'--'` .

Les informations suivantes sont destinées aux utilisateurs de MySQL avec des versions antérieures à la version 3.23.3 :

Si vous avez un programme SQL dans un fichier texte qui contient des commentaires au format `'--'`, il est recommandé d'utiliser l'utilitaire `replace` pour assurer la conversion en caractères `'#'` :

```
shell> replace " --" " #" < text-file-with-funny-comments.sql \
       | mysql database
```

au lieu du classique :

```
shell> mysql database < text-file-with-funny-comments.sql
```

Vous pouvez aussi éditer le fichier de commande ``lui−même'' pour remplacer les commentaires `'--'` par des commentaires `'#'` :

```
shell> replace " --" " #" -- text-file-with-funny-comments.sql
```

Puis, rétablissez−les avec :

```
shell> replace " #" " --" -- text-file-with-funny-comments.sql
```

1.8.6 Comment MySQL gère les contraintes

Comme MySQL permet de travailler avec des moteurs de tables transactionnels ou pas, les contraintes sont gérées un peu différemment sur MySQL que sur les autres bases de données.

Nous devons gérer le cas où vous avez modifié beaucoup de lignes avec une table non−transactionnelles, qui ne peut annuler les modifications en cas d'erreur.

La philosophie de base est d'essayer de détecter autant d'erreur que possible au moment de la compilation, mais d'essayer de réparer les erreurs à l'exécution. Nous y arrivons dans la plupart des cas, mais pas tout le temps. Ce qui est prévu pour un futur proche .

La solution de base de MySQL est d'interrompre la commande au milieu, ou de faire de son mieux pour réparer le problème et continuer.

Voici ce qui se passe dans les différents types de contraintes.

1.8.6.1 Contrainte avec PRIMARY KEY / UNIQUE

Normalement, vous allez obtenir une erreur lorsque vous essayerez d'insérer `INSERT` ou modifier `UPDATE` une ligne qui causera une violation de clé primaire, unique ou étrangère. Si vous utilisez un moteur transactionnelle, comme InnoDB, MySQL va immédiatement annuler la transaction. Si vous utilisez un moteur non−transactionnel, MySQL va s'arrêter à la mauvaise ligne, et laisser les dernières lignes intactes.

Pour rendre la vie plus facile, MySQL a ajouté le support de l'option `IGNORE` aux commandes qui peuvent rencontrer un problème de clé (comme `INSERT IGNORE ...`). Dans ce cas, MySQL va ignorer les problèmes de clé et la ligne, et continuer à traiter les lignes suivantes. Vous pouvez obtenir la liste des alertes avec la fonction `mysql_info()` et, dans les prochaines versions de MySQL 4.1, vous pourrez aussi les voir avec la commande `SHOW WARNINGS` . `mysql_info()` . Syntaxe de `SHOW WARNINGS` .

Notez que pour le moment, seules les tables `InnoDB` supportent les clés étrangères. Contraintes `FOREIGN KEY` . Le support des clés étrangères des tables `MyISAM` sont prévues pour la version 5.0.

1.8.6.2 Contraintes avec NOT NULL et DEFAULT

Pour être capable de supporter facilement les tables non−transactionnelles, tous les champs de MySQL ont des valeurs par défaut.

Si vous insérez une valeur ``invalide'' dans une colonne, comme `NULL` dans une colonne `NOT NULL` , ou une valeur numérique trop grand dans une colonne numérique, MySQL inscrit dans la colonne la ``meilleure valeur possible'', sans produire d'erreur :

- Si vous stockez un chiffre hors de l'intervalle de validité d'une colonne numérique, MySQL stocke à la place zéro, la plus petite valeur possible, ou bien la plus grande valeur possible.
- Pour les chaînes, MySQL stocke la chaîne vide ou bien la plus grande chaîne qui peut être stockée dans cette colonne.
- Si vous essayez de stocker une chaîne qui ne commence pas par un chiffre dans une colonne numérique, MySQL stocke 0.
- Si vous essayez de stocker `NULL` dans une colonne qui n'accepte pas la valeur `NULL` , MySQL stocke 0 ou `''` (la chaîne vide). Ce dernier comportement peut, pour des insertions de ligne unique, être modifié par l'option de compilation `-DDONT_USE_DEFAULT_FIELDS` . Les options habituelles de `configure` . Cela fait que les commandes `INSERT` génèreront une erreur à moins que vous ne spécifiez explicitement les valeurs pour toutes les colonnes qui requièrent une valeur non− `NULL` .
- MySQL vous permet de stocker des dates incorrectes dans les colonnes de type `DATE` et `DATETIME` (comme `'2000-02-31'` ou `'2000-02-00'`). L'idée est que ce n'est pas le travail du serveur SQL de valider les dates. Si MySQL peut stocker une valeur et relire exactement la même valeur, MySQL la stockera. Si la date est totalement erronée (hors de l'intervalle de validité), la valeur spéciale `'0000-00-00'` est stockée dans la colonne.

La raison de cette règle ci−dessus est que nous ne pouvons pas vérifier ces conditions avant que la requête ne soit exécutée. Si nous rencontrons un problème après la modification de quelques lignes, nous ne pourront pas annuler la modification, car la table ne le supporte peut−être pas. L'alternative qui consiste à s'arrêter c'est pas envisageable non plus, car nous aurions alors fait la moitié du travail, ce qui sera alors le pire scénario. Dans ce cas, il vaut mieux faire "du mieux possible", et continuer comme si rien n'était arrivé. En MySQL 5.0, nous envisageons d'améliorer cela en fournissant des alertes de conversions automatique, ainsi qu'une option pour vous permettre d'annuler la commande si elle n'utilise que des tables transactionnelles.

Ceci signifie qu'il ne faut pas compter sur MySQL pour vérifier le contenu des champs, mais de gérer cela au niveau de l'application.

1.8.6.3 Constante avec ENUM et SET

En MySQL 4.x `ENUM` n'est pas une véritable contrainte, mais simplement un moyen plus efficace de stocker des champs qui peuvent prendre un nombre limité de valeurs différentes. C'est la même raison pour laquelle `NOT NULL` n'est pas respecté. Contrainte avec `NOT NULL` et `DEFAULT` .

Si vous insérez une valeur invalide dans un champs `ENUM` , la colonne prendra la valeur réservée `0` , qui sera représentée par une chaîne vide, en mode chaîne. Le type `ENUM` .

Si vous insérez une mauvais option dans un ensemble `SET` , la valeur sera ignorée. Le type `SET` .

1.8.7 Erreurs connues, et limitations de MySQL

1.8.7.1 Erreurs connues en 3.23 et corrigées ultérieurement

Les erreurs suivantes sont connues mais restent non corrigées en MySQL 3.23, car pour les corriger, il nous faudrait modifier trop de code : cela risquerait d'introduire des bugs bien pire. Ces bugs sont considérés comme ``non nuisibles'' ou ``supportables''.

- Il est possible de rencontrer un blocage en utilisant la commande `LOCK TABLE` sur de multiples tables, puis, avec la même connexion, faire un `DROP TABLE` sur l'une d'entre elle, alors qu'un autre thread essai de verrouiller la table. Il est toutefois possible d'utiliser la commande `KILL` sur l'un des threads en question, pour résoudre ce problème. Corrigé en 4.0.12.
- `SELECT MAX(key_column) FROM t1,t2,t3...` où l'une des tables est vide ne retourne pas `NULL` mais plutôt le valeur maximale de la colonne. Corrigé en 4.0.11.
- `DELETE FROM heap_table` sans clause `WHERE` ne fonctionne pas sur une table `HEAP` .

1.8.7.2 Erreurs de la version 4.0, corrigées plus tard

Voici la liste des bugs/erreurs qui ne sont pas corrigées en MySQL 4.0 car cette correction prendrait trop de manipulations, qui risqueraient d'introduire encore d'autres bugs. Les bugs sont aussi classé comme ``non fatal'' ou ``supportable.''

- Dans une `UNION` , le premier `SELECT` définit le type, et les propriétés `max_length` et `NULL` pour les colonnes du résultat. En MySQL 4.1.1, ces propriétés sont définies comme la combinaison de toutes les parties de l' `UNION` .
- Dans une commande `DELETE` sur de nombreuses tables, vous ne pouvez pas utiliser les alias pour effacer dans une table. Ceci est corrigé en MySQL 4.1.
- Vous ne pouvez pas mélanger des clauses `UNION ALL` et `UNION DISTINCT` dans la même requête. Si vous utilisez l'option `ALL` dans une des clauses `UNION` , elle est utilisée pour toutes ces clauses.

1.8.7.3 Bugs connus / limitations de MySQL

Les problèmes suivants sont connus, et sont en tête de liste pour être corrigés :

- Il n'est pas possible de mélanger `UNION ALL` et `UNION DISTINCT` dans la même requête. Si vous utilisez `ALL` pour `UNION` alors il faut l'utiliser partout.
- Si un utilisateur a une transaction longue, et qu'un autre utilisateur efface une table qui est modifiée par la même transaction, il y a quelques chances que le log binaire n'enregistre pas la commande `DROP TABLE` avant que la table ne soit utilisée par la transaction elle−même. Nous envisageons de corriger cela en version 5.0, en forçant `DROP TABLE` a attendre jusqu'à ce que la table ne soit plus utilisée par la transaction.
- Lors de l'insertion d'un grand entier (valeur entre 2^63 et 2^64−1) dans une colonne de type décimal ou chaîne, il sera enregistré comme une valeur négative, car le nombre est considéré comme un entier signé dans ce contexte. Il est prévu de corriger cela en 4.1.
- `FLUSH TABLES WITH READ LOCK` ne bloque pas `CREATE TABLE` ou `COMMIT` , ce qui peut causer des problèmes avec la position du log lors d'une sauvegarde complète des tables et du log binaire.
- `ANALYZE TABLE` sur une table de type `BDB` , peut rendre la table inutilisable, dans certains cas, jusqu'au prochain redémarrage de `mysqld` . Lorsque cela survient, vous rencontrez les erreurs suivantes dans le fichier d'erreur MySQL :
  ```
  001207 22:07:56  bdb:  log_flush: LSN past current end-of-log
  ```
- MySQL accepte les parenthèses dans la clause `FROM` , mais les ignore silencieusement. La raison de l'absence d'erreur est que de nombreux clients qui génèrent des requêtes, ajoutent les parenthèses dans la clause `FROM` même si elles sont inutiles.
- Concaténer plusieurs `RIGHT JOINS` ou combiner des jointures `LEFT` et `RIGHT` dans la même requête ne donnera pas de résultat correct si MySQL ne génère que des lignes `NULL` pour la table précédent le `LEFT` ou avant la jointure `RIGHT` . Cela sera corrigé en 5.0, en même temps que le support des parenthèses pour la clause `FROM` .
- N'exécutez pas de commande `ALTER TABLE` sur une table `BDB` sur laquelle vous avez exécuté des transactions à plusieurs commandes, jusqu'à ce que ces transactions soient achevées : la transaction sera probablement ignorée.

- `ANALYZE TABLE` , `OPTIMIZE TABLE` et `REPAIR TABLE` peuvent causer des problèmes sur les tables avec lesquelles vous utilisez la commande `INSERT DELAYED` .
- Faire un `LOCK TABLE ...` et `FLUSH TABLES ...` ne vous garantit pas qu'il n'y a pas une transaction en court sur la table.
- Les tables `BDB` sont lentes à ouvrir. Si vous avez de nombreuses tables `BDB` dans une base, cela prendra du temps au client `mysql` pour accéder à la base si vous n'utilisez pas l'option `-A` , ou si vous utilisez la commande `rehash` . C'est particulièrement vrai si vous n'avez pas de cache de table important.
- La réplication utilise un log de niveau requête : le maître écrit les requêtes exécutées dans le log binaire. C'est une méthode de log rapide, compacte et efficace, qui fonctionne à la perfection dans la plupart des situations. Même si nous n'avons jamais vu d'occurrence de ce problème, il est théoriquement possible pour les données du maître et de l'esclave de différer si une requête non−déterministe est utilisée pour modifier les données, c'est à dire si elle est laissé au bon vouloir de l'optimiseur, ce qui n'est pas une bonne pratique même sans la réplication. Par exemple :
    - Des commandes `CREATE ... SELECT` ou `INSERT ... SELECT` qui insèrent zéro ou `NULL` valeurs dans la colonne `AUTO_INCREMENT` .
    - `DELETE` si vous effacez des lignes dans une table qui a une propriété `ON DELETE CASCADE` .
    - Les commandes `REPLACE ... SELECT` , `INSERT IGNORE ... SELECT` , si vous avez des clés en double, dans les données insérées.

  IF et seulement si ces requêtes n'ont pas de clause `ORDER BY` , qui garantisse un ordre déterministe .Effectivement, par exemple, pour les commandes `INSERT ... SELECT` sans clause `ORDER BY` , le `SELECT` peut retourner les lignes dans un ordre différent, ce qui aura pour résultat de donner des rangs différents et donnera des numéros d'identifiants différents aux colonnes `auto_increment`), en fonction des choix fait par les optimiseurs du maître et de l'esclave. Une requête sera optimisée différemment sur l'esclave et sur le maître si :
    - Les fichiers utilisés par les deux requêtes ne sont pas exactement les mêmes. Par exemple, `OPTIMIZE TABLE` a été exécuté sur le maître et pas sur l'esclave (pour corriger cela, depuis MySQL 4.1.1, `OPTIMIZE` , `ANALYZE` et `REPAIR` sont aussi écrits dans le log binaire).
    - La table est stockées sur un moteur de stockage différent sur le maître et sur l'esclave : c'est possible d'utiliser des moteurs de tables différents. Par exemple, le maître utiliser `InnoDB` et l'esclave MyISAM, car l'esclave a moins d'espace disque.
    - Les tailles de buffer MySQL (`key_buffer_size` , etc.) sont différentes sur le maître et sur l'esclave.
    - Le maître et l'esclave utilisent des versions différentes de MySQL, et le code de l'optimiseur est différent entre ces versions.

  Ce problème peut aussi affecter la restauration de base, utilisant `mysqlbinlog` ou `mysql` .

  Le plus simple pour éviter ces problèmes dans tous les cas est d'ajouter toujours une clause `ORDER BY` aux requêtes non−déterministe, pour s'assure que les lignes sont traitées dans le même ordre. Dans le futur, MySQL va ajouter automatiquement une clause `ORDER BY` si nécessaire.

Les problèmes suivants sont connus et seront corrigés en leur temps :

- `mysqlbinlog` n'efface pas les fichiers temporaires laissés après une commande `LOAD DATA INFILE` . L'utilitaire binaire `mysqlbinlog` .
- Il n'est pas possible de renommer une table temporaire.
- Lors de l'utilisation de la fonction `RPAD` , ou de toute autre fonction de chaîne qui peut ajouter des espaces à droite de la chaîne, dans une requête qui utilise une table temporaire pour la résolution, alors toutes les chaînes verront leurs espaces terminaux être supprimés. Voici un exemple d'une telle requête :

```
SELECT RPAD(t1.field1, 50, ' ') AS f2, RPAD(t2.field2, 50, ' ') AS f1 FROM table1 as t1 LEFT JOIN table2 AS t2 ON
t1.record=t2.joinID ORDER BY t2.record;
```

Le résultat final de ceci est que vous ne pourrez pas obtenir les espaces à gauche dans ces chaînes.

Le comportement décrit ci−dessus existe dans toutes les versions de MySQL.

La raison à cela est due au fait que les tables de type HEAP, qui sont utilisées en premier comme table temporaires, ne sont pas capables de gérer des colonnes de type VARCHAR.

Ce comportement sera corrigé dans l'une des versions de la série des 4.1.

- A cause de la méthode de stockage des tables de définitions de fichiers, il n'est pas possible d'utiliser le caractère 255 (`CHAR(255)`) dans les noms des tables, colonnes ou énumérations. Il est prévu de corriger de problème dans les versions version 5.1, lorsque nous aurons établi un nouveau format de définition des fichiers.
- Lorsque vous utilisez la commande `SET CHARACTER SET` , il n'est pas possible d'utiliser les caractères traduits dans les noms de bases, de tables ou de colonnes.
- Il n'est pas possible d'utiliser `_` ou `%` avec la commande `ESCAPE` dans la clause `LIKE... ESCAPE` .
- Si vous avez une colonne de type `DECIMAL` avec un nombre stocké dans un autre format (+01.00, 1.00, 01.00), `GROUP BY` peut considérer ces valeurs comme différentes.
- Lorsque `DELETE FROM merge_table` est utilisé sans la clause `WHERE` , elle va simplement effacer le fichier de la table, et ne pas effacer les tables associées.
- Vous ne pouvez pas compiler le serveur dans un autre dossier lorsque vous utilisez les `MIT-pthreads` . Comme cela requiert une modification des `MIT-pthreads` , nous ne corrigerons pas ce problème. Remarques sur MIT−pthreads .
- Les valeurs de type `BLOB` ne peuvent pas être utilisées ``correctement'' dans les clauses `GROUP BY` ou `ORDER BY` ou `DISTINCT` . Seuls, les `max_sort_length` premiers octets (par défaut, 1024) seront utilisés pour les comparaisons de `BLOB` . Ceci peut être modifié avec l'option `-O max_sort_length` de `mysqld` . Un palliatif à ce problème est d'utiliser une sous partie de chaîne : `SELECT DISTINCT LEFT(blob,2048) FROM tbl_name` .
- Les calculs sont faits avec des `BIGINT` ou `DOUBLE` (les deux sont normalement de 64 bits). La précision dépend alors de la fonction utilisée. La règle générale est que les fonctions de bits utilisent la précision des `BIGINT` , `IF` et `ELT()` utilisent la précision des `BIGINT` ou `DOUBLE` , et les autres utilisent la précision des `DOUBLE` . Il faut donc éviter d'utiliser les entiers non signés de grande taille, surtout s'ils dépassent la taille de 63 bits (9223372036854775807) pour toute autre fonction que les champs de bits ! La version 4.0 gère bien mieux les `BIGINT` que la 3.23.
- Toutes les colonnes de type chaînes, hormis les `BLOB` et `TEXT` , voient automatiquement leurs caractères blancs finaux supprimés. Pour le type `CHAR` c'est correct, et c'est considéré comme une fonctionnalité par la norme ANSI SQL92. Le hic est que pour le serveur MySQL les colonnes `VARCHAR` sont traitées de la même façon.
- Vous ne pouvez avoir que des colonnes de taille 255 pour les `ENUM` et `SET` .
- Avec les fonctions d'agrégation `MIN()` , `MAX()` et compagnie, MySQL compare actuellement les colonnes de type `ENUM` et `SET` par leur valeur de chaîne, plutôt que par leur position relative dans l'ensemble.
- `safe_mysqld` redirige tous les messages de `mysqld` vers le log `mysqld` . Le problème est que si vous exécutez `mysqladmin refresh` pour fermer et ouvrir à nouveau l'historique, `stdout` et `stderr` sont toujours redirigés vers l'ancien log. Si vous utilisez `--log` , vous devriez éditer `safe_mysqld` pour envoyer les messages vers `'hostname'.err` au lieu de `'hostname'.log` , de façon à pouvoir facilement récupérer la place de l'ancien log, en effaçant les vieux, et en exécutant `mysqladmin refresh` .
- Dans la commande `UPDATE` , les colonnes sont modifiées de gauche à droite. Si vous faite référence à une colonne modifiée, vous obtiendrez sa valeur modifiée, plutôt que sa valeur originale. Par exemple :

```
mysql> UPDATE tbl_name SET KEY=KEY+1,KEY=KEY+1;
```

  Cette commande va modifier la colonne `KEY` avec `2` au lieu de `1` .
- Vous ne pouvez pas utiliser les tables temporaires plus d'une fois dans la même requête. Par exemple, cette commande ne fonctionne pas :

```
mysql> SELECT * FROM temporary_table, temporary_table AS t2;
```

* `RENAME` ne fonctionne pas avec les tables `TEMPORARY` , ou les tables utilisées dans un rassemblement (`MERGE`).
* L'optimiseur peut gérer la clause `DISTINCT` différemment si vous utilisez des colonnes cachées dans une jointure. Dans une jointure, les colonnes cachées sont comptées comme une partie du résultat (même si elles ne sont pas montrées), tandis que dans les requêtes normales, les colonnes cachées ne participent pas aux `DISTINCT` . Nous allons probablement modifier ceci dans le futur, pour ne jamais exploiter les colonnes cachées avec `DISTINCT` .

  Voici un exemple :

  ```
  SELECT DISTINCT mp3id FROM band_downloads
        WHERE userid = 9 ORDER BY id DESC;
  ```

  et

  ```
  SELECT DISTINCT band_downloads.mp3id
        FROM band_downloads,band_mp3
        WHERE band_downloads.userid = 9
        AND band_mp3.id = band_downloads.mp3id
        ORDER BY band_downloads.id DESC;
  ```

  Dans le second cas, MySQL 3.23.x pourrait vous donner deux lignes identiques dans le résultat (car les lignes cachées `id` diffèrent).Notez que cela n'arrive que pour les requêtes où vous n'avez pas de colonnes de la clause ORDER BY dans le résultat, ce que vous ne pourriez pas faire en ANSI SQL.
* Comme le serveur MySQL vous permet de travailler avec des tables qui ne supportent pas les transactions, et donc, l'annulation `rollback` , certains comportements sont différents avec MySQL d'avec d'autres serveurs SQL. C'est nécessaire pour s'assurer que MySQL n'a jamais besoin d'annuler une commande SQL. Cela peut sembler un peu étrange au moment ou les colonnes doivent être vérifiées par l'application, mais cela vous fournit une accélération notable, à cause d'optimisations qui ne pourraient pas avoir lieu ailleurs.Si vous donnez une valeur incorrecte à une colonne, MySQL va stocker le `meilleur code possible` dans la colonne, au lieu d'annuler la transaction :
    * Si vous essayez de stocker une valeur qui est hors de l'intervalle de validité dans une colonne numérique, MySQL va stocker la plus petite ou la plus grande valeur qu'il connaisse dans cette colonne.
    * Si vous essayez de stocker une chaîne qui ne commence pas par un chiffre dans une colonne numérique, MySQL va stocker 0.
    * Si vous essayez de stocker la valeur `NULL` dans une colonne qui n'accepte pas la valeur `NULL` , le serveur MySQL va stocker 0 ou `''` (chaîne vide) à la place : ce comportement peut être modifié avec l'option de compilation −DDONT_USE_DEFAULT_FIELDS).
    * MySQL vous autorise le stockage de dates erronées dans les colonnes de type `DATE` et `DATETIME` (comme 2000−02−31 ou 2000−02−00). L'idée est que ce n'est pas au serveur SQL de faire le travail de validation. Si MySQL peut stocker une date, et relire exactement cette date, alors MySQL va stocker cette date. Si la date est totalement fausse (hors de l'intervalle de validité du serveur), la valeur spéciale 0000−00−00 sera utilisée.
    * Si vous utilisez une valeur non supportée avec une colonne de type `ENUM` , la valeur stockée sera la chaîne vide, de valeur numérique 0.
    * Si vous utilisez une valeur invalide dans une colonne de type `SET` , la valeur sera ignorée.
* Si vous exécutez une `PROCEDURE` sur une requête qui retourne un résultat vide, dans certains cas, `PROCEDURE` ne transformera pas les colonnes.
* La création de table de type `MERGE` ne vérifie pas si les tables sous−jacentes sont de type compatible.

* Le serveur MySQL ne supporte pas encore les valeurs Server `NaN` , `-Inf` et `Inf` pour les doubles. Utiliser ces valeurs générera des problèmes lorsque vous essayerez d'exporter et d'importer des données. Comme solution temporaire, vous pouvez remplacer `NaN` par `NULL` (si possible) et `-Inf` et `Inf` par les valeurs maximales possibles des colonnes `double` .
* Si vous utilisez la commande `ALTER TABLE` pour ajouter un index de type `UNIQUE` à un table utilisée dans un rassemblement de tables `MERGE` , puis que vous utilisez `ALTER TABLE` pour ajouter un index normal à la table `MERGE` , l'ordre des clés sera différent pour les tables s'il y avait déjà une ancienne clé qui n'était pas unique. Ceci est dû au fait que `ALTER TABLE` place les clés `UNIQUE` avant les clés normales, pour être capable de détecter les clés doublons plus vite.

Les bogues suivants sont connus dans les anciennes versions de MySQL :

* Vous pouvez obtenir un thread gelé si vous utilisez la commande `DROP TABLE` sur une table qui fait partie des tables verrouillées par `LOCK TABLES` .
* Dans les cas suivants, vous pouvez obtenir un crash :
    * Le gestionnaire d'insertions retardées a déjà des insertions en attente pour une table.
    * `LOCK table` avec `WRITE` .
    * `FLUSH TABLES` .
* Pour les versions de MySQL avant la 3.23.2, une commande `UPDATE` qui modifiait une clé avec la clause `WHERE` sur la même clé, pouvait échouer car la même clé était utilisée pour rechercher les lignes et la même ligne pouvait être trouvée plusieurs fois :
```
UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100;
```
  Un palliatif est :
```
mysql> UPDATE tbl_name SET KEY=KEY+1 WHERE KEY+0 > 100;
```
  Cela fonctionnera, car MySQL ne va pas utiliser d'index sur une expression dans la clause `WHERE` .
* Avant la version 3.23 de MySQL, tous les types numériques étaient traités comme des champs à virgule fixe. Cela signifie que vous deviez spécifier le nombre de décimales que le champ devait avoir. Tous les résultats étaient retournés avec le nombre correct de décimales.

Pour les bogues spécifiques aux systèmes d'exploitation, voyez la section sur la compilation et le port. Installer MySQL à partir des sources . Port vers d'autres systèmes .

2 Installer MySQL

Ce chapitre décrit comment obtenir et installer MySQL :

- **Déterminez si votre plate−forme est supportée.** Notez que tous les systèmes ne supportent pas MySQL de la même façon. MySQL est plus robuste et efficace que sur d'autres. Voyez Plate−formes supportées par MySQL pour plus de détails.
- **Choisissez une distribution à installer.** Plusieurs versions de MySQL sont disponibles, dans pluiseurs formats. Vous pouvez choisir une version preparée avec des exécutables pré−compilés, ou bien une version source. En cas de doute, utilisez la version binaire. Nous fournissons aussi un accès public à notre serveur de développement pour tester le nouveau code. Pour déterminer quelle version et quel type utiliser, voyez Choisir quelle distribution installer .
- **Téléchargez la distribution que vous souhaitez.** Pour une liste de site sur lesquels vous pouvez télécharger MySQL, voyez Obtenir MySQL . Vous pouvez vérifier l'intégrité de votre téléchargement en utilisant les instructions de Vérifier l'intégrité du paquet avec MD5 ou `GnuPG` .
- **Installez la distribution.** Pour les distributions binaires, voyez I Installer MySQL sous les autres systèmes Unix . Pour les distributions source, utilisez Installer MySQL à partir des sources . Chaque jeu d'instruction inclut une section spécifique aux plate−formes.

  Note : si vous envisagez de changer la version d'une installation existante de MySQL vers une nouvelle version, plutôt que d'installer MySQL pour la première fois, voyez la section Mise à jour ou rétrogradage de MySQL pour des informations sur les mises à jour, et sur les problèmes que vous pourriez rencontrer.

  Si vous rencontrez les problèmes d'installation, voyez la section Remarques spécifiques aux systèmes d'exploitation pour des informations sur les solutions aux proiblèmes spécifiques des plates−formes.
- **Pour la procédure post−installation, voyez** Configuration et tests consécutifs à l'installation . Ces procedures s'appliquent aussi bien à la distribution binaire que la distribution source. Cette section décrit aussi comment sécuriser les comptes initiaux MySQL, **qui n'ont pas de mot de passe** jusqu'à ce que vous leur assigniez un.
- Si vous voulez exécuter des scripts de tests MySQL, le support Perl de MySQL doit être disponible. Commentaires sur l'installation de Perl .

2.1 Notes générales à propos de l'installation

Avant d'installer MySQL, vous devez :

- Déterminer si MySQL fonctionne ou pas sur votre plate−forme.
- Choisir une distribution.
- Télécharger la distribution et vérifier son intégrité.

Cette section contient les informations nécessaires pour réaliser ces étapes. Après cela, vous pouvez utiliser les autres instructions des autres chapitres, pour installer la distribution.

2.1.1 Systèmes d'exploitation supportés par MySQL

Nous utilisons `GNU Autoconf` , alors il est possible de porter MySQL sur tous les systèmes modernes qui utilisent les threads Posix et un compilateur C++. Pour compiler uniquement le client, un compilateur C++ est simplement nécessaire. Nous utilisons et développons le logiciel nous−mêmes, en commençant par Sun Solaris (Versions 2.5 − 2.7) et SuSE Linux version 7.x.

Notez que pour de nombreux systèmes d'exploitation, le support natif des threads ne fonctionne qu'avec les dernières versions. MySQL a été compilé avec succès sur les combinaisons système d'exploitation/paquet de threads suivants :

- AIX 4.x, 5.x avec les threads natifs. Notes sur IBM−AIX .
- Amiga.
- BSDI 2.x avec le paquet `MIT-pthreads` . Notes relatives aux versions 2.x de BSD/OS .
- BSDI 3.0, 3.1 et 4.x avec les threads natifs. Notes relatives aux versions 2.x de BSD/OS .
- DEC Unix 4.x avec les threads natifs. Notes sur Alpha−DEC−UNIX Notes (Tru64) .
- FreeBSD 2.x avec le paquet `MIT-pthreads` . Notes relatives à FreeBSD .
- FreeBSD 3.x et 4.x avec les threads natifs. Notes relatives à FreeBSD .
- FreeBSD 4.x avec `LinuxThreads` . Notes relatives à FreeBSD .
- HP−UX 10.20 avec les threads DCE ou avec le paquet `MIT-pthreads` . Notes sur HP−UX Version 10.20 .
- HP−UX 11.x avec les threads natifs Notes sur HP−UX Version 11.x .
- Linux 2.0+ avec `LinuxThreads` 0.7.1+ ou `glibc` 2.0.7+. Notes relatives à Linux (toutes les versions de Linux) .
- Mac OS X. Notes relatives à Mac OS X .
- NetBSD 1.3/1.4 Intel et NetBSD 1.3 Alpha (requiert GNU make). Notes relatives à NetBSD .
- Novell NetWare 6.0. Installer MySQL sous NetWare .
- OpenBSD > 2.5 avec les threads natifs. OpenBSD < 2.5 avec le paquet `MIT-pthreads` . Notes relatives à la version 2.5 de OpenBSD .
- OS/2 Warp 3, FixPack 29 et OS/2 Warp 4, FixPack 4. Notes relatives à OS/2 .
- SCO OpenServer avec un port récent du paquet FSU Pthreads. Notes sur SCO .
- SCO UnixWare 7.1.x. Notes sur SCO UnixWare Version 7.1.x .
- SGI Irix 6.x avec les threads natifs. Notes sur SGI Irix .
- Solaris 2.5 et plus récent, avec les threads natifs sur SPARC et x86. Notes relatives à Solaris .
- SunOS 4.x avec le paquet `MIT-pthreads` . Notes relatives à Solaris .
- Tru64 Unix
- Windows 9x, Me, NT, 2000 et XP. Installer MySQL sous Windows .

Notez que toutes les plates−formes ne sont pas équipées de la même façon pour faire fonctionner MySQL. Les capacités d'une plate−forme pour supporter de fortes charges avec MySQL est déterminé par ceci :

- Stabilité générale de la bibliothèque de threads. Une plate−forme qui a une excellente réputation en général, mais une bibliothèque de threads instable, dont le code est utilisé par MySQL, même si le reste est parfait, fera de MySQL une application instable.
- La capacité du noyau et/ou de la bibliothèque de threads de profiter des capacités multi−processeurs, symétrique ou pas. En d'autres termes, lorsqu'un processus crée un thread, il doit être possible pour ce thread de s'exécuter sur différents processeurs.
- La capacité du noyau et/ou de la bibliothèque de threads de faire fonctionner de nombreux threads qui posent et lèvent des verrous mutex en peut de temps, fréquemment, sans changement de contexte excessif. En d'autres termes, si l'implémentation de `pthread_mutex_lock()` est trop soucieux du temps CPU, cela va ralentir sérieusement MySQL. Si ce problème n'est pas réglé, ajouter des processeurs supplémentaires va finalement ralentir MySQL.
- Performance et stabilité générale du système de fichiers.
- La capacité du système d'exploitation de gérer de grands fichiers, et de le faire efficacement, si vos tables sont grandes.
- Notre niveau d'expertise avec la plate−forme, chez MySQL AB. Si vous connaissons bien

une plate−forme, vous pourront introduire des optimisations et des corrections spécifiques à la plate−forme, et activé lors de la compilation. Nous pouvons aussi fournir des conseils judicieux pour configurer votre système optimalement pour MySQL.

- Le temps de tests que vous avons consacré à des configurations similaires, en interne.
- Le nombre d'utilisateur de MySQL qui font fonctionner MySQL avec succès sur cette plate−forme, avec des configurations similaires. Si ce nombre est grand, les chances de rencontrer un problème spécifique sont faibles.

En se basant sur les critères précédents, les meilleures plates−formes pour MySQL sont x86 avec SuSE Linux 7.1, noyau 2.4, et ReiserFS (ou toute autre distribution Linux similaire) et SPARC avec Solaris 2.7 ou 2.8. FreeBSD vient en troisième, mais nous espérons bien le voir rejoindre le groupe de tête, une fois que la bibliothèque de threads sera améliorée. Nous espérons aussi être bientôt capables d'ajouter les autres plates−formes sur laquelle MySQL compile, et fonctionne correctement, mais pas toujours le bon niveau de stabilité et de performances. Cela réclame des efforts de notre part, en coopération avec les développeurs de ces plates−formes. Si vous êtes intéressés par l'amélioration de ces composants, et que vous êtes en position pour influencer le développement, demandez des instructions détaillées à MySQL en envoyant un email aux listes internes. Les listes de diffusion MySQL .

Notez bien que la comparaison précédente ne signifie pas qu'un système d'exploitation est meilleur que l'autre, en général. Nous avons classé les systèmes en fonction de leur capacité à faire fonctionner un système MySQL, et nous nous limitons à cette comparaison. Avec cela en tête, le résultat de cette comparaison serait différent si nous y ajoutions d'autres problèmes. Et dans certains cas, la seule raison qui fait qu'un OS est meilleur que l'autre est parce que nous y avons consacré plus de temps, pour optimiser et tester. Nous nous bornons à exprimer notre point de vue pour vous aider à décider quelle plate forme choisir pour votre serveur MySQL.

2.1.2 Choisir votre version de MySQL

Lorsque vous vous préparez à installer MySQL, vous devez décider quelle version utiliser. Le développement de MySQL procède par série de versions, et vous pouvez prendre celle qui vous convient le mieux. Après avoir décidé de la version à installer, vous pouvez choisir le format de distribution : elles sont compilées ou au format source.

2.1.2.1 Quelle version de MySQL utiliser ?

La première décision à prendre est de savoir si vous voulez utiliser la dernière version de développement ou la dernière version stable :

- MySQL 5.0 est la nouvelle version de développement, et les nouvelles fonctionnalités sont activement développées. Jusque récemment, elle n'était disponible qu'en avant−première, sous BitKeeper. Une version alpha a été publiée depuis, pour permettre la diffusion large de la version, à des fins de tests.
- MySQL 4.1 est la version de développement, qui propose de nouvelles fonctionnalités majeures. Elle est toujours en version alpha. Les sources et le binaire sont disponibles pour tests et développement.
- MySQL 4.0 est la version stable courante, pour la production. Les nouvelles versions publiées sont des corrections de bogues. Aucune nouvelle fonctionnalité ne sera ajoutée, pour ne pas diminuer la stabilité du code.
- MySQL 3.23 est l'ancienne version de production. Cette série est retirée, et les nouvelles versions ne feront que corriger les bogues critiques.

Nous ne croyons pas au gel complet d'une version, et cela nous laisse de la place pour les corrections de bogues et les fonctionnalités qui ``doivent être faites.'' ``Un peu gelé'' signifie que nous pourrions ajouter de petites touches, qui ``n'affecterons pas ce qui fonctionne déjà, presque sûrement.'' Naturellement, les corrections de bogues des séries précédentes se propage aux nouvelles versions.

En règle générale, si vous utilisez MySQL pour la première fois ou si vous essayer de le porter vers un système pour lequel il n'existe pas de distribution binaire, nous vous recommandons d'utiliser la dernière version stable (actuellement la version 4.0). Notez que toutes les versions de MySQL sont passées aux bancs de test MySQL avant chaque sortie (même les versions de développement).

D'autre part, si vous utilisez un vieux système et que vous voulez procéder à une mise à jour, sans pour autant risquer de mettre à jour sans raison, vous devriez mettre à jour vers la dernière version de la même branche que celle que vous êtes en train d'utiliser (dans le cas où un numéro de version supérieur existe). Nous avons essayé de résoudre uniquement les bogues fatals et de produire des correctifs petits et sûrs pour cette version.

Si vous voulez utiliser de nouvelles versions qui ne sont pas présentes dans la version de production, vous pouvez utiliser la version de développement. Notez que les versions de développement ne sont pas aussi stables que les versions de production.

Si vous voulez utiliser les toutes dernières sources, qui contiennent tous les patches courants, et les corrections de bogues, vous pouvez utiliser notre entrepôt BitKeeper. Il n'y a pas de ``versions'' en tant que telle, mais des paquets, sur lesquels le code futur est basé.

La politique de nommage de MySQL utilise des numéros de version qui consiste en trois nombres suivis d'un suffixe. Par exemple, une version nommée `mysql-3.21.17-beta` doit être interprétée de la façon suivante :

- Le premier nombre (`3`) décrit le format de fichier. Toutes les versions 3 ont le même format de fichier.
- Le second nombre (`21`) correspond au niveau de version. Normalement, il y a le choix entre deux d'entre eux. L'un correspond à la version/branche stable (actuellement `23`) et l'autre se réfère à la branche de développement (actuellement `4.0`). Normalement, les deux versions sont stables, mais la version de développement peut comporter des lacunes, manquer de documentation sur des nouvelles fonctionnalités, ou peut ne pas compiler sur certains systèmes.
- Le troisième nombre (`17`) est le numéro de version au sein du niveau de version. Celui−ci est incrémenté à chaque nouvelle publication. En temps normal, vous souhaiterez utiliser la dernière version du niveau de version que vous avez choisi.

Pour chaque modification mineure, le dernier nombre de la version est incrémenté. Lorsque les nouvelles fonctionnalités sont majeures, ou que des incompatibilités mineures apparaissent avec les anciennes versions, le deuxième chiffre est incrémenté. Lorsque le format de fichier change, le premier chiffre est incrémenté.

Les noms de versions inclut aussi un suffixe qui indique le niveau de stabilité de la version. Une série progresse avec différents suffixes, qui indique sa stabilité. Les suffixes possibles sont :

- `alpha` indique que la publication contient de grandes portions de nouveau code qui n'a pas été testé à 100%. Les bogues connus (d'ordinaire, il n'y en a aucun) doivent être documentés dans la section nouveautés. Historique de MySQL . Il existe aussi de nouvelles commandes et extensions dans la plupart des versions alpha. Du développement actif qui inclut des changements majeurs dans le code peut concerner les versions alpha, mais tout sera testé avant de faire une publication. Il ne devrait pas y avoir de bogues connus dans les publications de MySQL.

- `beta` signifie que tout le nouveau code a été testé. Aucune fonctionnalité majeure qui pourrait causer corruption du code n'est ajoutée. Il ne doit pas y avoir un seul bogue connu. Une version alpha passe en `beta` quand il n'y a pas eu de bogue fatal rapporté depuis au moins un mois et que nous ne prévoyons pas de nouvelle fonctionnalité qui pourrait corrompre d'anciennes commandes.
- `gamma` est une version bêta qui existe depuis un certain temps et qui semble fonctionner correctement. Seulement des changements mineurs sont effectués. C'est ce que de nombreuses autres compagnies appellent une publication.
- S'il n'y a pas de suffixe, cela signifie que la version fonctionne depuis un certain temps sur différents sites avec aucun rapport de bogue autre que des bogues spécifiques à une plate−forme. Seuls des corrections critiques sont appliquées à la publication. C'est ce que l'on appelle une version stable.

MySQL utilise un schéma de nommage qui est légèrement différent des autres produits. En général, il est plutôt sûr d'utiliser une des versions qui est disponible depuis quelques semaines, sans avoir été remplacée par une nouvelle version de la même série.

Toutes les versions de MySQL passent par nos tests et bancs d'essais standards pour nous assurer qu'elles peuvent être utilisées sans danger. Les séries de tests s'améliorent en permanence car les tests standards sont étendus dans le temps pour traquer tous les bogues précédemment trouvées.

Notez bien que toutes les versions de MySQL ont été testées au moins avec :

Une batterie de tests internes
: Le dossier `mysql-test` contient de nombreux cas de tests. Nous utilisons ces tests pour virtuellement tous les systèmes d'exploitation. Voyez Suites de tests MySQL pour plus d'informations sur ces fichiers.

Les bancs de tests MySQL
: Ils effectuent une série de requêtes communes. C'est aussi un test pour savoir si le dernier processus d'optimisation rend le code plus rapide. La suite de tests comparatifs de MySQL .

Le test `crash-me`
: Il tente de déterminer de quelles fonctionnalités disposent les bases de données et quelles en sont les limites. La suite de tests comparatifs de MySQL .

Un autre test provient du fait que nous avons la version la plus récente de MySQL dans notre propre environnement de production interne, sur au moins une machine. Nous avons plus de 100 Go de données à manipuler.

2.1.2.2 Choisir le format de distribution

Après avoir choisi votre version de MySQL, il faut décider si vous voulez utiliser les versions binaires ou source. Dans la plupart des cas, vous choisirez une version binaire, si elle existe pour votre plate−forme. Les distributions binaires sont disponibles en format natif pour de nombreuses plates−formes, comme les paquets `RPM` de Linux ou les paquets `DMG` pour Mac OS X. Les distributions ont aussi disponibles sous formes d'archives Zip ou `tar` compressées.

Les raisons de choisir une distribution binaires sont :

- Les distributions binaires sont généralement plus faciles à installer que les distributions source.

* Pour satisfaire différents niveaux de besoin, nous fournissons deux versions binaires : une version compilée avec des moteurs de stockage non−transactionnels (petits et rapides), et une version compilée avec les extensions les plus importantes, comme les transactions. Les deux versions sont compilées à partir des mêmes sources. Tous les clients natifs MySQL peuvent se connecter aux serveurs MySQL, quelque soit leur version.La distribution MySQL maximale est suffixée avec `-max` et est configurée avec les mêmes options que `mysqld-max` . `mysqld-max` , la version étendue du serveur `mysqld` .

  Si vous voulez installer le `RPM MySQL-Max` , vous devez commencer par installer le `RPM MySQL-server` .

Dans certaines circonstances, il est préférable d'installer MySQL à partir de la distribution source :

* Vous voulez installer MySQL dans un dossier spécial. Les distributions standards sont ``prêtes à exécuter'' depuis n'importe quel dossier, mais vous voudrez peut être avoir plus de libertés pour dispatcher les composants de MySQL.
* Vous voulez configurer `mysqld` avec certaines extensions qui ne font pas parties des distributions binaires. Voici les extensions les plus courantes, que vous souhaiterez utiliser :
  * `--with-innodb` (par défaut pour MySQL 4.0 et plus récent)
  * `--with-berkeley-db` (disponible pour quelques plates−formes)
  * `--with-raid`
  * `--with-libwrap`
  * `--with-named-z-libs` (disponible pour certains binaires)
  * `--with-debug[=full]`
* Vous devez configurez `mysqld` sans certaines fonctionnalités qui font partie de la configuration standard. Par exemple, les distributions sont normalement compilées avec le support de tous les jeux de caractères. Si vous voulez rendre le serveur MySQL plus compact, vous pouvez recompiler MySQL avec uniquement les jeux de caractères dont vous avez besoin.
* Si vous avez un compilateur spécial, comme `pgcc` , ou que vous voulez utiliser des options particulières de votre compilateur pour optimiser MySQL pour votre architecture. Les distributions binaires sont compilées avec les options qui doivent fonctionner sur une large gamme de processeurs.
* Vous voulez utiliser les toutes dernières versions de MySQL, issues du serveur BitKeeper, pour avoir accès à toutes les corrections de bugs archivées. Par exemple, si vous avez découvert un bug, le correctif sera archivé dans le serveur de sources, et vous pourrez y accéder là. Le correctif n'apparaîtra pas avant la prochaine publication de MySQL.
* Vous voulez lire et modifier le code C et C++ de MySQL. Pour cela, obtenez une distribution source, car le code source est toujours le code ultime.
* Les distributions sources contiennent plus de tests et d'exemples que les distributions binaires.

2.1.2.3 Quand et comment sont publiées les nouvelles versions

MySQL évolue rapidement ici, à MySQL AB, et nous voulons le partager avec les autres utilisateurs de MySQL. Nous essayons de faire une nouvelle version à chaque fois que nous avons implanté des fonctionnalités qui seront utiles à d'autres.

Nous essayons aussi d'aider les utilisateurs dont les requêtes sont faciles à programmer. Nous prenons en considération tout ce que nos clients nous demandent, et nous accordons une attention particulière à nos clients qui ont pris une licence e−mail étendue.

Personne n'est obligé de télécharger une nouvelle version. La sections News vous indiquera si la nouvelle version contient une fonctionnalité que vous attendez. Historique de MySQL .

Nous utilisons la politique suivante, lors de la mise à jour de MySQL :

* Pour chaque modification mineure, le dernier numéro de la chaîne de version est incrémenté. Lorsqu'il y a des nouvelles fonctionnalités importantes ou des incompatibilité mineures avec la version précédente, nous incrémentons le chiffre du milieu. Lorsque le format de fichier change, le premier numéro est incrémenté.
* Des versions tables et testées sont publiées une à deux fois dans l'année, mais si de petits bogues apparaissent, une version qui ne va corriger que ces bogues sera publiée.
* Des versions fonctionnelles avec des corrections de bogues pour les vieilles versions sont publiées toutes les 1 à 8 semaines.
* Les distributions binaires de certaines plates−formes seront compilées par nos soins pour les versions majeures. D'autres personnes font des versions binaires pour d'autres systèmes, mais probablement moins fréquemment.
* Nous rendons généralement public les correctifs, une fois que nous avons découverts de petits bogues. Ils sont postés sur le serveur de versions BitKeeper et seront ajoutés à la prochaine version.
* Si il y a un bogue fatal dans une version, nous publierons une nouvelle version aussitôt que possible. Nous apprécions que les autres éditeurs fasse la même chose.

2.1.2.4 Politique de publication : aucun bug connu dans les versions

Nous mettons beaucoup d'efforts et de temps à la publication de version sans bugs. A notre connaissance, nous n'avons jamais publié une version de MySQL qui contienne un bug fatal **connu** et reproductible.

Un bug fatal est un problème qui fait planter MySQL en utilisation normale, fournit des réponses erronées à des requêtes classiques, ou a des problèmes de sécurité.

Nous documentons tous les problèmes ouverts, bugs et tout ce qui dépend des choix de conceptions. Erreurs connues et manques de MySQL .

Nous avons pour but de corriger tout ce qui peut être corrigé, sans risquer la stabilité des versions de MySQL. Dans certains cas, cela signifie que nous pouvons corriger une erreur dans la version de développement, mais pas dans la version stable. Naturellement, nous documentons ces problèmes, pour que les utilisateurs soient avertis.

Voici une description de notre processus de publication :

* Nous surveillons les bugs sur les listes de support utilisateur, les listes externes et la base de données de bugs sur le site http://bugs.mysql.com/ .
* Tous les bugs rapportés pour les versions en production entrent dans la base de données des bugs.
* Lorsque nous corrigeons un bug, nous essayons de réaliser un cas de test, que nous incluons dans notre système de tests, pour nous assurer que le bugs ne reviendra jamais (environ 90% des bugs ont des cas de tests).
* Nous créons aussi des cas de tests pour toutes les nouvelles fonctionnalités que nous voulons ajouter à MySQL.
* Avant de commencer à compiler une nouvelle version de MySQL, nous nous assurons que tous les bugs reproductibles pour les versions anciennes (3.23.x, 4.0.x, etc...) sont corrigés. Si le bug ne peut être corrigé (pour des raisons de choix de conception), nous le documentons dans le manuel. Erreurs connues et manques de MySQL .
* Nous compilons MySQL sur toutes les plates−formes pour lesquelles nous distribuons des paquets binaires (plus de 15 plates−formes à ce jour), et nous exécutons notre suite de tests et notre suite de performances sur chacune d'entre elles.
* Nous ne publions pas un paquet binaire sur une plate−forme pour laquelle la suite de test ou

de performances échoue. Si c'est une erreur générale, nous la corrigeons, et nous recommençons les tests sur toutes les plates−formes, à partir de zéro.
- Si nous recevons, durant le temps de compilation et de tests qui peut prendre deux à trois jours, un rapport de bugs concernant un bug fatal (par exemple, un bogue qui crée un coredump), nous le corrigeons, et nous recommençons le processus de tests.
- Après avoir publié les paquets binaires sur http://www.mysql.com/ , nous envoyons une annonce par email aux listes de diffusion. Les listes de diffusion MySQL . Le message d'annonce contient une liste de toutes les changements de la version, et de tous les bugs connus. La section des problèmes connus, '`known problems`', dans les notes de publications n'a été utilisé que dans quelques versions.
- Pour donner rapidement accès aux dernières fonctionnalités de MySQL, nous réalisons une publication de MySQL toutes les 4 à 5 semaines. http://downloads.mysql.com/snapshots.php .
- Si, après une publication, nous recevons des rapports de bugs qui prouvent qu'il y a, malgré tout, un bug critique dans une version spécifique à une plate−forme, nous le corrigeons rapidement, et nous annonçons une version '`a`' pour la plate−forme. Grâce à notre large communauté d'utilisateurs, les problèmes sont trouvés rapidement.
- Nos résultats de bonne publication sont excellents. Dans les dernières 150 versions, nous avons dû reprendre la compilation moins de 10 fois (dans trois des cas, le bug était dû à `glibc` sur l'une de nos machines de tests, qu'il a été difficile de trouver.

2.1.2.5 Binaires compilés par MySQL AB

MySQL AB, propose un jeu de distributions binaires de MySQL qui sont compilés sur nos machines, ou les machines auxquelles nos clients nous ont gracieusement donné accès.

En plus des versions binaires adaptées à chaque plate−forme, (Installation rapide et standard de MySQL .), nous proposons aussi des distributions binaires au format `.tar.gz` .

Pour les distributions Windows, voyez Installer MySQL sous Windows .

Ces distributions sont générées avec le script `Build-tools/Do-compile` qui compile le code source, et crée l'archive `tar.gz` en utilisant le script `scripts/make_binary_distribution` .

Ces archives sont configurés et compilées avec les options suivantes. Cette information peut aussi être obtenue en lisant les variables `COMP_ENV_INFO` et `CONFIGURE_LINE` dans le script `bin/mysqlbug` , disponible dans toutes les distributions binaires `tar` .

Les programmes suivants ont été compilés par les équipes de MySQL AB :

Linux 2.4.xx x86 avec `gcc` *2.95.3 :*

```
CFLAGS="-O2 -mcpu=pentiumpro" CXX=gcc CXXFLAGS="-O2 -mcpu=pentiumpro -felide-constructors" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--enable-assembler --disable-shared --with-client-ldflags=-all-static --with-mysqld-ldflags=-all-static
```

Linux 2.4.xx Intel Itanium 2 avec `ecc` *(Intel C++ Itanium Compiler 7.0) :*

```
CC=ecc CFLAGS="-O2 -tpp2 -ip -nolib_inline" CXX=ecc CXXFLAGS="-O2 -tpp2 -ip -nolib_inline" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
```

Linux 2.4.xx Intel Itanium avec `ecc` *(Intel C++ Itanium Compiler 7.0) :*

```
CC=ecc CFLAGS=-tpp1 CXX=ecc CXXFLAGS=-tpp1 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile
```

Linux 2.4.xx alpha avec `ccc` *(Compaq C V6.2−505 / Compaq C++ V6.3−006) :*

```
CC=ccc CFLAGS="-fast -arch generic" CXX=cxx CXXFLAGS="-fast -arch generic -noexceptions -nortti" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-mysqld-ldflags=-non_shared --with-client-ldflags=-non_shared --disable-shared
```

Linux 2.x.xx ppc avec `gcc` *2.95.4 :*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared --with-embedded-server --with-innodb
```

Linux 2.4.xx s390 avec `gcc` *2.95.3 :*

```
CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors" ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile --disable-shared
--with-client-ldflags=-all-static --with-mysqld-ldflags=-all-static
```

Linux 2.4.xx x86_64 (AMD64) avec `gcc` 3.2.1 :

```
CXX=gcc ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client
--enable-local-infile --disable-shared
```

Sun Solaris 8 x86 avec `gcc` 3.2.3 :

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared --with-innodb
```

Sun Solaris 8 SPARC avec `gcc` 3.2 :

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=no
--with-named-curses-libs=-lcurses --disable-shared
```

Sun Solaris 8 SPARC 64−bit avec `gcc` 3.2 :

```
CC=gcc CFLAGS="-O3 -m64 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -m64 -fno-omit-frame-pointer
-felide-constructors -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=no
--with-named-curses-libs=-lcurses --disable-shared
```

Sun Solaris 9 SPARC avec `gcc` 2.95.3 :

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-curses-libs=-lcurses
--disable-shared
```

Sun Solaris 9 SPARC avec `cc-5.0` (Sun Forte 5.0) :

```
CC=cc-5.0 CXX=CC ASFLAGS="-xarch=v9" CFLAGS="-Xa -xstrconst -mt -D_FORTEC_ -xarch=v9" CXXFLAGS="-noex -mt
-D_FORTEC_ -xarch=v9" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=no
--enable-thread-safe-client --disable-shared
```

IBM AIX 4.3.2 ppc avec `gcc` 3.2.3 :

```
CFLAGS="-O2 -mcpu=powerpc -Wa,-many " CXX=gcc CXXFLAGS="-O2 -mcpu=powerpc -Wa,-many -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-named-z-libs=no --disable-shared
```

IBM AIX 4.3.3 ppc avec `xlC_r` (IBM Visual Age C/C++ 6.0) :

```
CC=xlc_r CFLAGS="-ma -O2 -qstrict -qoptimize=2 -qmaxmem=8192" CXX=xlC_r CXXFLAGS ="-ma -O2 -qstrict -qoptimize=2
-qmaxmem=8192" ./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-named-z-libs=no --disable-shared --with-innodb
```

IBM AIX 5.1.0 ppc avec `gcc` 3.3 :

```
CFLAGS="-O2 -mcpu=powerpc -Wa,-many" CXX=gcc CXXFLAGS="-O2 -mcpu=powerpc -Wa,-many -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--with-server-suffix="-pro" --enable-thread-safe-client --enable-local-infile --with-named-z-libs=no
--disable-shared
```

IBM AIX 5.2.0 ppc avec `xlC_r` (IBM Visual Age C/C++ 6.0) :

```
CC=xlc_r CFLAGS="-ma -O2 -qstrict -qoptimize=2 -qmaxmem=8192" CXX=xlC_r CXXFLAGS="-ma -O2 -qstrict -qoptimize=2
-qmaxmem=8192" ./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-named-z-libs=no --disable-shared --with-embedded-server --with-innodb
```

HP−UX 10.20 pa−risc1.1 avec `gcc` 3.1 :

```
CFLAGS="-DHPUX -I/opt/dce/include -O3 -fPIC" CXX=gcc CXXFLAGS="-DHPUX -I/opt/dce /include -felide-constructors
-fno-exceptions -fno-rtti -O3 -fPIC" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-pthread --with-named-thread-libs=-ldce
--with-lib-ccflags=-fPIC --disable-shared
```

HP−UX 11.00 pa−risc avec `aCC` (HP ANSI C++ B3910B A.03.50) :

```
CC=cc CXX=aCC CFLAGS=+DAportable CXXFLAGS=+DAportable ./configure --prefix=/usr/local/mysql
--localstatedir=/usr/local/mysql/data --libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared --with-embedded-server --with-innodb
```

HP−UX 11.11 pa−risc2.0 64bit avec `aCC` (HP ANSI C++ B3910B A.03.33) :

```
CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

HP−UX 11.11 pa−risc2.0 32bit avec `aCC` (HP ANSI C++ B3910B A.03.33) :

```
CC=cc CXX=aCC CFLAGS="+DAportable" CXXFLAGS="+DAportable" ./configure --prefix=/usr/local/mysql
--localstatedir=/usr/local/mysql/data --libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared --with-innodb
```

HP−UX 11.22 ia64 64bit avec `aCC` (HP aC++/ANSI C B3910B A.05.50) :

```
CC=cc CXX=aCC CFLAGS="+DD64 +DSitanium2" CXXFLAGS="+DD64 +DSitanium2" ./configure --prefix=/usr/local/mysql
--localstatedir=/usr/local/mysql/data --libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared --with-embedded-server --with-innodb
```

Apple Mac OS X 10.2 powerpc avec `gcc` 3.1 :

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

FreeBSD 4.7 i386 avec `gcc` 2.95.4 :

```
CFLAGS=-DHAVE_BROKEN_REALPATH ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --with-named-z-libs=not-used --disable-shared
```

FreeBSD 4.7 i386 avec `LinuxThreads` et `gcc` 2.95.4 :

```
CFLAGS="-DHAVE_BROKEN_REALPATH -D__USE_UNIX98 -D_REENTRANT -D_THREAD_SAFE
-I/usr/local/include/pthread/linuxthreads" CXXFLAGS="-DHAVE_BROKEN_REALPATH -D__USE_UNIX98 -D_REENTRANT
-D_THREAD_SAFE -I/usr/local/include/pthread/linuxthreads" ./configure --prefix=/usr/local/mysql
--localstatedir=/usr/local/mysql/data --libexecdir=/usr/local/mysql/bin --enable-thread-safe-client
--enable-local-infile --enable-assembler --with-named-thread-libs="-DHAVE_GLIBC2_STYLE_GETHOSTBYNAME_R
-D_THREAD_SAFE -I /usr/local/include/pthread/linuxthreads -L/usr/local/lib -llthread -llgcc_r" --disable-shared
--with-embedded-server --with-innodb
```

QNX Neutrino 6.2.1 i386 avec `gcc` *2.95.3qnx−nto 20010315 :*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

Les paquets binaires suivants sont compilés sur des systèmes que des tiers prêtent gracieusement à MySQL AB. Notez que ces paquets sont fournis gracieusement. Comme MySQL AB n'a pas le contrôle complet sur ces systèmes, nous ne pouvons proposer qu'un support limité.

SCO Unix 3.2v5.0.6 i386 avec `gcc` *2.95.3 :*

```
CFLAGS="-O3 -mpentium" LDFLAGS=-static CXX=gcc CXXFLAGS="-O3 -mpentium -felide-constructors" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-named-z-libs=no --enable-thread-safe-client --disable-shared
```

SCO OpenUnix 8.0.0 i386 avec `CC` *3.2 :*

```
CC=cc CFLAGS="-O" CXX=CC ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --with-named-z-libs=no --enable-thread-safe-client
--disable-shared
```

Compaq Tru64 OSF/1 V5.1 732 alpha avec `cc/cxx` *(Compaq C V6.3−029i / DIGITAL C++ V6.1−027) :*

```
CC="cc -pthread" CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all" CXX="cxx -pthread"
CXXFLAGS="-O4 -ansi_alias -fast -inline speed -speculate all -noexceptions -nortti" ./configure
--prefix=/usr/local/mysql --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--with-prefix=/usr/local/mysql --with-named-thread-libs="-lpthread -lmach -lexc -lc" --disable-shared
--with-mysqld-ldflags=-all-static
```

SGI Irix 6.5 IP32 avec `gcc` *3.0.1 :*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --disable-shared
```

FreeBSD/sparc64 5.0 avec `gcc` *3.2.1 :*

```
CFLAGS=-DHAVE_BROKEN_REALPATH ./configure --prefix=/usr/local/mysql --localstatedir=/usr/local/mysql/data
--libexecdir=/usr/local/mysql/bin --with-extra-charsets=complex --enable-thread-safe-client --enable-local-infile
--disable-shared --with-innodb
```

Les options suivantes de compilations ont été utilisées pour les paquets binaires de MySQL, qui étaient fournis auparavant. Ces paquets ne sont plus mis à jours, mais les options de compilation sont conservées ici pour mémoire.

Linux 2.2.xx SPARC avec `egcs` *1.1.2 :*

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
--enable-thread-safe-client --enable-local-infile --enable-assembler --disable-shared
```

Linux 2.2.x avec x686 avec `gcc` *2.95.2 :*

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro -felide-constructors -fno-exceptions -fno-rtti"
./configure --prefix=/usr/local/mysql --enable-assembler --with-mysqld-ldflags=-all-static --disable-shared
--with-extra-charsets=complex
```

SunOS 4.1.4 2 sun4c avec `gcc` *2.7.2.1 :*

```
CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors" ./configure --prefix=/usr/local/mysql --disable-shared
--with-extra-charsets=complex --enable-assembler
```

SunOS 5.5.1 (et plus récents) sun4u avec `egcs` *1.0.3a or 2.90.27 or* `gcc` *2.95.2 et plus récents :*

```
CC=gcc CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" ./configure
--prefix=/usr/local/mysql --with-low-memory --with-extra-charsets=complex --enable-assembler
```

SunOS 5.6 i86pc avec `gcc` *2.8.1 :*

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-low-memory --with-extra-charsets=complex
```

BSDI BSD/OS 3.1 i386 avec `gcc` *2.7.2.1 :*

```
CC=gcc CXX=gcc CXXFLAGS=-O ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
```

BSDI BSD/OS 2.1 i386 avec `gcc` *2.7.2 :*

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
```

AIX 2 4 avec `gcc` *2.7.2.2 :*

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-extra-charsets=complex
```

Si vous avez des options plus optimales pour l'une des configurations précédemment listées, vous pouvez toujours nous en faire part sur la liste de distribution des développeurs. Les listes de diffusion MySQL .

Les distributions `RPM` antérieures à la version 3.22 de MySQL sont fournies par nos utilisateurs. A partir de la version 3.22, les `RPM` sont générés par nous chez MySQL AB.

Si vous voulez compiler une version de débogage de MySQL, vous devez ajouter `--with-debug` ou `--with-debug=full` aux lignes de configurations précédentes et effacer les options `-fomit-frame-pointer` .

2.1.3 Comment obtenir MySQL ?

Visitez le site de MySQL (http://www.mysql.com/) pour des informations à propos de la version courante et les instructions de téléchargement.

Notre miroir principal est situé sur http://mirrors.sunsite.dk/mysql/ .

Pour une liste complète et à jour des miroirs web/téléchargement de MySQL, voyez http://www.mysql.com/downloads/mirrors.html . Vous trouverez là des informations à propos des futurs miroirs et de quoi nous informer de la non−validité de l'un d'entre eux.

2.1.4 Vérifier l'intégrité des paquets avec MD5 ou GnuPG

Une fois que vous avez téléchargé le paquet MySQL qui vous convient, et avant de l'installer, vous devriez vous assurer qu'il est intact, et n'a pas été altéré.

MySQL AB propose deux moyens de vérifier l'intégrité :

- Signatures MD5
- Signatures chiffrées avec `GnuPG` , `GNU Privacy Guard`
- Pour les paquets `RPM` , le mécanisme de vérification d'intégrité intégré.

Les sections suivantes décrivent comment utiliser ces méthodes.

Dans le cas où vous vous apercevez que la somme de contrôle `MD5 checksum` ou la signature `GPG` ne correspond pas, essayez de télécharger à nouveau le même paquet, éventuellement depuis un autre miroir. Si vous échouez plusieurs fois à vérifier l'intégrité du paquet, faites nous part de votre problème, en incluant le nom complet du paquet désiré, et les sites de téléchargement que vous avez utilisé. Envoyez nous un courriel à l'adresse webmaster@mysql.com ou build@mysql.com .

2.1.4.1 Vérifier la signature MD5

Une fois que vous avez téléchargé le paquet, vous devez vérifier si la somme de contrôle MD5 correspond à celle qui est disponibles sur le site de MySQL. Chaque paquet a une somme de contrôle individuelle, que vous pouvez obtenir avec la commande suivante :

```
shell> md5sum <paquet>
```

Exemple :

```
 shell> md5sum mysql-standard-4.0.17-pc-linux-i686.tar.gz
60f5fe969d61c8f82e4f7f62657e1f06
               mysql-standard-4.0.17-pc-linux-i686.tar.gz
```

Ainsi, vous devez vérifier si la somme de contrôle résultante correspond à celle qui est imprimée sur la page de téléchargement, en dessous du paquet téléchargé.

Notez que tous les systèmes d'exploitation ne supportent pas la commande `md5sum` : sur certains, elle s'appelle simplement `md5` , sur d'autre, elle n'est pas du tout disponible. Sur Linux, elle a fait partie des `utilitaires texte GNU (GNU Text Utilities)` , qui sont disponibles pour toute une gamme de plates−formes. Vous pouvez télécharger le code source sur le site http://www.gnu.org/software/textutils/ . Si vous avez installé `OpenSSL` , vous pouvez utiliser la commande `openssl md5 <paquet>` à la place. Une implémentation DOS/Windows de la commande `md5` est disponible sur le site http://www.fourmilab.ch/md5/ .

2.1.4.2 Vérification de la signature avec GnuPG

Une méthode plus sûre pour vérifier l'intégrité d'un paquet est d'utiliser la signature `GnuPG` . C'est une méthode plus sûre que le MD5, mais elle requiert un peu plus de travail.

Depuis MySQL 4.0.10 (Février 2003), MySQL AB utilise `GNU Privacy Guard` (`GnuPG`), une alternative `Open Source` du très connu `Pretty Good Privacy` (`PGP`) par Phil Zimmermann. Voir http://www.gnupg.org/ . La plupart des distributions Linux dispose d'une version de `GnuPG` installée par défaut. Pour plus de détails sur `OpenPGP` , voir http://www.openpgp.org/ .

Pour vérifier la signature d'un paquet spécifique, vous devez obtenir en premier lieu une copie de la clé publique GPG. Vous pouvez soit la copier/coller directement depuis ce manuel, ou la demander sur le serveur http://www.keyserver.net/ .

```
 Key ID:
pub  1024D/5072E1F5 2003-02-03
     MySQL Package signing key (www.mysql.com) <build&#64;mysql.com>
Fingerprint: A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5

 Public Key (ASCII-armored):

 -----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

 mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZ
fw2vOUgCmYv2hW0hyDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRCRxAuAuVztHRcEAJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNW
hxwR9pRWVArNYJdDRT+rf2RUe3vpquKNQU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7ITnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAnWqcyefep
rv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q7TXlTUUwgUGFj
a2FnZSBzaWduaW5nIGtleSAod3d3Lm15c3FsLmNvbSkgPGJ1aWxkQG15c3FsLmNv
bT6IXQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAAAoJEIxxjTtQ
cuH1cY4AnilUwTXn8MatQOiG0a/bPxrvK/gCAJ4oinSNZRYTnblChwFaazt7PF3q
zIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAn1mxHijft00bKXvu
cSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJ
YiKJAAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8J
Eg2aLos+5zEYrB/LsrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWsEN/l
xaZoJYc3a6M02WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLmRDRi
Rjd1DTCHqeyX7CHhcghj/dNRlW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hkAWzE
7zaD5cH9J7yv/6xuZVw411x0h4UqsTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fm
Le11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRsKTfozBu74F47D8Ilbf5vSYHbuE5p
/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnlLzKUb/F5GwADBQf+Lwqq
a8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPDsDD9MZ1ZaSaf
anFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGoTbOW
I39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42Lmu
QT5NdKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt92
6s/ymfdf5HkdQ1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZ
Whe70YGNPw1yjWJT1IhMBBgRAgAMBQI+PqMdBQkJZgGAAAoJEIxxjTtQcuH17p4A
n3r1QpVC9yhnW2cSAjq+kr72GX0eAJ4295kl6NxYEuFApmr1+0uUq/SlsQ==
=YJkx
-----END PGP PUBLIC KEY BLOCK-----
```

Vous pouvez importer cette clé dans votre trousseau de clés publiques `GPG` avec la commande `gpg --import` . Par exemple, si vous avez sauvé la clé dans un fichier appelé `mysql_pubkey.asc` , la commande d'importation est :

```
 shell> gpg --import mysql_pubkey.asc
```

Voyez la documentation `GPG` pour plus de détails sur comment travailler avec les clés publiques.Une fois que vous avez téléchargé et importé la clé publique, vous pouvez télécharger le paquet MySQL et la signature qui lui est associée, sur la même page. Le fichier de signature a pour extension `.asc` . Par exemple :

Fichier de distribution Linux	mysql-standard-4.0.17-pc-linux-i686.tar.gz
Fichier de signature	mysql-standard-4.0.17-pc-linux-i686.tar.gz.asc

Assurez−vous que les deux fichiers sont stockés dans le même dossier, puis exécutez la commande suivante pour vérifier la signature du fichier :

```
 shell> gpg --verify <package>.asc
```

Exemple :

```
 shell> gpg --verify mysql-standard-4.0.17-pc-linux-i686.tar.gz.asc
gpg: Warning: using insecure memory!
```

```
gpg: Signature made Mon 03 Feb 2003 08:50:39 PM MET
using DSA key ID 5072E1F5
gpg: Good signature from
     "MySQL Package signing key (www.mysql.com) <build&#64;mysql.com>"
```

La mention " `Good signature` " (bonne signature) indique que le paquet est correct.

2.1.4.3 Contrôle d'intégrité avec RPM

Pour les paquets `RPM` , il n'y a pas de signature séparée : les paquets `RPM` disposent d'une signature `GPG` intégrée, et d'une somme de contrôle `MD5` . Vous pouvez les vérifier avec la commande suivante :

```
shell> rpm --checksig package_name.rpm
```

Exemple :

```
 shell> rpm --checksig MySQL-server-4.0.10-0.i386.rpm
MySQL-server-4.0.10-0.i386.rpm: md5 gpg OK
```

Note : si vous utilisez `RPM` 4.1 et qu'il se plaint que `(GPG) NOT OK (MISSING KEYS: GPG#5072e1f5)` (même si vous l'avez importé dans votre trousseau de clé), vous devez alors importer la clé dans votre trousseau `RPM` d'abord. `RPM` 4.1 n'utilise pas votre trousseau de clé GPG (ni GPG lui−même), car il entretient son propre trousseau de clé (car c'est une application de niveau système, et que le trousseau de clé est spécifique à chaque utilisateur). Pour importer la clé publique `mysql_pubkey.asc` MySQL dans votre trousseau de clés `RPM` , utilisez la commande suivante :

```
shell> rpm --import mysql_pubkey.asc
```

2.1.5 Dispositions d'installation

Cette section décrit les répertoires par défaut crées en installant les distributions binaires et les distributions de sources.

Sous Windows, le dossier d'installation par défaut est `C:\mysql` , qui a la structure suivante :

Dossier	Contenu du dossier
bin	Clients et serveur mysqld
data	Fichiers de log et bases de données
Docs	Documentation
examples	Programmes d'exemple et scripts
include	Fichiers d'inclusion (entêtes)
lib	Bibliothèques
scripts	Utilitaires
share	Fichiers de messages d'erreur

Les installations créée sur les distributions Linux `RPM` placent les fichiers dans les sous−dossiers suivants :

Dossier	Contenu du dossier
/usr/bin	Programmes clients
/usr/sbin	serveur mysqld
/var/lib/mysql	Fichiers de log et bases de données
/usr/share/doc/packages	Documentation
include	Fichiers d'inclusion (entêtes)
lib	Bibliothèques
/usr/share/mysql	Fichiers de messages d'erreurs et jeux de caractères
sql-bench	Suites de tests

Sous Unix, une archive `tar` avec la distribution binaire s'installe en la désarchivant dans le dossier d'installation que vous voulez (typiquement le dossier `/usr/local/mysql`) et crée les dossiers suivants au même endroit :

Dossier	Contenu du dossier
bin	Clients et serveur mysqld
data	Fichiers de log et bases de données
docs	Documentation, historique
include	Fichiers d'inclusion (entêtes)
lib	Bibliothèques

scripts	mysql_install_db
share/mysql	Fichiers de messages d'erreur
sql−bench	Suites de tests

Une distribution source est installée après compilation. Par défaut, les étapes d'installation installent les fichiers dans `/usr/local` , dans les sous−dossiers suivants :

Dossier	Contenu du dossier
bin	Programmes clients et scripts
include/mysql	Fichiers d'inclusion (entêtes)
info	Documentation
lib/mysql	Bibliothèques
libexec	The serveur mysqld
share/mysql	Fichiers de messages d'erreur
sql−bench	Suites de tests
var	Fichiers de log et bases de données

Dans le répertoire d'installation, les dispositions d'une installation des sources diffère d'une installation binaire des façons suivantes :

- Le serveur `mysqld` est installé dans le répertoire `libexec` plutôt que dans le répertoire `bin` .
- Le répertoire des données est `var` plutôt que `data` .
- `mysql_install_db` est installé dans le répertoire `/usr/local/bin` plutôt que dans `/usr/local/mysql/scripts` .
- Le répertoire des fichier d'entête et les répertoires des bibliothèques sont `include/mysql` et `lib/mysql` au lieu de `include` et `lib` .

Vous pouvez créer votre propre installation binaire à partir d'une distribution de sources compilées en exécutant le script `scripts/make_binary_distribution` .

2.2 Installation standard rapide de MySQL

Cette section couvre l'installation de MySQL sur les plate−formes pour lesquelles nous offrons un système d'installation spécifique. Cela s'appelle aussi une installation binaire. Toutefois, des installation binaires sont disponibles pour de nombreux autres plate−formes. Voyez Installer MySQL sous les autres systèmes Unix pour des instructions génériques, qui s'appliqueront aussi à ces plate−formes.

Voyez Installation générale pour plus d'informations sur les autres distributions binaires qui sont disponibles, et comment les obtenir.

2.2.1 Installer MySQL sous Windows

Le processus d'installation de MySQL sous Windows est le suivant :

- Installez la distribution.
- Configurez un fichier d'options si nécessaire.
- Sélectionnez un serveur à utiliser.
- Lancez le serveur.
- Assignez des mots de passe aux comptes MySQL initiaux.

MySQL pour Windows est disponible en deux formats :

- La distribution binaire, qui contient un programme d'installation, qui se charge de placer tout ce qui est nécessaire.
- La distribution source, qui contient tout le code et les fichiers de support pour compiler les exécutables avec VC++ 6.0.

En général, vous devriez utiliser la distribution binaire. Elle est plus facile à installer, et vous n'avez besoin d'aucun autre outil pour faire fonctionner MySQL.

Cette section décrit comment installer MySQL sur Windows en utilisant la distribution binaire. Pour installer une distribution source, voyez La distribution sous forme de sources pour Windows .

2.2.1.1 Prérequis système sur Windows

Pour utiliser MySQL sur Windows, vous avez besoin de :

- Un système d'exploitation 32 bits Windows, tels que 9x, Me, NT, 2000 ou XP. La famille NT (`Windows NT/2000/XP`) vous permet de faire fonctionner MySQL comme un service. Lancer MySQL comme un service Windows .
- Support du protocole TCP/IP.
- Une copie de la distribution binaire MySQL pour Windows, qui peut être téléchargé sur http://www.mysql.com/downloads/ . Obtenir MySQL .

  Note : les fichiers de la distribution sont fournis dans un format compressé, et nous vous recommandons d'utiliser le client FTP approprié, avec des capacités de reprise sur erreur, pour éviter les corruptions d'archive au cours du téléchargement.
- `WinZip` ou un programme `ZIP` pour décompresser l'archive `.zip` .
- Assez d'espace sur votre disque pour décompresser l'archive, et pour créer les bases selon vos besoins.
- Si vous envisagez d'utiliser MySQL via ODBC, vous aurez aussi besoin du pilote `MyODBC` . Interface ODBC .
- Si vous avez besoin de table d'une taille supérieure à 4 Go, installez MySQL sur un système de fichier NTFS ou plus récent. N'oubliez pas d'utiliser les options `MAX_ROWS` et `AVG_ROW_LENGTH` lorsque vous créez les tables. Syntaxe de `CREATE TABLE` .

2.2.1.2 Installation d'une distribution binaire sur Windows

Pour installer MySQL sur Windows, avec une distribution binaire, suivez cette procédure :

- Si vous travaillez avec les serveurs `Windows NT/2000/XP` , identifiez−vous en tant qu'administrateur.
- Si vous faites une mise à jour d'une version plus récente de MySQL, il est nécessaire d'arrêter le serveur. Si vous utilisez le serveur en tant que service, utilisez :
  ```
  C:\> NET STOP MySQL
  ```
  Si vous prévoyez d'utiliser un autre serveur après la mise à jour, par exemple si vous envisagez d'utiliser `mysqld-max` plutôt que `mysqld`), supprimez le service existant :
  ```
  C:\mysql\bin> mysqld --remove
  ```
  Vous pouvez réinstaller le service après la mise à jour.Si vous n'utilisez pas MySQL comme service, arrêtez le comme ceci :
  ```
  C:\mysql\bin> mysqladmin -u root shutdown
  ```
- Quittez le programme `WinMySQLAdmin` s'il fonctionnait.
- Décompressez les fichiers dans un emplacement temporaire.
- Exécutez le fichier `setup.exe` pour commencer l'installation. Si vous voulez l'installer à un autre emplacement que celui pas défaut `c:\mysql` , utilisez le bouton `Parcourir` pour spécifier un emplacement. Si vous ne voulez pas installer mySQL dans le dossier par défaut, vous devez spécifier son dossier à chaque démarrage du serveur. Le plus facile est de le configurer dans un fichier d'options, tel que décrit dans Préparer l'environnement MySQL sous Windows .
- Terminez le processus d'installation.

Note importante : Actuellement, les distribution Windows de MySQL 4.1 ne contiennent pas de programme d'installation. La distribution version 4.1 est un fichier `ZIP` que vous devez simplement décompresser dans le dossier d'installation de MySQL. Par exemple, pour installer `mysql-4.1.1-alpha-win.zip` dans `C:\mysql` , décompressez le fichier de distribution dans le disque `C:` , puis renommez le dossier résultant `mysql-4.1.1-alpha` en `mysql` .

Si vous mettez à jour vers MySQL version 4.1, depuis une version plus ancienne, vous souhaiterez préserver votre dossier de données `data` qui contient les tables de droits dans la base `mysql` ainsi que vos propres données. Avant d'installer 4.1, stoppez le serveur, et sauvez votre dossier `data` à un autre endroit. Puis, renommez le dossier `C:\mysql` ou supprimez le. Installez la version 4.1 telle que décrit dans le paragraphe précédent, et remplacez le nouveau dossier `data` par votre vieux dossier `data` . Lancez le nouveau serveur, et modifiez les tables de droits. Cela vous évitera la perte de votre base actuelle. Mise à jour des tables de droits .

2.2.1.3 Préparation de l'environnement MySQL de Windows

Si vous devez spécifier les options de démarrage lors du lancement de votre serveur, vous pouvez les indiquer en ligne de commande ou bien les placer dans un fichier d'options. Pour les options qui seront utilisées à chaque démarrage, il est plus pratique de les mettre dans le fichier de configuration MySQL. C'est particulièrement vrai dans les circonstances suivantes :

- Le répertoire d'installation ou de données est différent de ceux par défaut (`c:\mysql` et `c:\mysql\data`).
- Vous devez paramétrer la configuration du serveur. Par exemple, pour utiliser les tables transactionnelles `InnoDB` en MySQL version 3.23, vous devez manuellement créer deux nouveaux dossiers pour stocker les données `InnoDB` , données et log. Vous devez aussi ajouter des lignes de configurations supplémentaires dans le fichier d'options, comme décrit dans la section Options de démarrage `InnoDB` . Depuis MySQL 4.0, `InnoDB` crée un fichier de données et un fichier de log dans le dossier de données par défaut. Cela signifie que vous n'avez plus besoin de configurer explicitement `InnoDB` . Vous pouvez toujours le faire si vous le souhaitez, et un fichier d'options sera alors utile.

Sous Windows, l'installeur MySQL place le dossier de données dans le dossier d'installation de MySQL. Si vous souhaitez utiliser un autre dossier de données, dans une autre partie du disque, vous devez copier le contenu du dossier `data` dans le nouveau dossier. Par exemple, par défaut, l'installeur MySQL place les fichiers MySQL dans le dossier `C:\mysql` et le dossier de données dans le dossier `C:\mysql\data` . Si vous voulez utiliser le dossier de données `E:\mydata` , vous devez faire deux choses :

- Déplacer le dossier de données de `C:\mysql\data` à `E:\mydata` .
- Utiliser l'option de démarrage `--datadir` pour spécifier le nouveau dossier de données, à chaque fois que vous voulez lancer le serveur.

Lorsque le serveur MySQL démarre sur Windows, il recherche les options dans deux fichiers : `my.ini` dans le dossier Windows, et le fichier `C:\my.cnf` . Le dossier Windows est généralement `C:\WINDOWS` ou `C:\WinNT` . Vous pouvez déterminer sa localisation exacte à partir de la valeur de la variable d'environnement `WINDIR` :

```
C:\> echo %WINDIR%
```

MySQL recherche les options d'abord dans le fichier `my.ini` , puis dans le fichier `my.cnf` . Cependant, pour éviter les confusions, il est bon de n'utiliser qu'un seul des deux fichiers. Si votre PC utilise un fichier de boot où le lecteur `C:` n'est pas le disque de boot, votre seule option est d'utiliser le fichier `my.ini` . Quelque soit celui que vous utilisez, ce doit être un fichier de texte.Un fichier d'options peut être créé simplement et modifié avec un éditeur de texte quelconque, comme `Notepad` . Par exemple, si MySQL est installé dans le dossier `D:\mysql` et que le dossier de données est situé dans `D:\mydata\data` , vous pouvez créer un fichier d'options et configurer une section `[mysqld]` pour spécifier les valeurs de `basedir` et `datadir` :

```
 [mysqld]
# set basedir to installation path, e.g., c:/mysql
basedir=le_dossier_d'installation
# set datadir to location of data directory,
# e.g., c:/mysql/data or d:/mydata/data
datadir=le_dossier_des_données
```

Notez que les chemins de Windows doivent être spécifiés dans les fichiers d'options en utilisant des slash normaux (`/`) au lieu des anti−slash (`\`). Si vous utilisez des anti−slash, vous devez les doubler.Un autre moyen de gérer un fichier d'options est d'utiliser l'utilitaire `WinMySQLAdmin` . Vous pouvez trouver le logiciel `WinMySQLAdmin` dans le dossier `bin` de votre installation MySQL, ainsi qu'un fichier d'aide contenant les instructions pour l'utiliser. `WinMySQLAdmin` est capable d'éditer votre fichier d'objets, mais notez ceci :

- `WinMySQLAdmin` utilise uniquement le fichier `my.ini` .
- Si `WinMySQLAdmin` trouve le fichier `C:\my.cnf` , il va le renommer en `C:\my_cnf.bak` pour le désactiver.

Maintenant, vous êtes prêts à tester le serveur.

2.2.1.4 Choisir un serveur Windows

Depuis MySQL 3.23.38, la distribution Windows inclut le serveur MySQL classique et le serveur `MySQL−Max` . Voici la description de toutes les versions livrées :

Programme	Description
mysqld	Compilé avec les informations de débogage, la vérification d'allocation de mémoire, le support des liens symboliques et les tables InnoDB et BDB tables.
mysqld-opt	Optimisé. Depuis la version 4.0, InnoDB est activé. Avant la version 4.0, ce serveur incluait aucune tables transactionnelles.
mysqld-nt	Optimisé pour Windows NT/2000/XP avec support pour les pipes nommés.
mysqld-max	Optimisé, avec le support des liens symboliques et des tables InnoDB et BDB .
mysqld-max-nt	Comme mysqld-max , mais compilé avec le support des pipes nommés.

Tous les serveurs précédents sont optimisés pour les processeurs Intel modernes, mais ils fonctionneront sur toutes les architectures Intel de classe i386 et plus récent.

MySQL supporte TCP/IP sur toutes les plates−formes Windows. Les serveurs `mysqld−nt` et `mysql−max−nt` supportent les pipes nommés sur `NT` , 2000 et `XP` . Cependant, par défaut, MySQL utilise TCP/IP quelque soit la plate−forme. Les pipes nommés sont plus lents que TCP/IP.

Les pipes nommés sont sujets aux limitations suivantes :

- Depuis MySQL 3.23.50, les pipes nommés sont activés uniquement si vous lancez le serveur avec l'option `−−enable−named−pipe` . Il est nécessaire d'utiliser explicitement cette option, car des utilisateurs ont rencontré des problèmes lors de l'extinction du serveur MySQL, avec les pipes nommés.
- Les connexions par pipe nommés ne sont permises qu'avec `mysqld−nt` et `mysqld−max−nt` , et uniquement si le serveur fonctionne sur une version de Windows qui supporte les pipes nommés (NT, 2000, XP).
- Ces serveurs peuvent fonctionner avec Windows 98 et Me, mais ils requièrent la pile TCP/IP; le pipes nommés seront ignorés.
- Sur Windows 95, ces serveurs ne peuvent être utilisés.

Note: La plupart des exemples dans les prochaines sections, utilisent `mysqld` comme nom de serveur. Si vous choisissez un autre nom de serveur, comme `mysqld−opt` , assurez vous de bien faire les bons remplacements dans les commandes des exemples. Une bonne raison de choisir un nom de serveur différent est que `mysqld` contient le support complet du débogage, il utilise plus de mémoire et fonctionne plus lentement que les autres serveurs Windows.

2.2.1.5 Démarrer le serveur pour la première fois

Sur Windows 95, 98 et Me, les clients MySQL utilisent toujours TCP/IP pour se connecter au serveur. Cela permet à toute machine du réseau de se connecter à votre serveur MySQL. A cause de cela, assurez vous que le support TCP/IP est installé avant de lancer MYSQL. Vous pouvez trouver TCP/IP dans votre CD d'installation Windows.

Notez que si vous utilisez une vieille version Windows 95 (par exemple, OSR2), il est probable que vous ayez un vieux paquet Winsock; MySQL requiert Winsock 2! Vous pouvez télécharger un nouveau paquet Winsock sur http://www.microsoft.com/ . Windows 98 dispose de la bibliothèque Winsock 2, et il est donc inutile de la mettre à jour.

Sur les systèmes NT, comme Windows NT, 2000 ou XP, les clients ont deux options. Soit utiliser TCP/IP, soit les pipes nommés, si le serveur les supporte.

Pour des informations sur quel serveur choisir, voyez Préparer l'environnement MySQL sous Windows .

Cette section donne un aperçu général du lancement du serveur MySQL. La section suivante fournit des informations spécifiques pour une version particulière de Windows.

Les exemples de ces sections supposent que MySQL est installé dans son dossier par défaut : `C:\mysql` . Adaptez les noms de chemins si vous avez installé MySQL ailleurs.

Testez à partir d'une console DOS est la meilleure chose à faire car le serveur affiche des messages qui y apparaissent. Si quelque chose n'est pas bon dans votre configuration, ces messages vous aiderons à identifier et corriger le problème.

Assurez−vous d'être dans le répertoire où se situe le serveur, puis entrez cette commande :

```
C:\mysql\bin> mysqld --console
```

Vous devriez voir ce qui suit pendant le démarrage du serveur :

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started
```

Lorsque le serveur finit sa séquence de démarrage, vous devriez voir une ligne comme celle−ci, qui indique que le serveur est fonctionnel, et attend les connexions :

```
mysqld: ready for connections
Version: '4.0.14-log'  socket: ''  port: 3306
```

Le serveur va continuer à écrire dans la console les logs. Vous pouvez ouvrir une autre console pour utiliser un client.

Si vous omettez l'option `--console` , le serveur va écrire les logs dans un fichier du dossier de données

(`C:\mysql\data` par défaut). Le fichier d'erreurs est celui dont l'extension est `.err` .

Les comptes initiaux, qui sont dans les tables de droits de MySQL n'ont pas de mot de passe. Après le démarrage du serveur, vous devriez modifier leurs mots de passe avec les instructions de Configuration et tests consécutifs à l'installation .

Note : Les comptes sont listés dans les tables de droits MySQL qui n'ont pas de mot de passe initialement. Après avoir démarré le serveur, il est recommandé de les modifié, comme indiqué dans la documentation Configuration et tests consécutifs à l'installation .

2.2.1.6 Lancer MySQL en ligne de commande

Le serveur MySQL peut être lancé en ligne de commande, manuellement. Cela peut se faire sur toutes les versions de Windows.

Pour lancer le serveur `mysqld` depuis la ligne de commande, vous devez ouvrir une console (une fenêtre ``DOS'') et taper :

```
shell> C:\mysql\bin\mysqld
```

Sur les versions non−NT de Windows, cela va lancer le serveur `mysqld` en tâche de fond. C'est à dire, une fois que le serveur a démarrer, vous devez reprendre la main dans la fenêtre. Si vous lancez le serveur de cette manière sur Windows NT, 2000 ou XP, le serveur va fonctionne en tâche de fond et aucune ligne de commande ne réapparaîtra jusqu'à ce que le serveur ne s'arrête. A cause de cela, vous devrez ouvrir une autre fenêtre, pour lancer les clients, lorsque le serveur fonctionne.

Vous pouvez arrêter le serveur MySQL en exécutant cette commande :

```
shell> C:\mysql\bin\mysqladmin −u root shutdown
```

Cette commande appelle l'utilitaire d'administration `mysqladmin` , qui se connecte au serveur et lui dit de s'arrêter. La commande est exécutée en tant que `root` , qui est le compte d'administration par défaut dans le système de droits MySQL. Notez que les utilisateurs du système de droits MySQL sont totalement indépendant de ceux de Windows.

Si `mysqld` ne démarre pas, ouvrez le fichier de log pour voir si le serveur a noté un message qui vous indiquera la nature du problème. Le log du serveur est disponible dans le dossier `C:\mysql\data` . C'est le fichier avec un suffixe en `.err` . Vous pouvez aussi essayer de lancer le serveur avec la commande `mysqld --console` ; dans ce cas, vous pourrez voir des informations utiles dans la fenêtre de console, pour vous aider à résoudre le problème.

La dernière option est de lancer `mysqld` avec `--standalone --debug` . Dans ce cas, `mysqld` va écrire un fichier de trace, `C:\mysqld.trace` , qui devrait contenir les raisons des problèmes de `mysqld` . Créer des fichiers de traçage .

Utilisez `mysqld --help` pour afficher des informations sur les options que `mysqld` comprend!

2.2.1.7 Lancer MySQL comme un service Windows

Dans la famille NT (Windows NT, 2000 ou XP), la méthode recommandée pour faire fonctionner MySQL est de l'installer comme service Windows. Windows lance et arrête le serveur MySQL lorsque le système d'exploitation se lance ou s'arrête. Un serveur installé comme un service peut aussi être contrôlé en ligne de commande, avec la commande `NET` , ou avec l'utilitaire graphique

`Services` .

L'utilitaire `Services` (le gestionnaire Windows `Service Control Manager`) est disponible dans le panneau d'administration Windows (sous la section `Utilitaires d'administration` sous Windows 2000). Il est conseillé de fermer l'utilitaire `Services` lorsque vous faites une installation ou une suppression à partir de la ligne de commande : cela évite certaines erreurs étranges.

Pour faire fonctionner MySQL avec TCP/IP sous Windows NT 4, vous devez installer le service pack 3 ou plus récent.

Avant d'installer MySQL comme service Windows, vous devez commencer par arrêter le serveur en marche, avec cette commande :

```
C:\> C:\mysql\bin\mysqladmin -u root shutdown
```

Elle appelle l'utilitaire MySQL `mysqladmin` , qui se connecte au serveur et l'arrête. La commande se connecte en tant que `root` , qui est le compte d'administration par défaut. Notez que les utilisateurs du système de droits MySQL sont totalement indépendant de ceux de Windows.

Ensuite, installez le serveur comme un service :

```
C:\> mysqld --install
```

Si vous avez des problèmes d'installation de `mysqld` en tant que service en utilisant simplement le nom du serveur, essayez d'utiliser le chemin complet :

```
C:\> C:\mysql\bin\mysqld --install
```

Depuis MySQL MySQL 4.0.2, vous pouvez spécifier un nom de service personnalisé avec l'option `--install` . Depuis MySQL 4.0.3, vous pouvez spécifier en plus l'option `--defaults-file` après le nom du service, pour indiquer où le serveur doit lire les options au démarrage. Les règles qui détermine le nom du service et le fichier d'options à utiliser sont les suivantes :

- Si vous ne spécifiez pas de nom de service, le serveur utilise le nom de service par défaut de `MySQL` et le serveur lit les options du groupe `[mysqld]` dans le fichier d'options standard.
- Si vous spécifiez un nom de service après l'option `--install` , le server va ignorer le groupe d'options `[mysqld]` et lire les options dans le groupe du même nom que le nom du service. Le serveur lit ces options dans le fichier d'options standard.
- Si vous spécifiez une option `--defaults-file` après le nom du service, le serveur va ignorer les fichiers d'options standard et ne lire les options que dans le groupe `[mysqld]` .

Note : avant MySQL 4.0.17, un serveur installé comme service Windows avait des problèmes à se lancer si le chemin ou le nom du service contenait des espaces. Pour cette raison, évitez d'installer MySQL dans un dossier tel que `C:\Program Files` ou avec un nom qui contient des espaces.

Dans le cas général où vous installez le serveur avec l'option `--install` mais sans nom de service, le serveur est installé sous le nom de `MySQL` .

Un exemple plus complexe : voyez la commande suivante, qui peut être saisie sur une seule ligne :

```
C:\> C:\mysql\bin\mysqld --install mysql
         --defaults-file=C:\my-opts.cnf
```

Ici, le nom du service est donné après l'option `--install` . Si aucune option `--defaults-file` n'est donnée, cette commande aurait pour effet de faire lire au serveur le groupe `[mysql]` dans les fichiers d'options standard. Cela est une mauvaise idée, car ce groupe d'options est aussi celui du client `mysql` . Cependant, comme l'option `--defaults-file` est présente, le serveur lit les options uniquement dans le fichier indiqué, et uniquement dans le groupe d'options `[mysqld]` .

Vous pouvez aussi spécifier les options comme `` `Start parameters` '' dans l'utilitaire Windows `Services` avant de lancer le service.

Une fois que le serveur MySQL est installé, Windows va lancer automatiquement le service lorsque Windows se lance. Le service peut aussi être lancé immédiatement depuis l'utilitaire `Services` , ou avec la commande en ligne `NET START MySQL` . La commande `NET` n'est pas sensible à la casse.

Lorsqu'il fonctionne comme un service, `mysqld` n'a pas accès à la console Windows, et aucune message n'apparaîtra la. Si `mysqld` ne démarre pas, vérifiez dans le fichier d'erreurs si le serveur a inscrit des messages qui indiquent la cause du problème. Le fichier d'erreurs est situé dans le dossier `C:\mysql\data` . Il porte le suffixe `.err` .

Lorsque `mysqld` fonctionne comme un service, il peut être stoppé par l'utilitaire `Services` , la commande `NET STOP MySQL` , ou la commande `mysqladmin shutdown` . Si le service fonctionne lors de l'extinction de Windows, ce dernier va stopper automatiquement le serveur.

Depuis MySQL version 3.23.44, vous avez le choix d'installer le serveur comme un service `Manuel` , si vous ne voulez pas que le serveur soit lancé automatiquement au lancement du serveur Windows. Pour cela, utilisez l'option `--install-manual` plutôt que `--install` :

```
C:\> C:\mysql\bin\mysqld --install-manual
```

Pour supprimer un serveur qui a été installé comme service, commencez par l'arrêter s'il fonctionnait. Puis, utilisez l'option `--remove` pour le supprimer :

```
C:\> C:\mysql\bin\mysqld --remove
```

Pour les versions MySQL antérieure à la 3.23.49, un problème avec l'arrêt automatique réside dans le fait que Windows n'attend que quelques secondes avant l'extinction complète, et tue les processus si cette limite est dépassée. C'est la cause de problème potentiels (par exemple, le moteur `InnoDB` devra faire une restauration de base au prochain redémarrage). Depuis MySQL version 3.23.49, Windows attend suffisamment longtemps pour que le serveur s'arrête. Si vous remarquez que ce n'est pas suffisant pour votre installation, il est plus prudent de ne pas faire tourner MySQL comme un service. Au lieu de cela, lancez−le en ligne de commande, et stoppez le avec `mysqladmin shutdown` .L'augmentation du délai d'attente de Windows fonctionne avec Windows 2000 et XP. Elle ne fonctionne pas pour Windows NT, où Windows attend 20s l'extinction d'un service. Vous pouvez augmenter cette valeur par défaut en ouvrant la base de registres : `\winnt\system32\regedt32.exe` , et en éditant la valeur de `WaitToKillServiceTimeout` à `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control` é Spécifiez une nouvelle valeur plus grande, exprimée en millisecondes. Par exemple, la valeur de 120000 indique à Windows NT d'attendre 2 minutes (120 secondes).

Si vous ne voulez pas lancer `mysqld` comme un service, vous pouvez le lancer en ligne de commande, comme vous le faites sur les versions qui ne sont pas Windows NT. Pour des instructions, voyez Lancer MySQL en ligne de commande Windows .

2.2.1.8 Faire fonctionner MySQL sous Windows

Vous pouvez vérifier si MySQL fonctionne en exécutant les commandes suivantes :

```
 C:\> C:\mysql\bin\mysqlshow
C:\> C:\mysql\bin\mysqlshow -u root mysql
C:\> C:\mysql\bin\mysqladmin version status proc
C:\> C:\mysql\bin\mysql test
```

Si `mysqld` est lent à répondre aux connexions sur Windows 9x/Me, il y a probablement un problème avec vos DNS. Dans ce cas, démarrez `mysqld` avec `--skip-name-resolve` et utilisez `localhost` et les adresses IP dans les tables de droits MySQL. Vous pouvez aussi éviter les DNS lors de la connexion à un serveur `mysqld-nt` MySQL tournant sur NT/2000/XP en utilisant l'argument `--pipe` pour spécifier l'utilisation des tunnels nommés. Cela fonctionne pour la plupart des clients MySQL.

Vous pouvez forcer le client MySQL à utiliser des connexions par pipes nommés plutôt que TCP/IP en spécifiant l'option `--pipe` ou en spécifiant un point `.` comme hôte. Utilise les options de `--socket` pour spécifier le nom du pipe. En MySQL 4.1, vous devez utiliser `--protocol=PIPE` .

Il y a deux versions de l'outil en ligne de commande MySQL :

Binaire	Description
mysql	Compilé nativement sur Windows, ce qui fournit des possibilités très limitées d'édition de texte.
mysqlc	Compilé avec le compilateur Cygnus GNU et bibliothèques, ce qui fournit l'édition readline .

Si vous voulez utiliser `mysqlc.exe` , vous devez copier `C:\mysql\lib\cygwinb19.dll` dans un dossier ou `mysqlc` peut le trouver). Les distributions courantes de MYSQL inclut cette bibliothèque dans le même dossier que `mysqlc` (le dossier `bin` sous la base de données de votre installation MySQL). Si votre distribution n'a pas de bibliothèque `cygwinb19.dll` dans le dossier `bin` , recherchez la dans le dossier `lib` et copiez le dans votre dossier système Windows (`\Windows\system` ou équivalent).

2.2.1.9 MySQL pour Windows face à MySQL pour Unix

MySQL pour Windows a prouvé qu'il était très stable. Cette version de MySQL a les mêmes fonctionnalités que la version Unix, a quelques exceptions :

Windows 95 et les threads
: Windows 95 perd environs 200 octets de mémoire central lors de la création de chaque thread. Chaque connexion MySQL crée un nouveau thread, ce qui fait qu'il n'est pas recommandé d'exécuter `mysqld` pour des durées longues sur Windows 95 si votre serveur gère de nombreuses connexions. Les autres versions de Windows ne souffrent pas du même problème.

Nombre limités de ports de connexions
: Les systèmes Windows disposent d'environs 4,000 pour les connexions clientes, et après connexion, cela prend de 2 à 4 minutes avant qu'un port soit de nouveau utilisable. Dans des situations où les clients se connectent et se déconnectent à haute vitesse, il est possible que tous les ports soient utilisés, avant que les anciens ports ne redeviennent utilisables. Lorsque cela arrive, le serveur semblera inaccessible, même s'il fonctionne bien. Notez que des ports peuvent aussi être utilisées par d'autres applications, ce qui réduit encore le nombre de ports disponibles pour MySQL.

Lectures concurrentes
: MySQL dépends des fonctions `pread()` et `pwrite()` pour être capable de mêler des `INSERT` et des `SELECT` . Actuellement, nous utilisons les `mutexes` pour émuler les fonctions `pread()` / `pwrite()` . Nous allons, à long terme, remplacer ce niveau d'interface par une interface virtuelle de façon à ce que nous puissions utiliser l'interface `readfile()` / `writefile()` de `Windows NT/2000/XP` pour gagner de la vitesse. L'implémentation courante limite le nombre de fichiers ouverts par MySQL à 1024, ce qui signifie que vous ne pouvez pas utiliser d'aussi nombreux threads concurrents sur `Windows NT/2000/XP` que sur Unix.

`Blocking read`
: MySQL utilise une lecture bloquée pour chaque connexion. Cela signifie que :
  - Une connexion ne sera pas déconnectée automatiquement après 8 heures d'inactivité, comme c'est le cas sous Unix.
  - Si une connexion se bloque, il est impossible de la détruire sans tuer MySQL. `mysqladmin kill` ne fonctionne pas sur une connexion endormie.

`mysqladmin shutdown` ne peut pas s'exécuter tant qu'il y a des connexions qui dorment.

Nous envisageons de corriger ce problème, lorsque les développeurs Windows auront fourni un palliatif.

`DROP DATABASE`

Vous ne pouvez pas détruire une base qui est utilisée par un autre thread.

Interrompre MySQL depuis le gestionnaire de tâches

Vous ne pouvez pas tuer MySQL depuis le gestionnaire de tâche ou avec un utilitaire d'extinction de Windows 95. Vous devez l'éteindre avec `mysqladmin shutdown` .

Noms sensibles à la casse

Les noms de fichiers sont insensibles à la casse sous Windows, ce qui fait que les noms de tables et de bases ne sont pas sensibles à la casse pour MySQL sur Windows. La seule restriction est que les noms de tables et de bases doivent être donnés avec même casse dans le nom (tout en majuscules, ou en minuscules). Identifier Case Sensitivity .

Le caractère `'\'`

Les composants d'un chemin sont séparés par le caractère `'\'` sous Windows, qui est aussi le caractère de protection de MySQL. Si vous utilisez la commande `LOAD DATA INFILE` ou `SELECT ... INTO OUTFILE` , vous devez doubler le caractère `'\'` :

```
mysql> LOAD DATA INFILE "C:\\tmp\\skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

Alternativement, vous pouvez utiliser les noms de fichiers au format Unix, avec le caractère `'/'` :

```
mysql> LOAD DATA INFILE "C:/tmp/skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

Problèmes avec les pipes

Les pipes ne fonctionnent pas de manière fiables depuis la ligne de commande Windows. Si le pipe reçoit un caractère `^Z` / `CHAR(24)` , Windows va croire qu'il atteind la fin du fichier et arrête le programme.C'est un problème, notamment lorsque vous essayez d'utiliser le log binaire de cette manière :

```
mysqlbinlog binary-log-name | mysql --user=root
```

Si vous rencontrez un problème lors de l'utilisation du log, et que vous pensez que c'est à cause du caractère `^Z` / `CHAR(24)` , vous pouvez utiliser le palliatif suivant :

```
mysqlbinlog binary-log-file --result-file=/tmp/bin.sql
mysql --user=root --execute "source /tmp/bin.sql"
```

Cette dernière commande peut aussi être utilisée pour lire fiablement tout fichier SQL qui contient des données binaires.

L'erreur `Can't open named pipe`

Si vous utilisez MySQL 3.22 sous `NT` avec les derniers clients mysql, vous allez obtenir cette erreur :

```
error 2017: can't open named pipe to host: . pipe...
```

Ceci est dû au fait que les versions modernes de MySQL utilisent des pipes nommés sous `NT` , par défaut. Pour éviter cette erreur, vous devez utiliser l'option `--host=localhost` sur les nouveaux clients, ou bien créer le fichier d'options `C:\my.cnf` , qui contiendra les informations suivantes :

```
[client]
host = localhost
```

Depuis la version 3.23.50, les pipes nommés sont les seuls activés si `mysqld` est démarré avec `--enable-named-pipe` .

Erreur `Access denied for user`

Si vous rencontrez l'erreur `Access denied for user: 'utilisateur@unknown' to database 'mysql'` lors de l'accès au serveur MySQL sur la même machine, cela signifie que MySQL ne peut résoudre proprement votre nom d'hôte.

Pour corriger cela, vous devriez créer un fichier `\windows\hosts` dans l'information suivante :

```
127.0.0.1       localhost
```

`ALTER TABLE`
: Lorsque vous exécutez la commande `ALTER TABLE` , la table est verrouillé, empêchant les autres threads d'y accéder. Cela est lié au fait que sous Windows, vous ne pouvez pas effacer un fichier qui est en cours d'utilisation par d'autres threads : à l'avenir, vous pourrions trouver un moyen de contourner ce problème.

`DROP TABLE`
: La commande `DROP TABLE` sur une table qui est utilisée dans le cadre d'un `MERGE` ne fonctionne pas sous Windows, car le gestionnaire de `MERGE` garde la carte des tables cachée de la couche supérieure de MySQL. Comme Windows ne vous autorise pas à effacer des fichiers qui sont ouverts, vous devez d'abord vider de la mémoire toutes les tables du `MERGE` (avec la commande `FLUSH TABLES`) puis effacer la table `MERGE` avant d'effacer les tables. Nous allons corriger cela lorsque nous introduirons la notion de `VIEW` s.

`DATA DIRECTORY` *et* `INDEX DIRECTORY`
: Les directives `DATA DIRECTORY` et `INDEX DIRECTORY` de `CREATE TABLE` sont ignorées sous Windows, car Windows ne supporte pas les liens symboliques.

Voici quelques problèmes connus et pas encore corrigés, si jamais quelqu'un souhaite nous aider sur la version Windows :

- Ajouter des icônes pour le démarrage et l'arrêt de MySQL, dans l'installateur.
- Il serait vraiment pratique de pouvoir arrêter le processus `mysqld` depuis le gestionnaire de tâches. Pour le moment, il faut passer par `mysqladmin shutdown` .
- Le port de `readline` sur Windows pour pouvoir l'utiliser avec l'outil de ligne de commande `mysql` .
- Des versions graphiques des clients standards MySQL (`mysql` , `mysqlshow` , `mysqladmin` et `mysqldump`) seraient bien.
- Il serait bien si les fonctions de lecture et d'écriture sur les sockets de `net.c` pouvaient être interrompues. Cela rendrait possible l'arrêt des threads en court avec `mysqladmin kill` sous Windows.
- Ajouter des macros pour utiliser les méthodes rapides d'incrément/décrément compatibles avec les threads, fourni par Windows.

2.2.2 Installer MySQL sous Linux

Il est recommandé d'installer MySQL sous Linux en utilisant un fichier `RPM` . Les `RPM` de MySQL sont actuellement compilé sur une Red Hat en version 6.2, mais devraient fonctionner sur toute autre version de Linux qui supporte `rpm` et utilise `glibc` . Pour obtenir les paquets `RPM` , voyez la section Obtenir MySQL .

Note: les distributions `RPM` de MySQL sont souvent fournies par d'autres éditeurs. Soyez prévenus qu'elles peuvent contenir des fonctionnalités différents de celles proposées par MySQL AB, et les instructions de ce manuel ne s'appliquent pas forcément. Les instructions de l'éditeur doivent alors être utilisées.

Si vous avez des problèmes avec un fichier `RPM` , si vous obtenez par exemple l'erreur `` `Sorry, the host 'xxxx' could not be looked up` '', référez vous à Notes pour les distributions binaires de Linux .

Dans la plupart des cas, vous n'aurez besoin que d'installer les paquets du `serveur MySQL` et du `client MySQL` pour obtenir une installation MySQL fonctionnelle. Les autres paquets ne sont pas nécessaires pour une installation standard. Si vous voulez utiliser la version `MySQL Max` qui a des fonctionnalités supplémentaires, vous devez installer le `RPM MySQL-Max` . Cependant, il est recommandé de ne faire cela qu' **après** avoir installé le `RPM MySQL-server` . `mysqld-max` , la version étendue du serveur `mysqld` .

Si vous obtenez une message d'erreur de dépendance lors de l'installation des paquets MySQL 4.0 (par exemple, `` `error: removing these paquets would break dependencies: libmysqlclient.so.10 is needed by ...` ''), vous devriez aussi installer le paquet `MySQL-shared-compat` , qui inclut les bibliothèques partagées pour compatibilité ascendante (`libmysqlclient.so.12` pour MySQL 4.0 et `libmysqlclient.so.10` pour MySQL 3.23).

De nombreuses distributions Linux sont livrés avec MySQL 3.23, et elle sont dynamiquement liées à d'autres applications pour économiser de l'espace. Si ces bibliothèques partagées sont dans un paquet séparé (par exemple, `MySQL-shared`), il suffit de laisser le paquet installé, puis de mettre à jour le serveur et les clients qui sont statiquement liés à la bibliothèque, et ne dépendent pas des bibliothèques partagées. Pour les distributions qui incluent les bibliothèques partagées dans le même paquet que le serveur MySQL, (par exemple, Red Hat Linux), vous pouvez soit installer notre `RPM MySQL-shared` 3.23, soit utiliser le paquet `MySQL-shared-compat` .

Les fichiers `RPM` dont vous pourriez avoir besoin sont :

- `MySQL-server-VERSION.i386.rpm`

  Le serveur MySQL. Vous en aurez besoin à moins que vous ne vouliez que vous connecter à un serveur MySQL tournant sur une autre machine. Notez bien : Les fichiers du `RPM` de serveur étaient appelés `MySQL-VERSION.i386.rpm` avant MySQL 4.0.10. C'est à dire qu'ils n'avaient pas le mot `-server` dans leur nom.
- `MySQL-Max-VERSION.i386.rpm`

  Le serveur MySQL Max. Ce serveur a des capacités supplémentaires par rapport au serveur `MySQL-server` . Vous devez installer le `RPM MySQL-server` d'abord, parce que le `RPM MySQL-Max` dépend de lui.
- `MySQL-client-VERSION.i386.rpm`

  Les programmes clients MySQL standards. Vous avez certainement besoin d'installer ce paquet.
- `MySQL-bench-VERSION.i386.rpm`

  Tests et bancs d'essai. Nécessite Perl et les modules `RPM` msql et mysql.
- `MySQL-devel-VERSION.i386.rpm`

  Bibliothèques et fichiers d'inclusions dont vous aurez besoin pour compiler d'autres clients MySQL, tels que les modules Perl.
- `MySQL-shared-VERSION.i386.rpm`

  Ce paquet contient les bibliothèques partagées (`libmysqlclient.so*`) que certains langages et applications recherchent pour les charger dynamiquement, afin d'utiliser MySQL.
- `MySQL-shared-compat-VERSION.i386.rpm`

  Ce paquet inclut la bibliothèque partagée pour MySQL 3.23 et MySQL 4.0. Installez ce paquet au lieu de `MySQL-shared` , si vous avec des applications installée qui utilisent dynamiquement MySQL 3.23 mais que vous voulez passez à MySQL 4.0 sans briser les dépendances. Ce paquet est disponible depuis MySQL 4.0.13.
- `MySQL-embedded-VERSION.i386.rpm`

  La bibliothèque intégrée MySQL (depuis MySQL 4.0).
- `MySQL-VERSION.src.rpm` Celui−ci contient le code source de tous les paquets précédents. Il peut donc être utilisé pour construire des fichiers `RPM` pour d'autres architectures (par exemple, l'Alpha ou le SPARC).

Pour voir tous les fichiers présents dans un paquet `RPM` , lancez :

```
shell> rpm -qpl MySQL-VERSION.i386.rpm
```

Pour effectuer une installation standard minimale, lancez :

```
shell> rpm -i MySQL-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Pour installer uniquement le paquet du client MySQL, lancez :

```
shell> rpm −i MySQL−client−VERSION.i386.rpm
```

Les `RPM` fournissent une fonctionnalité qui permet de vérifier l'authenticité du paquet avant de l'installer. Si vous voulez en savoir plus sur cette fonctionnalité, voyez Vérifier l'intégrité du paquet avec MD5 ou `GnuPG` .

Le fichier `RPM` place les données dans `/var/lib/mysql` . Le `RPM` crée aussi les entrées appropriées dans `/etc/rc.d/` pour lancer le serveur automatiquement au démarrage. (Cela signifie que, si vous avez déjà effectué une installation auparavant, vous pouvez avoir besoin de faire une sauvegarde de vos fichiers de démarrage précédents si vous les changez, de façon à ne pas les perdre.) Voyez Lancer et arrêter automatiquement MYSQL pour plus d'informations sur comment démarrer automatiquement MySQL au lancement du serveur.

Si vous voulez installer le `RPM` MySQL sur une ancienne distribution MySQL, qui ne supporte pas les scripts d'initialisation de `/etc/init.d` (directement ou via un lien symbolique), vous devez créer un lien symbolique qui pointe sur le dossier où les scripts d'initialisation sont installés. Par exemple, si ce dossier est `/etc/rc.d/init.d` , utilisez une de ces commandes avant d'installer le `RPM` , pour créer `/etc/init.d` sous forme de lien symbolique, qui pointe ici :

```
shell> cd /etc; ln −s rc.d/init.d .
```

Cependant, toutes les distributions Linux courantes doivent supporter le nouveau dossier `/etc/init.d` , car c'est imposé par la compatibilité LSB (`Linux Standard Base`).Si les fichiers `RPM` que vous installez incluent `MySQL−server` , le démon `mysqld` devrait fonctionner après l'installation. Vous devriez être capable d'utiliser MySQL immédiatement.

Si quelque chose cloche, vous pouvez trouver plus d'information dans le chapitre d'installation binaire. Installer MySQL sous les autres systèmes Unix .

Note: Les comptes qui sont listés dans les tables de droits MySQL initiales n'ont pas de mot de passe. Après démarrage du serveur, il est recommandé de configurer ces mots de passe, en suivant les instructions de la section Configuration et tests consécutifs à l'installation .

2.2.3 Installer MySQL sur Mac OS X

Depuis MySQL 4.0.11, vous pouvez installer MySQL sur Mac OS X 10.2 ("Jaguar") avec le paquet binaire Mac OS X `PKG` au lieu d'utiliser la distribution binaire compressée. Notez que les anciennes versions de Mac OS X (i.e. 10.1.x) ne sont pas supportées par ce paquet.

Le paquet est situé dans une image disque (`.dmg`), que vous devez monter en double−cliquant son icône sur le `Finder` . Le disque devrait alors se monter, et afficher son contenu.

Pour télécharger MySQL, voyez Obtenir MySQL .

Note : avant de lancer l'installation, assurez vous qu'il n'y a pas de serveur MySQL en fonctionnement! Arrêtez tous les serveurs MySQL avant de continuer, soit en utilisant l'application manager (pour les serveurs Mac OS X) ou via `mysqladmin shutdown` en ligne de commande.

Pour installer le paquet MySQL, double−cliquez sur l'icône. Cela va lancer l'installeur de paquet MacOSX, qui vous guidera durant l'installation.

A cause d'un bug dans l'installeur de paquets MySQL, vous pourriez rencontrer le message d'erreur

```
You cannot install this software on this disk. (null)
```

dans le dialogue de sélection du disque de destination. Si cette erreur survient, cliquez sur le bouton de retour (`Go Back`) pour retourner à l'écran précédent. Puis, cliquez sur le bouton d'avance (`Continue`) pour passer à nouveau à la page de sélection des disques. Nous avons indiqué ce bug à Apple, qui travaille sur le sujet.

Le paquet Mac OS X de MySQL va s'installer lui−même dans le dossier `/usr/local/mysql-VERSION` et va aussi ajouter un lien symbolique `/usr/local/mysql` , qui pointe sur le nouveau dossier. Si un dossier appelé `/usr/local/mysql` existe déjà, il sera renommé en `/usr/local/mysql.bak` . De plus, il va installer les tables de droits MySQL en exécutant le script `mysql_install_db` après l'installation.

Le schéma d'installation est semblable à celui de la distribution binaire, tous les programmes MySQL sont situés dans le dossier `/usr/local/mysql/bin` . Les sockets MySQL sont installées dans le fichier `/etc/mysql.sock` par défaut. Plans d'installation .

L'installation requiert un compte nommé `mysql` (qui existe par défaut en Mac OS X 10.2 et plus récent).

Si vous utilisez Mac OS X Server, vous devez avoir déjà une version de MySQL installée :

Version Mac OS X Server	**Version MySQL**
10.2−10.2.2	3.23.51
10.2.3−10.2.6	3.23.53
10.3	4.0.14
10.3.2	4.0.16

Cette section du manuel couvre l'installation du paquet binaire MySQL pour Mac OS X uniquement. Assurez vous de bien lire l'aide d'Apple concernant l'installation de MySQL (Lancer le visualiseur d'aide, sélectionnez la rubrique "Serveur Mac OS X", et faites une recherche sur "MySQL", puis lisez l'entrée appelée " `Installing MySQL` ").

Notez bien que la version pre−installée de MySQL sur Mac OS X Server peut être lancée avec la commande `safe_mysqld` au lieu de `mysqld_safe` !

Si vous avez utilisé auparavant les paquets MySQL de Marc Liyanage pour Mac OS X, depuis le site de http://www.entropy.ch , vous pouvez simplement suivre le processus de mise à jour, en utilisant les conseils de ses pages.

Si vous faites une mise à jour depuis les versions de Marc, ou depuis une ancienne version de MySQL pour Mac OS X Server, avec le nouveau paquet officiel, vous devrez convertir les tables de droits. Mise à jour de Version 3.23 vers 4.0 .

Si vous voulez lancer automatiquement MySQL au démarrage du système, vous devez aussi installer le `MySQL Startup Item` . Depuis MySQL 4.0.15, il fait partie du disque d'installation MySQL pour Mac OS X dans un paquet séparé. Il suffit de double−cliquer sur l'icône `MySQLStartupItem.pkg` et de suivre les instructions pour l'installer.

Notez que le `MySQL Startup Item` ne doit être installé qu'une seule fois. Il n'y a pas besoin de me mettre à jour avec les versions de MySQL.

Le `MySQL Startup Item` est installé dans le dossier `/Library/StartupItems/MySQLCOM` . (Avant MySQL 4.1.2, le dossier était `/Library/StartupItems/MySQL` , mais cela créait un conflit avec le `MySQL Startup Item` du serveur `Mac OS X Server` .) Il ajoute la variable `MYSQLCOM=-YES-` au fichier de configuration `/etc/hostconfig` . Si vous voulez désactiver le démarrage automatique de MySQL, modifiez simplement la variable avec `MYSQLCOM=-NO-` .

Sur Mac OS X Server, l'installation par défaut de MySQL utilise la variable `MYSQL` dans `/etc/hostconfig` . Le `MySQL Startup Item` désactive cette variable en lui donnant la valeur de `MYSQL=-NO-` . Cela évite que des conflit de démarrage surviennent, si la variable `MYSQLCOM` est utilisée par `MySQL Startup Item` . Cependant, cela n'éteint pas un serveur MySQL en fonctionnement.

Après cette installation, vous pouvez lancer le serveur MySQL avec ces commandes, exécutées dans terminal. Notez bien que vous devez avoir des droits d'administrateur pour cela!

Si vous avez installé le `MySQL Startup Item` :

```
 shell> sudo /Library/StartupItems/MySQL/MySQL start
(Enter your password, if necessary)
(Press Control−D or enter "exit" to exit the shell)
```

Si vous n'avez pas installé le `MySQL Startup Item` :

```
 shell> cd /usr/local/mysql
shell> sudo ./bin/mysqld_safe
(Enter your password)
(Press CTRL+Z)
shell> bg
(Press CTRL+D to exit the shell)
```

Vous devez alors être capable de vous connecter au serveur MySQL, notamment en exécutant la commande `/usr/local/mysql/bin/mysql` . **Note :** cette procédure ne configure aucun mot de passe pour les comptes MySQL. Après avoir suivi cette procédure, lisez la section Configuration et tests consécutifs à l'installation , pour les instructions de post−installation et les tests.

Vous pouvez faire cela avec les commandes suivantes :

```
 /usr/local/mysql/bin/mysqladmin −u root password <password>
/usr/local/mysql/bin/mysqladmin −u root −h `hostname` password <password>
```

Vous pouvez aussi ajouter des alias à votre fichier de ressource Shell, pour accéder à `mysql` et `mysqladmin` depuis la ligne de commande :

```
 alias mysql '/usr/local/mysql/bin/mysql'
alias mysqladmin '/usr/local/mysql/bin/mysqladmin'
```

Alternativement, vous pouvez simplement ajouter `/usr/local/mysql/bin` à votre variable d'environnement `PATH` , par exemple, en ajoutant la ligne suivante dans votre fichier `$HOME/.tcshrc` :

```
 setenv PATH $&#123;PATH&#125;:/usr/local/mysql/bin
```

Si aucun fichier `.tcshrc` n'existe dans votre dossier d'utilisateur, créez le avec un éditeur texte.

Notez que l'installation d'un nouveau paquet MySQL ne supprimer pas le dossier d'une vieille installation : l'installeur de Mac OS X n'offre pas encore les fonctionnalités nécessaires pour mettre à jour proprement une vieille version.

Après avoir copié les fichiers de bases depuis votre ancien dossier vers le nouveau, et que vous avez réussi à redémarrer MySQL avec la nouvelle version, vous devriez pensez à supprimer les vieilles versions des fichiers, situées dans `/Library/Receipts/mysql-<version>.pkg` .

2.2.4 Installer MySQL sur NetWare

Porter `MySQL` sur `NetWare` a été un effort dirigé par `Novell` . Les clients Novell seront heureux de constater que NetWare 6.5 est distribué avec les exécutables MySQL et une licence de support commercial pour tous les serveurs qui fonctionnent sur cette version de NetWare.

MySQL pour NetWare est compilé avec une combinaison de `Metrowerks CodeWarrior pour NetWare` et de version spéciales de compilation de GNU `autotools` .

Les derniers paquets binaires pour NetWare sont disponibles sur le site http://dev.mysql.com/downloads/ . Obtenir MySQL .

Pour pouvoir héberger un serveur MySQL, un serveur NetWare doit avoir les pré−requis suivants.

- NetWare version 6.5, ou NetWare 6.0 avec Support Pack 3 installé (vous pouvez obtenir cela sur http://support.novell.com/filefinder/13659/index.html). Le système doit passer les pré−requis minimum de Novell pour faire tourner cette version de NetWare.
- Le sytème doit satisfaire les pre−requis minimum de NetWare.
- Les données MySQL, ainsi que les logiciels, doivent être installés dans un volume NSS; les volumes traditionnels ne sont pas supportés.

Les fichiers binaires de NetWare sont téléchargeables sur le site http://www.mysql.com/downloads/ .

Pour installer MySQL pour NetWare, utilisez la procédure suivante :

- Si vous mettez à jour une ancienne installation, stoppez le serveur MySQL. Vous pouvez le faire en console, avec la commande suivante :
  ```
  SERVER:  mysqladmin −u root shutdown
  ```
- Connectez vous sur le serveur depuis une machine cliente, avec un accès à l'endroit où vous voulez installer MySQL.
- Décompressez l'archive binaire `zip` sur le serveur. Assurez−vous d'autoriser les chemins utilisés dans l'archive zip. Il est sécuritaire d'utiliser le dossier `SYS:\` .Si vous mettez à jour une ancienne installation, vous pouvez copier le dossier de données (par exemple, `SYS:MYSQL\DATA`), ainsi que `my.cnf` si vous l'avez modifié. Vous pouvez alors effacer l'ancienne copie de MySQL.
- Vous pouvez renommer le dossier avec un nom plus cohérent, et facile à utiliser. Nous recommandons d'utiliser `SYS:MYSQL` ; les exemples du manuel feront références à ce dossier d'installation.
- Depuis la console du serveur, ajoutez un chemin de recherche pour le dossier contenant les `NLM` MySQL. Par exemple :
  ```
  SERVER:  SEARCH ADD SYS:MYSQL\BIN
  ```
- Installez la base de données initiale, si nécessaire, en exécutant le script `mysql_install_db` depuis la console.
- Lancez le serveur MySQL en utilisant le script `mysqld_safe` depuis la console.
- Pour finir l'installation, vous devriez aussi installer les commandes suivantes dans le fichier `autoexec.ncf` . Par exemple, si votre installation MySQL est dans le dossier `SYS:MYSQL` et que vous voulez que MySQL se lance automatiquement, vous pouvez ajouter ces lignes :
  ```
   #Starts the MySQL 4.0.x database server
  SEARCH ADD SYS:MYSQL\BIN
  MYSQLD_SAFE
  ```
  Si vous utilisez MySQL sur NetWare 6.0, nous vous recommandons fortement d'ajouter l'option `--skip-external-locking` à la ligne de commande :
  ```
   #Starts the MySQL 4.0.x database server
  SEARCH ADD SYS:MYSQL\BIN
  MYSQLD_SAFE --skip-external-locking
  ```
  Il est aussi nécessaire d'utiliser `CHECK TABLE` et `REPAIR TABLE` au lieu de `myisamchk` , car `myisamchk` utilise un verrouillage externe. Le verrouillage externe est reconnu pour poser des problèmes sur NetWare 6.0; ce problème a été supprimé sur NetWare 6.5. `mysqld_safe` sur NetWare fournit un écran de présence. Lorsque vous déchargez (extinction) le `NLM mysqld_safe` , l'écran ne se ferme pas par défaut. Au lieu de cela, il demande une action de l'utilisateur :
  ```
   *<NLM has terminated; Press any key to close the screen>*
  ```
  Si vous voulez que NetWare ferme automatiquement cet écrant, utilisez l'option `--autoclose` de `mysqld_safe` . Par exemple :
  ```
   #Starts the MySQL 4.0.x database server
  SEARCH ADD SYS:MYSQL\BIN
  MYSQLD_SAFE --autoclose
  ```

Le comportement de `mysqld_safe` sur NetWare est décrit plus en détails dans `safe_mysqld` , le script père de `mysqld` .

S'il y avait une installation MySQL sur le serveur, assurez vous de vérifier les commandes de démarrage de MySQL dans `autoexec.ncf` , et éditez ou effacez les autant que nécessaire.

Note : Les comptes qui sont stockés dans les tables de droits n'ont pas de mot de passe initial. Après avoir lancé le serveur, il est recommandé de leur donner des mots de passe en suivant les instructions de la section Configuration et tests consécutifs à l'installation .

2.2.5 Vue d'ensemble de l'installation rapide

Cette section couvre l'installation des distributions binaires de MySQL, qui sont fournies pour différentes plate−formes au format d'archive `tar` (les fichiers avec l'extension `.tar.gz`). Voyez Les compiliations de MySQL par MySQL AB pour une liste détaillée.

Pour télécharger une distribution source de MySQL, voyez Obtenir MySQL .

En plus de ces paquets génériques, nous offrons aussi des compilations spécifiques pour certaines plate−formes. Voyez Installation rapide et standard de MySQL pour plus d'information sur leur installation.

Vous avez besoin des utilitaires suivants pour installer une archive `tar` MySQL :

- GNU `gunzip` pour décompresser la distribution.
- Un utilitaire `tar` raisonnable pour ouvrir l'archive. GNU `tar` est reconnu pour cette tâche. Certains systèmes d'exploitation disposent d'une version pré−installée de `tar` qui posent des problèmes. Par exemple, Sun `tar` et Mac OS X `tar` ont des soucis avec les noms de fichiers longs. Dans ce cas, installez GNU `tar` . Sur Mac OS X, vous pouvez installer le logiciel pré−installé `gnutar` .

Si vous rencontrez des problèmes, **utilisez toujours** `mysqlbug` pour poser des questions à la liste MySQL. Même si le problème n'est pas un bug, `mysqlbug` rassemble des informations sur le système qui nous aiderons à résoudre votre problème. Si vous n'utilsiez pas `mysqlbug` , vous réduisez les chances de résolution de votre problème. Vous trouverez `mysqlbug` dans le dossier `bin` après avoir décompressé la distribution. Comment rapporter des bogues ou problèmes .

Les commandes de base que vous devez exécuter pour installer MySQL à partir des sources sont :

```
 shell> groupadd mysql
shell> useradd −g mysql mysql
shell> cd /usr/local
shell> gunzip < /path/to/mysql−VERSION−OS.tar.gz | tar xvf −
shell> ln −s full−path−to−mysql−VERSION−OS mysql
shell> cd mysql
shell> scripts/mysql_install_db
shell> chown −R root  .
shell> chown −R mysql data
shell> chgrp −R mysql .
shell> bin/mysqld_safe −−user=mysql &
```

Pour les versions de MySQL plus ancienne que la 4.0, remplacez `bin/safe_mysqld` par `bin/mysqld_safe` dans la commande finale. **Note :** cette procédure ne configure aucun mot de passe pour les comptes MySQL. Après avoir suivi cette procédure, lisez la section Configuration et tests consécutifs à l'installation , pour les instructions de post−installation et les tests.

Plus de détails suivent.

Pour installer une distribution binaire, suivez les étapes suivantes, puis reportez vous à Configuration et tests consécutifs à l'installation , pour la configuration post−installation et les tests :

- Ajoutez un utilisateur et un groupe avec les droits desquels `mysqld` fonctionnera :

```
shell> groupadd mysql
shell> useradd -g mysql mysql
```

Ces commandes ajoutent le groupe `mysql` group et l'utilisateur `mysql` . La syntaxe de `useradd` et de `groupadd` peut varier de façon significative suivant les versions d'Unix. Elles peuvent aussi s'appeler `adduser` et `addgroup` . Vous pouvez aussi souhaiter nommer le groupe et l'utilisateur autrement que `mysql` .

- Choisissez le dossier dans lequel vous voulez décompresser la distribution, et placez vous−y. Dans l'exemple suivant, nous allons décompresser la distribution dans le dossier `/usr/local` . Les instructions suivantes supposeront que vous avez les droits pour créer des dossiers de des fichiers dans `/usr/local` . Si ce dossier est protégé, vous aurez besoin des droits de `root` pour faire l'installation.

```
shell> cd /usr/local
```

- Téléchargez la distribution sur l'un des sites listé sur Obtenir MySQL .

  Les archives MySQL `tar` ont des noms de la forme `mysql-VERSION-OS.tar.gz` , où `VERSION` est le numéro de version (par exemple, `4.0.17`), et `OS` indique le système d'exploitation de la distribution (par exemple, `pc-linux-gnu-i586`). Pour une version donnée, les distributions binaires pour toutes les plate−formes sont compilées sur les mêmes sources MySQL.
- Décompressez la distribution dans le répertoire courant :

```
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
shell> ln -s full-path-to-mysql-VERSION-OS mysql
```

La commande `tar` crée un dossier appelé `mysql-VERSION-OS` . La commande `ln` crée un lien symbolique dans ce dossier. Cela vous laisse le moyen de transférer facilement votre installation dans le dossier `/usr/local/mysql` .

Avec GNU `tar` , il n'est pas nécessaire d'utiliser séparéement `gunzip` . Vous pouvez remplacer la première ligne par celle−ci pour décompresser et ouvrir l'archive dans le même temps :

```
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
```

- Placez−vous dans le répertoire racine de la distribution décompressée :

```
shell> cd mysql
```

Vous trouverez plusieurs fichiers et sous dossiers dans le dossier `mysql` . Le plus important pour l'installation sont les dossiers `bin` et `scripts` .

- Si vous n'avez jamais installé MySQL auparavant, vous devez créer les tables de droits :

```
shell> scripts/mysql_install_db
```

Notez que pour les versions de MySQL plus anciennes que la version 3.22.10, `mysql_install_db` laisse le serveur fonctionner après avoir créer les tables. Ce n'est plus vrai : vous devez redémarrer le serveur après avoir exécuté ce script.

- Changez le propriétaire du binaire pour `root` et le propriétaire des données pour l'utilisateur qui va faire tourner `mysqld` . En supposant que vous avez installé les données dans le dossier `/usr/local/mysql` , la commande est :

```
shell> chown -R root  .
shell> chown -R mysql data
shell> chgrp -R mysql .
```

La première commande change l'attribut `owner` du fichier pour `root` . La seconde change l'attribut `owner` du dossier de données pour l'utilisateur `mysql` . La troisième change l'attribut `group` pour le groupe `mysql` .

- Si vous voulez que MySQL démarre automatiquement après le démarrage de votre serveur, vous pouvez copier le fichier `support-files/mysql.server` là où votre serveur recherche les scripts de démarrage. Plus d'informations sur `support-files/mysql.server` sont disponibles dans Lancer et arrêter automatiquement MYSQL .
- Vous pouvez configurer de nouveaux comptes en utilisant le script `bin/mysql_setpermission` si vous installez les modules Perl `DBI` et `DBD::mysql` . Pour des instructions, voyez Commentaires sur l'installation de Perl .
- Si vous voulez utiliser `mysqlaccess` et avoir accès à la distribution MySQL dans un dossier non−standard, vous devez modifier le chemin où `mysqlaccess` va rechercher le client `mysql` . Editez le script `bin/mysqlaccess` à la ligne 18, environs. Recherchez une ligne qui ressemble à

ceci :

```
$MYSQL     = '/usr/local/bin/mysql';    # path to mysql executable
```

Changez le dossier pour qu'il prenne la valeur que du chemin où `mysql` est situé. Si vous le le faite pas, vous allez recevoir une erreur `Broken pipe` lorsque vous utilisez `mysqlaccess` .

Après que tout ait été ouvert et installé, il faut tester votre distribution :

Lancez le serveur MySQL avec les commandes suivantes :

```
shell> bin/mysqld_safe --user=mysql &
```

Pour les versions de MySQL plus ancienne que 4.0, remplacez `bin/safe_mysqld` par `bin/mysqld_safe` dans la commande.Puis, passez à `safe_mysqld` , le script père de `mysqld` , et Configuration et tests consécutifs à l'installation .

Note : cette procédure ne configure aucun mot de passe pour les comptes MySQL. Après avoir suivi cette procédure, lisez la section Configuration et tests consécutifs à l'installation , pour les instructions de post−installation et les tests.

2.3 Installation de MySQL avec une distribution source

Avant de procéder à l'installation à partir des sources, vérifiez auparavant que notre distribution binaire pour votre plate−forme ne fonctionne pas. Nous faisons un maximum d'efforts pour nous assurer que nos binaires sont compilés avec les meilleures options possibles.

Les distributions source de MySQL sont fournies sous forme d'archive `tar` , dont le nom est sous la forme `mysql-VERSION.tar.gz` , où `VERSION` est un nombre comme `4.0.20` }.

Pour télécharger une distribution source de MySQL, voyez Obtenir MySQL .

Vous avez besoin des outils suivants pour compiler et installer MySQL à partir des sources :

- GNU `gunzip` pour décompresser la distribution.
- Un programme `tar` pour désarchiver la distribution. GNU `tar` est connu pour fonctionner. Le `tar` de Sun connaît quelques problèmes.
- Un compilateur C++ ANSI fonctionnel. `gcc` >= 2.95.2, `egcs` >= 1.0.2 ou `egcs 2.91.66` , SGI C++, et SunPro C++ sont quelques−uns des compilateurs réputés pour fonctionner. `libg++` n'est pas nécessaire si vous utilisez `gcc` . `gcc` 2.7.x souffre d'un bogue qui l'empêche de compiler quelques fichiers C++ correctement écrits, tels que `sql/sql_base.cc` . Si vous disposez seulement de `gcc` 2.7.x, vous devez mettre à jour votre `gcc` afin de compiler MySQL. `gcc` 2.8.1 est aussi reconnu pour rencontrer des problèmes sur certaines plate−formes, il devrait donc être désactivé si un autre compilateur existe pour la plate−forme.

  `gcc` >= 2.95.2 est recommandé pour compiler MySQL dans ses versions 3.23.x.
- Un bon programme `make` . GNU `make` est une fois de plus recommandé et est quelquefois requis. Si vous rencontrez des problèmes, nous vous recommandons d'essayer GNU `make` 3.75 ou supérieur.

Si vous utilisez une version récente de `gcc` , suffisamment récente pour reconnaître l'option `-fno-exceptions` , il est **très important** que vous l'utilisiez. Sinon, vous risquez de compiler un binaire qui crashe aléatoirement. Nous recommandons donc l'utilisation de `-felide-constructors` et `-fno-rtti` en même temps que `-fno-exceptions` . En cas de doute, faites la chose suivante :

```
CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions \
```

```
-fno-rtti" ./configure --prefix=/usr/local/mysql --enable-assembler \
--with-mysqld-ldflags=-all-static
```

Sur la plupart des systèmes, il en résultera un binaire rapide et stable.

Si vous rencontrez des problèmes, **utilisez toujours** `mysqlbug` pour poster des questions sur les listes internes. Même si le problème n'est pas un bogue, `mysqlbug` rassemble des informations sur le système qui aidera les autres à résoudre votre problème. En n'utilisant pas `mysqlbug` , vous amoindrissez vos chances d'obtenir une solution à votre problème ! Vous trouverez `mysqlbug` dans le répertoire `scripts` après avoir désarchivé la distribution. Comment rapporter des bogues ou problèmes .

2.3.1 Installation depuis les sources : présentation

Les commandes de pages que vous devez exécuter pour installer la distribution source de MySQL sont :

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSION.tar.gz | tar -xvf -
shell> cd mysql-VERSION
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> cd /usr/local/mysql
shell> bin/mysql_install_db
shell> chown -R root  .
shell> chown -R mysql var
shell> chgrp -R mysql .
shell> bin/mysqld_safe --user=mysql &
```

Pour les versions de MySQL 4.0 ou plus récentes, remplacez `bin/safe_mysqld` par `bin/mysqld_safe` dans la commande finale.Si vous commencez avec un paquet `RPM` , commencez comme ceci :

```
shell> rpm --rebuild --clean MySQL-VERSION.src.rpm
```

Cela va compiler un paquet `RPM` binaire que vous pouvez installer. **Note :** cette procédure ne configure aucun mot de passe pour les comptes MySQL. Après avoir suivi la procédure, passez à la section Configuration et tests consécutifs à l'installation , pour les instructions de post−installation et de tests.

Une description plus détaillée suit.

- Ajoutez un utilisateur et un groupe pour `mysqld` comme ceci :
  ```
  shell> groupadd mysql
  shell> useradd -g mysql mysql
  ```
  Ces commandes ajoute le groupe `mysql` et l'utilisateur `mysql` . La syntaxe de `useradd` et `groupadd` peut différer légèrement suivant votre version d'Unix. Elles peuvent aussi s'appeler `adduser` et `addgroup` .Vous pouvez donner un autre nom à l'utilisateur et au groupe, à la place de `mysql` . Si vous le faîtes, adaptez les commandes dans les prochaines instructions.
- Choisissez un dossier dans lequel vous allez décompresser les sources de MySQL. Rendez vous dans ce dossier.
- Téléchargez une distribution sur un des sites listés sur Obtenir MySQL .
- Décompressez la distribution dans le dossier courant :
  ```
  shell> gunzip < /path/to/mysql-VERSION.tar.gz | tar xvf -
  ```
  Cette commande crée un dossier appelé `mysql-VERSION` .Avec GNU `tar` , il n'est pas besoin de faire un appel séparé à `gunzip` . Vous pouvez utilise cette commande alternative pour décompresser et extraire la distribution :
  ```
  shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
  ```
- Placez vous à la racine de la distribution :

```
shell> cd mysql-VERSION
```

Notez qu'actuellement, vous devez configurer et compiler MySQL depuis la racine de la distribution. Vous ne pouvez pas la compiler ailleurs.

* Configurer votre version et compilez le tout :

```
shell> ./configure --prefix=/usr/local/mysql
shell> make
```

Lorsque vous exécutez le code `configure` , vous pouvez aussi ajouter des options. Utilisez la commande `./configure --help` pour avoir une liste des options disponibles. Les options habituelles de `configure` , présente certaines options pratiques.

Si `configure` échoue et que vous allez envoyer un courriel aux listes MySQL pour demander de l'aide, ajouter surtout le contenu du fichier `config.log` qui vous semblent pertinentes. Incluez aussi les dernières lignes affichées par `configure` . Postez votre rapport de bug avec le script `mysqlbug` . Comment rapporter des bogues ou problèmes .

Si la compilation échoue, voyez la section Des problèmes à la compilation ? , pour avoir de l'aide immédiate sur les problèmes les plus courants.

* Installez la distribution :

```
shell> make install
```

Si vous voulez écrire un fichier d'option, utilisez un des fichiers présents dans le dossier `support-files` comme exemple. Par exemple,

```
shell> cp support-files/my-medium.cnf /etc/my.cnf
```

Il est possible que vous ayez à éxécuter certaines commandes en tant que `root` .Si vous voulez ajouter le support des tables `InnoDB` , vous devez éditer le fichier `/etc/my.cnf` , supprimer le caractère `#` dans les lignes d'options qui commencent par `innodb_...` , et donnez à ces options la valeur que vous souhaitez. Voyez Utiliser les fichiers d'options et Configuration `InnoDB` .

* Déplacez vous dans le dossier d'installation :

```
shell> cd /usr/local/mysql
```

* Si vous n'avez jamais installé MySQL auparavant, vous devez créer les tables de droits :

```
shell> bin/mysql_install_db
```

Si vous utilisez la commande en tant que `root` , il est recommandé d'utiliser l'option `--user` présentée. La valeurde l'option doit être le nom de l'utilisateur créé dans les premières étapes pour faire fonctionner le serveur. Si vous exécutez la commande après vous être connecté sous cet utilisateur, vous pouvez omettre l'utilisation de l'option `--user` .Notez que pour les versions de MySQL plus anciennes que la version 3.22.10, `mysql_install_db` laisse le serveur en fonctionnement après avoir créé les tables. Ce n'est plus vrai. Vous devez démarrer le serveur vous−même après avoir fait les dernières étapes d'installation.

* Changez le propriétaire des éxécutables pour les donner à `root` et le propriétaire des données por les donner à `mysqld` . En supposant que vous êtes dans le dossier d'installation, `/usr/local/mysql` , la commande ressemble à ceci :

```
shell> chown -R root  .
shell> chown -R mysql var
shell> chgrp -R mysql .
```

La première commande modifie l'attribut `owner` des fichiers pour les donner à `root` user. La seconde donner les fichiers de données à `mysql` . La troisième commande change le groupe `group` pour les donner au groupe `mysql` .

* Si vous voulez que MySQL se lance automatiquement au démarrage de votre serveur, vous pouvez copier le fichier `support-files/mysql.server` là ou votre système cherche les fichiers de démarrage. Plus d'informations sont disponibles dans le script `support-files/mysql.server` et dans la section Lancer et arrêter automatiquement MYSQL .
* Vous pouvez créer de nouveaux comptes en utilisant le script `bin/mysql_setpermission` si vous avez installé les modules Perl `DBI` et `DBD::mysql` . Pour les instructions, voyez Commentaires sur l'installation de Perl .

Après avoir installé tout, il est recommandé d'initialiser et tester votre installation avec cette commande :

```
shell> /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Pour les versions de MySQL plus ancienne que 4.0, remplacez `bin/safe_mysqld` par `bin/mysqld_safe` dans la commande.Si la commande échoue immédiatement avec le message `mysqld ended` , vous pouvez trouver certaines informations dans le fichier `mysql-data-directory/'hostname'.err` .

Plus d'informations sur `mysqld_safe` sont disponibles dans `safe_mysqld` , le script père de `mysqld` .

Note : les comptes qui sont listés dans les tables de droits MySQL n'ont pas de mot de passe initial. Après le démarrage du serveur, il est recommandé de configurer les mots de passe en utilisant les instructions de Configuration et tests consécutifs à l'installation .

2.3.2 Options habituelles de configure

Le script `configure` vous donne un bon moyen de contrôler la configuration de votre distribution MySQL. Habituellement, vous faites cela en spécifiant les options dans la ligne de commande de `configure` . Vous pouvez aussi affecter le comportement de `configure` en utilisant certaines variables d'environnement. Variables d'environnement MySQL . Pour une liste des options supportées par `configure` , exécutez cette commande :

```
shell> ./configure --help
```

Les options de `configure` les plus utilisées sont décrites ici :

- Pour ne compiler que les bibliothèques et programmes clients, et non le serveur, utilisez l'option `--without-server` :
  ```
  shell> ./configure --without-server
  ```
  Si vous n'avez pas de compilateur C++, `mysql` ne compilera pas (c'est le programme client qui requière C++). Dans ce cas, vous pouvez supprimer la partie de code dans `configure` qui vérifie l'existence d'un compilateur C++, puis exécuter `./configure` avec l'option `--without-server` . La compilation essaiera encore de construire `mysql` , mais vous pouvez ignorer les messages d'erreurs concernant `mysql.cc` . (Si `make` stoppe, essayez `make -k` pour dire de continuer même si on rencontre des erreurs.)
- Si vous voulez obtenir une bibliothèque MySQL intégrée (`libmysqld.a`) vous devez utiliser l'option `--with-embedded-server` .
- Si vous ne voulez pas que vos fichiers de log et bases de données soient dans `/usr/local/var` , utiliser une commande `configure` se rapprochant de l'une des commandes suivantes :
  ```
   shell> ./configure --prefix=/usr/local/mysql
  shell> ./configure --prefix=/usr/local \
             --localstatedir=/usr/local/mysql/data
  ```
  La première change le préfixe de l'installation pour que tout soit installé dans `/usr/local/mysql` au lieu de `/usr/local` par défaut. La seconde commande préserve le préfixe d'installation par défaut mais change le répertoire par défaut pour les bases de données (normalement `/usr/local/var`) en `/usr/local/mysql/data` . Après que vous ayez compilé MySQL, vous pouvez changer ces options dans les fichiers d'options. Utiliser les fichiers d'options .
- Si vous utilisez Unix et que vous voulez que la socket de MySQL soit à un autre endroit que celui par défaut (normalement `/tmp` ou `/var/run`) utilisez une commande `configure` comme celle−ci :
  ```
  shell> ./configure \
             --with-unix-socket-path=/usr/local/mysql/tmp/mysql.sock
  ```
  Notez que le fichier donné doit avoir un chemin absolu ! Vous pourrez aussi changer le chemin vers `mysql.sock` plus tard en utilisant les fichiers d'options de MySQL. Comment protéger `/tmp/mysql.sock` .
- Si vous voulez compiler des programmes liés statiquement (par exemple, pour créer une distribution binaire, pour obtenir plus de vitesse, ou pour résoudre des problèmes avec quelques distributions RedHat Linux), exécutez `configure` de la manière suivante :
  ```
  shell> ./configure --with-client-ldflags=-all-static \
             --with-mysqld-ldflags=-all-static
  ```

* Si vous utilisez `gcc` et n'avez pas `libg++` ou `libstdc++` d'installés, vous pouvez dire à `configure` d'utiliser `gcc` en tant que compilateur C++ :

```
shell> CC=gcc CXX=gcc ./configure
```

Quand vous utilisez `gcc` en tant que compilateur C++, aucune tentative de liaison avec `libg++` ou `libstdc++` ne sera effectuée. Il peut être bon d'utiliser cette méthode même si vous avez les bibliothèques citées, car quelques versions de celles−ci ont causé des problèmes à des utilisateurs MySQL par le passé.Voici quelques variables d'environnement à définir selon le compilateur que vous utilisez :

Dans la plupart des cas, vous pouvez obtenir un binaire MySQL raisonnablement optimal en utilisant les options de la table précédente et en ajoutant les options suivantes aux lignes de configuration :

```
 --prefix=/usr/local/mysql --enable-assembler \
--with-mysqld-ldflags=-all-static
```

En d'autres termes, la ligne de configuration ressemble à ce qui suit pour les versions récentes de gcc :

```
 CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro \
-felide-constructors -fno-exceptions -fno-rtti" ./configure \
--prefix=/usr/local/mysql --enable-assembler \
--with-mysqld-ldflags=-all-static
```

Les binaires que nous fournissons sur le site de MySQL à http://www.mysql.com/ sont tous compilés avec une optimisation totale et devraient être parfaits pour la plupart des utilisateurs. Les compiliations de MySQL par MySQL AB . Il y a quelques choses que vous pouvez modifier pour rendre le binaire encore plus rapide, mais cela est réservé aux utilisateurs avancés. Comment la compilation et les liaisons affectent la vitesse de MySQL .Si la génération échoue et produit des erreurs disant que votre compilateur ou outil de liaison n'est pas capable de créer la bibliothèque partagée `libmysqlclient.so.#` (`'#'` étant un numéro de version), vous pouvez contourner ce problème en donnant l'option `--disable-shared` à `configure` . Dans ce cas, `configure` ne générera pas de bibliothèque partagée `libmysqlclient.so.#` .
* Vous pouvez configurer MySQL pour qu'il n'utilise pas les valeurs `DEFAULT` pour les colonnes non− `NULL` (c'est à dire, les colonnes qui ne peuvent être `NULL`). Contrainte avec `NOT NULL` et `DEFAULT` .

```
shell> CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS ./configure
```

* Par défaut, MySQL utilise le jeu de caractères ISO−8859−1 (Latin1). Pour changer le jeu par défaut, utilisez l'option `--with-charset` :

```
shell> ./configure --with-charset=CHARSET
```

`CHARSET` peut être l'un des `big5` , `cp1251` , `cp1257` , `czech` , `danish` , `dec8` , `dos` , `euc_kr` , `gb2312` , `gbk` , `german1` , `hebrew` , `hp8` , `hungarian` , `koi8_ru` , `koi8_ukr` , `latin1` , `latin2` , `sjis` , `swe7` , `tis620` , `ujis` , `usa7` , ou `win1251ukr` . Le jeu de caractères utilisé pour les données et le stockage .

Depuis MySQL 4.1.1, la collation par défaut peut aussi être spécifiée. MySQL utilise la collation `latin1_swedish_ci` . Pour le changer, utilisez l'option `--with-collation` :

```
shell> ./configure --with-collation=COLLATION
```

Pour changer le jeu de caractères et la collation, utilisez les options `--with-charset` et `--with-collation` . La collation doit être valide pour le jeu de caractères spécifié. Utilisez la commande `SHOW COLLATION` pour déterminer les collations valides pour un jeu de caractères donné.

Si vous voulez convertir les caractères entre le serveur et le client, regardez du côté de la commande `SET CHARACTER SET` . Syntaxe de `SET` .

Attention : Si vous changez les jeux de caractères après avoir crée des tables, vous devrez exécuter `myisamchk -r -q --set-character-set=charset` sur chaque table. Vos index pourraient être stockés de manière incorrecte sinon. (Cela peut survenir si vous installez MySQL, créez quelques tables, puis reconfigurez MySQL pour qu'il utilise un jeu de caractères différent et le réinstallez.)

Avec l'option `--with-extra-charsets=LIST` vous pouvez définir les jeux de caractères additionnels à compiler dans le serveur.

Ici `LIST` est soit une liste de jeux de caractères séparés par des espaces, soit `complex` pour inclure tous les jeux de caractères ne pouvant être chargés dynamiquement, ou encore `all` pour inclure tous les jeux de caractères dans les binaires.
- Pour configurer MySQL avec le code de débogage, utilisez l'option `--with-debug` :
```
shell> ./configure --with-debug
```
  Cela alloue un vérificateur d'allocation de mémoire qui peut trouver quelques erreurs et qui fournit des informations sur ce qui se produit. Déboguer un serveur MySQL .
- Si vos programmes clients utilisent les threads, vous avez besoin de compiler une version sûre pour les threads de la bibliothèque du client MySQL avec l'option de configuration `--enable-thread-safe-client` . Cela créera une bibliothèque `libmysqlclient_r` avec laquelle vous devez lier vos applications threadées. Comment programmer une client threadé .
- Les options relatives à un système d'exploitation particulier peuvent être trouvées dans la section spécifique aux systèmes de ce manuel. Remarques spécifiques aux systèmes d'exploitation .

2.3.3 Installer à partir de l'arbre source de développement

Attention : Vous devez lire cette partie seulement si vous voulez nous aider à tester notre nouveau code. Si vous souhaitez seulement faire fonctionner MySQL sur votre système, vous devriez utiliser la distribution d'une version standard (que ce soit une distribution sous forme de sources ou de binaire).

Pour obtenir notre arbre source de développement le plus réent, suivez les instructions suivantes :

- Téléchargez `BitKeeper` à partir de http://www.bitmover.com/cgi−bin/download.cgi . Vous aurez besoin de `Bitkeeper` 2.0 ou supérieur pour accéder à notre dépôt.
- Suivez les instructions pour l'installer.
- Après avoir installé `BitKeeper` , commencez par vous déplacer dans le répertoire à partir duquel vous voulez travailler, et lancez l'une des commandes suivantes pour dupliquer la branche MySQL de votre choix :Pour dupliquer la branche 3.23, utilisez cette commande :
```
shell> bk clone bk://work.mysql.com:7000 mysql-3.23
```
  Pour dupliquer la branche 4.0, utilisez cette commande :
```
shell> bk clone bk://work.mysql.com:7001 mysql-4.0
```
  Pour dupliquer la branche 4.1, utilisez cette commande :
```
shell> bk clone bk://work.mysql.com:7004 mysql-4.1
```
  Pour dupliquer la branche 5.0, utilisez cette commande :
```
shell> bk clone bk://mysql.bkbits.net/mysql-5.0 mysql-5.0
```
  Dans l'exemple précédent, les sources seront respectivement placées dans les dossiers `mysql-3.23/` , `mysql-4.0/` , `mysql-4.1/` ou `mysql-5.0/` , de votre dossier courant.

  Si vous êtes derrière un firewall et que vous ne pouvez utiliser que des connexions HTTP, vous pouvez aussi accéder à `BitKeeper` via HTTP.

  Si vous devez utiliser un serveur proxy, assignez la variable d'environnement `http_proxy` pour qu'elle pointe sur votre proxy :

```
shell> export http_proxy="http://your.proxy.server:8080/"
```

Puis, remplacez le protocole `bk://` par `http://` lors de votre export. Par exemple :

```
shell> bk clone http://mysql.bkbits.net/mysql-4.1 mysql-4.1
```

Le premier téléchargement de l'arbre source peut prendre un certain temps, selon la vitesse de votre connexion. Soyez patients.

- Vous aurez besoin de GNU `make` , `autoconf` 2.53 (ou plus récent), `automake` 1.5, `libtool` 1.4 et `m4` pour lancer la prochaine série de commandes. Même si la plupart des systèmes d'exploitation sont livrés avec leur propre implémentation de `make` , les chances sont fortes pour que la compilation échoue avec des messages d'erreur étranges. Par conséquent, il est fortement recommandé d'utiliser GNU `make` (parfois aussi appelé `gmake`).Heureusement, d'autres systèmes d'exploitation sont livrés avec les utilitaires GNU, ou propose des paquets facilement installables. Dans tous les cas, vous pouvez les télécharger sur ces sites :
    - http://www.gnu.org/software/autoconf/
    - http://www.gnu.org/software/automake/
    - http://www.gnu.org/software/libtool/
    - http://www.gnu.org/software/m4/
    - http://www.gnu.org/software/make/

  Si vous essayez ded configurer MySQL 4.1 ou plus récent, vous aurez besoin de GNU `bison` 1.75 ou plus récent. Les anciennes versions de `bison` peuvent indiquer cette erreur :

```
sql_yacc.yy:#: fatal error: maximum table size (32767) exceeded
```

  Note : la taille maximale de la table n'est pas dépassée. L'erreur est causée par un bug dans les versions plus anciennes de `bison` .Les versions de MySQL avant la version 4.1 peuvent aussi se compiler avec d'autres versions de `yacc` (par exemple, BSD `yacc` 91.7.30). Pour les versions plus récentes, GNU `bison` est une obligation.

  Les commandes typiques nécessaires pour compiler MySQL sont présentées ci−dessous. La première commande `cd` change le dossier de travail : remplacez `mysql-4.0` avec le bon nom de dossier.

```
shell> cd mysql-4.0
shell> bk -r get -Sq
shell> aclocal; autoheader; autoconf;  automake;
shell> ./configure  # Ajoutez ici vos options favorites
shell> make
```

  Les lignes de commande qui passent dans les dossiers `innobase` et `bdb/dist` sont utilisées pour configurer `InnoDB` et Berkeley DB (`BDB`). Vous pouvez omettre ces lignes si vous n'avez pas besoin du support `InnoDB` ou `BDB` .

  Si vous obtenez des erreurs étranges pendant cette étape, vérifiez bien que vous avez vraiment installé `libtool` !

  Une collection de nos scripts de configuration les plus courants de trouve dans le sous−répertoire `BUILD/` . Si vous êtes fainéants, vous pouvez utiliser `BUILD/compile-pentium-debug` . Pour compiler sur une architecture différente, modifiez ce script en enlevant les drapeaux spécifiques au Pentium.
- Quand la compilation est achevée, lancez `make install` . Prenez garde sur des machines de production. Cette commande pourrait écraser votre installation actuelle. Si vous avez une autre installation de MySQL, nous vous recommandons de lancer `./configure` avec des valeurs des options `prefix` , `with-tcp-port` , et `unix-socket-path` différentes de celles de votre serveur de production.
- Torturez votre nouvelle installation et tentez de faire planter les nouvelles fonctionnalités. Commencez par lancer `make test` . Suites de tests MySQL .
- Si vous avez échoué avec l'étape `make` et que la distribution ne compile pas, envoyez un rapport sur le site http://bugs.mysql.com/ . Si vous avez installé la dernière version des indispensables outils GNU, et qu'ils échouent dans l'analyse de vos fichiers de configuration, envoyez aussi un rapport. D'autre part, si vous exécutez `aclocal` et que vous obtenez l'erreur

`command not found` ou un problème du même type, n'envoyez pas de rapport. A la place, assurez vous que les outils nécessaires sont bien installés et que votre variable `PATH` est configurée de telle façon que votre interpréteur de commandes les trouvent.
- Après la première opération `bk clone` pour obtenir l'arbre source, vous devez lancer régulièrement `bk pull` pour obtenir les mises à jour.
- Vous pouvez examiner l'historique des changements de l'arbre avec toutes les différences en utilisant `bk sccstool` . Si vous apercevez des différences anormales ou sur lesquelles vous avez des questions, n'hésitez pas à envoyer un e−mail aux listes internes. Les listes de diffusion MySQL . De même, si vous pensez avoir une meilleure méthode pour traiter un problème, envoyez un e−mail accompagné d'un patch à la même adresse. `bk diffs` vous fournira un patch après que vous ayez fait vos changements aux sources. Si vous n'avez pas le temps de coder votre idée, envoyez en juste une description.
- `BitKeeper` dispose d'une aide agréable à laquelle vous pouvez accéder via `bk helptool` .
- Veuillez noter que chaque commit (`bk ci` ou `bk citool`) postera un message avec un aperçu des changements à notre liste de diffusion interne, à la façon habituelle des propositions openlogging.org avec seulement les commentaires des changements. Généralement, vous n'aurez pas besoin d'utiliser commit (l'arbre public interdisant les `bk push`), mais plutôt d'utiliser la méthode `bk diffs` décrite plus haut.

Vous pouvez aussi naviguer dans les fichiers d'historiques, les commentaires et le code source en ligne. Par exemple, pour lire ses informations pour MySQL 4.1, allez à http://mysql.bkbits.net:8080/mysql−4.1 .

Le manuel est dans un module séparé, qui peut être obtenu comme ceci :

```
shell> bk clone bk://mysql.bkbits.net/mysqldoc mysqldoc
```

Il y a aussi des arbres BitKeeper pour `MySQL Control Center` et `Connector/ODBC` . Ils sont disponibles comme ceci :Pour obtenir le `MySQL Control Center` , utilisez cette commande :

```
shell> bk clone http://mysql.bkbits.net/mysqlcc mysqlcc
```

Pour obtenir le `Connector/ODBC` , utilisez cette commande :

```
shell> bk clone http://mysql.bkbits.net/myodbc3 myodbc3
```

2.3.4 Problèmes de compilation?

Tous les programmes MySQL compilent proprement chez nous, sans aucune alerte sur Solaris avec `gcc` . Sur d'autres systèmes, des alertes peuvent apparaître à cause de différences dans le système d'inclusions. Voyez Remarques sur MIT−pthreads pour les alertes qui peuvent apparaître avec `MIT-pthreads` . Pour d'autres problèmes, voyez la liste suivante.

La solution à de nombreux problèmes implique une nouvelle configuration. Si vous avez besoin de refaire une configuration voici quelques conseils généraux :

- Si `configure` est exécuté après une première exécution, il peut utiliser des informations qui ont été rassemblées durant une première invocation. Ces informations sont stockées dans le fichier `config.cache` . Lorsque `configure` est lancé, il commence par regarder dans ce fichier, et lire le contenu qui existe, en supposant que ces données sont toujours correctes. Cette supposition est invalide si vous faîtes une reconfiguration.
- Chaque fois que vous exécutez `configure` , vous devez exécuter à nouveau `make` pour recompiler. Toutefois, vous devrez peut être supprimer les vieux fichiers d'objets qui ont été compilé en utilisant différentes configurations précédentes.

Pour éviter d'utiliser de vieilles informations de configuration, ou des vieux fichiers d'objet, vous pouvez utiliser ces commandes, avant `configure` :

```
 shell> rm config.cache
shell> make clean
```

Alternativement, vous pouvez aussi utiliser `make distclean` .La liste suivante décrit certains problèmes lors de la compilation de MySQL, qui surviennent souvent :

- Si vous avez des problèmes lors de la compilation de `sql_yacc.cc` , comme ceux qui sont décrits ci−dessous, vous avez probablement été à court de mémoire ou d'espace de swap :

```
 Internal compiler error: program cc1plus got fatal signal 11
Out of virtual memory
Virtual memory exhausted
```

  Le problème est que `gcc` requiert de grandes quantité de mémoire pour compiler `sql_yacc.cc` avec les options `inline` . Essayez d'exécuter `configure` avec l'option `--with-low-memory` :

```
 shell> ./configure --with-low-memory
```

  Cette option ajoute `-fno-inline` dans la ligne de compilation, si vous utilisez `gcc` et `-O0` si vous utilisez autre chose. Vous pouvez essayer `--with-low-memory` même si il vous reste suffisamment de mémoire, et que vous ne pensez pas être limité. Ce problème a été observé sur des systèmes avec de généreuses configurations, et `--with-low-memory` résout ce problème.
- Par défaut, `configure` choisit `c++` comme compilateur, et GNU `c++` pour les liens avec `-lg++` . Si vous utilisez `gcc` , ce comportement peut poser les problèmes suivants :

```
 configure: error: installation or configuration problem:
C++ compiler cannot create executables.
```

  Vous pourriez aussi observer des problèmes durant la compilation, avec `g++` , `libg++` ou `libstdc++` .La cause de ces problèmes est que vous avez peut être que vous n'avez pas `g++` , ou que vous avez `g++` mais pas `libg++` , ou `libstdc++` . Regardez le fichier de log `config.log` . Il va sûrement contenir la raison exacte du mauvais fonctionnement de votre compilateur. Pour contourner ce problème, vous pouvez utiliser `gcc` comme compilateur C++. Essayez de modifier la variable d'environnement `CXX` avec la valeur "`gcc -O3`" . Par exemple :

```
 shell> CXX="gcc -O3" ./configure
```

  Cela fonctionne car `gcc` compile les sources C++ aussi bien que `g++` , mais il n'est pas lié avec `libg++` ou `libstdc++` par défaut.Un autre moyen pour régler ces problèmes, bien sur, est d'installer `g++` , `libg++` et `libstdc++` . Nous vous recommandons toutefois de ne pas utiliser `libg++` ou `libstdc++` avec MySQL car cela ne fera qu'accroître la taille de votre exécutable binaire, sans vous apporter d'avantages. Par le passé, certaines versions de ces bibliothèques ont posé des problèmes étranges aux utilisateurs MySQL.

  Utiliser `gcc` comme compilateur C++ est aussi nécessaire, si vous voulez compiler MySQL avec le support de RAID (voyez Syntaxe de `CREATE TABLE` pour plus d'information sur le type de table RAID), ou utilisez GNU `gcc` version 3 plus récent. Si vous avez des erreurs de compilation comme celles ci−dessous avec l'option `--with-raid` , essayez d'utiliser `gcc` comme compilateur C++ en définissant la variable d'environnement `CXX` ci−dessus :

```
 gcc -O3 -DDBUG_OFF -rdynamic -o isamchk isamchk.o sort.o  libnisam.a
../mysys/libmysys.a ../dbug/libdbug.a ../strings/libmystrings.a
 -lpthread -lz -lcrypt -lnsl -lm -lpthread
../mysys/libmysys.a(raid.o)(.text+0x79): In function
`my_raid_create'':: undefined reference to `operator new(unsigned)'
../mysys/libmysys.a(raid.o)(.text+0xdd): In function
`my_raid_create'':: undefined reference to `operator delete(void*)'
../mysys/libmysys.a(raid.o)(.text+0x129): In function
`my_raid_open'':: undefined reference to `operator new(unsigned)'
../mysys/libmysys.a(raid.o)(.text+0x189): In function
`my_raid_open'':: undefined reference to `operator delete(void*)'
../mysys/libmysys.a(raid.o)(.text+0x64b): In function
`my_raid_close'':: undefined reference to `operator delete(void*)'
collect2: ld returned 1 exit status
```

- Si votre compilation échoue avec des erreurs, ou si l'une des erreurs suivantes apparaît, vous devez changer la version de `make` en GNU `make` :

```
 making all in mit-pthreads
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
```

  ou :

```
 make: file `Makefile' line 18: Must be a separator (:
```

  ou :

```
 pthread.h: No such file or directory
```

  Solaris et FreeBSD sont connus pour avoir des problèmes avec `make` .GNU `make` version 3.75

est reconnu pour fonctionner.

- Si vous voulez définir des options supplémentaires qui seront utilisées par votre compilateur C ou C++, faites le en ajoutant ces options aux variables d'environnement `CFLAGS` et `CXXFLAGS` . Vous pouvez aussi spécifier le nom du compilateur via les variables `CC` et `CXX` . Par exemple :

```
 shell> CC=gcc
shell> CFLAGS=-O3
shell> CXX=gcc
shell> CXXFLAGS=-O3
shell> export CC CFLAGS CXX CXXFLAGS
```

  Voyez Les compiliations de MySQL par MySQL AB , pour avoir une liste des définitions des options disponibles sur divers systèmes.
- Si vous obtenez un message d'erreur comme celui−ci, vous devrez mettre à jour votre version de `gcc` :

```
 client/libmysql.c:273: parse error before `__attribute__'
```

  `gcc` 2.8.1 est connu pour fonctionner, mais nous recommandons l'utilisation de `gcc` 2.95.2 ou `egcs` 1.0.3a.
- Si vous obtenez des erreurs telles que celles qui sont affichées ci−dessous lors de la compilation de `mysqld` , c'est que `configure` n'a pas correctement détecter le dernier argument des fonctions `accept()` , `getsockname()` ou `getpeername()` :

```
 cxx: Error: mysqld.cc, line 645: In this statement, the referenced
     type of the pointer value "&length" is "unsigned long", which
     is not compatible with "int".
new_sock = accept(sock, (struct sockaddr *)&cAddr, &length);
```

  Pour corriger ce problème, éditez le fichier `config.h` (qui est généré par le fichier `configure`). Recherchez ces lignes :

```
 /* Define as the base type of the last arg to accept */
#define SOCKET_SIZE_TYPE XXX
```

  Remplacez `XXX` par `size_t` ou `int` , suivant votre système d'exploitation. Notez que vous devrez faire cette manipulation à chaque fois que vous exécuterez le script `configure` car `configure` regénére `config.h` .
- Le fichier `sql_yacc.cc` est généré à partir du fichier `sql_yacc.yy` . Normalement, le processus de création ne s'occupe pas de `sql_yacc.cc` , car MySQL en a déjà une copie. Cependant, si vous devez le recréer, vous pouvez rencontrer cette erreur :

```
 "sql_yacc.yy", line xxx fatal: default action causes potential...
```

  Cela indique que votre version de `yacc` est inadéquate. Vous devrez probablement réinstaller `bison` (la version GNU de `yacc`) et l'utiliser à la place.
- Sur Debian Linux 3.0, vous devez installer `gawk` au lieu du programme par défaut `mawk` si vous voulez compiler MySQL 4.1 ou plus récent avec le support `Berkeley DB` .
- Si vous avez besoin de déboguer `mysqld` ou un client MySQL, exécutez le script `configure` avec l'option `--with-debug` , puis recompilez vos clients avec la nouvelle bibliothèque. Déboguer un client MySQL .
- Si vous rencontrez une erreur de compilation sous Linux (e.g. SuSE Linux 8.1 ou Red Hat Linux 7.3) similaire à celle−ci :

```
 libmysql.c:1329: warning: passing arg 5 of `gethostbyname_r' from
incompatible pointer type
libmysql.c:1329: too few arguments to function `gethostbyname_r'
libmysql.c:1329: warning: assignment makes pointer from integer
without a cast
make[2]: *** [libmysql.lo] Error 1
```

  Par défaut, le script `configure` tente de déterminer le nombre correct d'argument en utilisant `g++` , le compilateur GNU C++. Ce test retourne des résultats erroné si `g++` n'est pas installé. Il y a deux façons de contourner le problème :
    - Assurez vous que GNU C++ `g++` est installé. Sur certains Linux, le paquet nécessaire est appelé `gpp` , et sur d'autres, c'est `gcc-c++` .
    - Utilisez `gcc` comme compilateur C++ en mettant donnant à la variable d'environnement `CXX` , la valeur de `gcc` :

    ```
     export CXX="gcc"
    ```

  Notez bien que vous devez lancer `configure` après cela.

2.3.5 Notes relatives aux MIT−pthreads

Cette section décrit quelques informations concernant l'utilisation des `MIT-pthreads` .

Notez que sur Linux vous **ne devez pas** utiliser les `MIT-pthreads` mais installer `LinuxThreads` ! Notes relatives à Linux (toutes les versions de Linux) .

Si votre système ne fournit pas un support natif des threads, vous aurez besoin de construire MySQL en utilisant le paquet des `MIT-pthreads` . Cela inclut les anciens systèmes FreeBSD, SunOS 4.x, Solaris 2.4 et plus ancien, et quelques autres systèmes. Plate−formes supportées par MySQL .

Notez qu'à partir de la version 4.0.2 de MySQL les `MIT-pthreads` ne font plus partie de la distribution des sources ! si vous avez besoin de ce paquet, vous pouvez l'obtenir sur http://www.mysql.com/Downloads/Contrib/pthreads−1_60_beta6−mysql.tar.gz

Après l'avoir récupéré, décompressez l'archive dans le répertoire racine de votre répertoire des sources de MySQL. Cela créera le répertoire `mit-pthreads` .

- Sur la plupart des systèmes, vous pouvez forcer l'utilisation des `MIT-pthreads` en exécutant `configure` avec l'option `--with-mit-threads` :
  ```
  shell> ./configure --with-mit-threads
  ```
  La compilation dans un dossier non−sources n'est pas supporté lors de l'utilisation des `MIT-pthreads` car nous voulons minimiser les changements de leur code.
- La vérification pour l'utilisation des `MIT-pthreads` ne survient que durant la partie du processus de configuration qui s'occupe du code du serveur. Si vous avez configuré la distribution en utilisant `--without-server` pour ne construire que le client, les clients ne sauront pas si les `MIT-pthreads` sont utilisés et utiliserons les socket Unix pour les connexions par défaut. Puisque les sockets Unix ne fonctionnent pas avec les `MIT-pthreads` sur certaines plate−formes, cela signifie que vous devrez utiliser `-h` ou `--host` quand vous exécuterez les programmes clients.
- Lorsque MySQL est compilé en utilisant les `MIT-pthreads` , le verrouillage système est désactivé par défaut pour des soucis de performances. Vous pouvez demander au serveur d'utiliser les verrous systèmes avec l'option `--external-locking` . Cela n'est requis que si vous avez besoin de faire fonctionner deux serveurs MySQL avec les mêmes données (non recommandé).
- De temps en temps, la commande `bind()` des pthreads n'arrive pas à attacher une socket sans afficher d'erreurs (du mois, sous Solaris). Le résultat est que toutes les connexions au serveur échouent. Par exemple :
  ```
   shell> mysqladmin version
  mysqladmin: connect to server at '' failed;
  error: 'Can't connect to mysql server on localhost (146)'
  ```
  La solution est de terminer le serveur `mysqld` et de le redémarrer. Cela ne nous est arrivé que quand nous avons forcé le serveur à se terminer et que nous l'avons redémarré immédiatement après.
- Avec les `MIT-pthreads` , l'appel système à `sleep()` ne peut pas être interrompu avec `SIGINT` (`break`). On ne s'en rend compte que quand on exécute `mysqladmin --sleep` . Vous devez attendre que l'appel système à `sleep()` se termine avant que le processus ne s'arrête.
- Lors de la liaison, vous pouvez obtenir des messages d'erreurs comme ceux−ci (du moins sur Solaris); ils peuvent être ignorés :
  ```
   ld: warning: symbol `_iob' has differing sizes:
      (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
  file /usr/lib/libc.so value=0x140);
      /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
  ld: warning: symbol `__iob' has differing sizes:
      (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
  file /usr/lib/libc.so value=0x140);
      /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
  ```
- D'autres avertissements peuvent être ignorés :
  ```
   implicit declaration of function `int strtoll(...)'
  implicit declaration of function `int strtoul(...)'
  ```

- Nous n'avons pas réussi à faire fonctionner readline avec les MIT−pthreads . (Cela n'est pas nécessaire, mais peut être utile à quelqu'un.)

2.3.6 La distribution source Windows

Ces instructions décrivent comment compiler MySQL version 4.1, depuis les sources, pour Windows. Les instructions sont fournies pour compiler les versions standards à partir des sources standards, ou depuis la version de développement de BitKeeper.

Note : :es instructions de ce document sont strictement destinées aux utilisateurs qui veulent tester MySQL sur Windows, à partir des toutes dernières sources de BitKeeper. Pour un serveur de production, MySQL vous recommande de ne pas compiler votre serveur vous−même. Normalement, il est mieux d'utiliser une distribution binaire précompilée, et optimisée pour l'utilisation sur Windows par MySQL AB. Les instructions d'installation pour les distributions binaires sont disponibles dans la section Installer MySQL sous Windows .

Pour compiler MySQL sur Windows depuis les sources, vous avez besoin des logiciels et ressources suivantes sur votre système :

- Le compilateur VC++ 6.0 (mis à jour avec le service pack 4 ou 5 , et paquet pre−processeur) Le paquet pre−processeur est nécessaire pour l'assembleur macro. Plus de détails à : http://msdn.microsoft.com/vstudio/downloads/updates/sp/vs6/sp5/faq.aspx .
- Environs 45 Mo d'espace disque.
- 64 Mo de RAM

Vous aurez besoin de la distribution source MySQL pour Windows. Il y a deux méthodes pour obtenir cette distribution pour MySQL 4.1 et supérieur :

- Télécharger une distribution source préparée par MySQL AB pour la version de MySQL que vous voulez. Les distributions sources pre−compilées sont disponibles pour les versions de MySQL publiées, et sont accessibles sur le site de http://www.mysql.com/downloads/ .
- Vous pouvez préparer votre propre distribution source vous−même, avec la dernière version disponible sur le serveur BitKeeper. Si vous voulez faire cela, vous devez créer la distribution sur un serveur Unix, et transférer l'archive sur votre système Windows. La raison est que certaines étapes de configuration requièrent des outils qui ne fonctionnent que sous Unix. L'approche BitKeeper requiert :
    - Un système fonctionnant sous Unix, ou un système équivalent, comme Linux.
    - BitKeeper 3.0 sur ce système. Vous pouvez obtenir BitKeeper sur http://www.bitkeeper.com/ .

Si vous utilisez une distribution source Windows vous pouvez passer directement à Compiler MySQL avec VC++ . Pour compiler depuis les sources BitKeeper, commencez par Créer une distribution MySQL pour Windows avec les dernières sources .

Si vous découvrez quelque chose qui ne fonctionne pas comme prévu, ou si vous avez des suggestions pour améliorer le processus actuel envoyez un message à la liste de diffusion dédiée à win32 . Les listes de diffusion MySQL .

2.3.6.1 Compiler MySQL avec VC++

Note: Les fichiers du projet VC++ de MySQL 4.1 et plus récent sont compatibles avec Microsoft Visual Studio 6.0 et plus récent (7.0/.NET) et sont testés par l'équipe de MySQL AB avant chaque publication.

Pour compiler MySQL, suivez ces instructions ci−dessous. Notez que les noms de menus sont laissés en anglais, et devront éventuellement être adaptés à votre interface.

- Créez un dossier de travail : par exemple, `workdir` .
- Décompressez la distribution source dans le dossier ci−dessus, en utilisant `WinZip` ou un autre utilitaire Windows qui sache lire les fichiers `.zip` .
- Lancez le compilateur VC++ 6.0.
- Dans le menu `File` , sélectionnez `Open Workspace` .
- Ouvrez le fichier `mysql.dsw` que vous trouverez dans le dossier de travail.
- Dans le menu `Build` , sélectionnez le menu `Set Active Configuration` .
- Cliquez dans le dialogue pour sélectionner `mysqld - Win32 Debug` , puis cliquez sur OK.
- Pressez `F7` pour lancer la compilation du serveur de débogage, des bibliothèques et des applications clients.
- Compilez la version que vous souhaitez, de la même façon.
- Les versions de débogage et les bibliothèques sont placées dans les dossiers `client_debug` et `lib_debug` . Les versions finales des programmes et bibliothèques sont placées dans les dossiers `client_release` et `lib_release` . Notez que si vous voulez compiler les versions de débogage et finales, vous pouvez utiliser l'option `` `build all` '' dans le menu `Build` .
- Testez le serveur. Le serveur compilé avec les instructions suivantes suppose que le dossier de base MySQL et le dossier de données sont situés dans les dossiers `C:\mysql` et `C:\mysql\data` , par défaut. Si vous voulez tester votre serveur, utilisez le chemin de votre dossier d'installation comme chemin racine. Vous pouvez faire cela en ligne de commande, avec les options `--basedir` et `--datadir` ou bien placez les bonnes options dans le fichier d'options (`C:\my.cnf` ou `my.ini` de votre dossier Windows). Si vous avez un dossier de données sur votre disque, vous pouvez spécifier son chemin.
- Lancez le serveur depuis le dossier `client_release` ou `client_debug` , suivant la version que vous voulez utiliser. Les instructions générales de lancement sont dans Installer MySQL sous Windows . Vous devrez adapter les instructions à votre configuration, si vous avez un dossier de base ou de données différents.
- Lorsque le serveur fonctionne en mode indépendant ou comme un service, suivant votre configuration, essayez de vous connecter avec le client interactif `mysql` , qui est placé dans le dossier `client_release` ou `client_debug` .

Lorsque vous êtes satisfait du fonctionnement de votre serveur, stoppez le. Puis installez MySQL comme ceci :

- Créez un dossier sur votre disque pour installer MySQL. Par exemple, nous pourrions l'installer dans `C:\mysql` . Voici les commandes :

  ```
  C:
  mkdir \mysql
  mkdir \mysql\bin
  mkdir \mysql\data
  mkdir \mysql\share
  mkdir \mysql\scripts
  ```

  Si vous voulez compiler d'autres clients et les lier à MySQL, il faudra créer d'autres dossiers :

  ```
  mkdir \mysql\include
  mkdir \mysql\lib
  mkdir \mysql\lib\debug
  mkdir \mysql\lib\opt
  ```

  Si vous voulez tester les performances de MySQL, créez ce dossier :

  ```
  mkdir \mysql\sql-bench
  ```

  Les tests de performances requièrent Perl.
- Depuis le dossier `workdir` , copiez dans le dossier `C:\mysql` les fichiers suivants :

  ```
  copy client_release\*.exe C:\mysql\bin
  copy client_debug\mysqld.exe C:\mysql\bin\mysqld-debug.exe
  xcopy scripts\*.* C:\mysql\scripts /E
  xcopy share\*.* C:\mysql\share /E
  ```

Si vous voulez compiler d'autres clients, et les lier avec MySQL, vous devrez aussi faire ceci :

```
copy lib_debug\mysqlclient.lib C:\mysql\lib\debug
copy lib_debug\libmysql.* C:\mysql\lib\debug
copy lib_debug\zlib.* C:\mysql\lib\debug
copy lib_release\mysqlclient.lib C:\mysql\lib\opt
copy lib_release\libmysql.* C:\mysql\lib\opt
copy lib_release\zlib.* C:\mysql\lib\opt
copy include\*.h C:\mysql\include
copy libmysql\libmysql.def C:\mysql\include
```

Si vous voulez tester les performances de MySQL, faites aussi ceci :

```
xcopy sql-bench\*.* C:\mysql\bench /E
```

Puis lancez votre serveur de la même façon que vous lancez l'exécutable MySQL classique. Installer MySQL sous Windows .

2.3.6.2 Compiler MySQL sur Windows à partir des sources de développement

Pour compiler la dernière version Windows à partir de sources disponibles dans le serveur BitKeeper, suivez les instructions suivantes. Notez que ces commandes doivent être exécutés sur un système fonctionnant sous Unix ou Linux. La procédure fonctionne très bien sous Linux, par exemple.

- Clonez les source issues de BitKeeper (version 4.1 ou plus récent). Pour plus d'informations sur comment cloner les sources sont disponibles dans la section Installer à partir des sources de l'arbre de développement .
- Configurez et compilez la distribution pour obtenir un serveur fonctionnel. Pour cela, vous pouvez lancez la commande suivante à la racine de vos sources :
  ```
  shell> ./BUILD/compile-pentium-max
  ```
- Après vous être assurés que le processus est complet et réussi, lancez l'utilitaire suivant depuis la racine de vos sources :
  ```
  shell> ./scripts/make_win_src_distribution
  ```
  Ce script crée un paquet source Windows, qui peut être utilisé sur votre système. Vous pouvez passer d'autres options à ce script, suivant vos besoins. Il accepte les options suivantes : Par défaut, `make_win_src_distribution` crée une archive zippée avec le nom `mysql-VERSION-win-src.zip` , où `VERSION` représente la version de votre source MySQL.
- Copiez ou téléchargez le paquet sur votre machine Windows. Pour le compiler, suivez les instructions de la section Compiler MySQL avec VC++ .

2.3.7 Compiler les clients MySQL sous Windows

Dans vos fichiers sources, vous devez inclure `windows.h` avant `mysql.h` :

```
#if defined(_WIN32) || defined(_WIN64)
#include <windows.h>
#endif
#include <mysql.h>
```

`my_global.h` inclut tous les autres fichiers nécessaires pour Windows (comme le fichier `windows.h`) si vous compilez votre programme sous Windows.Vous pouvez soit lier votre code avec la bibliothèque dynamique `libmysql.lib` , qui est juste une interface pour charger `libmysql.dll` à la demande, soit lier avec la bibliothèque statique `mysqlclient.lib` .

Notez que puisque les bibliothèques `mysqlclient` sont compilées en tant que bibliothèques threadées, vous devez aussi compiler votre code pour qu'il soit multi−threadé !

2.4 Procédure de post−installation

Il y a des manipulations importantes à faire après avoir installé MySQL. Par exemple, sous Unix, vous devez créer les tables de droits. Sur toutes les plate−forme, un point de sécurité important est que les comptes initiaux n'ont pas de mot de passe. Vous devez assigner les mots de passe pour éviter un accès indu au serveur MySQL.

Les sections suivantes décrivent les procédures de post installation sur Windows et pour les systèmes Unix. Une autre section, Lancer et résoudre des problèmes avec MySQL s'applique aux autres plate−formes : elle décrit ce que vous devez faire si vous avez des problèmes de lancement. La section Sécurisation des comptes MySQL initiaux s'applique aussi à toutes les plate−formes. Vous devez suivre les instructions pour vous assurer que vous avez bien protégé vos comptes MySQL en leur assignant un mot de passe.

Lorsque vous êtes prêts à créer d'autres comptes, vous pouvez trouver des informations sur le contrôle d'accès à MySQL et la gestion de comptes dans les sections Sécurité générale et accès à MySQL et Gestion des comptes d'utilisateur MySQL .

2.4.1 Post−installation sous Windows

Sous Windows, la table de droits n'a pas besoin d'être créée. Les distributions MySQL pour Windows incluent les tables de droits pré−configurées dans la base `mysql` , dans le dossier de données `data` . Cependant, vous devez assigner des mots de passe aux comptes.

Avant de donner des mots de passe aux comptes, vérifiez que le serveur fonctionne avec un client. Assurez vous que le serveur fonctionne (Lancer MySQL pour la première fois), puis utilisez les commandes suivantes pour vérifier que vous pouvez lire des données sur le serveur. Lr résultat doit être proche de celui présenté ici :

```
 C:\> C:\mysql\bin\mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
| test      |
+-----------+

 C:\> C:\mysql\bin\mysqlshow mysql
Database: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+

 C:\> C:\mysql\bin\mysql -e "SELECT Host,Db,User FROM db" mysql
+------+-------+------+
| host | db    | user |
+------+-------+------+
| %    | test% |      |
+------+-------+------+
```

Si vous avez une version fonctionnelle de Windows qui supporte les services, et que vous voulez que MySQL fonctionne automatiquement au lancement de Windows, voyez la section Lancer MySQL comme un service Windows .

2.4.2 Procédures de post−installation sous Unix

Une fois que vous avez installé MySQL sur Unix, vous devez initialiser les tables de droits, lancer le serveur, et vous assurer que tout fonctionne bien. Vous pouvez aussi configurer le démarrage et l'extinction automatique du serveur, lorsque votre serveur se lance et s'arrête.

Sous Unix, les tables de droits sont configurées avec le programme `mysql_install_db` . Pour certaines méthodes d'installation, ce programme est utilisé automatiquement :

- Si vous installez MySQL sous Linux avec un `RPM` , le serveur `RPM` exécute `mysql_install_db` .
- Si vous installez MySQL sous Mac OS X en utilisant la distribution `PKG` , l'installeur exécute `mysql_install_db` .

Sinon, vous devez lancer manuellement `mysql_install_db` .

La procédure suivante décrit comment initialiser les tables de droits (si cela n'a pas été fait), puis comment lancer le serveur. Vous trouverez aussi des suggestions de commandes pour tester l'accessibilité du serveur. Pour des informations sur le démarrage et l'extinction automatique du serveur, voyez Lancer et arrêter automatiquement MYSQL .

Une fois que vous avez exécuté la procédure, et que le serveur fonctionne, vous devez assigner des mots de passe aux comptes créés par `mysql_install_db` . Les instructions pour faire cela Sécurisation des comptes MySQL initiaux .

Dans les exemples ici, le serveur fonctionne avec l'utilisateur `mysql` . On suppose donc que ce compte existe. Créez ce compte, s'il n'existe pas, ou bien utilisez le nom que vous avez choisi.

- Changez de dossier de travail, pour vous mettre à la racine de l'installation MySQL, représentée ici par `BASEDIR` :

  ```
  shell> cd BASEDIR
  ```

  `BASEDIR` vaut probablement `/usr/local/mysql` ou `/usr/local` . Les étapes suivantes supposent que vous êtes dans ce dossier.
- Si nécessaire, lancez le programme `mysql_install_db` pour configurer les tables de droits initiales, qui déterminent les utilisateurs qui sont autorisées à se connecter au serveur. Vous devez faire cela si vous avez installé le programme avec une distribution qui ne lance pas ce programme pour vous.Typiquement, `mysql_install_db` doit être utilisé uniquement à la première installation, et vous pouvez éviter cette étape si vous faites une mise à jour. Cependant, `mysql_install_db` n'écrase pas les tables de droits : vous pouvez l'utiliser, en cas de doute.

  Pour initialiser les tables de droits, utilisez une des commandes suivantes, en fonction de la localisation de `mysql_install_db` dans le dossier `bin scripts` :

  ```
  shell> bin/mysql_install_db --user=mysql
  shell> scripts/mysql_install_db --user=mysql
  ```

  Le script `mysql_install_db` crée la base `mysql` qui contient les tables de droits, et la base `test` que vous pouvez utiliser pour les tests avec MySQL. Ce script va aussi créer l'utilisateur `root` et un compte anonyme. Ces deux comptes sont créés sans mot de passe. Une description des droits initiaux sont présentés dans la section Sécurisation des comptes MySQL initiaux . Le script `mysqld_safe` lance le serveur `mysqld` . Avant la version 4.0, utilisez `safe_mysqld` au lieu de `mysqld_safe` .

  Il est important de vous assurer que les dossiers et les fichiers appartiennent au compte `mysql` pour que le serveur puisse lire et écrire dedans. Pour cela, l'option `--user` doit être utilisée comme présenté si vous utilisez `mysql_install_db` comme `root` . Sinon, il est recommandé d'exécuter le script lorsque vous êtes connectés en tant que `mysql` : dans ce cas, vous pouvez omettre l'option `--user` .

`mysql_install_db` crée de nombreuses tables dans la base `mysql` : `user` , `db` , `host` , `tables_priv` , `columns_priv` , `func` , et même d'autres, en fonction des versions de MySQL.

Si vous ne voulez pas de la base `test` , vous pouvez la supprimer avec `mysqladmin -u root drop test` au redémarrage du serveur.

Si vous avez des problèmes avec `mysql_install_db` , voyez Problèmes avec `mysql_install_db` .

Il y a d'autres alternatives pour lancer `mysql_install_db` tel que fournit par la distribution MySQL :

- Si vous voulez que les droits initiaux soient différents des valeurs par défaut, vous pouvez modifier le script `mysql_install_db` avant de l'exécuter. Cependant, une technique préférable est d'utiliser `GRANT` et `REVOKE` pour changer les droits après avoir configuré les tables de droits. En d'autres termes, vous pouvez lancer `mysql_install_db` , puis utiliser `mysql -u root mysql` pour vous connecter au serveur en tant que `root` MySQL, pour émettre les commandes `GRANT` et `REVOKE` .Si vous voulez installer MySQL sur de nombreuses machines avec les mêmes droits, vous pouvez mettre les commandes `GRANT` et `REVOKE` dans un fichier de script, et exécuter ce script avec `mysql` après avoir utilisé `mysql_install_db` . Par exemple :

  ```
   shell> bin/mysql_install_db --user=mysql
  shell> bin/mysql -u root < your_script_file
  ```

  Comme cela, vous pouvez éviter les problèmes de répétition manuelle entre chaque machine.
- Il est possible de re−créer entièrement les tables de droits après les avoir créées. Vous pouvez le faire si vous apprenez comment utiliser `GRANT` et `REVOKE` et que vous avez fait tellement de modifications après `mysql_install_db` que vous voulez recommencer à zéro.

  Pour re−créer les tables de droits, supprimez les fichiers `.frm` , `.MYI` et `.MYD` dans le dossier contenant les tables `mysql` . C'est le dossier appelé `mysql` dans le dossier de données, qui est listé dans le dossier `datadir` lorsque vous utilisez la commande `mysqld --help` . Puis, utilisez à nouveau le script `mysql_install_db` .

  Note : pour les versions MySQL antérieure à la version 3.22.10, vous ne devez pas supprimer les fichiers `.frm` . Si vous les supprimez accidentellement, essayez de les retrouver et de les remettre dans le dossier `mysql` depuis votre distribution MySQL, avant d'utiliser `mysql_install_db` .
- Vous pouvez lancer `mysqld` manuellement en utilisant l'option `--skip-grant-tables` et en ajoutant les droits `mysql` :

  ```
   shell> bin/mysqld_safe --user=mysql --skip-grant-tables &
  shell> bin/mysql mysql
  ```

  Depuis `mysql` , exécutez manuellement les commandes SQL contenues dans le script `mysql_install_db` . Assurez vous que vous utilisez `mysqladmin flush-privileges` ou `mysqladmin reload` après, pour dire au serveur de relire les tables de droits.Notez que si vous n'utilisez pas `mysql_install_db` , vous devez remplir les tables manuellement, et en plus, vous devez commencer par les créer.

- Lancez le serveur `MySQL` :

```
 shell> bin/mysqld_safe --user=mysql &
```

Pour les versions de MySQL antérieure à 4.0, remplacez `bin/safe_mysqld` par `bin/mysqld_safe` dans cette commande.Il est important de vous assurez que le dossier de base de données et les fichiers de `mysql` sont accessibles, pour que le serveur puisse y lire et écrire. Pour cela, l'option `--user` peut être utilisée avec `mysql_install_db` si vous l'exécutez en tant que `root` . Sinon, vous devez exécuter le script lorsque vous êtes identifié comme `mysql` , auquel cas, vous pouvez omettre l'option `--user` .

D'autres instructions pour faire fonctionner MySQL en tant qu'utilisateur sans droits sont données dans la section Comment faire tourner MySQL en tant qu'utilisateur normal .

Si vous n'avez pas créé les tables de droits, l'erreur suivante sera inscrite dans le fichier de log d'erreur de votre serveur :

```
mysqld: Can't find file: 'host.frm'
```

Si vous avez des problèmes au démarrage du serveur, voyez Lancer et résoudre des problèmes avec MySQL .

- Utilisez `mysqladmin` pour vérifier que le serveur fonctionne. La commande suivante vous permet de faire un test simple pour vérifier que le serveur est actif et qu'il répond aux connexions :

```
shell> bin/mysqladmin version
shell> bin/mysqladmin variables
```

Le résultat de `mysqladmin version` varie légèrement, suivant votre plates−formes et votre version de MySQL, mais il doit être proche de ceci :

```
shell> bin/mysqladmin version
mysqladmin  Ver 8.40 Distrib 4.0.18, for linux on i586
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license
 Server version          4.0.18-log
Protocol version        10
Connection              Localhost via Unix socket
TCP port                3306
UNIX socket             /tmp/mysql.sock
Uptime:                 16 sec

 Threads: 1  Questions: 9  Slow queries: 0
Opens: 7  Flush tables: 2  Open tables: 0
Queries per second avg: 0.000
Memory in use: 132K  Max memory used: 16773K
```

Pour voir ce que vous pouvez faire d'autre avec `mysqladmin` , utilisez l'option `--help` .

- Vérifiez que vous pouvez éteindre le serveur :

```
shell> BINDIR/mysqladmin -u root shutdown
```

- Vérifiez que vous pouvez relancer le serveur. Pour cela, utilisez `mysqld_safe` ou `mysqld` directement. Par exemple :

```
shell> BINDIR/mysqld_safe --log &
```

Si `mysqld_safe` échoue, essayez de l'exécuter directement depuis le dossier d'installation MySQL (si vous n'y êtes pas déjà). Si cela ne fonctionne toujours pas, voyez Lancer et résoudre des problèmes avec MySQL .

- Exécutez les tests simples pour vérifier que le serveur fonctionne. Le résultat devrait être proche de celui−ci :

```
shell> BINDIR/mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
+-----------+

 shell> BINDIR/mysqlshow mysql
Database: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+

 shell> BINDIR/mysql -e "SELECT host,db,user FROM db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

Il y a aussi une suite de tests dans le dossier `sql-bench` (sous le dossier d'installation MySQL) que vous pouvez utiliser pour comparer les performances de MySQL sur différentes plates−formes. La suite de tests utilise le module Perl `DBI` pour assurer une interface avec les autres bases de données. Les modules Perl suivants sont aussi nécessaires pour exécuter la suite de tests :

```
DBI
DBD::mysql
Data::Dumper
Data::ShowTable
```

Ces modules sont disponibles sur la bibliothèque `CPAN` http://www.cpan.org/ . Installer Perl sur Unix .

Le dossier `sql-bench/Results` contient des résultats de tests sur différentes plates−formes et bases. Pour lancer tous les tests, exécutez ces commandes :

```
shell> cd sql-bench
shell> run-all-tests
```

Si vous n'avez pas de dossier `sql-bench` , vous avez probablement installé MySQL avec un fichier `RPM` , différent du `RPM` source. Le `RPM` source inclut le dossier `sql-bench`) Dans ce as, vous devez d'abord installer la suite de tests avant de l'utiliser. Depuis MySQL version 3.22, il y a des fichiers de tests `RPM` appelé `mysql-bench-VERSION-i386.rpm` , qui contiennent le code et les données.

Si vous avec une distribution source, il y a aussi des tests dans le sous−dossier `tests` . Par exemple, pour lancer `auto_increment.tst` , faîtes :

```
shell> BINDIR/mysql -vvf test < ./tests/auto_increment.tst
```

Les résultats attendus des tests sont disponibles dans le fichier `./tests/auto_increment.res` .

- A ce point, vous devez avoir un serveur fonctionnel. Cependant, les comptes initiaux n'ont pas de mot de passe : il est recommandé de leur assigner des mots de passe en suivant les instructions de la section Sécurisation des comptes MySQL initiaux .

2.4.2.1 Problèmes d'exécution de mysql_install_db

Le but du script `mysql_install_db` est de générer un nouveau système de droits pour MySQL. Il ne modifiera aucune autre donnée! Il ne fera rien du tout si vous avez des tables de droits installées.

Si vous voulez recréer vos tables de droits, vous devez éteindre le serveur `mysqld` , s'il fonctionnait. Puis, renommez le dossier `mysql` dans le dossier de données, sauvez le, et exécutez le script `mysql_install_db` . Par exemple :

```
shell> mv mysql-data-directory/mysql mysql-data-directory/mysql-old
shell> mysql_install_db --user=mysql
```

Cette section liste les problèmes que vous pourriez rencontrer lors de l'exécution du script `mysql_install_db` :

`mysql_install_db` *n'installe pas les tables de droits*
: Vous réalisez que `mysql_install_db` n'arrive pas à installer les tables de droits, et se termine sur ce message :

```
starting mysqld daemon with databases from XXXXXX
mysql daemon ended
```

Dans ce cas, examinez le fichier de log très attentivement! Le fichier de log est situé dans le dossier `XXXXXX` indiqué dans le message d'erreur, et il indiquera pourquoi `mysqld` n'a pas démarré. Si vous ne comprenez pas ce qui est arrivé, incluez le log dans votre message, lors de l'envoi du rapport de bugs avec `mysqlbug` ! Comment rapporter des bogues ou problèmes .

Un démon `mysqld` *fonctionne déjà*

Dans ce cas, vous n'avez probablement pas exécuté `mysql_install_db` du tout. Vous avez exécuté `mysql_install_db` une fois, lorsque vous avez installé MySQL pour la première fois.

Installer un second démon `mysqld` n'est pas possible lorsque le premier fonctionne.

Cela arrive lorsque vous avez une installation MySQL pre−existantes, mais que vous voulez installer une autre version ailleurs (par exemple, pour faire des tests ou simplement pour avoir deux installations). Généralement, le problème survient lorsque le second serveur est démarré, et qu'il essaie d'utiliser les mêmes ports et sockets que le premier. Dans ce cas, vous recevez des message d'erreur tels que :

```
Can't start server: Bind on TCP/IP port:
Address already in use
Can't start server: Bind on unix socket...
```

Pour des instructions sur la configuration de serveurs multiples, voyez la section Utiliser plusieurs serveurs sur la même machine .

`You don't have write access to /tmp`

Si vous n'avez pas les droits d'accès suffisant pour créer un fichier de socket à l'endroit prévu ou les permissions pour créer un fichier temporaire dans `/tmp,` vous allez avoir une erreur lors de l'utilisation de `mysql_install_db` ou avec `mysqld` .

Vous pouvez spécifier une socket différente et un dossier temporaire différent avec les options suivantes :

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

`some_tmp_dir` doit être le chemin complet d'un dossier dans lequel vous avez les droits en écriture.

Après cela, vous devriez être capable d'exécuter `mysql_install_db` et lancer le serveur avec ces commandes :

```
shell> bin/mysql_install_db --user=mysql
shell> bin/mysqld_safe --user=mysql &
```

Si `mysql_install_db` est situé dans le dossier `scripts` , modifiez la première commande pour utiliser `scripts/mysql_install_db` .

Voyez Comment protéger `/tmp/mysql.sock` . Variables d'environnement MySQL .

2.4.2.2 Lancer et arrêter MySQL automatiquement

Généralement, vous démarrez le serveur `mysqld` par l'un de ces moyens :

- En appelant `mysqld` directement. Cela fonctionne sur toutes les plates−formes.
- En lançant le serveur MySQL comme un service Windows. Cela fonctionne sur les versions de Windows qui supportent les services : comme Windows NT, 2000 et XP. Le service peut être configuré pour démarrer automatiquement au lancement de Windows, ou manuellement, à la demande. Pour des instructions, reportez vous à Lancer MySQL comme un service Windows .
- En appelant `mysqld_safe` , qui essaie de déterminer les options correctes avant de lancer `mysqld` . Ce script est utilisé sur les systèmes Unix BSD. Il est aussi appelé par `mysql.server` . `safe_mysqld` , le script père de `mysqld` .
- En appelant `mysql.server` . Ce script sert principalement au moment du démarrage et de l'extinction du système, sur les systèmes qui utilisent un dossier de processus programmés System V, où il est généralement enregistré sous le nom de `mysql` . Le script `mysql.server` lance le serveur en appelant `mysqld_safe` . Le script de démarrage `mysql.server` .

- Sur Mac OS X, vous pouvez installer un paquet indépendant appelé `MySQL Startup Item` pour activer le lancement automatique de MySQL au démarrage. Le `Startup Item` lance le serveur en appelant `mysql.server` . Voir Installer MySQL sous Mac OS X pour plus de détails.

Les scripts `mysql.server` et `safe_mysqld` et le `StartUp Item` de `Mac OS X` peuvent être utilisés pour démarrer le serveur automatiquement au moment du démarrage du serveur. `mysql.server` peut aussi servir à arrêter le serveur.

Le script `mysql.server` peut servir à démarrer ou arrêter le serveur en l'appelant avec les arguments `start` ou `stop` :

```
shell> mysql.server start
shell> mysql.server stop
```

Avant que `mysql.server` ne démarre le serveur, il change de dossier pour aller dans le dossier d'installation et appelle `safe_mysqld` . Si vous voulez que le serveur fonctionne sous un nom d'utilisateur spécifique, ajoutez l'option `user` appropriée dans le groupe `[mysqld]` du fichier `/etc/my.cnf` , tel que présenté ultérieurement dans cette section (il est possible que vous ayez besoin d'éditer `mysql.server`). Vous pourriez avoir à éditer `mysql.server` si vous avez une installation binaire dans une situation non standard. Modifiez la commande `cd` avec le dossier correct, avant qu'il n'exécute `safe_mysqld` . Si vous voulez que le serveur fonctionne avec un utilisateur spécifique, ajouter l'option `user` appropriée dans le fichier `/etc/my.cnf` , tel que présenté ultérieurement dans cette section.

`mysql.server stop` arrête le serveur en lui envoyant un signal. Vous pouvez éteindre le serveur manuellement avec la commande `mysqladmin shutdown` .

Pour lancer et arrêter automatiquement MYSQL sur votre serveur, vous devez ajouter les commandes de lancement et d'arrêt dans les bons endroits de vos fichiers `/etc/rc*` .

Notez que si vous utilisez des paquets Linux `RPM` (`MySQL-server-VERSION.rpm`), le script `mysql.server` est installé sous le nom `/etc/init.d/mysql` . Vous n'avez pas à l'installer manuellement. Voyez Installer MySQL sous Linux pour plus d'informations sur les paquets Linux `RPM` .

Certaines éditeurs fournissent des paquets `RPM` qui installent un script de démarrage sous un autre nom, comme `mysqld` .

Si vous installez MySQL depuis une distribution source, ou en utilisant une distribution binaire qui n'installe pas automatiquement le script `mysql.server` , vous pouvez l'installer manuellement. Le script est disponible dans le dossier `support-files` du dossier d'installation MySQL, ou dans le dossier source de MySQL.

Pour installer `mysql.server` manuellement, copiez le dans le dossier `/etc/init.d` sous le nom `mysql` , puis rendez−le exécutable. Pour cela, placez vous dans le dossier où `mysql.server` est stocké, et utilisez ces commandes :

```
shell> cp mysql.server /etc/init.d/mysql
shell> chmod +x /etc/init.d/mysql
```

Les anciens systèmes Red Hat utilisent le dossier `/etc/rc.d/init.d` plutôt que `/etc/init.d` . Adaptez les commandes précédentes. Alternativement, créez un lien symbolique `/etc/init.d` qui pointe sur `/etc/rc.d/init.d` :

```
shell> cd /etc
shell> ln -s rc.d/init.d .
```

Après installation du script, les commandes doivent être activées pour fonctionner au lancement du système, sur votre système d'exploitation. Sous Linux, vous pouvez utiliser `chkconfig` :

```
shell> chkconfig --add mysql
```

Sur certains systèmes Linux, les commandes suivantes sont aussi nécessaires pour activer totalement le script `mysql` :

```
shell> chkconfig --level 345 mysql on
```

Sous FreeBSD, les scripts de démarrage vont généralement dans le dossier `/usr/local/etc/rc.d/` . La page de manuel `rc(8)` indique que les scripts de ce dossier ne sont exécutés que si leur nom est de la forme `*.sh` . Tout autre fichier de ce dossier sera alors ignoré. En d'autres termes, vous devez installer le script `mysql.server` sous le nom `/usr/local/etc/rc.d/mysql.server.sh` pour activer le démarrage automatique.

Alternativement à la configuration précédente, certains systèmes d'exploitation utilisent aussi `/etc/rc.local` ou `/etc/init.d/boot.local` pour lancer des services supplémentaires au démarrage. Pour lancer MySQL avec cette méthode, vous pouvez ajouter une commande comme celle−ci au fichier de démarrage :

```
/bin/sh −c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'
```

Pour les autres systèmes, consultez la documentation de votre système d'exploitation pour savoir comment installer un script de démarrage.

Vous pouvez aussi ajouter des options à `mysql.server` via le fichier global `/etc/my.cnf` file. Un fichier `/etc/my.cnf` typique peut ressembler à ceci :

```
 [mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

 [mysql.server]
basedir=/usr/local/mysql
```

Le script `mysql.server` comprend les options suivantes : `basedir` , `datadir` et `pid-file` . Si spécifiées, elles **doivent** être placées dans un fichier d'option, et non pas en ligne de commande. `mysql.server` comprend les options de ligne de commande `start` et `stop` .

La table suivante montre quels groupes d'options chaque scripts de démarrage utilise :

Script	Groupe d'options
mysqld	mysqld et server
mysql.server	mysql.server , mysqld , et server
safe_mysqld	mysql.server , mysqld , et server

`[mysqld-major-version]` signifie que des groupes ayant des noms tels que `[mysqld-4.0]` , `[mysqld-4.1]` , et `[mysqld-5.0]` seront lus par les serveurs de versions 4.0.x, 4.1.x, 5.0.x, etc. Cette fonctionnalité a été ajoutée en MySQL 4.0.14. Elle sert à spécifier des options qui ne seront lues que par des serveurs dont les versions sauront les comprendre.

A des fins de compatibilité ascendante, `mysql.server` lit aussi le groupe d'options `[mysql_server]` et `mysqld_safe` lit le groupe d'options `[safe_mysqld]` . Cependant, il est recommandé de modifier vos fichiers de configuration pour utiliser les groupes `[mysql.server]` et `[mysqld_safe]` à la place.

Utiliser les fichiers d'options .

2.4.2.3 Problèmes de démarrage du serveur MySQL

Si vous avez des problèmes pour lancer le serveur, voici quelques pistes que vous pouvez essayer :

- Spécifiez toutes les options spéciales nécessaires aux moteurs de tables que vous utilisez.

- Assurez vous que le serveur sait où trouver le dossier de données.
- Assurez vous que le serveur peut utiliser le dossier de données. Le propriétaire et les droits du dossier de données et son contenu doivent être accessibles au serveur, en lecture et écriture.
- Vérifiez le log d'erreurs pour voir pourquoi le serveur ne démarre pas.
- Vérifiez que les interfaces réseau sont accessibles au serveur.

Certains moteurs de stockage ont des options qui contrôlent leur comportement. Vous devrez créer un fichier d'options `my.cnf` et y configurer celles des moteurs que vous voulez utiliser. Si vous allez utiliser des tables qui supportent les transactions (`InnoDB` , `BDB`), assurez vous qu'elles sont bien configurées comme vous le souhaitez.

- Si vous utilisez les tables `InnoDB` , voyez les options de démarrages spécifiques à `InnoDB` . En `MySQL` version 3.23, vous devez configurer `InnoDB` explicitement ou le serveur ne pourra pas démarrer. Depuis `MySQL` 4.0, `InnoDB` utiliser des valeurs par défaut pour sa configuration, si vous n'en spécifiez pas. Configuration `InnoDB` .
- Si vous utilisez les tables `BDB` (Berkeley DB), vous devez vous familiariser avec les différentes options spécifiques à `BDB` . Options de démarrage `BDB` .

Lorsque le démon `mysqld` démarre, il change le dossier de travail par le dossier de données. C'est là qu'il doit trouver les fichiers de log, et le fichier pid (ID de processus), ainsi que les dossiers de bases.

Le chemin du dossier de données est codé en dur lorsque la distribution est compilée. Cependant, si `mysqld` cherche le dossier de données ailleurs que là où il est vraiment, il ne va pas fonctionner correctement. Vous pouvez lire les chemins par défaut en invoquant `mysqld` avec l'option `--verbose` ou `--help` . Avant `MySQL` 4.1, omettez `--verbose` .

Si les valeurs par défaut ne correspondent pas à votre installation `MySQL` , vous pouvez les modifier en spécifiant des options de ligne de commande pour `mysqld` et `mysqld_safe` . Vous pouvez aussi lister les options dans un fichier d'options.

Pour spécifier la localisation du dossier de données explicitement, utilisez l'option `--datadir` . Cependant, vous pouvez spécifier à `mysqld` le chemin du dossier de base sous lequel `MySQL` est installé, et il va rechercher le dossier de données là. Vous pouvez faire cela avec l'option `--basedir` .

Pour vérifier l'effet de ces options, appelez `mysqld` avec ces options, suivies de `--verbose` et `--help` . Par exemple, si vous modifiez le chemin pour celui dans lequel `mysqld` est installé, alors vous pouvez utiliser la commande suivante, et vous verrez l'effet sur le démarrage du serveur avec une installation de base `/usr/local` :

```
shell> ./mysqld --basedir=/usr/local --verbose --help
```

Vous pouvez spécifier d'autres options comme `--datadir` , mais notez que `--verbose` et `--help` doivent être les dernières options. Avant `MySQL` 4.1, omettez l'option `--verbose` .Une fois que vous déterminez les configurations que vous voulez, lancez le serveur avec `--verbose` et `--help` .

Si votre démon `mysqld` fonctionne déjà, vous pouvez connaître les chemins de configuration avec la commande :

```
shell> mysqladmin variables
```

ou :

```
shell> mysqladmin -h host_name variables
```

`host_name` est le nom de l'hôte `MySQL` .Si vous avez une erreur `Errcode 13` (ce qui signifie `Permission denied`) lorsque vous démarrez `mysqld` , cela signifie que les droits d'accès au serveur ou son contenu ne sont pas bons. Dans ce cas, vous devez modifier les droits sur les dossiers et fichiers que le serveur va utiliser. Vous pouvez aussi lancer le serveur en tant que `root` , mais cela pose des problèmes de sécurité, et il vaut mieux l'éviter.

Sous Unix, vérifiez l'existence du dossier de données et vérifiez le nom du propriétaire du dossier de données et de son contenu. Par exemple, si le dossier est `/usr/local/mysql/var` , utilisez cette commande :

```
shell> ls −la /usr/local/mysql/var
```

Si le dossier, ses sous−dossiers ou ses fichiers ne sont pas au nom du compte qui fait tourner le serveur, changez le propriétaire avec cette commande :

```
shell> chown −R mysql /usr/local/mysql/var
shell> chgrp −R mysql /usr/local/mysql/var
```

Quelque soit la méthode que vous utilisez pour démarrer le serveur, si elle échoue, vérifiez le fichier de log d'erreurs pour savoir pourquoi. Les fichiers de log sont situés dans le dossier de données (typiquement `/usr/local/mysql/data` pour une distribution binaire, `/usr/local/var` pour une distribution source, et `\mysql\data\mysql.err` sous Windows). Regardez dans le dossier de données et recherchez des fichiers de la forme `host_name.err` et `host_name.log` ou `host_name` est le nom de votre serveur. Vérifiez alors les dernières lignes de ce fichier :

```
shell> tail host_name.err
shell> tail host_name.log
```

Recherchez des lignes comme celles−ci :

```
000729 14:50:10  bdb:  Recovery function for LSN 1 27595 failed
000729 14:50:10  bdb:  warning: ./test/t1.db: No such file or directory
000729 14:50:10  Can't init databases
```

Cela signifie que vous n'avez pas démarré `mysqld` avec `--bdb-no-recover` et Berkeley DB a trouvé une erreur dans les fichiers de log lorsqu'il a essayé de restaurer votre base. Pour pouvoir continuer, vous devez déplacer le vieux fichier de log Berkeley DB vers un autre dossier, pour l'examiner plus tard. Les fichiers de logs sont nommés `log.0000000001` , et ce nombre augmente au fil du temps.

Si vous exécutez `mysqld` avec les tables `BDB` et que `mysqld` fait des core dumps au démarrage, c'est peut être que vous avez des problèmes avec le fichier de restauration de `BDB` . Dans ce cas, essayez de démarrer `mysqld` avec `--bdb-no-recover` . Si cela aide, vous devriez alors retirer tous les fichiers de log `log.*` du dossier de données, et démarrer `mysqld` à nouveau.

Si vous obtenez l'erreur suivant, cela signifie que d'autres programmes (ou un autre serveur `mysqld`) fonctionne déjà avec le port TCP/IP ou la socket que `mysqld` essaie d'utiliser :

```
Can't start server: Bind on TCP/IP port: Address already in use
Can't start server : Bind on unix socket...
```

Utilisez `ps` pour vous assurer que vous n'avez pas d'autre serveur `mysqld` qui fonctionne. Si c'est le cas, éteignez le serveur avant de lancer `mysqld` à nouveau. Si un autre serveur fonctionne, et que vous voulez vraiment en avoir plusieurs, voyez la section Utiliser plusieurs serveurs sur la même machine .)Si vous ne pouvez pas trouver d'autre serveur en fonctionnement, essayer d'exécuter la commande `telnet votre-nom-d-hote numero-de-port-tcp` puis pressez la touche 'Entrée' plusieurs fois. Si vous n'obtenez pas de message d'erreur comme `telnet: Unable to connect to remote host: Connection refused` , alors un autre processus utilise le port TCP/IP de `mysqld` . Vous devrez alors rechercher le programme qui utilise ce port, et le désactiver, ou bien dire à `mysqld` d'écouter sur un autre port avec l'option `--port` . Dans ce cas, vous devrez aussi spécifier le numéro de port à tous les clients qui se connecte au serveur via TCP/IP.

Une autre raison d'inaccessibilité du port est que vous avez un coupe−feu qui fonctionne, et qui bloque ces port. Pour cela, modifiez la configuration du coupe−feu pour libérer l'accès au port.

Si `safe_mysqld` démarre le serveur, mais que vous n'arrivez pas à vous y connecter, vous devriez vous assurer que vous avez une entrée dans le fichier `/etc/hosts` qui ressemble à ceci :

```
127.0.0.1       localhost
```

Ce problème survient uniquement sur les systèmes qui n'ont pas une bibliothèque de threads fonctionnels, ou pour lesquels `MySQL` a été configuré pour utiliser les `MIT-pthreads` .Si vous n'arrivez toujours pas à lancer `mysqld` , vous pouvez essayer de générer un fichier de traces avec l'option `--debug` . Créer des fichiers de traçage .

2.4.3 Création des premiers droits MySQL

Le processus d'installation de MySQL passe par la création de la base de données `mysql` , qui contient les tables de droits :

- La distribution Windows contient des tables de droits pre−initialisées automatiquement.
- Sous Unix, les tables de droits sont remplies par le programme `mysql_install_db` . Certains assistants d'installation le font pour vous. Les autres vous imposent de le faire manuellement. Pour plus de détails, voyez Procédure de post−installation sur Unix .

Le script `mysql_install_db` démarre le serveur `mysqld` et initialise les tables de droits, avec les paramètres suivants :

- Deux comptes MySQL `root` sont créés en tant qu'administrateurs ayant tous les droits. Le mot de passe de l'utilisateur initial `root` est vide, ce qui permet à n'importe qui de se connecter en tant que `root` **sans mot de passe** , pour profiter de tous les droits.
  - ♦ Sous Windows, un compte `root` permet de se connecter depuis l'hôte local, et l'autre depuis n'importe quel hôte.
  - ♦ Sous Unix, les deux comptes `root` sont destinés à être utilisés depuis le compte local. Les connexions doivent être faîtes en spécifiant le nom d'hôte `localhost` , ou le véritable nom d'hôte, ou l'adresse IP.
- Deux comptes utilisateur anonyme sont créés, qui peuvent faire ce qu'ils veulent avec toutes les tables dans la base de données `'test'` ou commençant par `'test_'` . Cela signifie qu'un utilisateur peut se connecter sans mot de passe et être traité comme un utilisateur anonyme.
  - ♦ Sous Windows, un compte anonyme sert depuis l'hôte local. Ce compte a tous les droits, comme `root` . L'autre sert aux connexions depuis les hôtes, et a tous les droits pour les bases `test` ou commençant par `test` .
  - ♦ Sous Unix, les deux comptes anonymes servent depuis l'hôte local. Les connexions doivent être faîtes en spécificant le nom d'hôte `localhost` , ou le véritable nom d'hôte, ou l'adresse IP. Ces comptes ont tous les droits dans les bases `test` ou dont le nom commence par `test_` .

Comme indiqué, aucun des comptes initiaux n'a de mot de passe. Cela signifie que votre installation MySQL n'est pas protégée jusqu'à ce que vous y remédiez :

- Si vous voulez éviter que les clients ne se connectent en tant qu'utilisateur anonyme sans mot de passe, vous devez assigner des mots de passe à ces comptes, ou bien les supprimer.
- Vous devez assigner des mots de passe aux comptes `root` .

Les instructions suivantes décrivent comment configurer les mots de passe pour les comptes initiaux, en commençant par les comptes anonymes, puis pour les comptes `root` . Remplacez `` `nouveau_mot` '' dans les exemples ci−dessous par le nouveau mot de passe que vous voulez utiliser. Ces instructions montrent aussi comment supprimer les comptes anonymes.

Vous pouvez différer le changement de mot de passe jusqu'à plus tard, pour pouvoir utiliser cette configuration pour les tests. Cependant, n'oubliez pas de le faire avant de mettre votre serveur en production.

Pour assigner des mots de passe aux comptes anonymes, vous pouvez utiliser les commandes `SET PASSWORD` et `UPDATE` . Dans les deux cas, assurez−vous de chiffrer les mots avec la fonction `PASSWORD()` .

Pour utiliser `SET PASSWORD` sur Windows, faîtes :

```
 shell> mysql -u root
mysql> SET PASSWORD FOR ''&#64;'localhost' = PASSWORD('nouveau_mot');
mysql> SET PASSWORD FOR ''&#64;'%' = PASSWORD('nouveau_mot');
```

Pour utiliser `SET PASSWORD` sur Unix, faîtes :

```
 shell> mysql -u root
mysql> SET PASSWORD FOR ''&#64;'localhost' = PASSWORD('nouveau_mot');
mysql> SET PASSWORD FOR ''&#64;'host_name' = PASSWORD('nouveau_mot');
```

Dans la seconde commande `SET PASSWORD` , remplacez `host_name` par le nom de l'hôte du serveur. C'est le nom qui sera spécifié dans la colonne `Host` de la ligne du compte `root` , et qui n'est pas `localhost` . Si vous ne savez pas quel nom d'hôte c'est, utilisez cette commande avant d'utiliser `SET PASSWORD` :

```
 mysql> SELECT Host, User FROM mysql.user;
```

Recherchez une ligne qui contient `root` dans la colonne `User` et quelque chose d'autre que `localhost` dans la colonne `Host` . Puis, utilisez la valeur de `Host` dans la seconde commande `SET PASSWORD` .L'autre moyen d'assigner des mots de passe à un compte anonyme est d'utiliser la commande `UPDATE` pour modifier la table `user` directement. Connectez vous en tant que `root` et envoyez une commande `UPDATE` qui assigne une valeur à la colonne `Password` dans les lignes appropriée de la table `user` . La procédure est la même sous Unix et sous Windows. La commande suivante `UPDATE` met à jour le mot de passe pour les deux comptes en même temps :

```
 shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('nouveau_mot')
    ->     WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

Après voir fait la mise à jour des mots de passe dans la table `user` avec la commande `UPDATE` , vous devez demander au serveur de relire les tables de droits, avec `FLUSH PRIVILEGES` . Sinon, les modifications ne seront pas prises en compte avant le prochain redémarrage du serveur.Si vous préférez supprimer les comptes anonymes, faîtes ceci :

```
 shell> mysql -u root
mysql> DELETE FROM mysql.user WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

La commande `DELETE` s'applique à Windows et Unix. Sous Windows, si vous voulez supprimer uniquement les comptes anonymes qui ont les mêmes droits que `root` , faîtes ceci :

```
 shell> mysql -u root
mysql> DELETE FROM mysql.user WHERE Host='localhost' AND User='';
mysql> FLUSH PRIVILEGES;
```

Ce compte permet un accès anonyme avec les pleins droits : le supprimer améliore la sécurité.

Vous pouvez assigner les mots de passe au compte `root` de nombreuses façons. La discussion suivante montre trois méthodes :

- Utiliser la commande `SET PASSWORD`
- Utiliser la commande en ligne `mysqladmin`
- Utiliser la commande `UPDATE`

Pour assigner un mot de passe avec la commande `SET PASSWORD` , connectez vous en tant que `root` et faites deux commandes `SET PASSWORD` . Assurez vous de chiffrer le mot de passe avec `PASSWORD()` .

Pour Windows, faîtes ceci :

```
 shell> mysql -u root
mysql> SET PASSWORD FOR 'root'&#64;'localhost' = PASSWORD('nouveau_mot');
mysql> SET PASSWORD FOR 'root'&#64;'%' = PASSWORD('nouveau_mot');
```

Pour Unix, faîtes ceci :

```
 shell> mysql -u root
mysql> SET PASSWORD FOR 'root'&#64;'localhost' = PASSWORD('nouveau_mot');
mysql> SET PASSWORD FOR 'root'&#64;'host_name' = PASSWORD('nouveau_mot');
```

Dans la seconde commande `SET PASSWORD` , remplacez `host_name` par le nom de l'hôte du serveur. C'est le même nom que celui qui a été utilisé pour les comptes anonymes.

Pour assigner un mot de passe à `root` en utilisant `mysqladmin` , exécutez les commandes suivantes :

```
 shell> mysqladmin -u root password "nouveau_mot"
shell> mysqladmin -u root -h host_name password "nouveau_mot"
```

Ces commandes s'appliquent à Windows et à Unix. Dans la seconde commande, remplacez `host_name` par le nom du serveur hôte. Les guillemets doubles autour du mot de passe ne sont pas nécessaires, mais vous devez les utiliser si vous avez des espaces ou d'autres caractères spéciaux.

Si vous utilisez un serveur d'une **très** vieille version de MySQL, la commande `mysqladmin` va échouer avec un message d'erreur : `parse error near 'SET password'` . La solution à ce problème est de changer la version du serveur MySQL.

Vous pouvez aussi utiliser `UPDATE` pour modifier directement la table `user` . La commande `UPDATE` suivante assigne un mot de passe aux comptes `root` en même temps :

```
 shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('nouveau_mot')
    ->     WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```

La commande `UPDATE` s'applique à Windows et à Unix.Après modification des mots de passe, vous devrez les fournir à chaque connexion au serveur. Par exemple, si vous voulez utiliser la commande `mysqladmin` pour éteindre le serveur, vous devez utiliser une commande de cette forme :

```
 shell> mysqladmin -u root -p shutdown
Enter password: (enter root password here)
```

Note : si vous perdez votre mot de passe `root` après l'avoir configuré, la procédure pour le remettre à zéro est présentée dans la section Comment modifier un mot de passe Root oublié .Pour créer de nouveaux comptes, utilisez la commande `GRANT` . Pour les instructions, voyez Ajouter de nouveaux utilisateurs à MySQL .

2.5 Changer de version de MySQL

Vous pouvez toujours les fichiers de structures et de données entre les différentes versions de MySQL. La version de base actuelle est la version 3. Si vous changez le jeu de caractères lors de l'utilisation de MySQL (ce qui va aussi affecter le tri), vous devez exécuter la commande `myisamchk -r -q --set-character-set=charset` sur toutes les tables. Sinon, vos index ne seront pas correctement triés.

Les instructions suivantes sont un pense−bête de tout ce à quoi vous devez penser lors d'une mise à jour :

- Lisez le fichier de modification pour la version que vous voulez utiliser, afin de voir les nouvelles fonctionnalités que vous pourrez utiliser. Par exemple, en passant de MySQL 4.1 en 5.0, lisez l'historique de la version 5.0. Historique de MySQL .
- Avant de faire une mise à jour, faîtes une mise à jour de votre base.
- Si vous utilisez `MySQL Server` sur Windows, voyez Mise à jour de MySQL sous Windows .
- Une mise à jour peut impliquer la modification des tables de droits, dans la base `mysql` . Certaines colonnes ou tables peuvent être ajoutées, pour supporter de nouvelles fonctionnalités. Pour tirer partie de ces fonctionnalités, assurez vous de mettre à jour vos tables. La procédure de migration est présentée dans Mise à jour des tables de droits .

- Si vous utilisez la réplication, voyez Mettre à jour une architecture de réplication pour savoir comment mettre à jour votre architecture de réplication.
- Si vous installez une distribution MySQL−Max qui inclut le serveur mysqld−max , puis que vous passez à une version non−Max de MySQL, mysqld_safe va tenter d'utiliser l'ancien serveur mysqld−max . Si vous faites une telle mise à jour, vous devez supprimer manuellement l'ancien serveur mysqld−max pour vous assurer que mysqld_safe utilise le nouveau mysqld .

Si vous avez peur des nouvelles versions, vous pouvez toujours renommer votre vieux mysqld avec un nom comme mysqld−ancienne_version . Si votre nouveau serveur mysqld se comportement bizarrement, vous pourrez toujours l'éteindre, et redémarrer avec votre vieux mysqld !

Lorsque vous faites une évolution de version, vous devriez toujours faire une sauvegarde de vos anciennes données.

Si après un changement de version, vous rencontrez des problèmes avec les clients recompilés, comme Commands out of sync ou des core dumps inopinés, vous avez probablement utiliser un vieux fichier d'entête ou une vieille bibliothèque lors de la compilation de vos programmes. Dans ce cas, vérifiez la date de votre fichier mysql.h , et de votre bibliothèque libmysqlclient.a , pour vous assurer qu'ils proviennent bien de la nouvelle distribution MySQL. Si ce n'est pas le cas, recompilez vos programmes!

Si vous avez des problèmes comme si le nouveau serveur mysqld qui ne veut pas démarrer, ou que vous ne pouvez pas vous connecter sans un mot de passe, vérifiez que vous n'avez pas un vieux fichier my.cnf dans votre installation! Vous pouvez le vérifier comme ceci : program−name −−print−defaults . Si cette commande affiche autre chose que le nom du programme, vous avez un fichier my.cnf actif, qui perturbe vos opérations.

C'est une bonne idée que de reconstruire et re−installer le module Msql−Mysql à chaque fois que vous faites une nouvelle version de MySQL, en particulier si vous rencontrez des symptômes tels que les DBI qui font des core dump après votre mise à jour de MySQL.

2.5.1 Passer en de version 4.1 en version 5.0

En général, il faut suivre ces instructions pour passer en version 5.0, depuis la version 4.1 :

- Lisez la présentation de la version 5.0, pour savoir quelles évolutions et quelles nouveautés vous pourrez utiliser. Changements de la version 5.0.0 (Development) .
- Si vous utilisez MySQL Server sur Windows, voyez la section Mise à jour de MySQL sous Windows .
- MySQL 5.0 apporte les procédures stockées. Ce support requiert la table de droits proc dans la base mysql . Après la mise à jour du serveur, mettez aussi à jour les tables de droits pour vous assurer que la table proc existe. La procédure utilise le script mysql_fix_privilege_tables et est décrite dans la section Mise à jour des tables de droits .
- Si vous utilisez la réplication, voyez aussi la section Mettre à jour une architecture de réplication , pour la mise à jour de votre architecture.

2.5.2 Passer de la version 4.0 à la version 4.1

En général, vous devez suivre les instructions suivantes pour passer de MySQL 4.0 à 4.1 :

- Vérifiez la liste des modifications de cette section, vous voir si elles ont un impact sur vos applications.

- Lisez la liste des nouveautés de la version 4.1 pour identifier les nouvelles fonctionnalités significatives pour vos projets. Changements de la version 4.1.x (Alpha) .
- Si vous faites fonctionner MySQL sur Windows, voyez Mise à jour de MySQL sous Windows . **Note importante :** Les premières versions de MySQL 4.1 sur Windows ne contiennent pas de programme d'installation. Voyez Installer les binaires sous Windows pour des instructions sur l'installation d'une telle distribution.
- Après mise à jour du serveur, mettez à jour les tables de droits, pour accepter des colonnes `Password` plus grande. La procédure utilise le script `mysql_fix_privilege_tables` et est décrite dans la section Mise à jour des tables de droits . Les implications du changement de gestion du mot de passe est décrit plus loin dans cette section. Si vous ne le faîtes pas, MySQL ne pourra pas utiliser le protocole sécuritaire pour l'identification.
- Si vous utilisez la réplication, voyez la section Mettre à jour une architecture de réplication pour mettre à jour votre installation.
- Le moteur de table Berkeley DB table est passé en version DB 4.1 (depuis la 3.2), et il dispose d'un nouveau format de log. Si vous devez revenir en version 4.0, vous devrez utiliser `mysqldump` pour exporter vos tables `BDB` au format texte, et effacer tout les fichiers `log.XXXXXXXXXX` avant de redémarrer le serveur MySQL 4.0 et de réimporter les données.
- Le support des jeux de caractères a été amélioré. si vous avez des tables qui contiennent des données représentées dans un jeu de caractères que MySQL 4.1 supporte directement, vous pouvez convertir ces colonnes vers le bon jeu de caractères avec les instructions du chapitre Conversion des colonnes de caractères version 4.0 en format 4.1 .
- Si vous utilisez un vieux module `DBD-mysql` (`Msql-MySQL-modules`) vous devez mettre à jour le module `DBD-mysql` . Tout ce qui est plus récent que `DBD-mysql` 2.xx doit convenir.Si vous ne mettez pas à jour, certaines commandes telles que `DBI->do()` ne rapporteront pas correctement les erreurs.
- L'option `--defaults-file=option-file-name` vous donnera une erreur si le fichier d'option n'existe pas.

Plusieurs comportements visibles ont changé entre MySQL 4.0 et MySQL 4.1 pour corriger des bogues critiques et rendre MySQL plus compatible avec le standard SQL. Ces changements peuvent affecter votre application.

Certains des comportement 4.1 peuvent être testés en version 4.0 avant de passer à la 4.1. Nous avons ajouté l'option `--new` de démarrage de `mysqld` pour les versions supérieure à la 4.0.12. Options de ligne de commande `mysqld` .

Cette option vous donne le comportement de la version 4.1 pour les modifications les plus critiques. Vous pouvez aussi activer ces comportements pour une connexion particulière en utilisant la commande `SET @@new=1` , pour désactiver cette option avec `SET @@new=0` .

Si vous pensez que certains des changements de la version 4.1 vous affecteront, nous vous recommandons, avant de passer en version 4.1, de télécharger la dernière version 4.0, et de l'exécuter avec l'option `--new` en plus de vos configuration habituelles :

```
 [mysqld-4.0]
new
```

De cette manière, vous pouvez tester le comportement de la version 4.1 depuis votre serveur 4.0. Cela vous donnera le temps de supprimer les anomalies, et de passer sans problème à la version 4.1, ultérieurement. En faisant cela, vous n'allez pas rencontrer de bug accidentel lors du changement, que vous n'aurez pas corrigé grâce à `--new` .

Voici une liste complète, vous indiquant ce à quoi vous devez faire attention lors du changement de version :

Modification du serveur :

- Toutes les colonnes et tables ont désormais un jeu de caractères, qui apparaît dans le résultat de la commande `SHOW CREATE TABLE` et `mysqldump` . Jeux de caractères . (MySQL 4.0.6 et plus récent peuvent lire les nouveaux fichiers de dump, mais pas les plus anciennes versions de MySQL). Cela ne doit pas affecter les applications qui n'utilisent qu'un seul jeu de caractères.
- Le format de définition de table du fichier `.frm` a légèrement changé en version 4.1. Les versions de MySQL 4.0 à partir de la 4.0.11 peuvent lire le nouveau format `.frm` directement, mais les versions plus anciennes ne le peuvent pas. Si vous devez déplacer des tables de la version 4.1 vers une version 4.0.11, passez plutôt par `mysqldump` . `mysqldump` , exporter les structures de tables et les données .
- **Note importante :** si vous mettez à jour en version `InnoDB` −4.1.1 ou plus récent, il sera difficile de revenir à une version plus ancienne, 4.0 or 4.1.0! Ceci est dû aux versions de `InnoDB` qui ne reconnaissent pas les espaces de table multiples.
- Si vous utilisez plusieurs serveurs sur la même machine Windows, vous devriez utiliser l'option `--shared_memory_base_name` avec des valeurs différentes sur toutes les machines.
- L'interface des fonctions `UDF` agrégeantes a un peu changé. Vous devez commencer par déclarer une fonction `xxx_clear()` pour chaque fonction agrégeante `XXX()` .

Evolution du client :

- `mysqldump` dispose des options `--opt` et `--quote-names` , qui sont activées par défaut. Vous pouvez les désactiver avec `--skip-opt` et `--skip-quote-names` .

Evolution du SQL :

- La comparaison de chaînes fonctionne maintenant conformément au standard SQL : au lieu de supprimer les espaces de fin de chaîne avant la comparaison, nous complétons les chaînes courte avec des espaces. Le problème est que maintenant, `'a' > 'a\t'` , ce qui n'était pas le cas avant. Si vous avez des tables avec des colonnes `CHAR` ou `VARCHAR` dont le dernier caractères peut être de code `ASCII(32)` ou plus petit, vous devez utiliser la commande `REPAIR TABLE` ou `myisamchk` .
- Lorsque vous utilisez des commandes `DELETE` multi−tables, vous devez utiliser les alias de tables que vous voulez effacer, et non pas le véritable nom de la table. Par exemple, au lieu de :

```
DELETE test FROM test AS t1, test2 WHERE ...
```

  faîtes :

```
DELETE t1 FROM test AS t1, test2 WHERE ...
```

- `TIMESTAMP` est maintenant retourné comme une chaîne, au format `'YYYY-MM-DD HH:MM:SS'` . L'option `--new` peut être utilisée depuis la version 4.0.12, pour que le serveur adopte le comportement de la version 4.1 pour ce point. Si vous voulez recevoir la version entière de la valeur, comme en version 4.0, il suffit d'ajouter +0 à chaque colonne `TIMESTAMP` :

```
mysql> SELECT ts_col + 0 FROM tbl_name;
```

  La largeur d'affichage des colonnes `TIMESTAMP` ne sont plus supportées. Par exemple, si vous déclarez une colonne de type `TIMESTAMP(10)` , le nombre `(10)` est ignoré.Ces changements sont nécessaires pour respecter les standards SQL. Dans une future version, une autre modification aura lieu, mais restera compatible avec celle−ci : la taille de la valeur `TIMESTAMP` indiquera le nombre de chiffres voulu pour les fractions de secondes.
- Les valeurs binaires, comme `0xFFDF` , sont maintenant supposées être des chaînes et non pas des nombres. Cela corrige des problèmes avec les jeux de caractères, où il est plus pratique d'insérer une chaîne comme une chaîne binaire. Avec cette modification, vous devez utiliser la fonction `CAST()` si vous voulez comparer des valeurs binaires avec les entiers :

```
mysql> SELECT CAST(0xFEFF AS UNSIGNED INTEGER) < CAST(0xFF AS UNSIGNED INTEGER);
        -> 0
```

  Si vous n'utilisez pas `CAST()` , une comparaison lexicale de la chaîne aura lieu :

```
mysql> SELECT 0xFEFF < 0xFF;
        -> 1
```

Utiliser des chaînes binaires dans un contexte numérique, ou bien comparer des valeurs avec les opérateurs comme `=` devrait fonctionner comme auparavant. L'option `--new` peut être utilisée à partir de la version 4.0.13 pour que le serveur 4.0 se comporte comme le serveur 4.1.

- Les fonctions qui retournent des `DATE` , `DATETIME` , ou `TIME` sont désormais traitées lors de leur arrivée sur le client. Par exemple, en MySQL 4.1, vous obtenez le résultat suivant :

```
mysql> SELECT CAST("2001-1-1" as DATETIME);
   -> '2001-01-01 00:00:00'
```

  En MySQL 4.0, le résultat est différent :

```
mysql> SELECT CAST("2001-1-1" as DATETIME);
   -> '2001-01-01'
```

- Les valeurs `DEFAULT` ne peuvent plus être spécifiées pour les colonnes de type `AUTO_INCREMENT` . En 4.0, la clause `DEFAULT` est ignorée silencieusement. En 4.1, une erreur survient.
- `LIMIT` n'accepte plus les arguments négatifs. Utilisez 18446744073709551615 au lieu de −1.
- `SERIALIZE` n'est plus une option valide pour la variable `sql_mode` . Il faut utiliser la commande `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` à la place. `SERIALIZE` n'est plus valide comme option de `--sql-mode` pour `mysqld` , non plus. Utilisez `--transaction-isolation=SERIALIZABLE` .

Changement de l'interface C :

- Certaines fonctions C telles que `mysql_real_query()` retournent maintenant `1` en cas d'erreur, et non plus `-1` . Vous aurez peut être à changer certaines anciennes applications comme ceci :

```
 if (mysql_real_query(mysql_object, query, query_length) == -1)
&#123;
  printf("Erreur");
&#125;
```

  Modifiez le test de comparaison à 0 :

```
 if (mysql_real_query(mysql_object, query, query_length) != 0)
&#123;
  printf("Erreur");
&#125;
```

Gestion des mots de passe :

Le mécanisme de mot de passe a changé en version 4.1 pour assurer une meilleure sécurité, mais cela pose des problèmes de compatibilité, si vous avez encore des clients qui utilisent les bibliothèques 4.0 ou plus ancien. Il est probable que vous ayez de tels clients, s'ils se connectent depuis des serveurs distants qui n'ont pas encore adopté la version 4.0. La liste suivante présente les stratégies de mise à jour. Elle représentent différents compromis entre la compatibilité et la sécurité.

- Ne passez pas en version 4.1. Aucun comportement ne changera, mais vous ne pourrez pas utiliser les nouvelles fonctionnalités du protocole de la version 4.1. MySQL a amélioré le protocole client/serveur de la version 4.1, en ajoutant les commandes préparées et le support des jeux de caractères. Commandes préparées en C .
- Passez en version 4.1, utilisez le script `mysql_fix_privilege_tables` pour agrandir la colonne `Password` de la table `user` pour qu'elle puisse contenir les nouveaux hashs de mots de passe. Mais lancez le serveur avec l'option `--old-passwords` pour que les clients pre−4.1 puissent continuer d'utiliser leurs anciens comptes. Finalement, lorsque tous les clients seront passés en version 4.1, vous pourrez cesser d'utiliser l'option `--old-passwords` . Vous pouvez aussi changer les mots de passe de vos comptes MySQL pour adopter le nouveau format.
- Passez en version 4.1 et utilisez le script `mysql_fix_privilege_tables` pour aggrandir la colonne `Password` de la table `user` . Si vous savez que tous les clients sont passés en version 4.1, n'utilisez pas l'option `--old-passwords` . Au lieu de cela, changez les mots de passe de tous les comptes, pour qu'ils adoptent le nouveau format. Une installation 100% 4.1 est la plus sûre.

D'autres informations sur le nouvel algorithme de protection des mots de passe et les opérations les concernants sont disponibles dans la section Chiffrement des mots de passe en MySQL 4.1 . `Client does not support authentication protocol` .

2.5.3 Passer de la version 3.23 à la version 4.0

En général, ce que vous devez faire pour passer en version 4.0, à partir d'une version 3.23 :

- Vérifiez que les changements de la liste ci−dessous n'affectent pas votre application.
- Lisez les nouveautés de la version 4.0, pour savoir quelles nouvelles fonctionnalités vous allez découvrir en 4.0. Changements de la version 4.0.x (Production) .
- Après mise à jour, exécutez le script `mysql_fix_privilege_tables` pour ajouter de nouveaux droits et fonctionnalités à la table MySQL. Voyez Mise à jour des tables de droits .
- Editez les scripts de démarrage MySQL pour les fichiers de configuration pour ne plus utiliser les options abandonnées, listées ci−dessous.
- Convertissez vos vieilles tables `ISAM` en tables `MyISAM` avec la commande : `mysql_convert_table_format database` . Pour convertir toutes les tables d'une base de données, utilisez cette commande :

  ```
  shell> mysql_convert_table_format database db_name
  ```

  Notez que cela ne doit être fait que si toutes les tables de la base sont des tables `ISAM` ou `MyISAM` . Pour éviter de convertir toutes les tables d'une base au format `MyISAM` , vous pouvez explicitement utiliser les noms de vos tables `ISAM` après le nom de la base dans la commande. Vous pouvez aussi utiliser la commande `ALTER TABLE table_name TYPE=MyISAM` sur toutes les tables `ISAM` .Les tables individuelles peuvent être mises au format `MyISAM` en utilisant la commande `ALTER TABLE` suivante, pour chaque table :

  ```
  mysql> ALTER TABLE tbl_name TYPE=MyISAM;
  ```

  Pour connaître le type d'une table, utilisez cette commande :

  ```
  mysql> SHOW TABLE STATUS LIKE 'tbl_name';
  ```
- Assurez−vous que vous n'avez pas de client MySQL qui utilise des bibliothèques partagées (comme les modules Perl Msql−Mysql). Si vous en avez, vous devriez les recompiler car les structures utilisées dans `libmysqlclient.so` ont changées.
- Si vous utilisez MySQL sur Windows, voyez aussi Mise à jour de MySQL sous Windows .
- Si vous utilisez la réplication, voyez aussi Mettre à jour une architecture de réplication pour plus de détails sur la mise à jour de la réplication.

MySQL 4.0 va fonctionner même si vous ne suivez pas les instructions ci−dessus, mais il ne sera pas capable de profiter des nouveaux droits disponibles avec MySQL 4.0 et vous pourriez rencontrer des problèmes lors de l'évolution vers MySQL 4.1 ou plus récent. Les fichiers `ISAM` fonctionne toujours en MySQL 4.0 mais il est abandonné, et il sera désactivé en MySQL 5.0.

Les anciens clients doivent fonctionner avec le serveur version 4.0 sans aucun problème.

Même si vous suivez les instructions ci−dessus, vous pourrez retourner en version MySQL 3.23.52 ou plus récent, si vous rencontrez des difficultés avec MySQL 4.0. Dans ce cas, vous devez utiliser la commande `mysqldump` sur toutes les tables qui utilisent un index en texte plein, et restaurer ces tables en 3.23 (car la version 4.0 utilise un nouveau format pour les index en texte plein).

Voici une liste plus complète de points à contrôler lorsque vous passez à la version 4.0 :

- MySQL 4.0 a de très nombreux nouveaux droits dans la table `mysql.user` . Droits fournis par MySQL .

  Pour installer ces nouveaux droits, suivez la procédure dans Mise à jour des tables de droits . Jusqu'à ce que ce script soit exécuté, les utilisateurs auront les droits de `SHOW DATABASES` , `CREATE TEMPORARY TABLES` , et `LOCK TABLES` . Les droits de `SUPER` et `EXECUTE` héritent leur valeur du droit de `PROCESS` . `REPLICATION SLAVE` et `REPLICATION CLIENT` héritent leur valeur de `FILE` .

  Si vous avez un script qui crée automatiquement des nouveaux utilisateur, vous devez le modifier pour y inclure les nouveaux droits. Si vous n'utilisez pas la commande `GRANT` dans ces scripts, c'est une bonne idée que de les vérifier.

En version 4.0.2, l'option `--safe-show-database` est abandonnée (et ne fait plus rien du tout). Options de démarrage de `mysqld` qui concernent la sécurité .

Si vous obtenez des interdictions d'accès pour les nouveaux utilisateurs en version 4.0.2, vous devriez vérifier si vous avez besoin de nouveaux droits que vous n'utilisiez pas avant. En particulier, vous aurez besoin du droit de `REPLICATION SLAVE` (au lieu de `FILE`) pour les nouveaux esclaves.

- `safe_mysqld` a été renommé en `mysqld_safe` . Pour assurer la compatibilité ascendante, les distribution binaires vont inclure pour quelques temps un lien symbolique de `safe_mysqld` vers `mysqld_safe` .
- Le support `InnoDB` est désormais inclut par défaut dans la distribution binaire. Si vous compilez MySQL depuis les sources, et que vous voulez économiser de la mémoire, utilisez l'option `--skip-innodb` au démarrage du serveur. Pour compiler MySQL sans le support `InnoDB` , utilisez le script `configure` avec l'option `--without-innodb` .
- Les paramètres de démarrage `myisam_max_extra_sort_file_size` et `myisam_max_extra_sort_file_size` sont désormais exprimés en octets, et non plus en Mo, comme cela était le cas jusqu'en version 4.0.3).
- `mysqld` dispose maintenant de l'option `--temp-pool` activée par défaut, car cela donne de meilleures performances sur certains systèmes d'exploitation, et notamment Linux.
- Les options de démarrage `mysqld --skip-locking` et `--enable-locking` ont été renommées `--skip-external-locking` et `--external-locking` .
- Les systèmes externes de verrouillages des tables `MyISAM` / `ISAM` sont désormais désactivés par défaut. Vous pouvez les réactiver avec l'option `--external-locking` . Pour la plupart des utilisateurs, ce n'est jamais nécessaire.
- Les options de démarrage suivantes ont été renommées :

Ancien nom	Nouveau nom .
myisam_bulk_insert_tree_size	bulk_insert_buffer_size
query_cache_startup_type	query_cache_type
record_buffer	read_buffer_size
record_rnd_buffer	read_rnd_buffer_size
sort_buffer	sort_buffer_size
warnings	log-warnings
err-log	--log-error (for mysqld_safe)

Les options de démarrage `record_buffer` , `sort_buffer` et `warnings` vont encore fonctionner avec MySQL 4.0 mais elles sont obsolètes.

Changements SQL :

- Les variables SQL suivantes ont changé de nom.

Ancien nom	Nouveau nom .
SQL_BIG_TABLES	BIG_TABLES
SQL_LOW_PRIORITY_UPDATES	LOW_PRIORITY_UPDATES
SQL_MAX_JOIN_SIZE	MAX_JOIN_SIZE
SQL_QUERY_CACHE_TYPE	QUERY_CACHE_TYPE

Les anciens noms fonctionneront encore en MySQL 4.0, mais sont obsolètes.

- Vous devez utiliser la commande `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=#` au lieu de `SET SQL_SLAVE_SKIP_COUNTER=#` .
- `SHOW MASTER STATUS` retourne désormais une liste vide si les logs binaires ne sont pas activés.
- `SHOW SLAVE STATUS` retourne désormais une liste vide si l'esclave n'est pas initialisé.
- `SHOW INDEX` a 2 colonnes de plus (`Null` et `Index_type`) qu'il n'avait pas en version 3.23.
- Le format de `SHOW OPEN TABLE` a été changé.
- `ORDER BY col_name DESC` trie les valeurs `NULL` en dernier, depuis MySQL 4.0.11. En 3.23 et dans les premières versions de 4.0, ce n'était pas toujours cohérent.
- `CHECK` , `SIGNED` , `LOCALTIME` et `LOCALTIMESTAMP` sont des mots réservés.
- Les colonnes `DOUBLE` et `FLOAT` acceptent désormais l'option `UNSIGNED` pour le stockage (auparavant, `UNSIGNED` était ignoré pour ces colonnes).
- Le résultat de toutes les opérations sur les bits, `|` , `&` , `<<` , `>>` et `~` est maintenant non signé. Cela peut poser des problèmes si vous aviez un contexte dans lequel vous souhaitez un résultat signé. Fonctions de transtypage .

Note : lorsque vous utilisez la soustraction entre des entiers dont l'un est `UNSIGNED` , le résultat sera non signé! En d'autres termes, avant de passer à la version MySQL 4.0, vous devriez vérifier les situations où votre application soustrait une valeur d'un entier non signé, et que vous attendez une valeur négative, ou si vous soustrayez une valeur non signée d'une colonne. Vous pouvez désactiver ce comportement en utilisant l'option de démarrage `--sql-mode=NO_UNSIGNED_SUBTRACTION` lorsque vous démarrez `mysqld` . Fonctions de transtypage .

- Vous devriez utiliser des entiers pour stocker les valeurs dans les colonnes de type BIGINT (au lieu d'utiliser des chaînes, comme vous le faisiez en MySQL 3.23). Utiliser des chaînes va toujours fonctionner, mais passer des entiers est bien plus efficace.
- En version 3.23, `INSERT INTO ... SELECT` fonctionne toujours avec l'option `IGNORE` . En version 4.0.1, MySQL va s'arrêter (et peut être annuler la transaction) si vous ne spécifiez pas l'option `IGNORE` .
- Vous devriez utiliser la commande `TRUNCATE TABLE` lorsque vous voulez effacer toutes les lignes d'une table, et que vous ne souhaitez pas savoir combien de lignes ont été effacées de la table (car `TRUNCATE TABLE` est plus rapide que `DELETE FROM table_name`).
- Vous allez rencontrer une erreur si vous avez un verrou actif ou une transaction active, et que vous essayez d'utiliser les commandes `TRUNCATE TABLE` ou `DROP DATABASE` .
- Pour utiliser `MATCH ... AGAINST (... IN BOOLEAN MODE)` avec vos table,s vous devez les reconstruire avec `ALTER TABLE table_name TYPE=MyISAM` , **même** si la table est déjà au format `MyISAM` . Optimisation des recherches en texte intégral .
- `LOCATE()` et `INSTR()` sont sensibles à la casse, si l'un des arguments est une chaîne binaire. Sinon, ils sont insensibles à la casse.
- `STRCMP()` utilise désormais le jeu de caractères courant pour les comparaisons, ce qui signifie que le comportement par défaut des comparaisons est désormais insensible à la casse.
- `HEX(string)` retourne désormais les caractères convertis sous la forme d'une chaîne hexadécimale. Si vous voulez convertir un nombre en hexadécimal, vous devez vous assurer d'appeler `HEX()` avec un argument numérique.
- `RAND(seed)` retourne un nombre différent en version 4.0 qu'en version 3.23 : cela est fait pour différencier plus fortement `RAND(seed)` de `RAND(seed+1)` .
- Le type par défaut retourné par `IFNULL(A,B)` est maintenant le plus général des deux types `A` et `B` . (L'ordre est `STRING` , `REAL` puis `INTEGER`).

Changements de l'interface C :

- Les fonctions de l'ancienne API C API `mysql_drop_db` , `mysql_create_db` et `mysql_connect` ne sont plus supportées, à moins que vous ne compiliez MySQL avec `CFLAGS=-DUSE_OLD_FUNCTIONS` . Au lieu de cela, il sera plus sage de changer vos programmes, pour qu'il utilisent la nouvelle API 4.0.
- Dans la structure `MYSQL_FIELD` , `length` et `max_length` ont évolué de `unsigned int` en `unsigned long` . Cela ne va pas causer de problèmes, hormis le fait qu'ils peuvent générer des messages d'alerte lorsqu'ils sont utilisé comme argument de fonctions comme `printf()` .
- Les clients multi−threadés doivent utiliser `mysql_thread_init()` et `mysql_thread_end()` . Comment programmer une client threadé .

Autres changements :

- Si vous voulez recompiler le module Perl DBD::mysql, vous devez prendre les versions Msql−Mysql−modules 1.2218 ou plus récente, car les anciennes versions des module DBD utilisent une fonction `drop_db()` abandonnée.

2.5.4 Passer de la version 3.22 à la version 3.23

Les clients des versions 3.22 et 3.21 vont fonctionner sans problèmes avec la version 3.23 du serveur.

La liste suivante indique les points à vérifier lors de la migration :

Changement de tables :

- MySQL 3.23 supporte les nouvelles tables `MyISAM` et l'ancien type `ISAM` . Par défaut, toutes les nouvelles tables sont créées avec `MyISAM` à moins que vous ne lanciez `mysqld` avec l'option `--default-table-type=isam` . Vous n'avez pas à convertir les anciennes tables `ISAM` pour les utiliser avec MySQL 3.23. Vous pouvez les convertir les tables `ISAM` en `MyISAM` avec la commande `ALTER TABLE tbl_name TYPE=MyISAM` et le script Perl `mysql_convert_table_format` .
- Toutes les tables qui utilisent le jeu de caractères `tis620` doivent être corrigées avec `myisamchk -r` ou `REPAIR TABLE` .
- Si vous utilisez le jeu de caractères `allemand` pour les tris, vous devez réparer vos tables avec `isamchk -r` , car nous avons fait des modifications dans l'ordre de tri.

Changement au programme client :

- Le client MySQL `mysql` est démarré par défaut avec l'option option `--no-named-commands (-g)` . Cette option peut être désactivée avec `--enable-named-commands (-G)` . Cela peut causer des problèmes d'incompatibilité dans certains cas : par exemple, dans les scripts SQL qui utilisent des commandes nommées sans point virgule! Le format long de la commande devrait fonctionner correctement.
- Si vous voulez que les fichiers d'export de `mysqldump` soient compatibles entre les versions MySQL 3.22 et 3.23, vous ne devez pas utiliser l'option `--opt` ou `--all` de `mysqldump` .

Changements SQL :

- Si vous exécutez une commande `DROP DATABASE` sur un lien symbolique, le lien et la base originale seront effacés. Cela n'arrivait pas en 3.22 car `configure` ne détectait pas les appels à `readlink` .
- `OPTIMIZE TABLE` ne fonctionne que pour les tables `MyISAM` . Pour les autres types de tables, vous devez utiliser `ALTER TABLE` pour optimiser la table. Durant la commande `OPTIMIZE TABLE` , la table est verrouillée.
- Les fonctions de date qui travaillent sur des parties de date (comme `MONTH()`) vont désormais retourner 0 pour la date `0000-00-00` . (MySQL 3.22 retournait `NULL` .)
- Le type de retour par défaut de `IF()` dépendant maintenant des deux arguments, et plus seulement du premier.
- `AUTO_INCREMENT` ne fonctionne pas sur les nombres négatifs. La raison pour cela est que les nombres négatifs posaient des problèmes d'écrasement entre −1 et 0. `AUTO_INCREMENT` pour les tables `MyISAM` est maintenant géré à un niveau plus bas, et il est bien plus rapide. Pour les tables `MyISAM` , les anciens numéros ne sont plus réutilisés, même si vous effacez des lignes dans la table.
- `CASE` , `DELAYED` , `ELSE` , `END` , `FULLTEXT` , `INNER` , `RIGHT` , `THEN` et `WHEN` sont de nouveaux mots réservés.
- `FLOAT(p)` est maintenant un véritable type de nombre à virgule flottante, avec un nombre défini de décimales.
- Lors de la déclaration de `DECIMAL(length,dec)` , la taille de l'argument n'inclut plus une place pour le signe ou le séparateur décimal.
- Une chaîne `TIME` doit être fournie au format suivant : `[[[DAYS] [H]H:]MM:]SS[.fraction]` ou `[[[[[H]H]H]H]MM]SS[.fraction]` .
- `LIKE` compare maintenant les chaînes en appliquant les mêmes règles que `=` . Si vous voulez l'ancien comportement, vous pouvez compiler MySQL avec l'option `CXXFLAGS=-DLIKE_CMP_TOUPPER` .
- `REGEXP` est maintenant insensible à la casse pour les chaînes normales (non binaires).
- Quand vous vérifiez/réparez des tables, vous devez utiliser `CHECK TABLE` ou `myisamchk` pour les tables `MyISAM` (`.MYI`) et `isamchk` pour les tables `ISAM` (`.ISM`).
- Vérifiez tous vos appels à `DATE_FORMAT()` pour vous assurer qu'il y a un signe pourcentage `'%'`

avant chaque caractère de format (MySQL version 3.22 et plus récent avait déjà cette syntaxe, mais désormais '%' est obligatoire).
- En MySQL version 3.22, le résultat de SELECT DISTINCT ... était toujours trié. En version 3.23, vous devez spécifier la clause GROUP BY ou ORDER BY pour obtenir un résultat trié.
- SUM() retourne désormais NULL , au lieu de 0, si il n'y a pas de lignes à calculer. Ceci s'accorde avec la norme SQL.
- AND ou OR avec les valeurs NULL vont désormais retourner NULL au lieu de 0. Cela affecte surtout les requêtes qui utilisait NOT ou une expression AND/OR telle que NOT NULL = NULL .
- LPAD() et RPAD() vont réduire la taille de la chaîne résultante, si elle est plus grand que l'argument de taille.

Changement de l'interface C :

- mysql_fetch_fields_direct est maintenant une fonction (c'était une macro), qui retourne un pointeur sur MYSQL_FIELD au lieu de MYSQL_FIELD .
- mysql_num_fields() ne peut plus être utilisé sur les objets MYSQL* (c'est maintenant une fonction qui prend MYSQL_RES* comme argument. Il faut donc utiliser mysql_field_count() à la place).

2.5.5 Passer de la version 3.21 à la version 3.22

Rien qui n'affecte la compatibilité n'a changé entre les versions 3.21 et 3.22. Le seul problème courant est que les nouvelles tables qui sont créées avec le type DATE vont désormais utiliser le nouveau format de stockage. Vous ne pourrez pas accéder à ces nouveaux formats depuis les vieilles versions de mysqld .

Lors de la mise à jour en MySQL 3.23 depuis une ancienne version, suivez ces conseils :

- Après avoir installé MySQL version 3.22, vous devriez démarrer le nouveau serveur, et exécuter le script mysql_fix_privilege_tables . Il va ajouter les nouveaux droits à la commande GRANT . Si vous oubliez cela, vous obtiendrez des erreurs Access denied lorsque vous essayez d'utiliser les commandes ALTER TABLE , CREATE INDEX ou DROP INDEX . La procédure pour mettre à jour les tables de droits est décrite dans Mise à jour des tables de droits .
- L'interface C de mysql_real_connect() a changé. Si vous avez un vieux client qui appelle cette fonction, vous devez placer un 0 pour le nouvel argument db (ou réécrire le client pour qu'il envoie l'élément db , et accélère les connexions). Vous devez aussi appeler mysql_init() avant d'appeler mysql_real_connect() ! Ce changement a été fait pour permettre l'appel de la fonction mysql_options() , qui sauve les options dans la structure MYSQL .
- La variable key_buffer de mysqld a changé de nom, et est devenue key_buffer_size , mais vous pouvez toujours utiliser l'ancien nom dans vos fichiers de démarrage.

2.5.6 Passer de la version 3.20 à la version 3.21

Si vous avez une version de MySQL plus ancienne que la version 3.20.28 et que vous voulez passer à la version 3.21, vous devez suivre ces étapes :

Vous pouvez démarrer le serveur mysqld version 3.21 avec le script safe_mysqld --old-protocol pour l'utiliser avec les clients de la version 3.20. Dans ce cas, la fonction mysql_errno() des nouveaux clients ne sera pas fonctionnelle, et seul CR_UNKNOWN_ERROR (mais il fonctionne pour les erreurs client), et le serveur utilisera l'ancienne fonction password() plutôt que la nouvelle.

Si vous **n'utilisez pas** l'option --old-protocol avec mysqld , vous devez suivre ces instructions :

- Tous les clients doivent être recompilés. Si vous utilisez ODBC, vous devez obtenir le nouveau pilote `MyODBC` 2.x.
- Le script `scripts/add_long_password` doit être utilisé pour convertir le champs `Password` de la table `mysql.user` en `CHAR(16)` .
- Tous les mots de passe doivent être réassignés dans la table `mysql.user` pour utiliser les mots de 62 bits au lieu de 31 bits.
- Le format de table n'a pas changé, ce qui vous évite d'avoir à convertir des tables.

MySQL version 3.20.28 et plus récent peut gérer les nouvelles tables `user` sans affecter les clients. Si vous avez une version plus ancienne que la 3.20.28, les mots de passe ne seront plus valide, si vous convertissez la table `user` . Pour être tranquille, commencez par faire passer votre version à la 3.20.28 puis passez en version 3.21.

Le nouveau client fonctionne avec le serveur 3.20.x `mysqld` , alors si vous rencontrez des problèmes avec la version 3.21.x, vous pouvez toujours vous rabattre sur les vieux serveurs 3.20.x sans recompiler les clients.

Si vous n'utilisez pas l'option `--old-protocol` de `mysqld` , les vieux clients vont émettre une erreur :

```
ERROR: Protocol mismatch. Server Version = 10 Client Version = 9
```

La nouvelle interface Perl `DBI` / `DBD` supporte aussi l'ancienne interface `mysqlperl` . Le seul changement que vous devez faire si vous utilisez `mysqlperl` est de changer les arguments de la fonction `connect()` . Les nouveaux arguments sont : `host` , `database` , `user` , et `password` (les arguments `user` et `password` ont été échangés).Les modifications actuelles affectent les requêtes des anciennes applications :

- `HAVING` doit être spécifié avant la clause `ORDER BY` .
- Les paramètres de la fonction `LOCATE()` ont été échangés.
- Il y a de nouveaux mots réservés. Les plus notables sont `DATE` , `TIME` et `TIMESTAMP` .

2.5.7 Mettre à jour MySQL sous Windows

Lors de la mise à jour de MySQL sous Windows, suivez ces instructions :

- Téléchargez la dernière version de MySQL pour Windows.
- Cherchez un horaire dans la journée où l'utilisation du serveur est faible, et où une interruption de service est acceptable.
- Prévenez les utilisateurs que vous allez interrompre le serveur.
- Stoppez le serveur MySQL (par exemple, avec la commande `NET STOP MySQL` ou avec l'utilitaire de `Services` si vous utilisez MySQL sous forme de service, ou encore avec `mysqladmin shutdown`).
- Quittez le programme `WinMySQLAdmin` , s'il fonctionnait.
- Exécutez les scripts d'installation de la distribution Windows, en cliquant sur le bouton "Install" dans WinZip et en suivant les instructions d'installation. **Note importante :** Les premières versions des distributions Windows de MySQL 4.1 n contenait pas de programme d'installation. Voyez Installer les binaires sous Windows pour les instructions d'installation d'une telle distribution.
- Vous pouvez écraser votre précédente installation (généralement installée dans `C:\mysql`), ou l'installer dans un nouveau dossier, comme `C:\mysql4` . Supprimer l'ancienne version est recommandée.
- Relancez le serveur (par exemple, avec `NET START MySQL` si vous utilisez MySQL comme un service, ou directement avec la commande `mysqld`).
- Mettez à jour les tables de droits. La procédure est décrite dans la section Mise à jour des tables de droits .

Situations possibles :

```
 A system error has occurred.
System error 1067 has occurred.
The process terminated unexpectedly.
```

Cette erreur signifie que votre fichier d'options, qui est par défaut `C:\my.cnf` , contient une option qui n'est pas reconnue par MySQL. Vous pouvez vérifier que c'est le cas en renommant le fichier `my.cnf` en `my_cnf.old` , pour éviter que le serveur l'utilise. S'il démarre correctement, il vous faut alors identifier la partie du fichier d'options cause le problème. Créez un nouveau fichier `my.cnf` , puis déplacez progressivement toutes les parties de l'ancien fichier d'options, en redémarrant le serveur entre deux copies : vous allez identifier à coup sur le problème.

2.5.8 Mise à jour des tables de droits

Certaines versions introduisent des modifications dans la structure des tables de droits (les tables qui sont dans la base `mysql`), pour ajouter de nouveaux droits ou fonctionnalités. Pour vous assurer que vos tables de droits sont à jour lorsque vous changez de version de `MySQL` , il est recommandé de les mettre aussi à jour.

Sous Unix ou ses équivalent, la mise à jour des tables de droits se fait en exécutant le script `mysql_fix_privilege_tables` :

```
shell> mysql_fix_privilege_tables
```

Vous devez exécuter ce script lorsque le serveur fonctionne. Le script tente de se connecter au serveur local avec le compte `root` . Si votre compte `root` requiert un mot de passe, indiquez−le en ligne de commande. Depuis `MySQL` 4.1 et plus récent, spécifiez le mot de passe comme ceci :

```
shell> mysql_fix_privilege_tables --password=root_password
```

Pour les versions antérieures à la version `MySQL` 4.1, spécifiez le mot de passe comme ceci :

```
shell> mysql_fix_privilege_tables root_password
```

L script `mysql_fix_privilege_tables` effectue les manipulations nécessaires pour convertir vos tables de droits au format courant. Vous pouvez apercevoir des alertes `Duplicate column name` durant l'exécution du script : ces alertes peuvent être ignorées sans danger.Après avoir exécuté le script, arrêtez le serveur, et relancez le.

Sous Windows, il n'existe pas de moyen facile de modifier les tables de droits jusqu'à `MySQL` 4.0.15. Depuis la version 4.0.15 on, la distribution `MySQL` inclut un script SQL `mysql_fix_privilege_tables.sql` que vous pouvez utiliser avec le client `mysql` . Si votre installation `MySQL` est située dans le dossier `C:\mysql` , la commande a utiliser ressemble à celle−ci :

```
C:\mysql\bin> mysql -u root -p mysql

mysql> SOURCE C:\mysql\scripts\mysql_fix_privilege_tables.sql
```

Si votre installation est située dans un autre dossier, vous devrez adapter les chemins.

La commande `mysql` vous demandera le mot de passe pour le compte `root` : saisissez−le lorsqu'il est demandé.

Comme pour la procédure Unix, vous pouvez voir apparaître des alertes `Duplicate column name` durant le traitement de votre base `mysql` par le script `mysql_fix_privilege_tables.sql` : elles peuvent être ignorées.

Après avoir exécuté le script arrêtez le serveur, et relancez−le.

2.5.9 Migrer depuis une autre architecture

Si vous utilisez MySQL version 3.23, vous pouvez copier les fichiers `.frm` , `.MYI` et `.MYD` entre les différentes architectures qui supportent le même format de nombre à virgule flottante (MySQL prend en charge les échanges d'octets). Tables `MyISAM` .

Les données MySQL des tables `ISAM` et les fichiers d'index (`.ISD` et `*.ISM` , respectivement) sont dépendantes de l'architecture, et dans certains cas, dépendantes du système d'exploitation. Si vous voulez déplacer des applications vers une autre machine qui a une autre architecture, ou un autre système d'exploitation que votre machine courante, il est recommandé de ne pas faire une simple copie de base en copiant les fichiers vers leur nouvelle destination. Utilisez plutôt `mysqldump` .

Par défaut, `mysqldump` va créer un fichier de requêtes SQL. Vous pouvez alors transférer le fichier sur une autre machine, et le fournir comme script à un client `mysql` .

Essayez la commande `mysqldump --help` pour voir quelles options sont disponibles. Si vous envoyez les données vers une nouvelle version de MySQL, il recommandé d'utiliser l'option `mysqldump --opt` pour obtenir un export compact et plus rapide.

Le plus facile (mais pas le plus rapide) pour déplacer une base de données entre deux machines et d'exécuter les commandes suivantes sur la machine qui héberge la base :

```
 shell> mysqladmin -h 'other hostname' create db_name
shell> mysqldump --opt db_name \
       | mysql -h 'other hostname' db_name
```

Si vous voulez copier la base depuis une machine distante sur un réseau lent, vous pouvez utiliser :

```
 shell> mysqladmin create db_name
shell> mysqldump -h 'other hostname' --opt --compress db_name \
       | mysql db_name
```

Vous pouvez aussi stocker le résultat dans un fichier, et transférer le fichier sur la machine de destination, puis charger ce fichier dans le serveur. Par exemple, vous pouvez exporter la base vers un fichier source comme ceci :

```
 shell> mysqldump --quick db_name | gzip > db_name.contents.gz
```

Le fichier créé est compressé. Transférez le fichier contenant le contenu de votre base sur la machine de destination, puis utilisez ces commandes :

```
 shell> mysqladmin create db_name
shell> gunzip < db_name.contents.gz | mysql db_name
```

Vous pouvez aussi utiliser `mysqldump` et `mysqlimport` pour accomplir cette opération. Pour les grandes tables, c'est bien plus rapide que d'utiliser simplement `mysqldump` . Dans les commandes suivantes, `DUMPDIR` représente le chemin complet du dossier que vous utilisez pour stocker le résultat de `mysqldump` .Premièrement, créez un dossier pour les fichiers d'exportation, puis faites l'export :

```
 shell> mkdir DUMPDIR
shell> mysqldump --tab=DUMPDIR db_name
```

Puis transférez les fichiers du dossier `DUMPDIR` dans un dossier correspondant, dans la machine de destination, puis chargez ces fichiers dans MySQL comme ceci :

```
 shell> mysqladmin create db_name           # Création de la base
shell> cat DUMPDIR/*.sql | mysql db_name   # Création des tables dans la base
shell> mysqlimport db_name DUMPDIR/*.txt   # Chargement des données dans les tables
```

N'oubliez pas non plus de copier le contenu de votre base `mysql` car c'est là que résident les droits (`user` , `db` , `host`). Vous devrez alors exécuter les commandes en tant que `root` MySQL sur la nouvelle machine, jusqu'à ce que vous ayez réinstallé `mysql` .

Après l'importation de la base `mysql` sur la nouvelle machine, exécutez la commande `mysqladmin flush-privileges` pour que le serveur relise les droits.

2.6 Notes spécifiques aux systèmes d'exploitation

2.6.1 Notes relatives à Linux (toutes versions)

Cette section présente les problèmes que nous avons rencontré avec Linux. Les premières sections décrivent les problèmes liés aux opérations générales, les problèmes qui arrivent avec les distributions binaires ou source, puis les problèmes de post−installation. Les autres sections discutent de problèmes qui surviennent sur des plates−formes spécifiques.

Notez que la plupart des problèmes surviennent sur les anciennes versions de Linux. Si vous utilisez une version récente, vous ne les verrez jamais ou presque.

2.6.1.1 Notes sur Linux

MySQL requiert au moins Linux Version 2.0.

Attention : Nous avons rencontré d'étranges problèmes avec Linux 2.2.14 et MySQL sur des architectures SMP. Nous avons aussi des rapports d'utilisateurs MySQL qui ont rencontré de sérieux problèmes de stabilité avec le noyau 2.2.14. Si vous utilisez ce noyau, il est recommandé de passer en version 2.2.19 (ou plus récente) ou en 2.4. Si vous avez un serveur multi−processeurs, vous devriez considérer sérieusement la migration vers la version 2.4 car elle vous apportera un gain de vitesse significatif. Votre système sera aussi plus stable.

Lorsque vous utilisez les `LinuxThreads` , vous verrez un minimum de trois processus `mysqld` . Ce sont en fait des threads. Il y a un thread de gestion `LinuxThreads` , un thread pour gérer les connexions et un thread pour gérer les alertes et signaux.

2.6.1.2 Notes relatives à Linux pour les distributions binaires

MySQL requière au moins la version 2.0 de Linux.

Attention : Certains utilisateurs de MySQL nous ont avertis qu'ils ont rencontré de graves problèmes de stabilité avec MySQL et le noyau 2.2.14 de Linux. Si vous utilisez ce noyau, vous devez mettre à jour à la 2.2.19 (ou plus récent) ou a un noyau 2.4. Si vous utilisez un ordinateur multi−processeurs, vous devriez sérieusement songer à passer au noyau 2.4 qui vous apportera de grandes performances niveau vitesse.

La version binaire est liée avec `−static` , ce qui signifie que normalement vous n'avez pas besoin de vous soucier des versions des bibliothèques système que vous avez. Vous n'avez pas besoin d'installer `LinuxThreads` non plus. Un programme lié avec `−static` est légèrement plus grand qu'un programme liée dynamiquement mais aussi un peu plus rapide (3−5%). Un problème, toutefois, est que vous ne pouvez utiliser de fonctions définies par l'utilisateur avec un programme lié statiquement. Si vous allez écrire ou utiliser des fonctions `UDF` (c'est réservé aux développeurs C ou C++), vous devez compiler MySQL vous−même, en utilisant les liaisons dynamiques.

Si vous utilisez un système basé sur `libc` (au lieu de `glibc2`), vous aurez probablement quelques problèmes de résolution des noms d'hôtes et des problèmes avec `getpwnam()` avec les versions binaires. (Cela vient du fait que `glibc` dépend malheureusement de quelques bibliothèques externes pour résoudre les noms d'hôtes et `getpwent()` , même quand elle est compilée avec `−static`). Dans ce cas, vous obtiendrez probablement l'erreur suivante quand vous exécuterez `mysql_install_db` :

```
Sorry, the host 'xxxx' could not be looked up
```

ou l'erreur suivante quand vous essayez de démarrer `mysqld` avec l'option `--user` :

```
getpwnam: No such file or directory
```

Vous pouvez résoudre ce problème de la façon suivante :

- Obtenez une distribution des sources MySQL (un distribution `RPM` ou le `tar.gz`) et installez la à la place.
- Exécutez `mysql_install_db --force` ; cela n'exécutera pas le test `resolveip` dans `mysql_install_db` . Le mauvais côté est que vous ne pourrez pas utiliser de noms d'hôtes dans les tables de droits; vous devez utiliser les adresses IP à la place (sauf pour `localhost`). Si vous utilisez une vielle version de MySQL qui ne supporte pas `--force` , vous devez supprimer le test `resolveip` dans `mysql_install` à l'aide d'un éditeur.
- Démarrez `mysqld` avec `su` au lieu d'utiliser `--user` .

Le binaire Linux−Intel et les `RPM` de MySQL sont configurés pour la vitesse la plus grande possible. Nous essayons toujours d'utiliser le compilateur le plus rapide disponible.

Le support Perl de MySQL requière la version 5.004_03 de Perl ou plus récent.

Sur quelques version de Linux 2.2, vous pouvez obtenir l'erreur `Resource temporarily unavailable` quand vous faites beaucoup de nouvelles connexions à un serveur `mysqld` en utilisant TCP/IP.

Le problème est que Linux possède un délai entre votre fermeture de la socket TCP/IP et sa libération par le système. Vu qu'il y a un nombre fini de places pour les branchements TCP/IP, vous obtiendrez l'erreur précédente si vous essayez de faire beaucoup de connexions TCP/IP en peu de temps, comme quand vous exécutez le benchmark MySQL `test-connect` via TCP/IP.

Nous avons envoyé des questions plusieurs fois à propos de ce problème à différentes listes de diffusions Linux mais n'avons jamais réussi à résoudre ce problème proprement.

Le seul correctif connu pour ce problème est d'utiliser des connexions persistantes dans vos clients ou d'utiliser les sockets, si vous utilisez le serveur de bases de données et le client sur la même machine. Nous espérons que le noyau de `Linux 2.4` corrigera ce problème bientôt.

2.6.1.3 Notes sur la distribution source de Linux

Les notes suivantes concernant `glibc` ne s'appliquent que si vous compilez vous−mêmes `MySQL` . Si vous utilisez Linux sur une machine x86, dans la plupart des cas, il est mieux d'utiliser notre bibliothèque. Nous compilons nos exécutables avec les meilleures versions corrigées de `glibc` que nous pouvons trouver, avec les meilleures options de compilation possible, pour faire que notre serveur supporte les hautes charges. Pour un utilisateur typique, même pour une configuration avec de nombreuses connexions ou des tables dépassant les 2 Go, notre programme est meilleur choix. Après avoir lu ce texte, si vous doutez toujours, testez notre compilation pour voir si elle répond à vos besoins. Si vous pensez qu'elle n'est pas à la hauteur, alors essayez de compiler par vous−mêmes. Dans ce cas, nous apprécierons de savoir ce que vous avez fait, pour pouvoir améliorer notre propre version.

`MySQL` utilise les `LinuxThreads` sur Linux. Si vous utilisez une vieille version de Linux qui n'a pas `glibc2` , vous devrez installer `LinuxThreads` avant de compiler `MySQL` . Vous pouvez obtenir `LinuxThreads` sur http://dev.mysql.com/downloads/os−linux.html .

Notez que les versions de `glibc` avant et incluant la version 2.1.1 ont un bug critique dans la gestion de `pthread_mutex_timedwait()` , qui est utilisée lorsque vous envoyez des commandes `INSERT DELAYED` . Nous vous recommandons de ne pas utiliser `INSERT DELAYED` avant de mettre à jour `glibc` .

Notez que le noyau Linux et la bibliothèque `LinuxThreads` sont limitées par défaut à 1024. Si vous envisagez d'avoir plus de 1000 connexions simultanées, vous aurez besoin de faire des changement dans `LinuxThreads` :

- Augmentez `PTHREAD_THREADS_MAX` dans `sysdeps/unix/sysv/linux/bits/local_lim.h` à 4096 et réduisez `STACK_SIZE` dans `linuxthreads/internals.h` à 256 ko. Les chemins sont relatifs à la racine de `glibc` . (Notez que MysQL ne sera pas stable autour de 600 à 1000 connexions si `STACK_SIZE` vaut les 2 Mo par défaut.
- Recompilez `LinuxThreads` pour avoir un nouveau `libpthread.a` , et recompilez `MySQL` avec.

La page http://www.volano.com/linuxnotes.html contient des informations supplémentaires pour contourner cette limite de `LinuxThreads` .

Il y a un autre problème qui limite considérablement les performances de `MySQL` , notamment sur des systèmes multi−processeurs. L'implémentation mutex de `LinuxThreads` dans `glibc` 2.1 est très mauvaise pour les programmes ayant de nombreux threads qui gardent le mutex pour une courte période de temps. Cela produit un résultat paradoxal : si vous compilez `MySQL` avec un version originale de `LinuxThreads` , supprimer des processeurs de votre architecture va améliorer les performances. Nous avons fait un correctif de `glibc` 2.1.3 pour corriger ce comportement (http://www.mysql.com/Downloads/Linux/linuxthreads−2.1−patch).

Avec `glibc` 2.2.2, `MySQL` 3.23.36 va utiliser un mutex adaptatif, qui est bien meilleur que la version corrigée de `glibc` 2.1.3. Soyez prévenu, que dans certaines conditions, le code courant du mutex de `glibc` 2.2.2 surchauffe, ce qui bride `MySQL` . La probabilité de rencontrer cette condition sera réduite en donnant à `mysqld` la plus haute priorité. Nous avons été capable de corriger le problème avec un correctif, disponible à http://www.mysql.com/Downloads/Linux/linuxthreads−2.2.2.patch . Il combine la correction avec l'augmentation du nombre limite de threads et de la taille de pile. Vous devez l'appliquer dans le dossier `linuxthreads` avec `patch -p0 </tmp/linuxthreads-2.2.2.patch` . Nous espérons qu'il sera inclut dans une version future de `glibc` 2.2. Dans tous les cas, si vous compilez avec `glibc` 2.2.2, vous devrez corriger `STACK_SIZE` et `PTHREAD_THREADS_MAX` . Nous espérons que les valeurs par défaut seront corrigées et remplacées par des valeurs plus raisonnables pour les configurations à haut rendement de `MySQL` : les manipulations futures pour compiler `MySQL` seront alors réduites à `./configure; make; make install` .

Nous recommandons que vous utilisiez ces correctifs pour compiler une version spéciale statique de `libpthread.a` et que vous l'utilisiez pour compiler `MySQL` . Nous savons que les correctifs sont sécuritaires pour `MySQL` , et améliore significativement les performances, mais nous ne pouvons pas nous avancer pour les autres applications. Si vous compilez d'autres applications qui requièrent les `LinuxThreads` avec la version statique corrigée de la bibliothèque, faîtes le à vos risques et périls.

Si vous rencontrez des problèmes étranges durant l'installation de `MySQL` , ou si vous voyez les utilitaires se geler, il est très probable que vous ayez un problème de compilateur ou de bibliothèque. Dans ce cas, utiliser notre binaire résoudra vos problèmes.

Si vous compilez vos propres clients `MySQL` , vous pouvez rencontrer l'erreur suivante durant l'exécution :

```
ld.so.1: fatal: libmysqlclient.so.#:
open failed: No such file or directory
```

Ce problème peut être évité avec les méthodes suivantes :

* Compilez le client avec l'option `-Wl,r/full/path/to/libmysqlclient.so` plutôt que `-Lpath`).
* Copiez `libmysqclient.so` dans `/usr/lib` .
* Ajoutez le chemin du dossier où `libmysqlclient.so` est situé dans la variable d'environnement `LD_RUN_PATH` avant de lancer votre client.

Si vous utilisez le compilateur Fujitsu (`fcc/FCC`), vous aurez des problèmes pour compiler `MySQL` car le fichier d'entête Linux est très orienté `gcc` . La ligne de configuration `configure` devrait fonctionner avec `fcc/FCC` :

```
 CC=fcc CFLAGS="-O -K fast -K lib -K omitfp -Kpreex -D_GNU_SOURCE \
    -DCONST=const -DNO_STRTOLL_PROTO" \
CXX=FCC CXXFLAGS="-O -K fast -K lib \
    -K omitfp -K preex --no_exceptions --no_rtti -D_GNU_SOURCE \
    -DCONST=const -Dalloca=__builtin_alloca -DNO_STRTOLL_PROTO \
    '-D_EXTERN_INLINE=static __inline'" \
./configure \
    --prefix=/usr/local/mysql --enable-assembler \
    --with-mysqld-ldflags=-all-static --disable-shared \
    --with-low-memory
```

/

2.6.1.4 Notes de post−installation pour Linux

`mysql.server` est stocké dans le dossier `support-files` dans le dossier d'installation MySQL, ou dans le dossier des sources. Vous pouvez l'installer dans `/etc/init.d/mysql` pour assurer le démarrage et l'extinction automatique de MySQL. Lancer et arrêter automatiquement MYSQL .

Si MySQL n'arrive pas à ouvrir assez de fichiers, ou à créer assez de connexions, il se peut que vous n'ayez pas configuré Linux pour qu'il gère assez de fichiers.

Dans Linux 2.2 ou plus, vous pouvez connaître le nombre de gestionnaires de fichiers alloués en faisant :

```
 shell> cat /proc/sys/fs/file-max
shell> cat /proc/sys/fs/dquot-max
shell> cat /proc/sys/fs/super-max
```

Si vous avez plus de 16 Mo de mémoire, vous devez ajouter quelque chose comme ce qui suit dans vos scripts d'initialisation (`/etc/init.d/boot.local` sur SuSE Linux) :

```
 echo 65536 > /proc/sys/fs/file-max
echo 8192 > /proc/sys/fs/dquot-max
echo 1024 > /proc/sys/fs/super-max
```

Vous pouvez aussi exécuter les commandes précédentes à partir de la ligne de commande en tant que root, mais les changements seront perdus au prochain redémarrage de l'ordinateur.

Vous pouvez sinon définir ces paramètres lors du démarrage de la machine en utilisant l'outil `sysctl` , qui est utilisé par plusieurs distributions Linux (SuSE l'a aussi ajouté, à partir de SuSE Linux 8.0). Ajoutez simplement les valeurs suivantes dans un fichier nommé `/etc/sysctl.conf` :

```
 # Increase some values for MySQL
fs.file-max = 65536
fs.dquot-max = 8192
fs.super-max = 1024
```

Il est recommandé d'ajouter aussi la ligne suivante dans le fichier `/etc/my.cnf` :

```
 [mysqld_safe]
open-files-limit=8192
```

Cela va autoriser le serveur à un maximum de 8192 de connexions et fichiers ouvertes simultanément.La constante `STACK_SIZE` des `LinuxThreads` contrôle l'espacement des piles de threads dans l'espace d'adressage. Elle doit être assez grande pour qu'il y ait plusieurs chambres pour la pile de chaque thread individuel, mais assez petite pour empêcher les piles de certains threads d'agir sur les données globales de `mysqld` . Malheureusement, l'implémentation Linux de `mmap()` , comme nous l'avons découvert, va libérer une région réservée, si vous lui demandez de libérer une adresse déjà utilisée, détruisant les données de la page, au lieu de retourner une erreur. Donc, la sécurité de `mysqld` et des autres applications qui dépendent d'un comportement civils du code qui gère les threads. L'utilisateur doit s'assurer que le nombre de threads fonctionnant simultanément est suffisamment bas pour éviter d'entrer dans la pile globale. Avec `mysqld` , vous devez suivre cette règle de bon fonctionnement en donnant une valeur raisonnable à `max_connections` .

Si vous compilez MySQL vous−mêmes, vous pouvez corriger `LinuxThreads` pour améliorer l'utilisation de la pile. Notes pour les distributions sources de Linux . Si vous ne voulez pas corriger `LinuxThreads` , vous ne devez pas dépasser 500 pour la valeur de `max_connections` . Cela devrait même être moins si vous avez un tampon de clefs assez large, de grosses tables heap, ou d'autres choses qui peuvent faire allouer beaucoup de mémoire à `mysqld` , ou si vous utilisez un noyau 2.2 avec un patch 2G. Si vous utilisez notre binaire ou `RPM` 3.23.25 ou plus, vous pouvez mettre `max_connections` à 1500 sans problèmes, en supposant que vous n'avez ni de grosses tables heap ni grands tampons de clefs. Plus vous réduirez `STACK_SIZE` dans `LinuxThreads` plus les threads créés seront sûrs. Nous recommandons une valeur entre 128 ko et 256 ko.

Si vous utilisez beaucoup de connexions simultanées, vous pouvez souffrir d'une ``fonctionnalité'' du noyau 2.2, qui tente d'éviter les DOS par fork en pénalisant les processus qui forkent ou qui clonent des fils. Cela fait que MySQL ne se comporte pas bien si vous augmentez le nombre de clients simultanés. Sur les systèmes mono−processeurs, nous avons vu des symptômes sous la forme de ralentissement : il prenait un très long temps pour se connecter (parfois une minute), et il fallait autant de temps pour terminer le processus. Sur les systèmes multi−processeurs, nous avons observé une décroissance graduelle des performances des requêtes chez de nombreux clients. Durant nos recherches pour corriger le problème, nous avons reçu un patch d'un client qui prétendait avoir résolu le problème pour son site. Ce patch est disponible sur http://www.mysql.com/Downloads/Patches/linux−fork.patch . Nous avons maintenant fait des tests exhaustifs de ce patch en développement et en production. Il a amélioré significativement les performances sans causer de problèmes, et nous l'avons recommandé à nos utilisateurs qui fonctionnent avec des serveurs chargés et un noyau 2.2.

Ce problème a été réglé avec le noyau 2.4 : si vous n'êtes pas satisfait avec les performances courantes de votre système, au lieu de le corriger, passez donc votre noyau 2.2 en 2.4. Sur les systèmes multi−processeurs, la mise à jour vous donnera d'ailleurs un regain de puissance, en plus de corriger le bug.

Nous avons testé MySQL sur des noyaux 2.4 et sur des machines bi−processeurs, et nous avons trouvé que MySQL se comporte **beaucoup** mieux. Il n'y avait pratiquement pas de ralentissement de requêtes même avec 1000 client, et gain de puissance était de 180% (calculé avec le ratio de vitesse maximale divisé par la vitesse moyenne d'un client). Nous avons observé des résultats similaires sur une machine quadri−processeurs : virtuellement aucun ralentissement alors que le nombre de clients est monté jusqu'à 1000, et le gain de puissance a atteind 300%. En se basant sur ces résultats, pour un serveur haute performances multi−processeurs, nous vous recommandons de passer en noyau 2.4.

Nous avons découvert qu'il est essentiel de faire fonctionner les processus `mysqld` avec la priorité maximal sur le noyau 2.4 pour atteindre les meilleures performances. Cela peut se faire en ajoutant la commande `renice -20 $$` dans `mysqld_safe` . Durant nos tests sur une machine quadri−processeurs, augmenter la priorité a engendré 60% d'amélioration avec 400 clients.

Nous essayons aussi de rassembler plus d'informations sur comment MySQL se comporte sur un système 2.4 quadri− ou octo−processeurs. Si vous avez accès a de telles données, envoyez nous un email à benchmarks@mysql.com avec les résultats. Nous allons les étudier pour les inclure dans le manuel.

Si vous voyez un processus `mysqld` mort avec `ps` , c'est que vous avez découvert un bug dans MySQL ou qu'une des tables est corrompue. Que faire si MySQL crashe constamment .

Pour obtenir un `core dump` sur Linux si `mysqld` se termine avec un signal `SIGSEGV` , vous pouvez lancer `mysqld` avec l'option `--core-file` . Notez que vous aurez probablement à augmenter la taille du fichier core en ajoutant la commande `ulimit -c 1000000` à `mysqld_safe` ou en lançant `mysqld_safe` avec `--core-file-size=1000000` . `safe_mysqld` , le script père de `mysqld` .

2.6.1.5 Notes relatives à Linux x86

MySQL requière la version 5.4.12 de `libc` ou plus récent. Il est connu pour fonctionner avec `libc` 5.4.46. La version 2.0.6 de `glibc` ou plus récente devrait aussi fonctionner. Il y a eu quelques problèmes avec les `RPM` de `glibc` de Red Hat, et donc, si vous avez des problèmes, vérifiez s'il existe des mises à jour ! Les `RPM` de `glibc` 2.0.7−19 et 2.0.7−29 sont connus pour fonctionner.

Si vous utilisez `gcc` 3.0 ou plus récent pour compiler MySQL, vous devez installer la bibliothèque `libstdc++v3` avant de compiler MySQL; si vous ne le faites pas vous obtiendrez une erreur à propos d'un symbole `__cxa_pure_virtual` manquant durant la liaison! Pour corriger ce problème, lancez `mysqld` avec l'option `--thread-stack=192K` . Utilisez la syntaxe `-O thread_stack=192K` avant MySQL 4.) La taille de la pile est maintenant par défaut pour les versions MySQL 4.0.10 et plus récente, alors vous ne devriez pas rencontrer de problème.

Si vous utilisez `gcc` 3.0 et plus récent pour compiler MySQL, vous devez installer la bibliothèque `libstdc++v3` avant de compiler MySQL; si vous ne le faites pas, vous aurez des erreurs à propos de `__cxa_pure_virtual` qui manque, durant la résolution des symboles.

Sur quelques vieilles distributions de Linux, `configure` peut produire une erreur comme celle qui suit :

```
 Syntax error in sched.h. Change _P to __P in the
/usr/include/sched.h file.
See the Installation chapter in the Reference Manual.
```

Faites ce que le message d'erreur dit et ajoutez un `_` à la macro `_P` qui n'en a qu'un, puis essayez à nouveau.Vous pouvez obtenir quelques avertissements en compilant; celles qui suivent peuvent être ignorées :

```
 mysqld.cc -o objs-thread/mysqld.o
mysqld.cc: In function `void init_signals()':
mysqld.cc:315: warning: assignment of negative value `-1' to
`long unsigned int'
mysqld.cc: In function `void * signal_hand(void *)':
mysqld.cc:346: warning: assignment of negative value `-1' to
`long unsigned int'
```

Si `mysqld` provoque toujours un plantage au démarrage, le problème peut être que vous avez un vieux `/lib/libc.a` . Renommez le, puis supprimez `sql/mysqld` et faites à nouveau un `make install` puis réessayez. Ce problème a été reporté sur quelques installations de Slackware.Si vous obtenez l'erreur suivante en liant `mysqld` , cela signifie que votre `libg++.a` n'est pas installé correctement :

```
 /usr/lib/libc.a(putc.o): In function `_IO_putc':
putc.o(.text+0x0): multiple definition of `_IO_putc'
```

Vous pouvez éviter d'utiliser `libg++.a` en exécutant `configure` comme suit :

```
 shell> CXX=gcc ./configure
```

Si `mysqld` se plante immédiatement, et que vous utilisez Red Hat Version 5.0, avec une version de `glibc` plus ancienne que 2.0.7−5, il est recommandé d'installer les patchs `glibc` . Il y a beaucoup d'informations à ce sujet dans les archives courriel, disponibles sur http://lists.mysql.com/ .

2.6.1.6 Notes relatives à Linux SPARC

Sur quelques implémentations, `readdir_r()` est cassé. Le symptôme est que `SHOW DATABASES` retourne toujours un résultat vide. Cela peut être corrigé en supprimant `HAVE_READDIR_R` de `config.h` après avoir configuré et avant de commencer à compiler.

2.6.1.7 Notes relatives à Linux Alpha

La version 3.23.12 de MySQL est la première version de MySQL à être testée sur Linux−Alpha. Si vous voulez utiliser MySQL sur Linux−Alpha, vous devez vous assurer d'avoir cette version ou une version plus récente.

Nous avons testé MySQL sur Alpha avec nos tests de performance et notre suite de tests : tout semble fonctionner correctement.

Nous construisons actuellement les paquets binaires de MySQL sur `SuSE Linux 7.0` pour `AXP` , `kernel 2.4.4-SMP` , `Compaq C compiler (V6.2-505)` et `Compaq C++ compiler (V6.3-006)` sur une machine `Compaq DS20` avec un processeur `Alpha EV6` .

Vous pouvez trouver les précédents compilateurs sur http://www.support.compaq.com/alpha−tools/). En utilisant ces compilateurs, au lieu de `gcc` , nous obtenons environs 9 à 14% d'améliorations des performances avec MySQL.

Notez que jusqu'aux versions 3.23.52 et 4.0.2 de MySQL nous avons optimisé l'application pour le processeur courant seulement (en utilisant l'option de compilation `-fast`); cela signifiait que vous ne pouviez utiliser nos binaires si vous n'aviez pas un processeur Alpha EV6.

Avec les versions suivantes nous avons ajouté l'option `-arch generic` à nos options de compilation, ce qui assure que le binaire fonctionne sur tout les processeurs Alpha. Nous compilons aussi statiquement pour éviter les problèmes de bibliothèques.

```
 CC=ccc CFLAGS="-fast -arch generic" CXX=cxx \
CXXFLAGS="-fast -arch generic -noexceptions -nortti" \
./configure --prefix=/usr/local/mysql --disable-shared \
--with-extra-charsets=complex --enable-thread-safe-client \
--with-mysqld-ldflags=-non_shared --with-client-ldflags=-non_shared
```

Si vous voulez utiliser `egcs` la ligne de configuration suivante a fonctionné pour nous :

```
 CFLAGS="-O3 -fomit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
--disable-shared
```

Quelques problèmes connus lors de l'utilisation de MySQL sur Linux−Alpha:

- Le débogage d'applications threadées comme MySQL ne fonctionnera pas avec `gdb 4.18` . Vous devez télécharger et utiliser `gdb 5.1` à la place !
- Si vous essayez de lier statiquement `mysqld` en utilisant `gcc` , l'image résultante videra son noyau (`core dump`) au démarrage. En d'autres termes, **n'utilisez pas** `--with-mysqld-ldflags=-all-static` avec `gcc` .

2.6.1.8 Note relative à Linux PowerPC

MySQL devrait fonctionner sur MkLinux avec le dernier paquet `glibc` (testé avec `glibc` 2.0.7).

2.6.1.9 Notes relatives à Linux MIPS

Pour faire fonctionner MySQL sur Qube2, (Linux Mips), vous aurez besoin de la bibliothèque `glibc` la plus récente (`glibc-2.0.7-29C2` est connue pour marcher). Vous devez aussi utiliser le compilateur `egcs` C++ (`egcs-1.0.2-9` , `gcc 2.95.2` ou plus récent).

2.6.1.10 Notes relatives à Linux IA64

Pour pouvoir compiler MySQL sous Linux IA64, nous utilisons les lignes de compilation suivante : En utilisant `gcc-2.96` :

```
 CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
"--with-comment=Official MySQL binary" --with-extra-charsets=complex
```

Sur IA64 les binaires des clients MySQL utilisent des bibliothèques partagées. Cela signifie que si vous installez notre distribution binaire à un autre endroit que `/usr/local/mysql` vous devez modifier le fichier `/etc/ld.so.conf` ou ajouter le chemin vers le répertoire où vous avez `libmysqlclient.so` à la variable d'environnement `LD_LIBRARY_PATH` .

Problèmes en reliant avec la bibliothèque du client MySQL .

2.6.2 Notes relatives à Mac OS X

Sous Mac OS X, `tar` ne sais pas gérer les noms de fichiers longs. Si vous avez besoin de décompresser la distribution `.tar.gz` , utilisez `gnutar` .

2.6.2.1 Mac OS X 10.x

MySQL devrait fonctionner sans problème avec les versions Mac OS X 10.x (Darwin). Vous n'avez pas besoin du patch pour les pthreads sur cet OS!

Cela s'applique aussi à Mac OS X 10.x Server. La compilation pour la plate−forme serveur est identique à la compilation pour pa version client de Mac OS X. Toutefois, notez que MySQL est pre−installé sur la version `Server` !

Notre exécutable Mac OS X est compilé sous Darwin 6.3 avec les options de `configure` suivantes :

```
 CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
--with-extra-charsets=complex --enable-thread-safe-client \
--enable-local-infile --disable-shared
```

Installer MySQL sous Mac OS X .

2.6.2.2 Mac OS X Server 1.2 (Rhapsody)

Avant d'essayer de configurer MySQL sur `Mac OS X Server` , vous devez d'abord installer le paquet pthread. Ce n'est plus nécessaire avec les versions récentes du serveur.

Installer MySQL sous Mac OS X .

2.6.3 Notes pour Solaris

Sous Solaris, vous pouvez rencontrer des problèmes avant même d'avoir désarchivé la distribution MySQL! Le programme `tar` de Solaris ne peut pas manipuler de noms de fichiers longs, provoquant les messages suivants quand vous décompressez MySQL :

```
x mysql-3.22.12-beta/bench/Results/ATIS-mysql_odbc-NT_4.0-cmp-db2,\
informix,ms-sql,mysql,oracle,solid,sybase, 0 bytes, 0 tape blocks
tar: directory checksum error
```

Dans ce cas, vous devez utiliser GNU `tar` (`gtar`) pour désarchiver la distribution. Vous pouvez en trouver une copie précompilée pour Solaris sur http://www.mysql.com/Downloads/ .La gestion native des threads Sun fonctionne uniquement depuis Solaris 2.5. Pour les versions 2.4 et antérieures, MySQL utilisera automatiquement les `MIT-pthreads` . Remarques sur MIT−pthreads .

Vous pouvez rencontrer les erreurs suivantes lors du configure:

```
checking for restartable system calls... configure: error can not run test
programs while cross compiling
```

Cela signifie que l'installation de votre compilateur est défectueuse! Dans ce cas, vous devez mettre à jour votre compilateur en faveur d'une version plus récente. Vous pouvez aussi résoudre le problème en innérant la ligne suivante dans le fichier `config.cache` :

```
ac_cv_sys_restartable_syscalls=$&#123;ac_cv_sys_restartable_syscalls='no'&#125;
```

Si vous utilisez Solaris sur une architecture SPARC, nous recommandons `gcc` 2.95.2 comme compilateur. Vous pouvez le trouver sur http://gcc.gnu.org/ . Notez que `egcs` 1.1.1 et `gcc` 2.8.1 ne fonctionnent pas correctement sur SPARC!La ligne `configure` recommandée dans le cas de l'utilisation de `gcc` 2.95.2 est:

```
CC=gcc CFLAGS="-O3" \
CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory --enable-assembler
```

Si vous avez une machine UltraSPARC, vous pouvez gagner 4% de performances supplémentaires en ajoutant " `-mcpu=v8 -Wa,-xarch=v8plusa` " à CFLAGS et CXXFLAGS.Si vous utilisez le compilateur Forte 5.0 (et supérieur) de Sun, vous pouvez lancer `configure` de la façon suivante :

```
CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt" \
CXX=CC CXXFLAGS="-noex -mt" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Vous pouvez créer un binaire 64 bits avec :

```
CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt -xarch=v9" \
CXX=CC CXXFLAGS="-noex -mt -xarch=v9" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Lors de bancs de tests MySQL, nous avons gagné 4% en vitesse sur une UltraSPARC en utilisant Forte 5.0 en mode 32 bits plutôt que gcc 3.2 avec les marqueurs −mcpu.

Si vous créez un binaire 64 bits, il est de 4% plus lent que le binaire 32 bits, mais en contrepartie vous pouvez gérer davantage de treads et de mémoire.

Si vous rencontrez des problèmes avec `fdatasync` ou `sched_yield` , vous pouvez les résoudre en ajoutant `LIBS=-lrt` à la ligne configure.

Le paragraphe suivant ne s'applique qu'aux compilateurs plus anciens que WorkShop 5.3 :

Vous pouvez avoir à modifier le script `configure` et changer la ligne :

```
#if !defined(__STDC__) || __STDC__ != 1
```

en :

```
#if !defined(__STDC__)
```

Si vous activez `__STDC__` avec l'option `-Xc` , le compilateur Sun ne peut pas compiler avec le fichier d'entêtes `pthread.h` de Solaris. C'est un bogue de Sun (compilateur ou fichier d'inclusion défectueux).

Si `mysqld` génère les messages d'erreur suivants lorsque vous le lancez, cela est du au fait que vous avez compilé MySQL avec le compilateur de Sun sans activer l'option multi−threads (`-mt`):

```
libc internal error: _rmutex_unlock: rmutex not held
```

Ajoutez `-mt` à `CFLAGS` et `CXXFLAGS` puis réessayez.

Si vous utilisez la version SFW de gcc (fournie avec Solaris 8), vous devez ajouter `/opt/sfw/lib` à la variable d'environnement `LD_LIBRARY_PATH` avant de lancer le configure.

Si vous utilisez le gcc disponible sur `sunfreeware.com` , vous pouvez rencontrer de nombreux problèmes. Vous devriez recompiler gcc et les GNU binutils sur la machine à partir de laquelle vous les utiliserez, afin d'éviter tout souci.

Si vous obtenez l'erreur suivante lorsque vous compilez MySQL avec `gcc` , cela signifie que votre `gcc` n'est pas configuré pour votre version de Solaris :

```
shell> gcc -O3 -g -O2 -DDBUG_OFF  -o thr_alarm ...
./thr_alarm.c: In function `signal_hand':
./thr_alarm.c:556: too many arguments to function `sigwait'
```

La meilleure chose à faire dans ce cas est d'obtenir la version la plus récente de `gcc` et de compiler avec votre `gcc` actuel! Au moins pour Solaris 2.5, la plupart des versions binaires de `gcc` ont d'anciens fichiers d'inclusion inutilisables qui planteront les programmes qui utilisent les threads (ainsi probablement d'autres programmes)!

Solaris ne fournit pas de versions statiques de toutes les bibliothèques système (`libpthreads` et `libdl`), vous ne pouvez donc pas compiler MySQL avec `--static` . Si vous tentez de le faire, vous obtiendrez l'erreur :

```
ld: fatal: library -ldl: not found
undefined reference to `dlopen'
cannot find -lrt
```

Si de nombreux processus essaient de se connecter très rapidement à `mysqld` , vous verrez cette erreur dans le journal MySQL :

```
Error in accept: Protocol error
```

Pour éviter cela, vous pouvez lancer le serveur avec l'option `--set-variable back_log=50` . Veuillez noter que `--set-variable` est déprécié depuis MySQL 4.0, utilisez uniquement `--back_log=50` .Si vous liez votre propre client MySQL, vous pouvez avoir l'erreur suivante quand vous le lancez :

```
ld.so.1: ./my: fatal: libmysqlclient.so.#:
open failed: No such file or directory
```

Le problème peut être évité avec l'une des méthodes suivantes :

- Liez le client avec le marqueur suivant (à la place de `-Lpath`) : `-Wl,r/full-path-to-libmysqlclient.so` .
- Copiez `libmysqclient.so` dans `/usr/lib` .
- Ajoutez le chemin du répertoire où `libmysqlclient.so` est installé à la variable d'environnement `LD_RUN_PATH` avant de lancer votre client.

Si vous avez des soucis avec configure qui essaie de lier avec `-lz` et que vous n'avez pas installé `zlib` , vous avez deux solutions :

- Si vous voulez utiliser le protocole compressé de communication, vous devrez vous procurer et installer zlib sur ftp.gnu.org.
- Configurez avec `--with-named-z-libs=no` .

Si vous utilisez gcc et rencontrez des problèmes en chargeant la fonction `UDF` dans MySQL, essayez d'ajouter `-lgcc` à la ligne de liaison de la fonction `UDF` .

Si vous voulez que MySQL se lance automatiquement, vous pouvez copier `support-files/mysql.server` dans `/etc/init.d` et créer un lien symbolique pointant dessus et s'appelant `/etc/rc3.d/S99mysql.server` .

Comme Solaris ne supporte pas les fichiers core pour les applications `setuid()` , vous ne pouvez pas obtenir un fichier core de `mysqld` si vous utilisez l'option `--user` .

2.6.3.1 Notes relatives à Solaris 2.7/2.8

Vous pouvez normalement utiliser les binaires Solaris 2.6 sur Solaris 2.7 et 2.8. La plupart des fonctionnalitées de Solaris 2.6 s'appliquent aussi à Solaris 2.7 et 2.8.

Notez que la version 3.23.4 de MySQL et plus doivent être capables de détecter automatiquement les nouvelles versions de Solaris et d'activer les parades pour résoudre les problèmes suivants !

Solaris 2.7 / 2.8 ont quelques bogues dans les fichiers inclus. Vous verrez peut−être l'erreur suivante en utilisant `gcc` :

```
 /usr/include/widec.h:42: warning: `getwc' redefined
/usr/include/wchar.h:326: warning: this is the location of the previous
definition
```

Si cela arrive, vous pouvez faire ce qui suit pour résoudre ce problème :

Copiez `/usr/include/widec.h` vers `.../lib/gcc-lib/os/gcc-version/include` et changez la ligne 41 de :

```
#if     !defined(lint) && !defined(__lint)
```

en :

```
#if     !defined(lint) && !defined(__lint) && !defined(getwc)
```

Alternativement, vous pouvez éditer directement le fichier `/usr/include/widec.h` . De toutes façons, après avoir apporté la correction, vous devez effacer `config.cache` et exécuter `configure` à nouveau !Si vous obtenez des erreurs comme celles qui suivent quand vous exécutez `make` , c'est parce que `configure` n'a pas détecté le fichier `curses.h` (probablement à cause de l'erreur dans `/usr/include/widec.h`) :

```
 In file included from mysql.cc:50:
/usr/include/term.h:1060: syntax error before `,'
/usr/include/term.h:1081: syntax error before `;'
```

La solution est de faire l'une des choses qui suit :

- Configurez avec `CFLAGS=-DHAVE_CURSES_H CXXFLAGS=-DHAVE_CURSES_H ./configure` .
- Editez `/usr/include/widec.h` comme indiqué plus haut et ré−exécutez configure.
- Effacez la ligne `#define HAVE_TERM` di fichier `config.h` et exécutez `make` à nouveau.

Si vous obtenez une erreur disant que votre programme de liaison ne peut trouver `-lz` lors de la liaison du programme de votre client, le problème est probablement que votre fichier `libz.so` est installé dans `/usr/local/lib` . Vous pouvez corriger ceci en utilisant l'une des méthodes suivantes :

- Ajoutez `/usr/local/lib` à `LD_LIBRARY_PATH` .
- Ajoutez un lien vers `libz.so` à partir de `/lib` .
- Si vous utilisez Solaris 8, vous pouvez installer la zlib optionnelle à partir de votre CD Solaris 8.
- Configurez MySQL avec l'option `--with-named-z-libs=no` .

2.6.3.2 Remarques pour Solaris x86

Sous Solaris 2.8 sur x86, `mysqld` va crasher (core dump) si vous l'exécutez 'strip'.

Si vous utilisez `gcc` ou `egcs` sous Solaris x86 et que vous rencontrez des problèmes avec des coredumps, lorsqu'il y a de la charge, il est recommandé d'utiliser la commande de `configure` suivante :

```
 CC=gcc CFLAGS="-O3 -fomit-frame-pointer -DHAVE_CURSES_H" \
CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors -fno-exceptions \
-fno-rtti -DHAVE_CURSES_H" \
./configure --prefix=/usr/local/mysql
```

Cela va éviter les problèmes avec la bibliothèque `libstdc++` et avec les exceptions C++.

Si cela ne vous aide pas, il est recommandé de compiler une version de débogage, et de l'exécuter avec un fichier de trace sous `gdb` . Déboguer mysqld sous gdb .

2.6.4 Notes relatives à BSD

Cette section fournit des informations pour les différentes variétés de BSD, ainsi que les versions spécifiques de celles−ci.

2.6.4.1 Notes relatives à FreeBSD

FreeBSD 4.x est recommandé pour exécuter MySQL vu que le paquet des threads est plus intégré.

La façon la plus facile et la plus conseillée d'installer est d'utiliser les ports du serveur et du client MySQL disponibles sur http://www.freebsd.org/ .Les utiliser vous donnera :

- Un MySQL fonctionnant avec toutes les optimisations connues pour votre version active de FreeBSD.
- Configuration et construction automatique.
- Scripts de démarrage installés dans `/usr/local/etc/rc.d` .
- La possibilité de voir tous les fichiers installés avec `pkg_info -L` .
- La possibilité de les effacer tous avec `pkg_delete` si vous ne voulez plus de MySQL sur cette machine.

Il est recommandé d'utiliser les `MIT-pthreads` sur FreeBSD 2.x et les threads natifs sur les versions 3 et plus. Il est possible de faire fonctionner le tout avec les threads natifs sur les dernières versions 2.2.x mais vous rencontrerez probablement des problèmes en coupant `mysqld` .

Malheureusement, certains appels systèmes sur FreeBSD ne sont pas encore totalement compatibles avec les threads. Le cas le plus notable est la fonction `gethostbyname()` qui est utilisée par MySQL pour convertir des noms d'hôtes en adresse IP. Dans certaines circonstances, le processus `mysqld` va soudainement prendre 100% du processeur, et ne plus répondre. Si vous rencontrez cette situation, essayez de relancer MySQL avec l'option `--skip-name-resolve` .

Alternativement, vous pouvez compiler MySQL sur FreeBSD 4.x avec la bibliothèque `LinuxThreads` , qui évite les quelques problèmes que l'implémentation thread native de FreeBSD a. Pour une bonne comparaison entre `LinuxThreads` et les threads natifs, voyez l'article de Jeremy Zawodny `FreeBSD or Linux for your MySQL Server?` à http://jeremy.zawodny.com/blog/archives/000697.html .

Un problème connu lors de l'utilisation des LinuxThreads sur FreeBSD est que wait_timeout ne fonctionne pas (probablement un problème de gestion des signaux sous FreeBSD/ LinuxThreads). Cela devrait être corrigé en FreeBSD 5.0. Le symptôme est que les connexions persistantes vont rester à bloquées très longtemps sans se refermer.

Le processus de compilation de MySQL requiert GNU make (gmake) pour fonctionner. Si vous voulez compiler MySQL vous devez d'abord installer GNU make .

La méthode recommandée pour compiler et installer MySQL sur FreeBSD avec gcc (2.95.2 et plus récent) est :

```
 CC=gcc CFLAGS="-O2 -fno-strength-reduce" \
    CXX=gcc CXXFLAGS="-O2 -fno-rtti -fno-exceptions \
    -felide-constructors -fno-strength-reduce" \
    ./configure --prefix=/usr/local/mysql --enable-assembler
gmake
gmake install
cd /usr/local/mysql
bin/mysql_install_db --user=mysql
bin/mysqld_safe &
```

Si vous remarquez que configure va utiliser MIT-pthreads , il faut alors lire les notes MIT-pthreads . Remarques sur MIT−pthreads .

Si vous avez une erreur durant make install qui dit qu'il ne peut trouver /usr/include/pthreads , configure n'a pas détecté l'absence de MIT-pthreads . Pour corriger le problème, supprimez config.cache , puis relancez configure avec l'option --with-mit-threads .

Assurez−vous que votre configuration de la résolution des noms est bonne. Sinon, vous airez peut−être quelques problèmes lors de la connexion à mysqld . Assurez−vous que l'entrée localhost dans le fichier /etc/hosts est correcte (sinon, vous aurez des problèmes pour vous connecter à la base de données). Le fichier doit commencer par une ligne similaire à :

```
 127.0.0.1       localhost localhost.votre.domaine
```

FreeBSD est aussi connu pour avoir une petite limite de gestionnaires de fichiers par défaut. Fichier non trouvé . Décommentez la section ulimit -n dans safe_mysqld ou enlevez la limite pour l'utilisateur mysqld dans /etc/login.conf (et regénèrez le avec cap_mkdb). Assurez−vous aussi de définir la classe appropriée pour cet utilisateur dans le fichier des mots de passe si vous n'utilisez pas celui par défaut. (utilisez : chpass nom−utilisateur−mysqld). safe_mysqld , le script père de mysqld .Si vous avez beaucoup de mémoire, vous devriez penser à recompiler le noyau pour permettre à MySQL d'utiliser plus de 512 Mo de RAM. Regardez l' option MAXDSIZ dans le fichier de configuration de LINT pour plus d'informations.

Si vous avez des problèmes avec la date courante dans MySQL, configurer la variable d'environnement TZ aidera sûrement. Variables d'environnement MySQL .

Pour obtenir un système sécurisé et stable, vous ne devez utiliser que les noyaux FreeBSD marqués -RELEASE .

2.6.4.2 Notes concernant NetBSD

Pour compiler sur NetBSD vous aurez besoin de GNU make . Sinon, la compilation stoppera lorsque make essayera d'exécuter lint sur les fichiers C++.

2.6.4.3 Notes relatives à OpenBSD 2.5

Dans la version 2.5 de OpenBSD, vous pouvez compiler MySQL avec les threads natifs avec les options suivantes :

```
CFLAGS=-pthread CXXFLAGS=-pthread ./configure --with-mit-threads=no
```

2.6.4.4 Notes relatives à OpenBSD 2.8

Nos utilisateurs nous ont informé que OpenBSD 2.8 comporte un bogue des threads qui pose quelques problèmes avec MySQL. Les développeurs d'OpenBSD ont résolu ce problème, mais depuis le 25 janvier 2001 ce n'est disponible que dans la branche ``−current''. Les symptômes de ce bogue sont : réponses lentes, beaucoup de charge, grande utilisation du CPU, et crashes.

Si vous obtenez une erreur comme `Error in accept:: Bad file descriptor` ou erreur 9 en essayant d'ouvrir les tables ou les dossiers, le problème est probablement que vous n'avez pas alloué assez de descripteurs de fichiers à MySQL.

Dans ce cas, essayez de démarrer `safe_mysqld` en tant que root avec les options suivantes :

```
shell> mysqld_safe --user=mysql --open-files-limit=2048 &
```

2.6.4.5 Notes relatives aux versions 2.x de BSD/OS

Si vous obtenez l'erreur suivante lors de la compilation de MySQL, votre valeur de `ulimit` pour la mémoire virtuelle est trop petite :

```
 item_func.h: In method `Item_func_ge::Item_func_ge(const Item_func_ge &)':
item_func.h:28: virtual memory exhausted
make[2]: *** [item_func.o] Error 1
```

Essayez d'utiliser `ulimit -v 80000` et exécutez `make` à nouveau. Si cela ne fonctionne pas et que vous utilisez `bash` , essayez de passer à `csh` ou `sh` ; quelques utilisateurs de BSDI ont reporté des problèmes avec `bash` et `ulimit` .

Si vous utilisez `gcc` , vous aurez peut−être aussi à utiliser l'option `--with-low-memory` de `configure` pour pouvoir compiler `sql_yacc.cc` .

Si vous avez des problèmes avec la date courante dans MySQL, configurer la variable `TZ` vous aidera probablement. Variables d'environnement MySQL .

2.6.4.6 Notes relatives aux versions 3.x de BSD/OS

Mettez à jour à la version 3.1 de BSD/OS. Si cela n'est pas possible, installez le patch BSDIpatch M300−038.

Utilisez la commande suivante lors de la configuration de MySQL :

```
shell> env CXX=shlicc++ CC=shlicc2 \
      ./configure \
          --prefix=/usr/local/mysql \
          --localstatedir=/var/mysql \
          --without-perl \
          --with-unix-socket-path=/var/mysql/mysql.sock
```

Ce qui suit fonctionne aussi :

```
shell> env CC=gcc CXX=gcc CXXFLAGS=-O3 \
      ./configure \
```

```
        --prefix=/usr/local/mysql \
        --with-unix-socket-path=/var/mysql/mysql.sock
```

Vous pouvez changer les répertoires si vous voulez, ou utiliser les valeurs par défaut en ne spécifiant pas de chemins.

Si vous avez des problèmes de performances alors que la charge est petite, essayez d'utiliser l'option `--skip-thread-priority` de `mysqld` ! Cela exécutera tous les threads avec la même priorité; Sur la version 3.1 de BSDI, cela donne de meilleures performances (en attendant que BSDI corrige sont gestionnaire de threads).

Si vous obtenez l'erreur `virtual memory exhausted` durant la compilation, vous devez essayer en utilisant `ulimit -v 80000` et exécutant `make` à nouveau. Si cela ne fonctionne pas et que vous utilisez `bash` , essayez de passer à `csh` ou `sh` ; quelques utilisateurs de BSDI ont reporté des problèmes avec `bash` et `ulimit` .

2.6.4.7 Notes relatives aux versions 4.x de BSD/OS

Les versions 4.x de `BSDI` ont quelques bogues relatifs aux threads. Si vous voulez utiliser MySQL sur ce système, vous devez installer tous les patches liés aux threads. vous devez au moins installer `M400-023` .

Sur quelques systèmes avec une version 4.x de `BSDI` , vous pouvez rencontrer des problèmes avec les bibliothèques partagées. Le symptôme est que vous ne pouvez utiliser aucun programme client, comme par exemple, `mysqladmin` . Dans ce cas, vous devez le reconfigurer pour qu'il n'utilise pas les bibliothèques partagées avec l'option `--disable-shared` de configure.

Quelques utilisateurs ont eu avec `BSDI` 4.0.1 un problème faisant qu'après un bout de temps, le binaire `mysqld` ne peut plus ouvrir de tables. Cela est du au fait qu'un bogue relatif au système ou à la bibliothèque fait changer de répertoire à `mysqld` sans qu'on ne l'ait demandé !

La solution est soit de mettre à jour vers la version 3.23.34 ou de supprimer la ligne `#define HAVE_REALPATH` de `config.h` après avoir exécuté `configure` et avant d'exécuter `make` .

Notez que ce qui précède signifie que vous ne pouvez pas créer de lien symbolique sur un dossier de bases de données vers un autre dossier de bases de données ou lier une table symboliquement vers une autre base de données sur `BSDI` ! (Créer un lien symbolique vers un autre disque fonctionne).

2.6.5 Notes sur les autres Unix

2.6.5.1 Notes relatives à la version 10.20 de HP−UX

Il y a quelques petits problèmes que vous pourrez rencontrer lors de la compilation de MySQL sur HP−UX. Nous recommandons l'utilisation de `gcc` au lieu du compilateur natif de HP−UX, car `gcc` produit un meilleur code !

Nous recommandons l'utilisation de gcc 2.95 sur HP−UX. N'utilisez pas les options de haute optimisation (comme −O6) car cela pourrait ne pas être sûr sur HP−UX.

La ligne de configuration suivante devrait fonctionner avec gcc 2.95 :

```
 CFLAGS="-I/opt/dce/include -fpic" \
CXXFLAGS="-I/opt/dce/include -felide-constructors -fno-exceptions \
-fno-rtti" CXX=gcc ./configure --with-pthread \
```

```
--with-named-thread-libs='-ldce' --prefix=/usr/local/mysql --disable-shared
```

La ligne de configuration suivante devrait fonctionner avec gcc 3.1 :

```
 CFLAGS="-DHPUX -I/opt/dce/include -O3 -fPIC" CXX=gcc \
CXXFLAGS="-DHPUX -I/opt/dce/include -felide-constructors -fno-exceptions \
-fno-rtti -O3 -fPIC" ./configure --prefix=/usr/local/mysql \
--with-extra-charsets=complex --enable-thread-safe-client \
--enable-local-infile  --with-pthread \
--with-named-thread-libs=-ldce --with-lib-ccflags=-fPIC
--disable-shared
```

2.6.5.2 HP−UX Version 11.x Notes

Pour les version 11.x de HP−UX nous recommandons MySQL 3.23.15 ou plus récent.

A cause de quelques bogues critiques dans les bibliothèques standard de HP−UX, vous devez installer les patches suivants avant d'essayer de faire fonctionner MySQL sous HP−UX 11.0 :

```
 PHKL_22840 Streams cumulative
PHNE_22397 ARPA cumulative
```

Cela résoudra le problème de l'obtention de `EWOULDBLOCK` à partir de `recv()` et `EBADF` à partir de `accept()` dans les applications threadées.

Si vous utilisez `gcc` 2.95.1 sur un système HP−UX 11.x non−patché, vous obtiendrez l'erreur :

```
 In file included from /usr/include/unistd.h:11,
                 from ../include/global.h:125,
                 from mysql_priv.h:15,
                 from item.cc:19:
/usr/include/sys/unistd.h:184: declaration of C function ...
/usr/include/sys/pthread.h:440: previous declaration ...
In file included from item.h:306,
                 from mysql_priv.h:158,
                 from item.cc:19:
```

Le problème est que HP−UX ne définit pas `pthreads_atfork()` avec consistence. Il possède des prototypes en conflit dans `/usr/include/sys/unistd.h` :184 et `/usr/include/sys/pthread.h` :440 (détails ci−dessous).

Une solution est de copier `/usr/include/sys/unistd.h` dans `mysql/include` et éditer `unistd.h` en le changeant pour qu'il corresponde à la définition dans `pthread.h` . Voici les modifications :

```
 extern int pthread_atfork(void (*prepare)(), void (*parent)(),
                                          void (*child)());
```

est remplacée par

```
 extern int pthread_atfork(void (*prepare)(void), void (*parent)(void),
                                          void (*child)(void));
```

Après cela, la ligne de configuration suivante devrait fonctionner :

```
 CFLAGS="-fomit-frame-pointer -O3 -fpic" CXX=gcc \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti -O3" \
./configure --prefix=/usr/local/mysql --disable-shared
```

Si vous utilisez MySQL 4.0.5 avec le compilateur HP−UX, vous pouvez utiliser : (testé avec cc B.11.11.04):

```
 CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure --with-extra-character-set=complex
```

Vous pouvez ignorer toutes les erreurs de ce type :

```
 aCC: warning 901: unknown option: `-3': use +help for online documentation
```

Si vous obtenez l'erreur suivante de `configure` :

```
 checking for cc option to accept ANSI C... no
configure: error: MySQL requires a ANSI C compiler (and a C++ compiler).
Try gcc. See the Installation chapter in the Reference Manual.
```

Vérifiez que le chemin vers le compilateur KRne précède pas le chemin vers le compilateur C et C++ HP−UX.Une autre raison qui pourrait vous empêcher de compiler, et le fait de n'avoir pas définit l'option `+DD64` ci−dessus.

2.6.5.3 Notes relatives à IBM−AIX

La détection automatique de `xlC` est absente de Autoconf, ce qui fait qu'un commande `configure` comme celle qui suit est requise lors de la compilation de MySQL (Cet exemple utilise le compilateur IBM) :

```
 export CC="xlc_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192 "
export CXX="xlC_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192"
export CFLAGS="-I /usr/local/include"
export LDFLAGS="-L /usr/local/lib"
export CPPFLAGS=$CFLAGS
export CXXFLAGS=$CFLAGS

 ./configure --prefix=/usr/local \
        --localstatedir=/var/mysql \
        --sysconfdir=/etc/mysql \
        --sbindir='/usr/local/bin' \
        --libexecdir='/usr/local/bin' \
        --enable-thread-safe-client \
        --enable-large-files
```

Ce sont les options utilisées pour compiler la distribution de MySQL qui peut être trouvée sur http://www−frec.bull.com/ .

Si vous changez le `-O3` en `-O2` dans la ligne précédente, vous devez aussi enlever l'option `-qstrict` (c'est une limitation du compilateur IBM C).

Si vous utilisez `gcc` ou `egcs` pour compiler MySQL, vous **devez** utiliser l'option `-fno-exceptions` , vu que la gestion des exceptions de `gcc` / `egcs` n'est pas sûre pour les threads ! (Cela est testé avec `egcs` 1.1.) Il y a aussi quelques problèmes connus avec l'assembleur d'IBM, qui peuvent lui faire générer du mauvais code lors de son utilisation avec gcc.

Nous recommandons la ligne de `configure` suivante avec `egcs` et `gcc 2.95` sur AIX :

```
 CC="gcc -pipe -mcpu=power -Wa,-many" \
CXX="gcc -pipe -mcpu=power -Wa,-many" \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory
```

Le `-Wa,-many` est nécessaire pour que la compilation se passe sans problèmes. IBM est au courant de ce problème mais n'est pas pressé de le corriger à cause de l'existence du palliatif. Nous ne savons pas si `-fno-exceptions` est requise avec `gcc 2.95` , mais comme MySQL n'utilise pas les exeptions et que l'option en question génère un code plus rapide, nous vous recommandons de toujours utiliser cette option avec `egcs / gcc` .

Si vous obtenez un problème avec le code de l'assembleur essayez en changeant l'option −mcpu=xxx pour l'adapter à votre processeur. Le plus souvent, on a besoin de power2, power, ou powerpc, et sinon 604 ou 604e. Je ne suis pas positif mais je pense que l'utilisation de "power" sera sûre la plupart du temps, même sur une machine power2.

Si vous ne savez pas quel est votre processeur, exécutez "uname −m", cela vous renverra une chaîne comme "000514676700", avec un format xxyyyyyymmss où xx et ss sont toujours des zéros, yyyyyy est un identifiant unique du système et mm est l'identifiant du CPU Planar. Une liste de ces valeurs peut être trouvée sur http://publib.boulder.ibm.com/doc_link/en_US/a_doc_lib/cmds/aixcmds5/uname.htm . Cela vous donnera un type et un modèle de machine que vous pouvez utiliser pour déterminer quel type de processeur vous avez.

Si vous avez des problèmes avec les signaux (MySQL se termine de manière imprévue lors des montées en charge) vous avez peut−être trouvé un bogue du système avec les threads et les signaux. Dans ce cas, vous pouvez demander à MySQL de ne pas utiliser les signaux en configuration avec :

```
shell> CFLAGS=-DDONT_USE_THR_ALARM CXX=gcc \
      CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti \
      -DDONT_USE_THR_ALARM" \
      ./configure --prefix=/usr/local/mysql --with-debug --with-low-memory
```

Cela n'affecte pas les performances de MySQL, mais comporte un effet secondaire faisant en sorte que vous ne pourrez tuer les clients en état ``sleeping'' sur une connexion avec `mysqladmin kill` ou `mysqladmin shutdown` . A la place, le client se terminera lorsqu'il émmetra sa prochaine commande.

Sur quelques versions de AIX, lier avec `libbind.a` fait vider son noyau à `getservbyname` (core dump). Il s'agit d'un bogue AIX et doit être remonté à IBM.

Pour AIX 4.2.1 et gcc vous devez apporter les modifications suivantes :

Après la configuration, éditez `config.h` et `include/my_config.h` et changez la ligne qui comporte

```
#define HAVE_SNPRINTF 1
```

en

```
#undef HAVE_SNPRINTF
```

Et finalement, dans `mysqld.cc` vous devez ajouter un prototype pour initgoups.

```
 #ifdef _AIX41
extern "C" int initgroups(const char *,int);
#endif
```

Si vous avez besoin d'allouer beaucoup de mémoire au processus mysqld, il ne suffit pas de configurer 'ulimit −d unlimited'. Vous aurez aussi à configurer dans `mysqld_safe` quelque chose comme :

```
export LDR_CNTRL='MAXDATA=0x80000000'
```

Vous trouverez plus d'informations sur l'utilisation d'une grande quantité de mémoire sur : http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/genprogc/lrg_prg_support.htm .

2.6.5.4 Notes relatives à SunOS 4

Avec SunOS 4, les `MIT-pthreads` sont requis pour compiler MySQL, ce qui signifie que vous aurez besoin de GNU `make` .

Quelques systèmes SunOS 4 ont des problèmes avec les bibliothèques dynamiques et `libtool` . Vous pouvez utiliser la ligne suivante de `configure` pour éviter ce problème :

```
shell> ./configure --disable-shared --with-mysqld-ldflags=-all-static
```

Lors de la compilation de `readline` , vous pouvez obtenir des avertissements à propos de définitions dupliquées. Vous pouvez les ignorer.Lors de la compilation de `mysqld` , il y aura quelques avertissements `implicit declaration of function` . Vous pouvez les ignorer.

2.6.5.5 Notes pour Alpha−DEC−UNIX (Tru64)

Si vous utilisez `egcs` 1.1.2 sur Digital Unix, vous devez passer à `gcc` 2.95.2, car `egcs` connaît de sérieux bugs sur DEC!

Lorsque vous compilez des programmes threadés sous Digital Unix, la documentation recommande l'utilisation de l'option `-pthread` avec `cc` et `cxx` et les bibliothèques `-lmach -lexc` (en plus de `-lpthread`). Vous devriez exécuter le script `configure` comme ceci :

```
CC="cc -pthread" CXX="cxx -pthread -O" \
./configure --with-named-thread-libs="-lpthread -lmach -lexc -lc"
```

Lorsque vous compilez `mysqld` , vous pouvez voir apparaître des alertes comme celles−ci :

```
mysqld.cc: In function void handle_connections()':
mysqld.cc:626: passing long unsigned int *' as argument 3 of
accept(int,sockadddr *, int *)'
```

Vous pouvez les ignorer tranquillement. Elles apparaissent car `configure` ne peut détecter que des erreurs, et pas des alertes.

Si vous démarrez le serveur directement en ligne de commande, vous pouvez rencontrer des problèmes d'interruption si vous vous déconnectez. Lorsque vous vous déconnectez, les processus en cours reçoivent le signal `SIGHUP` . Si c'est le cas, essayez de démarrer le serveur comme ceci :

```
shell> nohup mysqld [options] &
```

`nohup` fait que la commande suivante va ignorer les signaux `SIGHUP` envoyés par le terminal. Alternativement, vous pouvez démarrer le serveur avec le script `safe_mysqld` , qui appelle le démon `mysqld` avec l'option `nohup` pour vous. `safe_mysqld` , le script père de `mysqld` .

Si vous avez des problèmes pour compiler `mysys/get_opt.c` , vous pouvez simplement supprimer la ligne `#define _NO_PROTO` au début du fichier!

Si vous utilisez le compilateur `cc` de `Compaq` , la ligne de configuration suivante devrait fonctionner :

```
CC="cc -pthread"
CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host"
CXX="cxx -pthread"
CXXFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host \
-noexceptions -nortti"
export CC CFLAGS CXX CXXFLAGS
./configure \
--prefix=/usr/local/mysql \
--with-low-memory \
--enable-large-files \
--enable-shared=yes \
--with-named-thread-libs="-lpthread -lmach -lexc -lc"
gnumake
```

Si vous avez un problème avec `libtool` , lorsque vous compilez les bibliothèques partagées, ou lorsque vous compilez `mysql` , vous devriez pouvoir résoudre ce problème avec :

```
cd mysql
/bin/sh ../libtool --mode=link cxx -pthread  -O3 -DDBUG_OFF \
-O4 -ansi_alias -ansi_args -fast -inline speed \
-speculate all \ -arch host  -DUNDEF_HAVE_GETHOSTBYNAME_R \
-o mysql  mysql.o readline.o sql_string.o completion_hash.o \
../readline/libreadline.a -lcurses \
../libmysql/.libs/libmysqlclient.so  -lm
cd ..
gnumake
gnumake install
scripts/mysql_install_db
```

2.6.5.6 Notes pour Alpha−DEC−OSF/1

Si vous avez des problèmes de compilation et que le `CC` de DEC et `gcc` sont installés, essayez d'utiliser le script `configure` comme ceci :

```
CC=cc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Si vous avez des problèmes avec le fichier `c_asm.h` , vous pouvez créer un fichier inerte `c_asm.h` avec :

```
touch include/c_asm.h
CC=gcc CFLAGS=-I./include \
CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Notez que les problèmes suivants avec le programme `ld` peuvent être corrigés en téléchargeant le dernier kit de patch de DEC (Compaq) à : http://ftp.support.compaq.com/public/unix/ .

Su OSF/1 V4.0D et avec le compilateur "DEC C V5.6−071 on Digital Unix V4.0 (Rev. 878)" le compilateur présente un comportement étrange (`undefined asm symbols`). `/bin/ld` apparaît aussi comme incorrect (problèmes avec des erreurs `_exit undefined` survenant lors du link de `mysqld`). Sur ce système, nous avons réussi à compiler MySQL avec le script `configure` suivant, après avoir remplacé `/bin/ld` par la version de OSF 4.0C:

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
```

Avec le compilateur Digital "C++ V6.1−029", la ligne suivante doit fonctionner :

```
 CC=cc -pthread
CFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
       -arch host
CXX=cxx -pthread
CXXFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
         -arch host -noexceptions -nortti
export CC CFLAGS CXX CXXFLAGS
./configure --prefix=/usr/mysql/mysql --with-mysqld-ldflags=-all-static \
           --disable-shared --with-named-thread-libs="-lmach -lexc -lc"
```

Avec certaines versions de OSF/1, la fonction `alloca()` est buguée. Corrigez cela en supprimant la ligne du fichier `config.h` qui définit `'HAVE_ALLOCA'` .La fonction `alloca()` a aussi un prototype incorrect dans `/usr/include/alloca.h` . L'alerte en résultant peut être ignorée.

Le script `configure` va utiliser automatiquement les bibliothèques de threads suivantes : `--with-named-thread-libs="-lpthread -lmach -lexc -lc"` .

En utilisant `gcc` , vous pouvez aussi essayer le script `configure` avec ceci :

```
shell> CFLAGS=-D_PTHREAD_USE_D4 CXX=gcc CXXFLAGS=-O3 ./configure ...
```

Si vous avez des problèmes avec les signaux (MySQL s'arrête inopinément sous forte charge), vous pouvez avoir rencontré un bogue de l'OS avec les threads, et les signaux. Dans ce cas, vous pouvez indiquer à MySQL de ne pas utiliser les signaux avec la configuration suivante :

```
shell> CFLAGS=-DDONT_USE_THR_ALARM \
      CXXFLAGS=-DDONT_USE_THR_ALARM \
      ./configure ...
```

Cela ne modifie pas les performances de MySQL, mais vous ne pourrez plus terminer les clients qui sont en mode ``sleeping'' sur une connexion avec la commande `mysqladmin kill` ou `mysqladmin shutdown` . Au lieu de cela, le client sera interrompu lorsqu'il émettra la prochaine commande.

Avec `gcc` 2.95.2, vous aurez probablement les problèmes de compilation suivants :

```
 sql_acl.cc:1456: Internal compiler error in `scan_region', at except.c:2566
Please submit a full bug report.
```

Pour corriger cela, vous devez aller dans le dossier `sql` et faire un ``copier coller'' de la dernière ligne `gcc` , tout en remplaçant le code `-O3` par le code `-O0` ou ajouter le code `-O0` immédiatement après `gcc` si vous n'avez aucune option `-O` sur votre ligne de compilation). Après cela, vous pouvez retourner au niveau de la racine de MySQL, et tenter à nouveau un `make` .

2.6.5.7 Notes relatives à SGI Irix

Si vous utilisez la version 6.5.3 d'Irix ou plus récente, `mysqld` ne pourra créer de threads que si vous l'exécutez en tant qu'utilisateur possédant le privilège `CAP_SCHED_MGT` (comme `root`) ou que vous donnez au serveur `mysqld` ce privilège avec la commande suivante :

```
shell> chcap "CAP_SCHED_MGT+epi" /opt/mysql/libexec/mysqld
```

Vous devrez peut−être supprimer quelques définitions dans `config.h` après avoir exécuté `configure` et avant de compiler.

Sur quelques implémentations d'Irix, la fonction `alloca()` ne marche pas. Si le serveur `mysqld` se stoppe sur quelques requêtes `SELECT` , supprimez les lignes de `config.h` qui définissent `HAVE_ALLOC` et `HAVE_ALLOCA_H` . Si `mysqladmin create` ne fonctionne pas, supprimez la ligne qui définit `HAVE_READDIR_R` dans `config.h` . Vous devrez peut−être supprimer la ligne de `HAVE_TERM_H` aussi.

SGI recommande que vous installiez tous les patches de cette page :
http://support.sgi.com/surfzone/patches/patchset/6.2_indigo.rps.html

Vous devrez, au moins, installer la dernière version du noyau, de `rld` et de `libc` .

Vous avez besoin de tous les patches POSIX sur cette page, pour le support des pthreads :

http://support.sgi.com/surfzone/patches/patchset/6.2_posix.rps.html

Si vous obtenez une erreur se rapprochant de la suivante lors de la compilation de `mysql.cc` :

```
"/usr/include/curses.h", line 82: error(1084): invalid combination of type
```

Tapez ce qui suit dans le répertoire racine de votre source MySQL :

```
shell> extra/replace bool curses_bool < /usr/include/curses.h \
> include/curses.h
shell> make
```

Un problème de planification a aussi été signalé. Si seul un thread est en cours, les choses ralentissent. Evitez cela en démarrant un autre client. Cela pourra accélérer l'exécution de l'autre thread de 2 à 10 fois. Ceci est un problème pas encore très clair avec les threads Irix; vous devrez improviser pour trouver des solutions en attendant que cela soit corrigé.

Si vous compilez avec `gcc` , vous pouvez utiliser la commande `configure` suivante :

```
CC=gcc CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql --enable-thread-safe-client \
--with-named-thread-libs=-lpthread
```

Sous Irix 6.5.11 avec les compilateurs natifs Irix C et C++ versions 7.3.1.2, ce qui suit est connu pour fonctionner :

```
CC=cc CXX=CC CFLAGS='-O3 -n32 -TARG:platform=IP22 -I/usr/local/include \
-L/usr/local/lib' CXXFLAGS='-O3 -n32 -TARG:platform=IP22 \
-I/usr/local/include -L/usr/local/lib' ./configure \
--prefix=/usr/local/mysql --with-innodb --with-berkeley-db \
--with-libwrap=/usr/local \
--with-named-curses-libs=/usr/local/lib/libncurses.a
```

2.6.5.8 Notes sur SCO

Le port actuel est testé uniquement sur les systèmes ``sco3.2v5.0.5'', ``sco3.2v5.0.6'' et ``sco3.2v5.0.7''. Il y a aussi eu des progrès sur le port vers ``sco 3.2v4.2''.

Pour le moment, le compilateur recommandé sur OpenServer est `gcc` 2.95.2. Avec lui, vous devriez être capable de compiler MySQL simplement avec :

```
CC=gcc CXX=gcc ./configure ... (options)
```

- Pour OpenServer 5.0.x, vous avez besoin de `gcc-2.95.2p1` ou plus récent, de Skunkware. http://www.sco.com/skunkware/ puis recherchez dans les paquets OpenServer ou par FTP sur ftp2.caldera.com, dans le dossier `pub/skunkware/osr5/devtools/gcc` .
- Vous avez besoin de `GCC 2.5.x` pour ce produit, et du système de développement. Ils sont nécessaires sur cette version de Unix SCO. Vous ne pouvez pas simplement utiliser GCC Dev.

* Vous devriez installer le paquet `FSU Pthreads` et l'installer. Il peut être trouvé à l'adresse : http://moss.csc.ncsu.edu/~mueller/ftp/pub/PART/pthreads.tar.gz . Vous pouvez aussi obtenir un paquet precompilé sur http://www.mysql.com/Downloads/SCO/FSU−threads−3.5c.tar.gz .
* Les `FSU Pthreads` peuvent être compilé sur SCO Unix 4.2 avec TCP/IP. Ou OpenServer 3.0 ou Open Desktop 3.0 (OS 3.0 ODT 3.0), avec le `SCO Development System` , installé avec le bon port de GCC 2.5.x ODT ou OS 3.0 avec le bon port de GCC 2.5.x Il y a beaucoup de problèmes si vous n'utilisez pas le bon port. Le port de ce produit requiert le `SCO Unix Development` . Sans cela, il vous manque des bibliothèques et le linker nécessaire.
* Pour compiler `FSU Pthreads` sur votre système, faîtes ceci :
    * Exécutez `./configure` dans le dossier `threads/src` et sélectionnez l'option SCO OpenServer. Cette commande copie `Makefile.SCO5` dans le fichier `Makefile` .
    * Exécutez `make` .
    * Pour installer le paquet dans le dossier par défaut `/usr/include` , identifiez vous comme root, puis utilisez `cd` pour vous placer dans le dossier `thread/src` , et faites `make install` .
* N'oubliez pas d'utiliser GNU `make` lors de la compilation de MySQL.
* Si vous ne lancez pas `mysqld_safe` en tant que `root` , vous obtiendrez probablement un maximum de 110 fichiers ouverts par processus. `mysqld` vous le dira dans les logs.
* Avec SCO 3.2V5.0.5, il est recommandé d'utiliser `FSU Pthreads` version 3.5c ou plus récent. Il est recommandé d'utiliser `gcc` 2.95.2 ou plus récent!

  La commande `configure` devrait fonctionner :

```
shell> ./configure --prefix=/usr/local/mysql --disable-shared
```

* Avec SCO 3.2V4.2, il est recommandé d'utiliser `FSU Pthreads` version 3.5c ou plus récent. La commande `configure` devrait fonctionner :

```
shell> CFLAGS="-D_XOPEN_XPG4" CXX=gcc CXXFLAGS="-D_XOPEN_XPG4" \
       ./configure \
           --prefix=/usr/local/mysql \
           --with-named-thread-libs="-lgthreads -lsocket -lgen -lgthreads" \
           --with-named-curses-libs="-lcurses"
```

  Vous pourriez rencontrer des problèmes avec certains fichiers à inclure. Dans ce cas, vous pouvez trouver des fichiers spécifiques pour SCO à l'adresse http://www.mysql.com/Downloads/SCO/SCO−3.2v4.2−includes.tar.gz . Il suffit de décompresser le fichier dans le dossier `include` de votre dossier source MySQL.

Notes de développement SCO :

* MySQL doit automatiquement détecter le paquet `FSU Pthreads` et l'utiliser pour compiler `mysqld` avec `-lgthreads -lsocket -lgthreads` .
* Les bibliothèques de développement SCO sont re−entrantes avec les `FSU Pthreads` . SCO affirme que ses bibliothèques sont ré−entrantes, donc elles sont aussi ré−entrantes avec les `FSU Pthreads` . `FSU Pthreads` sur OpenServer essaie d'utiliser les concepts SCO pour rendre ses bibliothèques ré−entrantes.
* `FSU Pthreads` (tout au moins, la version de http://www.mysql.com/) est livré avec GNU `malloc` . Si vous rencontrez des problèmes avec l'utilisation de la mémoire, assurez−vous que le fichier `gmalloc.o` est inclus dans `libgthreads.a` et `libgthreads.so` .
* Avec les `FSU Pthreads` , les appels système suivants sont compatibles avec les pthreads : `read()` , `write()` , `getmsg()` , `connect()` , `accept()` , `select()` et `wait()` .
* Le patch CSSA−2001−SCO.35.2 (le patch est nommé `erg711905-dscr_remap security patch` (version 2.0.0)) bloque les `FSU Pthreads` et rend `mysqld` instable. Vous devez le supprimer si vous voulez faire fonctionner `mysqld` sur une machine OpenServer 5.0.6.
* SCO fournit des patches pour son système d'exploitation à l'adresse ftp://ftp.sco.com/pub/openserver5 pour OpenServer 5.0.x
* SCO fournit des patches de sécurités et la bibliothèque `libsocket.so.2` à l'adresse ftp://ftp.sco.com/pub/security/OpenServer et ftp://ftp.sco.com/pub/security/sse pour OpenServer 5.0.x

- Patch de sécurité avant OSR506. De même, le patch telnetd de ftp://stage.caldera.com/pub/security/openserver/ ou ftp://stage.caldera.com/pub/security/openserver/CSSA−2001−SCO.10/ ainsi que libsocket.so.2 et libresolv.so.1 ont des instructions concernant leur installation sur un système pre−OSR506.C'est probablement une bonne idée que d'installer les patches de sécurité ci−dessus avant de compiler et d'utiliser MySQL.

2.6.5.9 Notes sur SCO UnixWare Version 7.1.x

Sur UnixWare 7.1.0, vous devez utiliser une version de MySQL au moins aussi récente que la 3.22.13 pour avoir les correctifs sur cet OS.

Nous avons réussi à compiler MySQL avec la commande de configuration suivante configure sur UnixWare Version 7.1.x:

```
CC=cc CXX=CC ./configure --prefix=/usr/local/mysql
```

Si vous voulez utiliser gcc , vous devez utiliser gcc 2.95.2 ou plus récent.

```
CC=gcc CXX=g++ ./configure --prefix=/usr/local/mysql
```

SCO fournit des patchs pour son OS à ftp://ftp.sco.com/pub/unixware7 pour UnixWare 7.1.1 et 7.1.3 et à ftp://ftp.sco.com/pub/openunix8 pour OpenUNIX 8.0.0. SCO fournit des informations sur les correctifs de sécurité à ftp://ftp.sco.com/pub/security/OpenUNIX pour OpenUNIX et à ftp://ftp.sco.com/pub/security/UnixWare pour UnixWare.

2.6.6 Notes relatives à OS/2

MySQL utilise un certain nombre de fichiers ouverts. A cause de cela, vous devez ajouter un ligne se rapprochant de la suivante dans votre fichier CONFIG.SYS :

```
SET EMXOPT=-c -n -h1024
```

Si vous ne le faites pas, vous obtiendrez probablement l'erreur :

```
File 'xxxx' not found (Errcode: 24)
```

Lors de l'utilisation de MySQL avec OS/2 Warp 3, FixPack 29 ou plus est requis. Avec OS/2 Warp 4, FixPack 4 ou plus est requis. C'est un besoin de la bibliothèque des Pthreads. MySQL doit être installé sur une partition qui supporte les noms de fichiers longs, tel que HPFS, FAT32, etc.Le script INSTALL.CMD doit être exécuté à partir du CMD.EXE d'OS/2 et ne fonctionnera probablement pas avec des substituts tels que 4OS2.EXE .

Le script scripts/mysql-install-db a été renommé. Il est maintenant nommé install.cmd et est un script REXX, qui mettra en place les configurations de sécurité par défaut de MySQL et créera les icônes WorkPlace Shell pour MySQL.

Le support des module dynamiques est compilé, mais n'est pas assez testé. Les modules dynamiques doivent être compilés en utilisant la bibliothèque pthreads.

```
 gcc -Zdll -Zmt -Zcrtdll=pthrdrtl -I../include -I../regex -I.. \
    -o exemple udf_exemple.cc -L../lib -lmysqlclient udf_exemple.def
mv exemple.dll exemple.udf
```

Note : A cause des limitations de OS/2, les noms des modules UDF ne doivent pas dépasser 8 caractères. Les modules sont stockés dans le répertoire /mysql2/udf ; le script safe-mysqld.cmd placera ce répertoire dans la variable d'environnement BEGINLIBPATH . Lors de l'utilisation des modules UDF , les extensions spécifiées sont ignorées. Elle est supposée être `.udf` . Par exemple, sous Unix, le module partagé peut se nommer `exemple.so` et vous chargeriez une de ses fonctions de la façon suivante :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "exemple.so";
```

Sous OS/2, me module s'appellera `exemple.udf` , mais vous n'aurez pas à spécifier son extension :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "exemple";
```

2.6.7 Notes relatives à BeOS

Nous sommes vraiment intéressés par le port de MySQL sur BeOS, mais malheureusement, nous n'avons personne qui s'y connaisse en BeOS ou qui ait le temps de s'en occuper.

Nous sommes intéressés par quelqu'un qui serait prêt à faire le port, et nous l'aiderions pour toutes les questions techniques qu'il pourrait se poser durant le processus.

Nous avons déjà eu des contacts avec des développeurs BeOS qui ont dit que MySQL était porté à 80% sur BeOS, mais nous n'avons plus entendu parler d'eux depuis.

2.7 Commentaires sur l'installation de Perl

Le support MySQL par Perl est fournis grâce à l'interface `DBI` / `DBD` . Cette interface requiert Perl Version 5.6.0 ou plus récent. Elle **fonctionnera pas** si vous avez une autre version plus ancienne de Perl.

Si vous voulez utiliser les transactionsa vec Perl `DBI` , vous devez installer `DBD::mysql` version 1.2216 ou plus récent. La version 2.9003 ou plus récent est recommandée.

Notez que si vous utilisez la bibliothèque client MySQL 4.1, vous devrez utiliser `DBD::mysql` 2.9003 ou plus récent.

Depuis la version 3.22.8, le support de Perl n'est plus inclut dans les distribution de MySQL. Vous pouvez obtenir les modules requis sur le site http://search.cpan.org pour Unix, ou sur le site de ActiveState (fichiers `ppm`) pour Windows. La section suivante décrit comment faire.

Le support de MySQL par Perl doit être installé si vous voulez exécuter les scripts de tests de performances. La suite de tests comparatifs de MySQL .

2.7.1 Installer Perl sur Unix

Le support Perl de MySQL requièrt que vous ayez installé le support de programmation de clients pour MySQL. Si vous avez installé MySQL à partir de fichiers `RPM` , les programmes clients sont dans le `RPM` client, mais le support de la programmation de clients est dans le `RPM` des développeurs. Assurez−vous d'avoir installé le dernier `RPM` .

Si vous voulez installer le support Perl, les fichiers dont vous avez besoin sont disponible sur la bibliothèque CPAN (Comprehensive Perl Archive Network) à http://search.cpan.org .

Le moyen le plus facile pour installer le module Unix de Perl et d'utilise le module `CPAN` . Par exemple :

```
 shell> perl −MCPAN −e shell
cpan> install DBI
cpan> install DBD::mysql
```

L'installation de `DBD::mysql` effectue de nombreux tests. Ces tests requièrent une connexion local au serveur MySQL, en tant qu'utilisateur anonyme, sans mot de passe. Si vous avez supprimé l'accès anonyme, ou assigné des mots de passe, les tests échoueront. Vous pouvez utiliser `force install DBD::mysql` pour ignorer ces tests. `DBI` requiert le module `Data::Dumper` . Il peut être déjà installé. Si non, vous devez l'installer avant d'installer `DBI` .

Il est aussi possible de télécharger la distribution module sous la forme d'une archive `tar` compressée, et de compiler manuellement les modules. Par exemple, pour décompresser et construire la distribution `DBI` , utilisez cette procédure :

- Décompressez la distribution dans le dossier courant :
  ```
  shell> gunzip < Data-Dumper-VERSION.tar.gz | tar xvf -
  ```
  Cette commande crée un dossier appelé `Data-Dumper-VERSION` .
- Mettez vous dans le répertoire racine de la distribution décompressée :
  ```
  shell> cd DBI-VERSION
  ```
- Construisez la distribution et compilez tout :
  ```
  shell> perl Makefile.PL
  shell> make
  shell> make test
  shell> make install
  ```

La commande `make test` est importante car elle vérifie que le module fonctionne. Notez que quand vous exécutez cette commande durant l'installation de `Msql-Mysql-modules` pour tester le code de l'interface, le serveur MySQL doit être en marche sinon le test échouera.

Il est bon de reconstruire et réinstaller la distribution `Msql-Mysql-modules` à chaque fois que vous réinstallez une nouvelle version de MySQL, particulièrement si vous avez des problèmes avec vos scripts `DBI` après avoir avoir mis à jour MySQL.

Si vous n'avez pas le droit d'installer des modules Perl dans le dossier système ou que vous voulez installer des modules locaux de Perl, la référence suivante pourra vous aider : http://www.iserver.com/support/contrib/perl5/modules.html

Regardez le paragraphe ``Installing New Modules that Require Locally Installed Modules.''

2.7.2 Installer ActiveState Perl sur Windows

Pour installer le module `DBD` MySQL avec ActiveState Perl sous Windows, vous devez faire ce qui suit :

- Obtenez ActiveState Perl à partir de http://www.activestate.com/Products/ActivePerl/ et installez le.
- Ouvrez un terminal DOS.
- Si requis, définissez la variable HTTP_proxy. Par exemple, vous pouvez faire :
  ```
  set HTTP_proxy=my.proxy.com:3128
  ```
- Démarrez le programme PPM :
  ```
  C:\> c:\perl\bin\ppm.pl
  ```
- Installez `DBI` , si ce n'est pas déjà fait :
  ```
  ppm> install DBI
  ```
- Si cela fonctionne, exécutez la commande suivante :
  ```
  install \
  ftp://ftp.de.uu.net/pub/CPAN/authors/id/JWIED/DBD-mysql-1.2212.x86.ppd
  ```

Ce qui suit devrait fonctionner avec la version 5.6 d'ActiveState Perl.

Si ce qui précède ne veut pas fonctionner, vous devez à la place installer le pilote `MyODBC` et vous connecter au serveur MySQL via ODBC :

```
 use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn","$utilisateur","$motdepasse") ||
  die "Obtenu l'erreur $DBI::errstr lors de la connexion à $dsn\n";
```

2.7.3 Problèmes lors de l'utilisation des interfaces Perl DBI et DBD

Si Perl vous informe qu'il ne peut trouver le module `../mysql/mysql.so` , il se trouve probablement que Perl n'arrive pas à trouver la bibliothèque partagée `libmysqlclient.so` .

Vous pouvez corriger cela en suivant l'une des méthodes suivantes :

- Compilez la distribution `Msql-Mysql-modules` avec `perl Makefile.PL -static -config` au lieu de `perl Makefile.PL` .
- Copiez `libmysqlclient.so` dans le dossier où se situent vos autres bibliothèques partagées (souvent `/usr/lib` ou `/lib`).
- Modifiez l'option `-L` utilisée pour compiler `DBD::mysql` pour refléter le chemin correct de `libmysqlclient.so` .
- Sous Linux vous pouvez ajouter le chemin vers le dossier dans lequel se trouve `libmysqlclient.so` au fichier `/etc/ld.so.conf` .
- Ajoutez le chemin complet vers le dossier où se situe `libmysqlclient.so` à la variable d'environnement `LD_RUN_PATH` .

Notez que vous aurez aussi besoin de modifier les options `-L` s'il y a d'autres bibliothèques que le linker ne peut trouver. Par exemple, si le linker ne peut trouver `libc` comme il est dans `/lib` et que la commande de link spécifie `-L/usr/lib` , modifiez l'option `-L` en `-L/lib` ou ajoutez l'option `-L/lib` à la commande de link existante.

Si vous obtenez l'erreur suivante de `DBD-mysql` , vous utilisez probablement `gcc` (ou un vieux binaire compilé avec `gcc`) :

```
 /usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Ajoutez `-L/usr/lib/gcc-lib/... -lgcc` à la commande de liaison lorsque la bibliothèque `mysql.so` est construite (vérifiez l'affichage de `make` concernant `mysql.so` quand vous compilez le client Perl). L'option `-L` doit spécifier le chemin vers le dossier où se situe `libgcc.a` sur votre système.

Une autre cause du problème peut être que Perl et MySQL ne sont pas tous deux compilés avec `gcc` . Dans ce cas là, vous devrez faire en sorte qu'ils le soient.

Si vous obtenez les erreurs suivantes de la part de `Msql-Mysql-modules` quand vous exécutez ces tests :

```
 t/00base............install_driver(mysql) failed:
Can't load '../blib/arch/auto/DBD/mysql/mysql.so' for module DBD::mysql:
../blib/arch/auto/DBD/mysql/mysql.so: undefined symbol:
uncompress at /usr/lib/perl5/5.00503/i586-linux/DynaLoader.pm line 169.
```

cela signifie que vous avez besoin d'inclure la bibliothèque dynamique, −lz, dans la ligne de liaison. Cela peut se faire en changeant ce qui suit dans `lib/DBD/mysql/Install.pm` :

```
 $sysliblist .= " -lm";
```

en

```
 $sysliblist .= " -lm -lz";
```

Après cela, vous **devez** exécuter 'make realclean' et reprendre l'installation dès le début.Si vous voulez installer `DBI` sur SCO, vous devez éditer le fichier `Makefile` de `DBI-xxx` et chaque sous−dossier.

Notez que `gcc` doit être en version 2.95.2 ou plus récente :

```
 ANCIEN:                               NOUVEAU:
CC = cc                               CC = gcc
CCCDLFLAGS = -KPIC -W1,-Bexport       CCCDLFLAGS = -fpic
CCDLFLAGS = -wl,-Bexport              CCDLFLAGS =

 LD = ld                               LD = gcc -G -fpic
LDDLFLAGS = -G -L/usr/local/lib       LDDLFLAGS = -L/usr/local/lib
LDFLAGS = -belf -L/usr/local/lib      LDFLAGS = -L/usr/local/lib

 LD = ld                               LD = gcc -G -fpic
OPTIMISE = -Od                        OPTIMISE = -O1

 OLD:
CCCFLAGS = -belf -dy -w0 -U M_XENIX -DPERL_SCO5 -I/usr/local/include

 NEW:
CCFLAGS = -U M_XENIX -DPERL_SCO5 -I/usr/local/include
```

Ceci est dû au fait que le chargeur dynamique de Perl ne va pas charger les modules `DBI` , s'ils sont compilés avec `icc` ou `cc` .

Si vous voulez utiliser le module de Perl sur un système qui ne supporte pas les liaisons dynamiques (comme Caldera/SCO) vous pouvez générer une version statique de Perl incluant `DBI` et `DBD-mysql` . L'approche est de générer une version de Perl avec le code de `DBI` lié et de l'installer au dessus de votre Perl courant. Puis vous utilisez cette version pour en créer à nouveau une autre qui comporte le code de `DBD` lié et d'installer cette version ci.

Sur Caldera (SCO), vous devez définir les variables d'environnement suivantes :

```
 shell> LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/usr/progressive/lib
```

ou :

```
 shell> LD_LIBRARY_PATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> LIBPATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> MANPATH=scohelp:/usr/man:/usr/local1/man:/usr/local/man:\
/usr/skunk/man:
```

D'abord, créez un Perl incluant un `DBI` lié statiquement en exécutant des commandes dans le dossier où se situe votre distribution `DBI` :

```
 shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Ensuite, vous devez installer le nouveau Perl. Les affichages de `make perl` vous indiqueront les commandes `make` exactes que vous aurez besoin d'exécuter pour faire l'installation. Sur Caldera (SCO), il s'agit de `make -f Makefile.aperl inst_perl MAP_TARGET=perl` .Puis, utilisé le Perl qui vient d'être créé pour en créer un nouveau qui inclut un `DBD::mysql` lié statiquement en exécutant ces commandes dans le dossier où votre distribution de `Msql-Mysql-modules` se situe :

```
 shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Finalement, vous devez installer ce nouveau Perl. Une fois de plus, l'affichage de `make perl` vous indiquera la commande à utiliser.

3 Tutoriels d'introduction

Ce chapitre fournit un tutoriel d'introduction à MySQL en montrant comment utiliser le client `mysql` pour créer et utiliser une simple base de données. `mysql` (quelques fois nommé ``moniteur terminal'' ou juste ``moniteur'') est un programme interactif qui vous permet de vous connecter à un serveur MySQL, exécuter des requêtes et voir les résultats. `mysql` peut aussi être utilisé en mode batch : vous placez vos requêtes dans un fichier, puis vous faites exécuter à `mysql` le contenu de ce fichier. Les deux manières d'utiliser `mysql` sont expliquées ici.

Pour voir une liste d'options fournies par `mysql` , invoquez−le avec l'option `--help` :

```
shell> mysql --help
```

Ce chapitre assume que `mysql` est installé sur votre machine et qu'un serveur MySQL est disponible pour que vous vous y connectiez. Si ce n'est pas le cas, contactez votre administrateur MySQL. (Si **vous** êtes l'administrateur, vous aurez besoin de consulter d'autres sections de ce manuel.)

Ce chapitre décrit le processus d'installation et d'utilisation d'une base de données en entier. Si vous n'êtes intéressés que par l'accès à une base de données existante, vous pouvez sauter les sections décrivant la création de la base et des tables.

Ce chapitre n'est qu'un tutoriel, beaucoup de détails ne sont pas approfondis. Consultez les sections appropriées du manuel pour plus d'informations sur les sujets abordés.

3.1 Connexion et déconnexion au serveur

Pour vous connecter au serveur, vous aurez dans la plupart des cas à fournir un nom d'utilisateur à MySQL, et, sûrement, un mot de passe. Si le serveur fonctionne sur une autre machine que la vôtre, vous devrez spécifier son adresse. Contactez votre administrateur pour connaître les paramètres à utiliser lors de la connexion (hôte, nom d'utilisateur, mot de passe à utiliser...). Une fois que vous aurez les bons paramètres, vous pourrez vous connecter de la façon suivante :

```
 shell> mysql -h hote -u utilisateur -p
Enter password: ********
```

`********` représente votre mot de passe, entrez−le lorsque `mysql` affiche `Enter password:` .Si tout fonctionne, vous devrez voir quelques informations d'introduction suivies d'une invite de commande `mysql>` :

```
 shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log

 Type 'help' for help.

 mysql>
```

L'invite vous dit que `mysql` attend que vous entriez des commandes.Quelques installations de MySQL autorisent les connexions anonymes au serveur tournant sur l'hôte local. Si c'est la cas sur votre machine, vous devriez arriver à vous connecter à ce serveur en invoquant la commande `mysql` sans aucune option :

```
 shell> mysql
```

Après vous être connecté avec succès, vous pouvez vous déconnecter à tout moment en entrant `QUIT` dans l'invite `mysql>` :

```
 mysql> QUIT
Bye
```

Vous pouvez aussi le faire en appuyant sur Ctrl−D.

La plupart des exemples dans les sections suivantes supposent que vous êtes connecté au serveur. Cela se voit à l'invite `mysql>` .

3.2 Entrer des requêtes

Assurez−vous d'être connecté au serveur, comme expliqué précédemment dans cette section. Faire ceci ne sélectionnera pas une base par lui même, mais c'est normal. A ce stade, il est important de découvrir la façon dont sont publiées les requêtes, pour ensuite pouvoir créer des tables, y insérer et rechercher des données. Cette section décrit les principes de base pour entrer une commande, en utilisant plusieurs requêtes que vous pouvez essayer pour vous familiariser avec la façon dont `mysql` fonctionne.

Voilà une commande simple qui demande au serveur de vous donner son numéro de version et la date courante. Entrez−la comme suit, juste après l'invite `mysql>` puis pressez Enter :

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

La requête révèle plusieurs choses à propos de `mysql` :

- Une commande consiste normalement en une commande SQL suivie d'un point−virgule. (Il y a quelques cas ou le point−virgule n'est pas requis. `QUIT` , mentionnée plus tôt, en fait partie. Nous verrons les autres plus tard.)
- Lorsque vous entrez une commande, `mysql` l'envoi au serveur pour l'exécution et affiche le résultat, puis affiche un autre `mysql>` pour indiquer qu'il attend une autre commande.
- `mysql` affiche le résultat des requêtes dans une table (lignes et colonnes). La première ligne contient le nom des colonnes. Les lignes suivantes constituent le résultat de la requête. Normalement, les titres des colonnes sont les noms des champs des tables de la base de données que vous avez récupérés. Si vous récupérez la valeur d'une expression au lieu d'une colonne (comme dans l'exemple précédent), `mysql` nomme la colonne en utilisant l'expression elle−même.
- `mysql` vous indique combien de lignes ont été retournées et combien de temps d'exécution la requête a pris, ce qui vous donnera une approximation des performances du serveur. Ces valeurs sont imprécises car elles représentent le temps logiciel (et non le temps processeur ou matériel), et qu'elles sont affectées par des facteurs tels que la charge du serveur ou l'accessibilité du réseau. (Dans un soucis de brièveté, la ligne contenant `` `rows in set` '' n'est plus montrée dans les exemples suivants de ce chapitre.)

Les mots−clef peuvent être entrés sous n'importe quelle forme de casse. Les requêtes suivantes sont équivalentes :

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Voilà une autre requête. Elle montre que vous pouvez utiliser `mysql` en tant que simple calculatrice :

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

Les commandes vues jusqu'à présent ont été relativement courtes, et tenaient sur une seule ligne. Vous pouvez même entrer plusieurs requêtes sur une seule ligne. Il suffit de terminer chacune d'elle par un point−virgule :

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION()    |
+--------------+
```

```
| 3.22.20a-log |
+--------------+

 +---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Une commande ne doit pas être obligatoirement sur une seule ligne ; les commandes qui exigent plusieurs lignes ne sont pas un problème. `mysql` détermine où se situe la fin de votre commande en recherchant le point−virgule de terminaison, et pas l'extrémité de la commande entrée. (Dans d'autres termes, `mysql` accepte des formats libres d'entrée : il collecte les lignes entrées mais ne les exécute qu'une fois le point−virgule trouvé.)

Voilà une seule requête sur plusieurs lignes :

```
 mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith&#64;localhost | 1999-03-18   |
+--------------------+--------------+
```

Dans cet exemple, notez comment l'invite change de `mysql>` à `->` après avoir entré la première ligne d'une requête multi−lignes. C'est la façon dont `mysql` indique qu'il n'a pas vu de requête complète et qu'il attend la fin de celle−ci. L'invite est votre ami en vous fournissant la rétroactivité. Si vous utilisez cette rétroactivité, vous vous rendrez toujours compte de ce que `mysql` attend.

Si vous décidez d'annuler une commande que vous êtes en train de taper, faites−le en entrant `\c` :

```
 mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Ici aussi, portez votre attention sur l'invite. Elle se transforme à nouveau en `mysql>` après que vous ayez entré `\c` , vous informant que `mysql` est prêt pour une nouvelle requête.

Le tableau suivant montre les différentes invites que vous pourrez voir et résume leur signification quand à l'état dans lequel se trouve `mysql` :

Invite	**Signification**
mysql>	Prêt pour une nouvelle commande.
->	En attente de la ou des lignes terminant la commande.
'>	En attente de la prochaine ligne collectant une chaîne commencée par un guillemet simple (''').
">	En attente de la prochaine ligne collectant une chaîne commencée par un guillemet double ('"').
`>	En attente de la prochaine ligne collectant une chaîne commencée par un guillemet oblique ('`').

Les commandes sur plusieurs lignes sont la plupart du temps des accidents, lorsque vous voulez faire une commande sur une seule ligne et que vous oubliez le point−virgule de fin. Dans ce cas, `mysql` attend la suite de votre saisie :

```
 mysql> SELECT USER()
    ->
```

Si cela vous arrive (vous pensez que votre requête est complète mais la seule réponse est l'invite `->`), il est fort probable que `mysql` attende le point−virgule. Si vous ne notez pas ce que l'invite vous indique, vous pourriez patienter pendant longtemps avant de réaliser ce que vous devez faire. Entrez un point−virgule pour compléter la requête, et `mysql` devrait l'exécuter :

```
 mysql> SELECT USER()
    -> ;
+--------------------+
```

```
| USER()              |
+--------------------+
| joesmith&#64;localhost |
+--------------------+
```

L'invite `'>` ainsi que `">` apparaissent durant l'entrée de chaîne. Dans MySQL, vous pouvez écrire une chaîne entourée du caractère `'` ou bien `"` (par exemple, `'Bonjour'` or `"Au Revoir"`), et `mysql` vous laisse entrer une chaîne qui peut être sur plusieurs lignes. Lorsque vous voyez une invite comme `'>` ou `">` , cela signifie que vous avez entré une ligne contenant le caractère `'` ou `"` , mais vous n'avez pas encore entré le caractère correspondant qui termine votre chaîne. C'est pratique si vous entrez réellement une chaîne à lignes multiples, mais est−ce probable ? Pas vraiment. Plus souvent, les invites `'>` et `">` indiquent que vous avez, par inadvertance, oublié un caractère de fermeture. Par exemple :

```
 mysql> SELECT * FROM ma_table WHERE nom = "Smith AND age < 30;
    ">
```

Si vous entrez cette requête `SELECT` , puis appuyez sur Enter et attendez le résultat, rien ne se passera. Au lieu de vous demander pourquoi la requête met si longtemps à s'exécuter, remarquez que l'invite de commande s'est transformée en `">` . Cela indique que `mysql` attend de voir la fin d'une chaîne de caractères non−terminée. (Voyez−vous l'erreur dans cette requête ? Il manque le second guillemet à la suite de `"Smith` .)

Que faire ? Le plus simple est d'annuler la commande. Toutefois, vous ne pouvez vous contenter de taper `\c` dans ce cas−là, car `mysql` l'interprète comme une partie de la chaîne qu'il est en train de collecter ! A la place, entrez le second guillemet (pour que `mysql` sache que vous avez fini la chaîne), puis entrez `\c` :

```
 mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
    "> "\c
mysql>
```

L'invite se change à nouveau en `mysql>` , indiquant que `mysql` est prêt pour une nouvelle requête.

Il est important de savoir ce que les invites `'>` et `">` signifient, car si vous avez entré par erreur une chaîne non terminée, toutes les lignes suivantes que vous entrerez seront ignorées par `mysql` , même une ligne contenant `QUIT` ! Cela peut prêter à confusion, spécialement si vous ne savez pas que vous devez fournir le guillemet fermant avant de pouvoir annuler la commande courante.

3.3 Création et utilisation d'une base de données

Maintenant que vous savez entrer des commandes, il est temps d'accéder à une base.

Supposons que vous avec plusieurs animaux chez vous (dans votre ménagerie) et que vous voulez garder diverses informations les concernant. Vous pouvez le faire en créant des tables pour stocker vos données et y charger vos informations. Vous pourrez alors répondre à différentes sortes de questions à propos de vos animaux en récupérant les données à partir des tables. Cette section vous montre comment :

- Créer une base de données
- Créer une table
- Charger des données dans vos tables
- Récupérer des données à partir des tables de différentes façons
- Utiliser plusieurs tables

La base de données de la ménagerie va être simple (délibérément), mais il n'est pas difficile de penser à des situations courantes de la vie où vous aurez à utiliser un tel type de base de données. Par exemple, une telle base pourrait être utilisée par un éleveur pour gérer sa boutique, ou par un vétérinaire pour garder des traces de ses patients. Une distribution de la ménagerie contenant quelques requêtes et des exemples de données utilisées dans la section suivante peuvent être trouvés sur le site web de MySQL. Ils sont disponibles au format compressé `tar` (http://www.mysql.com/Downloads/Contrib/Examples/menagerie.tar.gz) ou au format Zip (http://www.mysql.com/Downloads/Contrib/Examples/menagerie.zip).

Utilisez la commande `SHOW` pour trouver quelles bases existent déjà sur le serveur :

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

La liste des bases de données est probablement différente sur votre machine, mais les bases `mysql` et `test` y figurent sûrement. La base `mysql` est requise car elle gère les accès et les privilèges. La base `test` est souvent fournie pour que les utilisateurs y effectuent leurs tests.

Notez que vous ne pourrez voir toutes les bases de données si vous n'avez pas le privilège `SHOW DATABASES` . Syntaxe de `GRANT` et `REVOKE` .

Si la base de données `test` existe, essayez d'y accéder :

```
mysql> USE test
Database changed
```

Notez que `USE` , comme `QUIT` , ne requiert pas de point−virgule. (Vous pouvez terminer ces commandes avec un point−virgule ; cela ne posera pas de problèmes.) La commande `USE` est spéciale d'un autre point de vue : elle doit être donnée sur une seule ligne.Vous pouvez utiliser la base de données `test` (si vous y avez accès) pour les exemples qui suivent, mais tout ce que vous créerez dans cette base pourra être effacé par quiconque y a accès. Pour cette raison, vous feriez mieux de demander à votre administrateur MySQL la permission d'utiliser une base de données rien que pour vous. Supposez que vous voulez nommer la votre `menagerie` . L'administrateur a besoin d'exécuter une commande telle que :

```
mysql> GRANT ALL ON menagerie.* TO votre_nom_mysql;
```

où `votre_nom_mysql` est le nom d'utilisateur MySQL qui vous est assigné.

3.3.1 Créer et sélectionner une base de données

Si l'administrateur vous a créé une base de données lors du paramétrage de vos droits, vous pouvez commencer à l'utiliser. Sinon, vous aurez besoin de la créer par vous−même :

```
mysql> CREATE DATABASE menagerie;
```

Sous Unix, les noms des bases de données sont sensibles à la casse (ce qui diffère des mots réservés de SQL), ce qui fait que vous devez toujours vous référer à votre base de données avec `menagerie` , non avec `Menagerie` , `MENAGERIE` , ou d'autres variantes. Cela est aussi valable pour les noms de tables. (Sous Windows, cette restriction n'est pas appliquée, même si vous devez vous référer à une table ou une base de la même façon dans une même requête).

La création d'une base de données ne la sélectionne pas pour l'utilisation ; vous devez le faire explicitement. Pour rendre `menagerie` la base courante, utilisez cette commande :

```
mysql> USE menagerie
Database changed
```

Votre base a besoin d'être créée juste une fois, mais vous devez la sélectionner pour l'utiliser, chaque fois que vous débutez une session `mysql` . Vous pouvez le faire en publiant une requête `USE` comme ci−dessus. Sinon, vous pouvez sélectionner la base directement dans la ligne de commande lorsque vous invoquez `mysql` . Vous devez juste spécifier son nom après les paramètres de connexion dont vous avez besoin. Par exemple :

```
 shell> mysql −h hote −u utilisateur −p menagerie
Enter password: ********
```

Notez que `menagerie` n'est pas votre mot de passe dans la commande que nous venons de montrer. Si vous voulez le fournir dans la ligne de commande après l'option `−p` , vous devez le faire sans espace entre les deux (par exemple, tapez `−pmonmotdepasse` , et non `−p monmotdepasse`). Toutefois, mettre le mot de passe en ligne de commande n'est pas recommandé, car le faire permettrait à d'autres utilisateurs connectés sur votre machine de l'obtenir.

3.3.2 Création d'une table

Créer la base de données est la partie facile, mais jusque−là elle est vide, comme vous le montre `SHOW TABLES` :

```
 mysql> SHOW TABLES;
Empty set (0.00 sec)
```

La partie la plus difficile est le choix de la structure de la base de données : de quelles tables aurez vous besoin et quelles colonnes devront figurer dans chacune d'elles.

Vous voudrez une table qui contient un enregistrement pour chaque animal. On peut l'appeler la table `animal` , et elle devra contenir, au minimum, le nom de chaque animal. Puisque le nom tout seul n'est pas intéressant, la table devra contenir d'autres informations. Par exemple, si plus d'une personne de votre famille possède un animal, vous voudrez lister le nom du maître de chaque animal. Vous voudrez peut−être aussi enregistrer une description basique comme l'espèce ou le sexe.

Et pour l'âge ? C'est intéressant, mais n'est pas bon pour un stockage en base de données. L'âge change chaque jour, vous devrez donc mettre à jour vos enregistrements assez souvent. Il est préférable de stocker une valeur fixe, comme la date de naissance. Dans ce cas−là, à chaque fois que vous aurez besoin de l'âge, vous pourrez l'obtenir en faisant la différence entre la date courante et la date enregistrée. MySQL fournit des fonctions de calcul sur les dates, cela ne sera donc pas difficile. Enregistrer la date de naissance, au lieu de l'âge a d'autres avantages :

- Vous pouvez utiliser la base de données pour des tâches, comme la génération d'un rappel pour les prochains anniversaires d'animaux. (Si vous trouvez que ce type de requêtes est quelque peu idiot, notez que c'est la même question que vous vous poseriez dans le contexte d'une base de données d'affaires pour identifier les clients à qui vous aurez besoin d'envoyer un message de voeux, pour cette touche informatiquement assistée d'humanisme.)
- Vous pouvez calculer l'âge à partir d'autres dates que la date du jour. Par exemple, si vous stockez la date de la mort dans la base de données, vous pourrez facilement calculer l'âge qu'avait un animal à sa mort.

Vous trouverez probablement d'autres informations qui pourront être utiles dans la table `animal` , mais celles identifiés jusqu'à maintenant sont largement suffisantes pour l'instant : nom, maître, espèce, sexe, naissance, et mort.

Utilisez une requête `CREATE TABLE` pour spécifier la structure de votre table :

```
mysql> CREATE TABLE animal (nom VARCHAR(20), maitre VARCHAR(20),
    -> espece VARCHAR(20), sexe CHAR(1), naissance DATE, mort DATE);
```

`VARCHAR` est un bon choix pour les colonnes `nom` , `maitre` , et `espece` car leurs valeurs varient en longueur. La longueur de ces colonnes ne doit pas nécessairement être la même, et n'a pas besoin d'être forcement `20` . Vous pouvez choisir une taille entre `1` et `255` , celle qui vous semblera la plus raisonnable. (Si vous faites un mauvais choix et que vous vous apercevez plus tard que vous avez besoin d'un champ plus long, MySQL fournit la commande `ALTER TABLE` .)Le sexe des animaux peut être représenté de plusieurs façons, par exemple, `"m"` et `"f"` , ou bien `"male"` et `"femelle"` . Il est plus simple d'utiliser les caractères simples `"m"` et `"f"` .

L'utilisation du type de données `DATE` pour les colonnes `naissance` et `mort` est un choix plutôt judicieux.

Maintenant que vous avez créé une table, `SHOW TABLES` devrait produire de l'affichage :

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| animal              |
+---------------------+
```

Pour vérifier que la table a été créée de la façon que vous vouliez , utilisez la commande `DESCRIBE` :

```
mysql> DESCRIBE animal;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nom       | varchar(20) | YES  |     | NULL    |       |
| maitre    | varchar(20) | YES  |     | NULL    |       |
| espece    | varchar(20) | YES  |     | NULL    |       |
| sexe      | char(1)     | YES  |     | NULL    |       |
| naissance | date        | YES  |     | NULL    |       |
| mort      | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Vous pouvez utiliser `DESCRIBE` quand vous voulez, par exemple, si vous avez oublié les noms des colonnes dans votre table ou leurs types.

3.3.3 Charger des données dans une table

Après la création de votre table, vous aurez besoin de la remplir. Les commandes `LOAD DATA` et `INSERT` sont utiles pour cela.

Supposons que les enregistrements de vos animaux peuvent être décrits comme suit. (Observez que MySQL attend les dates au format `YYYY-MM-DD` ; cela peut différer de ce à quoi vous êtes habitué.)

nom	**maître**	**races**	**sexe**	**naissance**	**mort**
Fluffy	Harold	chat	f	1993-02-04	
Claws	Gwen	chat	m	1994-03-17	
Buffy	Harold	chien	f	1989-05-13	
Fang	Benny	chien	m	1990-08-27	
Bowser	Diane	chien	m	1998-08-31	1995-07-29
Chirpy	Gwen	oiseau	f	1998-09-11	
Whistler	Gwen	oiseau		1997-12-09	
Slim	Benny	serpent	m	1996-04-29	

Puisque vous commencez avec une table vide, il est facile de la remplir en créant un fichier texte contenant une ligne pour chaque animal que vous avez, puis charger son contenu à l'aide d'une seule commande.

Vous pouvez créer un fichier `pet.txt` contenant un enregistrement par ligne, avec les valeurs séparés par des tabulations, et ordonnées comme les champs l'étaient dans la requête `CREATE TABLE` . Pour les données manquantes (comme un sexe inconnu ou la date de mort d'un animal toujours en vie), vous pouvez utiliser les valeurs `NULL` . Pour les représenter dans votre fichier texte, utilisez `\N` . Par exemple, l'enregistrement de Whistler l'oiseau ressemblera à ça (l'espace entre les valeurs est une tabulation) :

nom	maître	race	sexe	naissance	mort
Whistler	Gwen	bird	\N	1997-12-09	\N

Pour charger le fichier `pet.txt` dans la table `pet` , utilisez cette commande :

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Notez que si vous créez un fichier sur Windows, avec un éditeur qui utilise des caractères de lignes comme `\r\n` , vous devez utiliser :

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Vous pouvez spécifier la valeur du séparateur de colonnes et le marqueur de fin de lignes explicitement dans la commande `LOAD DATA` si vous le voulez, mais les valeurs par défaut sont la tabulation et le retour à la ligne. Ceux−là sont suffisants pour que la commande lise le fichier `pet.txt` correctement.

Si la commande échoue, il est probable que votre installation MySQL n'a pas la possibilité d'accéder aux fichiers. Voyez Problèmes de sécurité avec `LOAD DATA LOCAL` pour plus d'informations sur comment modifier cela.

Lorsque vous voulez ajouter des enregistrements un par un, la commande `INSERT` est utile. Dans sa forme la plus simple, où vous spécifiez une valeur pour chaque colonne, dans l'ordre où les colonnes sont listées dans la requête `CREATE TABLE` . Supposons que Diane achète un nouvel hamster nommé Puffball. Vous pourriez ajouter ce nouvel enregistrement en utilisant un `INSERT` de la façon suivante :

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Notez que les chaînes de caractères et les valeurs de dates sont spécifiées en tant que chaînes protégées par des guillemets. De plus, avec `INSERT` vous pouvez insérer la valeur `NULL` directement pour représenter une valeur manquante. Vous n'utilisez pas `\N` comme vous le faites avec `LOAD DATA` .A partir de cet exemple, vous devriez être capable de voir qu'il y a beaucoup plus de commandes à taper lorsque vous utilisez la commande `INSERT` au lieu de `LOAD DATA` .

3.3.4 Récupérer des informations à partir d'une table

La commande `SELECT` est utilisée pour récupérer des informations à partir d'une table. La forme usuelle est :

```
 SELECT quoi_selectionner
FROM quel_table
WHERE conditions_a_satisfaire
```

`quoi_selectionner` indique ce que vous voulez voir. Cela peut être une liste de colonnes, ou `*` pour indiquer ``toutes les colonnes''. `quel_table` indique la table à partir de laquelle récupérer les données. La clause `WHERE` est optionnelle. Si elle est présente, `conditions_a_satisfaire` spécifie les conditions que les lignes doivent satisfaire pour être séléctionnées.

3.3.4.1 Sélectionner toutes les données

La plus simple forme de `SELECT` récupère toutes les données d'une table :

```
mysql> SELECT * FROM animal;
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | chat    | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | chien   | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | chien   | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | chien   | m    | 1998-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL       |
```

```
| Whistler | Gwen   | oiseau  | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | serpent | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Cette forme de `SELECT` est utile si vous voulez récupérer la table entière. par exemple, après l'avoir juste remplie avec vos données d'origine. Il apparaît alors qu'une erreur s'était glissée dans votre fichier de données : Bowser a l'air d'être né après sa mort ! En consultant le papier original de son pedigree, vous trouvez que la date correcte est 1989 et non pas 1998. Il y a au moins deux façons de corriger cela :

- Corriger le fichier `animal.txt` pour corriger l'erreur, puis vider et recharger à nouveau la table en utilisant `DELETE` et `LOAD DATA` :

  ```
   mysql> SET AUTOCOMMIT=1;  # Utilisé pour une recréation rapide de la table
  mysql> DELETE FROM animal;
  mysql> LOAD DATA LOCAL INFILE "animal.txt" INTO TABLE animal;
  ```

  Toutefois, si vous choisissez cette méthode, vous devrez aussi rentrer à nouveau l'enregistrement de Puffball.
- Corriger uniquement l'enregistrement erroné avec une requête `UPDATE` :

  ```
   mysql> UPDATE animal SET naissance = "1989-08-31" WHERE nom = "Bowser";
  ```

Comme nous l'avons montré, il est facile de récupérer toutes les données d'une table. Toutefois, vous ne voudrez sûrement pas le faire, surtout si la table devient imposante. A la place, vous serez plus intéressé par répondre à une question particulière, dans ce cas−là, vous spécifiez quelques contraintes pour les informations que vous voulez. Regardons quelques requêtes de sélection qui répondent à des questions à propos de vos animaux.

3.3.4.2 Sélectionner des lignes particulières

Vous pouvez sélectionner des lignes particulières de votre table. Par exemple, si vous voulez vérifier la modification que vous avez effectuée sur la date de naissance de Bowser, sélectionnez son enregistrement comme suit :

```
 mysql> SELECT * FROM animal WHERE nom = "Bowser";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

L'affichage confirme que la date est correcte maintenant : 1989, et non 1998.

La comparaison des chaînes de caractères se fait normalement avec sensibilité à la casse, vous pouvez donc spécifier le nom `"bowser"` , `"BOWSER"` , etc. Le résultat de la requête sera le même.

Vous pouvez spécifier des conditions sur toutes les colonnes, pas seulement `nom` . Par exemple, si vous voulez savoir quels animaux sont nés après 1998, testez la colonne naissance :

```
 mysql> SELECT * FROM animal WHERE naissance >= "1998-1-1";
+----------+--------+---------+------+-----------+-------+
| nom      | maitre | espece  | sexe | naissance  | mort  |
+----------+--------+---------+------+-----------+-------+
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL  |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL  |
+----------+--------+---------+------+-----------+-------+
```

Vous pouvez combiner plusieurs conditions, par exemple, pour trouver les chiennes :

```
 mysql> SELECT * FROM animal WHERE espece = "chien" AND sexe = "f";
+-------+--------+--------+------+-----------+-------+
| nom   | maitre | espece | sexe | naissance  | mort  |
+-------+--------+--------+------+-----------+-------+
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL  |
+-------+--------+--------+------+-----------+-------+
```

La requête précédente utilise l'opérateur logique `AND` . L'opérateur `OR` existe aussi :

```
mysql> SELECT * FROM animal WHERE espece = "serpent" OR espece = "oiseau";
+----------+--------+---------+------+------------+-------+
| nom      | maitre | espece  | sexe | naissance  | mort  |
+----------+--------+---------+------+------------+-------+
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL  |
| Whistler | Gwen   | oiseau  | NULL | 1997-12-09 | NULL  |
| Slim     | Benny  | serpent | m    | 1996-04-29 | NULL  |
+----------+--------+---------+------+------------+-------+
```

`AND` et `OR` peuvent être utilisés ensemble. Si vous le faites, une bonne idée est d'utiliser les parenthèses pour indiquer comment les conditions doivent être regroupées :

```
mysql> SELECT * FROM animal WHERE (espece = "chat" AND sexe = "m")
    -> OR (espece = "chien" AND sexe = "f");
+-------+--------+--------+------+------------+-------+
| nom   | maitre | espece | sexe | naissance  | mort  |
+-------+--------+--------+------+------------+-------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL  |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL  |
+-------+--------+--------+------+------------+-------+
```

3.3.4.3 Sélectionner des colonnes particulières

Si vous ne voulez pas voir les lignes entières de votre table, nommez les colonnes qui vous intéressent, en les séparant par des virgules. Par exemple, si vous voulez savoir quand vos animaux sont nés, sélectionnez les colonnes `nom` et `naissance` :

```
mysql> SELECT nom, naissance FROM animal;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Pour trouver qui possède les animaux, utilisez cette requête :

```
mysql> SELECT maitre FROM animal;
+--------+
| maitre |
+--------+
| Harold |
| Gwen   |
| Harold |
| Benny  |
| Diane  |
| Gwen   |
| Gwen   |
| Benny  |
| Diane  |
+--------+
```

Toutefois, remarquez que la requête récupère le champ `maitre` de chaque enregistrement, et certains apparaissent plus d'une fois. Pour minimiser l'affichage, récupérez chaque résultat unique une seule fois en ajoutant le mot−cle `DISTINCT` :

```
mysql> SELECT DISTINCT maitre FROM animal;
+--------+
| maitre |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

Vous pouvez utiliser une clause `WHERE` pour combiner la sélection des lignes avec celle des colonnes. Par exemple, pour obtenir les dates de naissance des chiens et chats uniquement, utilisez cette requête :

```
mysql> SELECT nom, espece, naissance FROM animal
    -> WHERE espece = "chien" OR espece = "chat";
+--------+--------+------------+
| nom    | espece | naissance  |
+--------+--------+------------+
| Fluffy | chat   | 1993-02-04 |
| Claws  | chat   | 1994-03-17 |
```

```
| Buffy  | chien  | 1989-05-13 |
| Fang   | chien  | 1990-08-27 |
| Bowser | chien  | 1989-08-31 |
+--------+--------+------------+
```

3.3.4.4 Trier les enregistrements

Vous avez sûrement noté dans les exemples précédents que les lignes de résultat sont affichées sans ordre particulier. Cependant, il est souvent plus facile d'examiner les résultats lorsqu'ils sont triés d'une manière significative. Pour trier un résultat, vous devez utiliser une clause `ORDER BY` .

L'exemple suivant présente les dates d'anniversaire des animaux, triées par date :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Sur les noms de colonnes, le tri − comme toutes les opérations de comparaison − est normalement exécuté sans tenir compte de la casse. Cela signifie que l'ordre sera indéfini pour les colonnes qui sont identiques, excepté leur casse. Vous pouvez forcer le tri sensible à la casse en utilisant la clause BINARY : `ORDER BY BINARY(champ)` . Pour trier dans l'ordre inverse, ajoutez le mot−clé `DESC` (décroissant) au nom de la colonne à trier :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance DESC;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Vous pouvez effectuer un tri sur plusieurs colonnes. Par exemple, pour trier par types d'animaux, puis par la date d'anniversaire des animaux, en plaçant les plus jeunes en premier, utilisez la requête suivante :

```
mysql> SELECT nom, espece, naissance FROM animaux ORDER BY espece, naissance DESC;
+----------+---------+------------+
| nom      | espece  | naissance  |
+----------+---------+------------+
| Chirpy   | oiseau  | 1998-09-11 |
| Whistler | oiseau  | 1997-12-09 |
| Claws    | chat    | 1994-03-17 |
| Fluffy   | chat    | 1993-02-04 |
| Fang     | chien   | 1990-08-27 |
| Bowser   | chien   | 1989-08-31 |
| Buffy    | chien   | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | serpent | 1996-04-29 |
+----------+---------+------------+
```

Notez que le mot−clé `DESC` est appliqué uniquement au nom de la colonne qui le précède (`naissance`) ; les valeurs `espece` continuent à être triées dans l'ordre croissant.

3.3.4.5 Calcul sur les Dates

MySQL fournit plusieurs fonctions que vous pouvez utiliser pour effectuer des calculs sur les dates, par exemple, pour calculer l'âge ou pour extraire des parties de date.

Pour déterminer quel âge a chacun de vos animaux, vous devez calculer la différence entre l'année en cours et l'année de naissance, puis soustraire à la date courante si la date du jour se produit plus tôt dans l'année civile que la date de naissance. La requête suivante montre, pour chaque animal, la date de naissance, la date courante, ainsi que l'âge en années.

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux;
+----------+------------+-------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+-------------+------+
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
+----------+------------+-------------+------+
```

Ici, `YEAR()` extrait l'année de la date et `RIGHT()` extrait les 5 caractères les plus à droite de la date qui représentent `MM-DD` (année civile). La partie de l'expression qui compare les valeurs de `MM-DD` évalue à 1 ou à 0, qui ajustent la différence d'année à la baisse, si `CURRENT_DATE` se produit plus au début de l'année que la `naissance` . L'expression complète est un peu plus fine en utilisant un alias (`age`) pour produire un nom de colonne un peu plus significatif.

La requête fonctionne, mais le résultat pourrait être lu plus facilement si les lignes étaient présentées dans le même ordre. Cela peut être obtenu en ajoutant une clause `ORDER BY nom` pour trier le résultat par nom :

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux ORDER BY nom;
+----------+------------+-------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+-------------+------+
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
+----------+------------+-------------+------+
```

Pour trier le résultat par l' `age` plutôt que par le `nom` , utilisez simplement une clause `ORDER BY` différente :

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux ORDER BY age;
+----------+------------+-------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+-------------+------+
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
+----------+------------+-------------+------+
```

Une requête similaire peut être utilisée pour déterminer l'âge qu'avait un animal à sa mort. Vous determinez les animaux qui le sont en regardant les valeurs `mort` qui ne valent pas `NULL` . Alors, pour ceux dont la valeur est non `NULL` , calculez la différence entre la `mort` et la `naissance` :

```
mysql> SELECT nom, naissance, mort,
    -> (YEAR(death)-YEAR(naissance)) - (RIGHT(mort,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux WHERE mort IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| nom    |naissance   | mort       | age  |
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 |    5 |
+--------+------------+------------+------+
```

Cette requête utilise `mort IS NOT NULL` plutôt que `mort <> NULL` parce que `NULL` est une valeur spéciale. Cela sera expliqué plus tard. Travailler avec les valeurs `NULL` .

Vous désirez savoir quels sont les animaux qui ont leur anniversaire le mois prochain ? Pour effectuer ce type de calculs, l'année et le jour ne sont pas utiles ; vous voulez simplement extraire le mois de la colonne `naissance` . MySQL fournit plusieurs fonctions d'extraction de parties de dates, comme `YEAR()` , `MONTH()` , et `DAYOFMONTH()` . `MONTH()` est la fonction appropriée dans notre cas. Pour voir comment cette fonction travaille, exécutez une requête simple qui retourne l' `naissance` et le `MONTH(naissance)` :

```
mysql> SELECT nom, naissance, MONTH(naissance) FROM animaux;
+----------+------------+------------------+
| nom      | naissance  | MONTH(naissance) |
+----------+------------+------------------+
| Fluffy   | 1993-02-04 |                2 |
| Claws    | 1994-03-17 |                3 |
| Buffy    | 1989-05-13 |                5 |
| Fang     | 1990-08-27 |                8 |
| Bowser   | 1989-08-31 |                8 |
| Chirpy   | 1998-09-11 |                9 |
| Whistler | 1997-12-09 |               12 |
| Slim     | 1996-04-29 |                4 |
| Puffball | 1999-03-30 |                3 |
+----------+------------+------------------+
```

Trouver les animaux qui ont leur anniversaire dans le mois suivant est aisé. Supposez que le mois courant est Avril. Donc, la valeur du mois est `4` et vous cherchez les animaux nés en Mai (mois 5) comme ceci :

```
mysql> SELECT nom, naissance FROM animaux WHERE MONTH(naissance) = 5;
+-------+------------+
| nom   |naissance   |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

Il y a une petite complication si le mois courant est Décembre, bien sûr. Vous ne pouvez pas uniquement ajouter 1 au numéro du mois courant (`12`) et chercher les animaux qui sont nés le mois numéro 13, parce qu'il n'existe pas. A la place, vous cherchez les animaux nés en Janvier (mois numéro 1).Vous pouvez toujours écrire une requête qui fonctionne quelque soit le mois courant. Comme cela, vous n'avez pas à utiliser un numéro de mois particulier dans votre requête. `DATE_ADD()` vous permet d'ajouter un intervalle de temps à une date donnée. Si vous ajoutez un mois à la valeur de `NOW()` , et que vous extrayez le mois à l'aide de `MONTH()` , le résultat produit le mois dans lequel vous devez chercher un anniversaire :

```
mysql> SELECT nom, naissance FROM animaux
    -> WHERE MONTH(naissance) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
```

Une manière différente d'arriver au même résultat est d'ajouter `1` pour trouver le mois prochain après le mois courant (après l'usage de la fonction (`MOD`) pour ajouter à la valeur du mois la valeur `0` si il est de `12`) :

```
mysql> SELECT nom, naissance FROM animaux
    -> WHERE MONTH(naissance) = MOD(MONTH(NOW()), 12) + 1;
```

Notez que `MONTH` retourne un nombre entre 1 et 12. `MOD(quelquechose,12)` retourne un nombre entre 0 et 11. Donc, l'addition doit être faite après l'utilisation de la fonction `MOD()` , sinon, nous aurions un intervalle entre Novembre (11) et Janvier (1).

3.3.4.6 Travailler avec la valeur NULL

La valeur `NULL` peut être surprenante jusqu'à ce que vous vous y habituiez. Conceptuellement, `NULL` représente une valeur qui manque, ou une valeur inconnue, et elle est traitée différemment des autres valeurs. Pour tester la présence de la valeur `NULL` , vous ne pouvez pas utiliser les opérateurs arithmétiques habituels comme `=` , `<` , ou `<>` . Pour le voir, il suffit d'essayer ceci :

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

Clairement, vous n'obtiendrez aucun résultat valable pour ces comparaisons. Utilisez les opérateurs `IS NULL` et `IS NOT NULL` à la place :

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

Notez que deux `NULL` sont considérés comme égaux lors que vous utilisez la clause `GROUP BY` .

Avec MySQL, 0 et `NULL` représentent le booléen faux, et tout le reste représente le booléen vrai. La valeur par défaut du booléen vrai issue d'une comparaison est 1.

Lorsque vous utilisez la clause `ORDER BY` , les valeurs `NULL` sont toujours triées en premier, même si vous utilisez l'attribut `DESC` .

Ce traitement particulier de `NULL` explique pourquoi, dans la section précédente, il était nécessaire de déterminer quel animal ne vivait plus en utilisant la fonction `mort IS NOT NULL` au lieu de `mort <> NULL` .

3.3.4.7 Recherche de modèles

MySQL fournit le standard SQL des recherches de modèles, basé sur une extension des expressions régulières similaires à celles utilisées par les utilitaires Unix comme `vi` , `grep` , et `sed` .

La recherche de modèles SQL vous permet d'utiliser le caractère `'_'` pour trouver n'importe quel caractère et le caractère `'%'` pour trouver un nombre arbitraire de caractères (y compris aucun caractère). Dans MySQL, la recherche de modèles est sensible à la casse par défaut. Quelques exemples vous sont présentés ici. Notez que vous n'utilisez ni `=` ni `<>` lorsque vous utilisez la recherche de modèles SQL ; utilisez les opérateurs de comparaison `LIKE` ou `NOT LIKE` à la place.

Pour trouver les noms commençant par la lettre `'b'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "b%";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Buffy  | Harold | chien  | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

Pour trouver les noms finissant par `'fy'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "%fy";
+--------+--------+---------+------+------------+-------+
| nom    | maitre | espece  | sexe | naissance  | mort  |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | chat    | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | chien   | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Pour trouver les noms contenant le caractères `'w'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "%w%";
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Bowser   | Diane  | chien   | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen   | oiseaux | NULL | 1997-12-09 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Pour trouver les noms contenant exactement 5 caractères, utilisez le caractère de recherche `'_'` :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "_____";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

L'autre type de recherche de modèles fourni par MySQL utilise les expressions régulières étendues. Lorsque vous testez une recherche avec ce type de modèle, utilisez les opérateurs `REGEXP` et `NOT REGEXP` (ou `RLIKE` et `NOT RLIKE` qui sont des synonymes).Quelques caractéristiques des expressions régulières étendues sont :

- Le caractère `'.'` trouve n'importe quel caractère.
- Une classe de caractères `'[...]'` trouve n'importe quel caractère contenu entre les crochets. Par exemple, la classe de caractères `'[abc]'` trouve le caractère `'a'` , `'b'` , ou `'c'` . Pour définir un intervalle de caractères, utilisez un trait d'union. La classe de caractères `'[a-z]'` trouvera n'importe quel caractère minuscule, tout comme la classe `'[0-9]'` trouvera n'importe quel nombre.
- Le caractère `'*'` trouvera aucune ou plus d'instances du caractère qui le précède. Par exemple, `'x*'` trouvera n'importe quel nombre de fois le caractère `'x'` , `'[0-9]*'` trouvera n'importe quel nombre, et `'.*'` trouvera n'importe quel nombre de fois n'importe quel caractère.
- Le modèle est trouvé s'il se produit n'importe où dans la valeur testée. (Les modèles SQL ne sont trouvés que s'ils sont présents en valeur entière.)
- Pour ancrer un modèle de sorte qu'il soit trouvé au début ou à la fin de valeur testée, utilisez `'^'` au début ou bien `'$'` à la fin du modèle.

Pour démontrer comment les expressions régulières fonctionnent, les requêtes `LIKE` vues précédemment ont été réécrites pour utiliser `REGEXP` .

Pour trouver les noms qui commencent par la lettre `'b'` , utilisez `'^'` pour trouver le début du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^b";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Buffy  | Harold | chien  | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

Avant la version 3.23.4 de MySQL, `REGEXP` était sensible à la casse, et la requête précédente ne retournait aucune ligne. Pour trouver la lettre `'b'` minuscule ou majuscule, utilisez cette requête à la place :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^[bB]";
```

Depuis MySQL 3.23.4, pour forcer `REGEXP` à être sensible à la casse, utilisez le mot−clé `BINARY` pour faire de la chaîne, une chaîne binaire. Cette requête trouvera uniquement la lettre minuscule `'b'` au début du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP BINARY "^b";
```

Pour trouver les noms finissant par `'fy'` , utilisez `'$'` pour trouver la fin du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "fy$";
+--------+--------+--------+-------+------------+------+
| nom    | maitre | espece | sexe  | naissance  | mort |
+--------+--------+--------+-------+------------+------+
| Fluffy | Harold | chat   | f     | 1993-02-04 | NULL |
| Buffy  | Harold | chien  | f     | 1989-05-13 | NULL |
+--------+--------+--------+-------+------------+------+
```

Pour trouver les noms contenant la lettre 'w' minuscule ou majuscule, utilisez la requête suivante :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "w";
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Bowser   | Diane  | chien   | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen   | oiseaux | NULL | 1997-12-09 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Parce qu'une expression régulière est trouvée si le modèle se trouve n'importe où dans la valeur, il n'est pas nécessaire dans la requête précédente de mettre un joker de chaque côté du modèle recherché pour trouver la valeur entière comme cela aurait été le cas en utilisant les modèles de recherche SQL.

Pour trouver les noms contenant exactement 5 caractères, utilisez '^' et '$' pour trouver le début et la fin du nom, et 5 instances de '.' au milieu :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^.....$";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

Vous pouvez aussi écrire la requête suivante en utilisant l'opérateur '{n' } ``répéter− n −fois'' :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^.&#123;5&#125;$";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

3.3.4.8 Compter les lignes

Les bases de données sont souvent employées pour répondre à la question : ``Combien de fois un certain type de données se trouve dans la table ?'' Par exemple, vous aimeriez savoir combien d'animaux vous avez, ou bien combien d'animaux chaque propriétaire possède, ou encore savoir différentes choses concernant vos animaux.

Savoir combien vous avez d'animaux revient à se poser la question : ``Combien de lignes y a−t−il dans la table animaux ?'' parce qu'il y a un enregistrement par animaux. La fonction COUNT() compte le nombre de résultats non NULL , donc, la requête pour compter les animaux ressemble à ceci :

```
mysql> SELECT COUNT(*) FROM animaux;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Vous pouvez trouver également les noms des propriétaires des animaux. Vous pouvez utiliser COUNT() si vous voulez trouver combien d'animaux possède chaque propriétaire :

```
mysql> SELECT maitre, COUNT(*) FROM animaux GROUP BY maitre;
+--------+----------+
| maitre | COUNT(*) |
+--------+----------+
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+
```

Notez l'utilisation de la clause GROUP BY pour grouper tous les enregistrements par propriétaire. Sans cela, vous auriez le message d'erreur suivant :

```
mysql> SELECT maitre, COUNT(maitre) FROM animaux;
ERROR 1140 at line 1: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

COUNT() et GROUP BY sont utiles pour caractériser vos données de diverses façons. Les exemples suivants montrent différentes manières pour obtenir des statistiques sur les animaux.Nombre d'animaux par espèce :

```
mysql> SELECT espece, COUNT(*) FROM animaux GROUP BY espece;
+---------+----------+
| espece  | COUNT(*) |
+---------+----------+
| oiseau  |        2 |
| chat    |        2 |
| chien   |        3 |
| hamster |        1 |
| serpent |        1 |
+---------+----------+
```

Nombre d'animaux par sexe :

```
mysql> SELECT sexe, COUNT(*) FROM animaux GROUP BY sexe;
+------+----------+
| sexe | COUNT(*) |
+------+----------+
| NULL |        1 |
| f    |        4 |
| m    |        4 |
+------+----------+
```

(Dans ce résultat, `NULL` indique les sexes inconnus.)

Nombre d'animaux par espèce et sexe :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| oiseau  | NULL |        1 |
| oiseau  | f    |        1 |
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
| hamster | f    |        1 |
| serpent | m    |        1 |
+---------+------+----------+
```

Vous n'avez pas besoin de rechercher une table entière quand vous employez `COUNT()` . Par exemple, la requête précédente, si vous voulez trouver uniquement les chiens et les chats, ressemble à cela :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux
    -> WHERE espece = "chien" OR espece = "chat"
    -> GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
+---------+------+----------+
```

Ou bien, si vous voulez trouver le nombre d'animaux par sexe, uniquement pour les animaux dont le sexe est connu :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux
    -> WHERE sexe IS NOT NULL
    -> GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| oiseau  | f    |        1 |
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
| hamster | f    |        1 |
| serpent | m    |        1 |
+---------+------+----------+
```

3.3.4.9 Utiliser plus d'une table

La table `animal` garde les enregistrements de vos animaux. Si vous voulez enregistrer d'autres informations concernant vos animaux, comme les événements de leurs vies, les visites chez le vétérinaire, ou encore lorsqu'ils ont mis bas, vous avez besoin d'une autre table. De quoi a besoin cette table ? Elle doit :

- Contenir le nom de l'animal pour savoir à quel animal cet événement se rattache.
- Une date pour savoir quand a eu lieu l'événement.
- Un champ qui décrit l'événement.
- Un champ de type événement, si vous voulez être capable de cataloguer les événements.

En prenant cela en considération, le code `CREATE TABLE` pour la table `evenement` doit ressembler à ceci :

```
mysql> CREATE TABLE evenement (nom VARCHAR(20), date DATE,
    -> type VARCHAR(15), remarque VARCHAR(255));
```

Tout comme la table `animal` , il est facile d'enregistrer les enregistrements initiaux en créant un fichier texte délimité par des tabulations, contenant l'information :

nom	date	type	remarque
Fluffy	1995-05-15	mise bas	4 chatons, 3 femelles, 1 mâles
Buffy	1993-06-23	mise bas	5 chiots, 2 femelles, 3 mâles
Buffy	1994-06-19	mise bas	3 chiots, 3 femelles
Chirpy	1999-03-21	vétérinaire	Redresser le bec
Slim	1997-08-03	vétérinaire	Cotes cassées
Bowser	1991-10-12	chenil	
Fang	1991-10-12	chenil	
Fang	1998-08-28	anniversaire	Don d'un nouvel objet de mastication
Claws	1998-03-17	anniversaire	Don d'un nouveau collier anti-puces
Whistler	1998-12-09	anniversaire	Premier anniversaire

Chargez ces enregistrements comme cela :

```
mysql> LOAD DATA LOCAL INFILE "evenement.txt" INTO TABLE evenement;
```

En se basant sur ce que vous avez appris des requêtes effectuées sur la table `animal` , vous devriez être capable de faire des recherches sur les enregistrements de la table `evenement` ; le principe est le même. Quand devez−vous vous demander si la table `evenement` est seule suffisante pour répondre à votre question ?

Supposez que vous voulez trouver l'âge de chaque animal lorsqu'il a mis bas. La table `evenement` indique quand cela s'est produit, mais pour le calcul de l'âge de la mère, vous avez besoin de sa date de naissance. Parce que ces informations sont stockées dans la table `animal` , vous avez besoin des deux tables pour cette requête :

```
mysql> SELECT animal.nom,
    -> (TO_DAYS(date) - TO_DAYS(naissance))/365 AS age,
    -> remarque
    -> FROM animal, evenement
    -> WHERE animal.nom = evenement.nom AND type = "mise bas";
+--------+------+-------------------------------+
| nom    | age  | remarque                      |
+--------+------+-------------------------------+
| Fluffy | 2.27 | 4 chatons, 3 femelles, 1 mâle |
| Buffy  | 4.12 | 5 chiots, 2 femelles, 3 mâles |
| Buffy  | 5.10 | 3 chiots, 3 femelles          |
+--------+------+-------------------------------+
```

Il y a plusieurs choses à noter concernant cette requête :

- La clause `FROM` liste les deux tables parce que la requête a besoin d'informations contenues dans ces deux tables.
- Lorsque vous combinez (joignez) des informations provenant de plusieurs tables, vous devez spécifier quels enregistrements d'une table peuvent être associés à quels enregistrements des autres tables. C'est aisé parce qu'elles ont toutes les deux une colonne `nom` . La requête utilise la clause `WHERE` pour faire correspondre les enregistrements des deux tables sur les valeurs de la colonne `nom` .
- Parce que la colonne `nom` apparaît dans les deux tables, vous devez être explicite concernant la table que vous utilisez lorsque vous vous référez à cette colonne. C'est fait en faisant précéder le nom de la colonne par le nom de la table.

Vous n'avez pas besoin de deux tables différentes pour effectuer une jointure. Quelques fois, c'est plus facile de joindre une table sur elle−même, si vous voulez comparer des enregistrements dans une table avec d'autres enregistrements de la même table. Par exemple, pour trouver des paires multiples parmi vos animaux, vous pouvez joindre la table `animal` sur elle−même pour trouver les paires mâles / femelles par rapport à l'espèce :

```
mysql> SELECT p1.nom, p1.sexe, p2.nom, p2.sexe, p1.espece
    -> FROM animal AS p1, animal AS p2
    -> WHERE p1.espece = p2.espece AND p1.sexe = "f" AND p2.sexe = "m";
+--------+------+--------+------+---------+
| nom    | sexe | nom    | sexe | espece  |
+--------+------+--------+------+---------+
| Fluffy | f    | Claws  | m    | chat    |
| Buffy  | f    | Fang   | m    | chien   |
| Buffy  | f    | Bowser | m    | chien   |
+--------+------+--------+------+---------+
```

Dans cette requête, nous avons spécifié des alias pour les noms de tables dans l'ordre de référence des colonnes et ainsi maintenir directement à quelle instance de la table chaque colonne est associée.

3.4 Obtenir des informations à propos des bases de données et des tables

Que faire si vous oubliez le nom d'une base de données ou d'une table, ou bien encore la structure d'une table donnée (par exemple, comment se nomment ses colonnes) ?

MySQL répond à ce problème en fournissant plusieurs commandes qui renvoient des informations à propos des tables et des bases de données les contenant.

Vous avez déjà vu `SHOW DATABASES` qui liste les bases de données gérées par le serveur. Pour trouver quelle base de données est actuellement sélectionnée, utilisez la fonction `DATABASE()` :

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Si vous n'avez encore sélectionné aucune base de données, le résultat est vide.

Pour trouver quelles sont les tables que la base contient (par exemple, quand vous n'êtes pas sûr du nom d'une table), utilisez cette commande :

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| evenement           |
| animal              |
+---------------------+
```

Si vous voulez en savoir d'avantage sur la structure d'une table, la commande `DESCRIBE` est utile ; elle fournit des informations sur chaque colonne de la table :

```
mysql> DESCRIBE animal;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nom       | varchar(20) | YES  |     | NULL    |       |
| maitre    | varchar(20) | YES  |     | NULL    |       |
| espece    | varchar(20) | YES  |     | NULL    |       |
| sexe      | char(1)     | YES  |     | NULL    |       |
| naissance | date        | YES  |     | NULL    |       |
| mort      | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

`Field` indique le nom de la colonne, `Type` est son type de données, `NULL` indique si la colonne peut contenir des valeurs `NULL` , `Key` indique si la colonne est indexée et `Default` spécifie la valeur par défaut de la colonne.Si vous avez des index sur une table, `SHOW INDEX FROM nom_de_table` vous fournira des informations sur elles.

3.5 Utilisation de mysql en mode batch

Dans les sections précédentes, vous avez utilisé `mysql` inter activement pour entrer vos requêtes et voir les résultats. Vous pouvez aussi utiliser `mysql` en mode batch. Pour ce faire, placez les commandes que vous voulez exécuter dans un fichier, puis dites à `mysql` de lire les entrées à partir de celui−ci :

```
shell> mysql < fichier-batch
```

Si vous utilisez `mysql` sous Windows et que vous avez des caractères spéciaux dans le fichier qui posent problèmes, vous pouvez faire :

```
dos> mysql -e "source fichier-batch"
```

Si vous devez spécifier les paramètres de connexion en ligne de commande, la commande ressemblera à ca :

```
shell> mysql -h hôte -u utilisateur -p < fichier-batch
Enter password: ********
```

Lorsque vous utilisez `mysql` de cette façon, vous créez un fichier de script, puis vous l'exécutez.Si vous voulez que le script continue, même si il y a des erreurs, vous devez utiliser l'option `--force` de la ligne de commande.

Pourquoi utilisez un script ? Voici quelques raisons :

- Si vous utilisez une requête de façon répétitive (c'est à dire, chaque jour, ou chaque semaine), en faire un script vous évitera de la réécrire chaque fois.
- Vous pouvez générer de nouvelles requêtes à partir de requêtes existantes et similaires en copiant et éditant des fichiers de scripts.
- Ce mode peut aussi être utile lors du développement d'une requête, particulièrement pour les commandes sur plusieurs lignes ou plusieurs séquences de commandes. Si vous commettez une erreur, vous n'avez pas à tout récrire. Editez juste votre script pour corriger l'erreur et dites à `mysql` de l'exécuter à nouveau.
- Si vous avez une requête qui produit beaucoup d'affichage, vous pouvez le rediriger vers un visualiseur plutôt que de le regarder défiler sur votre écran :
  ```
  shell> mysql < fichier-batch | more
  ```
- Vous pouvez capturer l'affichage dans un fichier pour un traitement ultérieur :
  ```
  shell> mysql < fichier_batch > mysql.out
  ```
- Vous pouvez distribuer votre script à d'autres personnes pour qu'elles l'exécutent.
- Quelques situations ne permettent pas une utilisation interactive, par exemple, quand vous exécutez une requête à partir d'une tâche `cron` . Dans ce cas, vous devez utiliser le mode batch.

Le format d'affichage par défaut est différent (plus concis) lorsque vous exécutez `mysql` en mode batch de celui utilisé inter activement. Par exemple, le résultat de `SELECT DISTINCT espece FROM animal` ressemble à ça inter activement :

```
+---------+
| espece  |
+---------+
| oiseau  |
| chat    |
| chien   |
| hamster |
| serpent |
+---------+
```

Mais à ça en mode batch :

```
 espece
oiseau
chat
chien
hamster
serpent
```

Si vous voulez le format d'affichage interactif en mode batch, utilisez `mysql -t` . Pour écrire les commandes exécutez dans la sortie, utilisez `mysql -vvv` .Vous pouvez aussi utiliser un script à partir de l'invite `mysql` en utilisant la commande `source` :

```
mysql> source nom_fichier;
```

3.6 Exemples de requêtes usuelles

Voilà des exemples qui vous serviront à résoudre les problèmes communs avec MySQL.

Certains exemples utilisent la table `shop` pour sauvegarder le prix de chaque article (numéro de l'élément) pour certains vendeurs (dealers). En supposant que chaque vendeur à un prix fixe pour chaque article, le couple (`article` , `dealer`) est une clef primaire pour les enregistrements.

Démarrez le client en ligne de commande `mysql` et sélectionnez une base de données :

```
mysql nom-base-données
```

(Dans la plupart des installations de MySQL, vous pouvez utiliser la base de données `test`).Vous pouvez créer la table d'exemple de la façon suivante :

```
CREATE TABLE shop (
article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
dealer  CHAR(20)                 DEFAULT ''     NOT NULL,
price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
PRIMARY KEY(article, dealer));

INSERT INTO shop VALUES
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
(3,'D',1.25),(4,'D',19.95);
```

Les données d'exemple sont :

```
mysql> SELECT * FROM shop;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.6.1 La valeur maximale d'une colonne

``Quel est le numéro du plus grand élément ?''

```
SELECT MAX(article) AS article FROM shop

+---------+
| article |
+---------+
|       4 |
+---------+
```

3.6.2 La ligne contenant le maximum d'une certaine colonne

``Trouvez le numéro, vendeur et prix de l'article le plus cher.''

En SQL−99 (et MySQL version 4.1), cela est facilement fait avec une sous−requête :

```
 SELECT article, dealer, price
FROM   shop
WHERE  price=(SELECT MAX(price) FROM shop);
```

En MySQL 4.0 ou plus ancien, vous devez le faire en deux temps :

- Obtenir le plus grand prix de la table avec une requête `SELECT` .

```
 mysql> SELECT MAX(price) FROM shop;
+------------+
| MAX(price) |
+------------+
|      19.95 |
+------------+
```

- Utiliser la valeur 19.95 avec la requête suivante :

```
 mysql> SELECT article, dealer, price
    -> FROM   shop
    -> WHERE  price=19.95;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

Une autre solution est de trier toutes les lignes en ordre décroissant, et de ne lire que la première ligne avec la clause `LIMIT` :

```
 SELECT article, dealer, price
FROM   shop
ORDER BY price DESC
LIMIT 1;
```

Note : s'il y a beaucoup d'articles chers (par exemple, chaque 19.95) la solution avec `LIMIT` n'en montre qu'un !.

3.6.3 Maximum d'une colonne par groupe

``Quel est le plus grand prix par article ?''

```
 SELECT article, MAX(price) AS price
FROM   shop
GROUP BY article

 +---------+-------+
| article | price |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

3.6.4 La ligne contenant la plus grande valeur d'un certain champ par rapport à un groupe

``Pour chaque article, trouvez le ou les vendeurs ayant le plus haut prix.''

En ANSI SQL, je l'aurais fait de cette façon avec une sous−requête :

```
 SELECT article, dealer, price
FROM   shop s1
WHERE  price=(SELECT MAX(s2.price)
              FROM shop s2
              WHERE s1.article = s2.article);
```

En MySQL il vaut mieux le faire en plusieurs étapes :

- Récupérer la liste de couples article et plus grand prix.
- Pour chaque article, récupérer la ligne qui a le plus grand prix stocké.

Cela se fait facilement avec une table temporaire :

```
CREATE TEMPORARY TABLE tmp (
       article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
       price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL);

LOCK TABLES shop read;

INSERT INTO tmp SELECT article, MAX(price) FROM shop GROUP BY article;

SELECT shop.article, dealer, shop.price FROM shop, tmp
WHERE shop.article=tmp.article AND shop.price=tmp.price;

UNLOCK TABLES;

DROP TABLE tmp;
```

Si vous n'utilisez pas une table `TEMPORARY` , vous devez aussi verrouiller celle−ci.

``Peut−on le faire avec une seule requête ?''

Oui, mais en utilisant une astuce inefficace que j'appelle ``astuce du `MAX−CONCAT` '' :

```
SELECT article,
      SUBSTRING( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 7) AS dealer,
  0.00+LEFT(      MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 6) AS price
FROM   shop
GROUP BY article;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | C      |  1.69 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

Le dernier exemple peut, bien sûr, être amélioré en découpant les colonnes concaténées dans le client.

3.6.5 Utiliser les variables utilisateur

Vous pouvez utiliser les variables utilisateur de MySQL pour garder des résultats en mémoire sans avoir à les enregistrer dans des variables temporaires du client. Variables définies par l'utilisateur .

Par exemple, pour trouver l'article avec le plus haut et le plus bas prix, vous pouvez faire :

```
mysql> SELECT &#64;min_price:=MIN(price),&#64;max_price:=MAX(price) FROM shop;
mysql> SELECT * FROM shop WHERE price=&#64;min_price OR price=&#64;max_price;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.6.6 Utiliser les clefs étrangères

Depuis la version 3.23.44 de MySQL, les tables `InnoDB` supportent les contraintes des clefs étrangères. Le moteur de tables `InnoDB` . Consultez aussi Clés étrangères .

Actuellement, vous n'avez pas besoin de clefs étrangères pour réaliser des jointures entre les tables. La seule chose que MySQL ne fait pas encore (avec les types autres que InnoDB), est CHECK pour s'assurer que que la clef que vous utilisez existe bien dans la ou les tables que vous référencez et il n'efface pas automatiquement les lignes d'une table avec une définition de clef étrangère. Si vous utilisez vos clefs comme une clef normale, tout marchera parfaitement :

```
 CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

 CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
    PRIMARY KEY (id)
);
 INSERT INTO person VALUES (NULL, 'Antonio Paz');

 INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());


 <P>
INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

 INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'red', LAST_INSERT_ID()),
(NULL, 'dress', 'blue', LAST_INSERT_ID()),
(NULL, 't-shirt', 'white', LAST_INSERT_ID());

 </P>

SELECT * FROM person;
+----+---------------------+
| id | name                |
+----+---------------------+
|  1 | Antonio Paz         |
|  2 | Lilliana Angelovska |
+----+---------------------+

 SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+
 SELECT s.* FROM person p, shirt s
 WHERE p.name LIKE 'Lilliana%'
   AND s.owner = p.id
   AND s.color <> 'white';

 +----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

3.6.7 Recherche sur deux clefs

MySQL n'optimise pas encore quand vous effectuez des recherches sur deux clefs différentes combinées avec OR (la recherche sur une clef avec différentes parties OR est elle pas mal optimisée) :

```
 SELECT champ1_index, champ2_index FROM test_table WHERE champ1_index = '1'
OR  champ2_index = '1'
```

La raison est que nous n'avons pas trouvé le temps suffisant pour parvenir à un moyen efficace de gérer cela dans un cas général. (En comparaison, la gestion de AND est maintenant complètement générale et fonctionne très bien.)

En MySQL 4.0, vous pouvez résoudre ce problème efficacement en utilisant une clause `UNION` qui combine le résultat de deux requêtes `SELECT` séparée.s Syntaxe de `UNION` . Chaque requête `SELECT` ne recherche qu'avec une seule clé, et peut être optimisée :

```
 SELECT field1_index, field2_index FROM test_table WHERE field1_index = '1'
UNION
SELECT field1_index, field2_index FROM test_table WHERE field2_index = '1';
```

Avant MYSQL 4.0, vous pouvez résoudre ce problème efficacement en utilisant une table temporaire (`TEMPORARY`). Ce type d'optimisation est très utile si vous utilisez des requêtes très complexes et que le serveur SQL fait une optimisation dans le mauvais ordre.

```
 CREATE TEMPORARY TABLE tmp
SELECT champ1_index, champ2_index FROM test_table WHERE champ1_index = '1';
INSERT INTO tmp
SELECT champ1_index, champ2_index FROM test_table WHERE champ2_index = '1';
SELECT * from tmp;
DROP TABLE tmp;
```

La méthode ci−dessus pour résoudre cette requête est en effet une `UNION` de deux requêtes. Syntaxe de `UNION` .

3.6.8 Calcul du nombre de visites par jour

Ce qui suit donne une idée d'une utilisation des fonctions de bits pour calculer le nombre de jours par mois où un utilisateur a visité une page web.

```
 CREATE TABLE t1 (year YEAR(4), month INT(2) UNSIGNED ZEROFILL,
            day INT(2) UNSIGNED ZEROFILL);
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),
            (2000,2,23),(2000,2,23);
```

La table d'exemple contient des valeurs au format année−mois−jour, qui représentent des visites d'utilisateurs sur la page. Pour déterminer le nombre de jour entre deux visites, utilisez la requête suivante :

```
 SELECT year,month,BIT_COUNT(BIT_OR(1<<day)) AS days FROM t1
      GROUP BY year,month;
```

Qui retourne :

```
 +------+-------+------+
| year | month | days |
+------+-------+------+
| 2000 |    01 |    3 |
| 2000 |    02 |    2 |
+------+-------+------+
```

Ce qui précède calcule le nombre de jours différents qui a été utilisé pour une combinaison année/mois, avec suppression automatique des doublons.

3.6.9 Utiliser AUTO_INCREMENT

L'attribut `AUTO_INCREMENT` peut être utilisé pour générer un identifiant unique pour les nouvelles lignes :

```
 CREATE TABLE animals (
            id MEDIUMINT NOT NULL AUTO_INCREMENT,
            name CHAR(30) NOT NULL,
            PRIMARY KEY (id)
            );
INSERT INTO animals (name) VALUES ("dog"),("cat"),("penguin"),
                                  ("lax"),("whale"),("ostrich");
SELECT * FROM animals;
```

Qui retourne :

```
+----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | lax     |
|  5 | whale   |
|  6 | ostrich |
+----+---------+
```

Vous pouvez obtenir la valeur utilisée de la clef `AUTO_INCREMENT` avec la fonction SQL `LAST_INSERT_ID()` ou la fonction d'API `mysql_insert_id()` .Note : Pour une insertion multi−lignes, `LAST_INSERT_ID()` / `mysql_insert_id()` retourneront la clef `AUTO_INCREMENT` de la **première** ligne insérée. Cela permet de reproduire les insertions multi−lignes sur d'autres services.

Pour les tables `MyISAM` et `BDB` vous pouvez spécifier `AUTO_INCREMENT` sur une colonne secondaire d'une clef multi−colonnes. Dans ce cas, la valeur générée pour la colonne auto−incrémentée est calculée de la façon suivante : `MAX(auto_increment_column)+1) WHERE prefix=given-prefix` . C'est utile lorsque vous voulez placer des données dans des groupes ordonnés.

```
CREATE TABLE animals (
            grp ENUM('fish','mammal','bird') NOT NULL,
            id MEDIUMINT NOT NULL AUTO_INCREMENT,
            name CHAR(30) NOT NULL,
            PRIMARY KEY (grp,id)
            );
INSERT INTO animals (grp,name) VALUES("mammal","dog"),("mammal","cat"),
                 ("bird","penguin"),("fish","lax"),("mammal","whale"),
                 ("bird","ostrich");
SELECT * FROM animals ORDER BY grp,id;
```

Qui retourne :

```
+--------+----+---------+
| grp    | id | name    |
+--------+----+---------+
| fish   |  1 | lax     |
| mammal |  1 | dog     |
| mammal |  2 | cat     |
| mammal |  3 | whale   |
| bird   |  1 | penguin |
| bird   |  2 | ostrich |
+--------+----+---------+
```

Notez que dans ce cas, la valeur d' `AUTO_INCREMENT` sera réutilisée si vous effacez la ligne avec la plus grande valeur d' `AUTO_INCREMENT` tous groupes confondus. Cela n'arrive jamais avec les tables `MyISAM` , dont les valeurs `AUTO_INCREMENT` ne sont jamais réutilisées.

3.7 Requêtes du projet Twin

A Analytikerna et Lentus, nous avons eu à mettre en place le partie système et base de données d'un grand projet de recherche. Ce projet est une collaboration entre l'institut de médecine environnementale de l'institut de Karolinska Stockholm et la section de recherche clinique d'âge et de psychologie à l'université de la Californie du sud.

Le projet nécessite une partie de récolte d'informations où tous les jumeaux en Suède de plus de 65 ans sont contactés par téléphone. Ceux qui répondent à certains critères sont admis à la seconde étape. Dans celle−ci, les jumeaux qui veulent participer rencontrent une équipe de médecins/infirmiers. Les examens incluent des examens physiques et neuropsychologiques, des tests en laboratoire, de la neuro−imagerie, des études psychologiques et de la collecte d'informations relatives à la famille. En plus de tout cela, les données à propos des facteurs de risques médicaux et environnementaux sont collectées.

Plus d'informations à propos de l'étude Twin peuvent être trouvées sur : http://www.imm.ki.se/TWIN/TWINUKW.HTM

La dernière partie de ce projet est administrée avec une interface web écrite en utilisant Perl et MySQL.

Chaque nuit, toutes les informations des interviews sont stockées dans une base de données MySQL.

3.7.1 Trouver tous les jumeaux répondant aux critères

La requête suivante a été utilisée pour déterminer qui participerait à la seconde partie du projet :

```
 SELECT
        CONCAT(p1.id, p1.tvab) + 0 AS tvid,
        CONCAT(p1.christian_name, " ", p1.surname) AS Name,
        p1.postal_code AS Code,
        p1.city AS City,
        pg.abrev AS Area,
        IF(td.participation = "Aborted", "A", " ") AS A,
        p1.dead AS dead1,
        l.event AS event1,
        td.suspect AS tsuspect1,
        id.suspect AS isuspect1,
        td.severe AS tsevere1,
        id.severe AS isevere1,
        p2.dead AS dead2,
        l2.event AS event2,
        h2.nurse AS nurse2,
        h2.doctor AS doctor2,
        td2.suspect AS tsuspect2,
        id2.suspect AS isuspect2,
        td2.severe AS tsevere2,
        id2.severe AS isevere2,
        l.finish_date
FROM
        twin_project AS tp
        /* For Twin 1 */
        LEFT JOIN twin_data AS td ON tp.id = td.id
                  AND tp.tvab = td.tvab
        LEFT JOIN informant_data AS id ON tp.id = id.id
                  AND tp.tvab = id.tvab
        LEFT JOIN harmony AS h ON tp.id = h.id
                  AND tp.tvab = h.tvab
        LEFT JOIN lentus AS l ON tp.id = l.id
                  AND tp.tvab = l.tvab
        /* For Twin 2 */
        LEFT JOIN twin_data AS td2 ON p2.id = td2.id
                  AND p2.tvab = td2.tvab
        LEFT JOIN informant_data AS id2 ON p2.id = id2.id
                  AND p2.tvab = id2.tvab
        LEFT JOIN harmony AS h2 ON p2.id = h2.id
                  AND p2.tvab = h2.tvab
        LEFT JOIN lentus AS l2 ON p2.id = l2.id
                  AND p2.tvab = l2.tvab,
        person_data AS p1,
        person_data AS p2,
        postal_groups AS pg
WHERE
        /* p1 gets main twin and p2 gets his/her twin. */
        /* ptvab is a field inverted from tvab */
        p1.id = tp.id AND p1.tvab = tp.tvab AND
        p2.id = p1.id AND p2.ptvab = p1.tvab AND
        /* Just the sceening survey */
        tp.survey_no = 5 AND
        /* Skip if partner died before 65 but allow emigration (dead=9) */
        (p2.dead = 0 OR p2.dead = 9 OR
         (p2.dead = 1 AND
          (p2.death_date = 0 OR
           (((TO_DAYS(p2.death_date) - TO_DAYS(p2.birthday)) / 365)
            >= 65))))
        AND
        (
        /* Twin is suspect */
        (td.future_contact = 'Yes' AND td.suspect = 2) OR
        /* Twin is suspect - Informant is Blessed */
        (td.future_contact = 'Yes' AND td.suspect = 1
                                 AND id.suspect = 1) OR
        /* No twin - Informant is Blessed */
        (ISNULL(td.suspect) AND id.suspect = 1
                            AND id.future_contact = 'Yes') OR
```

```
        /* Twin broken off - Informant is Blessed */
        (td.participation = 'Aborted'
         AND id.suspect = 1 AND id.future_contact = 'Yes') OR
        /* Twin broken off - No inform - Have partner */
        (td.participation = 'Aborted' AND ISNULL(id.suspect)
                                      AND p2.dead = 0))
        AND
        l.event = 'Finished'
        /* Get at area code */
        AND SUBSTRING(p1.postal_code, 1, 2) = pg.code
        /* Not already distributed */
        AND (h.nurse IS NULL OR h.nurse=00 OR h.doctor=00)
        /* Has not refused or been aborted */
        AND NOT (h.status = 'Refused' OR h.status = 'Aborted'
        OR h.status = 'Died' OR h.status = 'Other')
ORDER BY
        tvid;
```

Quelques explications :

`CONCAT(p1.id, p1.tvab) + 0 AS tvid`
: Nous voulons trier la concaténation de `id` et `tvab` dans un ordre numérique. Ajouter `0` au résultat force MySQL à le considérer comme un nombre.

colonne `id`
: Identifie une paire de jumeaux. C'est un clef dans toutes les tables.

colonne `tvab`
: Identifie un jumeau dans une paire. Valeur `1` ou `2` .

colonne `ptvab`
: Inverse de `tvab` . Si `tvab` est `1` c'est égal à `2` , et vice−versa. Elle existe pour diminuer la frappe et faciliter la tâche à MySQL lors de l'optimisation de la requête.

Cette requête montre, entre autres, comment faire pour consulter une table depuis cette même table en utilisant une jointure (`p1` et `p2`). Dans cet exemple, est utilisé pour chercher quel partenaire du projet est décédé avant l'âge de 65 ans. Si c'est le cas, la ligne n'est pas retournée.Tout ce qui précède existe dans toutes les tables avec des informations relatives aux jumeaux. Nous avons une clé sur les champs `id,tvab` (toutes les tables), et sur les champs `id,ptvab` (`person_data`) pour accélérer les requêtes.

Sur notre machine de production (un 200MHz UltraSPARC), cette requête retourne près de 150−200 lignes et prend moins d'une seconde.

Le nombre d'enregistrements dans les tables utilisées plus haut :

Table	**Lignes**
person_data	71074
lentus	5291
twin_project	5286
twin_data	2012
informant_data	663
harmony	381
postal_groups	100

3.7.2 Afficher une table avec l'état des paires de jumeaux

Chaque entrevue se finit avec un code d'état, appelé `event` . La requête ci−dessous montre comment afficher une table avec toutes les paires, rassemblées par code d'état. Elle indique combien de paires ont terminé, combien de paires ont à moitié terminé et combien on refusé, etc.

```
 SELECT
        t1.event,
        t2.event,
        COUNT(*)
FROM
        lentus AS t1,
        lentus AS t2,
        twin_project AS tp
WHERE
```

```
        /* Nous recherchons une paire à la fois */
        t1.id = tp.id
        AND t1.tvab=tp.tvab
        AND t1.id = t2.id
        /* On étudie toutes les données */
        AND tp.survey_no = 5
        /* Cela évite qu'un paire deviennen un doublon */
        AND t1.tvab='1' AND t2.tvab='2'
GROUP BY
        t1.event, t2.event;
```

3.8 Utilisation de MySQL avec Apache

Il existe des programmes vous permettant d'identifier vos utilisateurs à l'aide d'une base MySQL et qui vous permettent aussi de créer des journaux de log dans vos table MySQL.

Vous pouvez changer le format d'archivage d'Apache pour le rendre plus facilement lisible par MySQL en mettant ce qui suit dans le fichier de configuration d'Apache :

```
 LogFormat \
        "\"%h\",%&#123;%Y%m%d%H%M%S&#125;t,%>s,\"%b\",\"%&#123;Content-Type&#125;o\",  \
        \"%U\",\"%&#123;Referer&#125;i\",\"%&#123;User-Agent&#125;i\""
```

Avec MySQL, vous pouvez exécuter une requête de la sorte :

```
 LOAD DATA INFILE '/local/access_log' INTO TABLE table_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

4 Utiliser les programmes MySQL

Ce chapitre présente les programmes fournis par MySQL AB et les options disponibles pour faire fonctionner ces programmes.

La plupart des programmes ont des options qui leur sont spécifiques, mais la syntaxe de spécifications des options est similaires pour tous les programmes. Les derniers chapitres fournissent des détails sur chaque programme, y compris leurs options.

4.1 Présentation des logiciels MySQL

MySQL AB fournit différents types de logiciels :

Le serveur MYSQL et les scripts de démarrage :

`mysqld` est le serveur MySQL

`mysqld_safe` , `mysql.server` et `mysqld_multi` sont les scripts de démarrage.

`mysql_install_db` initialise le dossier de données et les premières bases.

Ces logiciels sont présentés en détail dans la section Administration de la base .

Logiciels clietns pour accéder au serveur :

`mysql` est un client en ligne de commande, pour éxécuter des commandes SQL, interactivement, ou en mode batch.

`mysqlcc` (MySQL Control Center) est un client interactif graphique, pour exécuter des commandes SQL, et administrer le serveur

`mysqladmin` est un client d'administration

`mysqlcheck` effectue les opérations de maintenance sur les tables

`mysqldump` et `mysqlhotcopy` font les sauvegardes de bases

`mysqlimport` importe des fichiers de données

`mysqlshow` affiche des informations sur les bases et les tables

Ces logiciels sont présentés en détails dans la section Clients MySQL et utilitaires .

Utilitaires qui fonctionnent indépendamment du serveur :

`myisamchk` effectue les opérations de maintenance des tables

`myisampack` produit des tables compressées, en lecture seule

`mysqlbinlog` est un outil pour traiter les fichiers de logs binaires

`perror` affiche le message associé à un code d'erreur

`myisamchk` est présenté dans la section Administration de la base . Les autres logiciels sont détaillés dans Clients MySQL et utilitaires .

La plupart des distributions MySQL incluent tous ces programmes, hormis ceux qui sont spécifiques à une plate−forme. Par exemple, les scripts de démarrage du serveur ne sont pas utilisés sur Windows. L'exception est le format `RPM` , qui est plus spécialisé. Il y a des `RPM` pour le serveur, d'autres pour les clients. Si vous pensez qu'il vous en manque un, voyez la section Installation de MySQL pour tout savoir sur les distributions et leur contenu. Il se peut alors que vous ayez un autre paquet à installer.

4.2 Appeler des programmes MySQL

Pour appeler un logiciel MySQL en ligne de commande (c'est à dire depuis un terminal), il suffit d'entrer le nom du programme, suivi d'options, et d'autres arguments, pour indiquer au programme ce que vous voulez faire. Les commandes suivantes montrent quelques appels simples. `` `shell>` '' représente l'invite de commande de votre interpretteur : il ne fait pas partie de la commande elle−même. Les invites de commandes classiques sont `$` pour `sh` et `bash` , `%` pour `csh` et `tcsh` , et `C:\>` pour

Windows `command.com` ou `cmd.exe` .

```
 shell> mysql test
shell> mysqladmin extended-status variables
shell> mysqlshow --help
shell> mysqldump --user=root personnel
```

Les arguments qui commencent par un tiret sont des options. Ils spécifient généralement le type de connexion que le serveur doit faire, ou lui donne un mode opératoire spécial. La syntaxe des options est décrite dans la section Options spécifiques des programmes .Les arguments (ceux qui ne commencent pas par un tiret), fournissent davantage d'informations au programme. Par exemple, `mysql` utilise le premier argument comme un nom de base de données : la commande `mysql test` indique que vous voulez utiliser la base de données `test` .

Les sections ultérieures décriront les programmes individuellement, avec leurs options spécifiques, et l'interpretation qu'ils font des arguments.

Certaines options sont communes à un grand nombre de programmes. Les plus communes sont `--host` , `--user` et `--password` qui spécifient respectivement le nom d'hôte du serveur MySQL, le nom d'utilisateur et le mot de passe du compte. Tous les clients MySQL comprennent ces options : ellent vous permettent d'indiquer sur quel serveur vous voulez travailler.

Vous pouvez avoir besoin d'appeler les logiciels MySQL en utilisant le chemin jusqu'au dossier `bin` dans lequel ils sont installé. Il est probable que dans ce cas, si vous rencontrez une erreur `` `program not found' `` lors de votre appel aux programmes hors du dossier `bin` . Pour rendre plus confortable l'utilisation de MySQL, vous pouvez ajouter le chemin vers `bin` dans votre variable d'environnement `PATH` . Puis, exécutez le programme en utilisant simplement son nom.

Consultez la documentation de votre interpreteur de ligne de commande pour savoir assigner la variable `PATH` . La syntaxe d'assignation des variables est spécifique à chaque terminal.

4.3 Spécifier des options aux programmes

Vous pouvez fournir des options aux programmes MySQL de différentes façons :

- En ligne de commande, après le nom du programme. C'est le plus courant pour les appels ponctuels du programme.
- Dans un fichier d'options, que le programme lit au démarrage. C'est le plus courant pour les programmes que vous voulez utiliser fréquemment.
- Dans les variables d'environnement. Ce sont des options pratiques si vous voulez les appliquer à chaque fois que vous lancez le programme, même si les fichiers d'options sont plus utilisés en pratique. (La section Plusieurs serveurs MySQL sous Unix présente une situation où les variables d'environnement peuvent être très utiles. Elle décrit une situation où vous pouvez utiliser les variables pour spécifier le numéro de port TCP/IP et le fichier de socket Unix pour le client et le serveur).

Les programmes MySQL déterminent quelles options sont disponibles en examinant d'abord les variables d'environnement, puis le fichier d'options et enfin, la ligne de commande. Si une option est spécifiées plusieurs fois, la dernière occurence sera utilisée. Cela signifie que les variables d'environnement on la plus faible prioritée, et que les options de ligne de commande ont la forte priorité.

La meilleure technique consiste à stocker les options dans un fichier d'options. Vous pourrez alors éviter de saisir les options en ligne de commande, mais vous pourrez les remplacer par d'autres valeurs ponctuellement, en utilisant la ligne de commande.

4.3.1 Options de ligne de commande de mysqld

Les options des programmes spécifiées en ligne de commande suivent ces règles :

- Les options sont données après la commande.
- Une option commence avec un ou deux tirets, suivant que c'est un nom long ou court. De nombreuses options ont les deux formats. Par exemple, `-?` et `--help` sont les formes courtes et longues de l'option qui demande à un programme d'afficher le message d'aide.
- Les noms des options sont sensibles à la casse. `-v` et `-V` sont valides tous les deux, mais ont des significations différentes. Elles correspondent aux formes courtes des options `--verbose` et `--version` .
- Certains options prennent une valeur en argument, après le nom de l'option. Par exemple, `-h localhost` et `--host=localhost` indique au client MySQL le nom d'hôte du serveur à utiliser. La valeur de l'option est le nom de l'hôte à utiliser.
- Pour une option longue qui prend une valeur, séparez l'option de la valeur avec le signe égal ('`=`'). Pour une option longue qui prend une valeur, séparez l'option de la valeur avec le signe espace. (`-hlocalhost` et `-h localhost` sont équivalents) Une exception à cette règle est l'option qui permet de spécifier le mot de passe MySQL. Cette option peut être donnée en format long comme `--password=pass_val` ou bien `--password` . Dans le dernier cas (sans mot de passe indiqué), le programme vous demandera interactivement un mot de passe. Le mot de passe peut aussi être configuré avec la forme courte `-ppass_val` , ou encore `-p` . Cependant, en forme courte, si le mot de passe est fourni, il doit suivre immédiatement la lettre, **sans espace** . La raison à cela est que le programme ne saura pas si l'argument suivant est le paramètre de l'option ou un autre argument. Par conséquent, les deux commandes suivantes ont deux significations très différente :

  ```
  shell> mysql -ptest
  shell> mysql -p test
  ```

  La première commande demande à `mysql` d'utiliser la valeur `test` comme mot de passe, mais ne spécifie pas de base de données par défaut. La seconde commande demande à `mysql` de demander le mot de passe à l'écran, et d'utiliser la base `test` comme base par défaut.

MySQL 4.0 a introduit une souplesse supplémentaire dans la manière de saisir des options. Ces modifications ont été ajoutée en MySQL 4.0.2. Certaines se rapportent à la façon de spécifier qu'une option est ``activée'' ou ``désactivée'', et d'autres se rapportent aux options qui sont disponibles dans une version, mais pas dans l'autre. Ces fonctionnalités sont décrites ultérieurement dans cette section. Un autre changement se rapport à la méthode de spécifications des variables de programme. La section Utiliser les options pour configurer les variables de programme présente en détails ce sujet.

Certaines options qui contrôlent le comportement du serveur peuvent être activées ou désactivées. Par exemple, le client `mysql` supporte l'option `--column-names` qui détermine si il faut afficher ou pas les noms des colonnes. Par défaut, cette option est activée. Cependant, vous pouvez la désactiver dans certaines situations, comme lorsque vous voulez envoyer le résultat de `mysql` dans un autre programme qui s'attend à ne recevoir que des données, et pas les entêtes.

Pour désactiver les nom des colonnes, vous pouvez spécifier l'option de trois manières différentes :

```
--disable-column-names
--skip-column-names
--column-names=0
```

Les préfixes `--disable` et `--skip` et le suffixe `=0` ont tous le même effet : ils désactivent l'option.

La forme ``active'' de l'option peut être spécifiée de ces trois manières :

```
--column-names
--enable-column-names
--column-names=1
```

Une autre modification au traitement des options, introduit en MySQL 4.0 est que vous pouvez utiliser le préfixe `--loose` pour les options de ligne de commande. Si une option est préfixée par `--loose` , le programme ne va pas se terminer avec une erreur, s'il ne reconnaît pas l'option, mais il va juste émettre une alerte :

```
shell> mysql --loose-no-such-option
mysql: WARNING: unknown option '--no-such-option'
```

Le préfixe `--loose` peut être utilise lorsque vous exécutez des programmes depuis plusieurs installations de MySQL sur la même machine : tout au moins, des serveurs aussi récents que la version 4.0.2. Ce préfixe est particulièrement utilise lorsque vous listes les options dans un fichier d'options. Une option peut ne pas être reconnue par une version du programme, avec le préfixe `--loose` , ou le préfixe `loose` dans un fichier d'options. Les versions du programme qui ne reconnaissent pas l'option émettront une alerte, mais ignoreront l'option. Cette stratégie requiert que les versions sont toutes en 4.0.2 ou plus récent, car les anciennes versions ne connaissent pas `--loose` .

4.3.2 Fichier d'options my.cnf

Les programmes MySQL peuvent lire des options de démarrage depuis un fichier d'options (aussi appelé fichier de configuration). Les fichiers d'options fournissent un moyen pratique de spécifier les options les plus courantes pour éviter de les saisir à chaque lancement du programme. Cette fonctionnalité est fournie depuis MySQL version 3.22.

Les programmes suivantes supportent les fichiers d'options : `myisamchk` , `myisampack` , `mysql` , `mysql.server` , `mysqladmin` , `mysqlbinlog` , `mysqlcc` , `mysqlcheck` , `mysqld_safe` , `mysqldump` , `mysqld` , `mysqlhotcopy` , `mysqlimport` et `mysqlshow` .

MySQL lit les fichiers d'options suivants sous Windows :

Fichier	Contenu
WINDIR\my.ini	Options globales
C:\my.cnf	Options globales

`WINDIR` représente votre dossier Windows. Il est généralement `C:\Windows` ou `C:\WinNT` . Vous pouvez déterminer sa localisation exacte à partir de la variable d'environnement `WINDIR` avec cette commande :

```
C:\> echo %WINDIR%
```

MySQL lit les options par défaut dans les fichiers suivants sous Unix :

Fichier	Objet
/etc/my.cnf	Options globales
DATADIR/my.cnf	Options spécifiques au serveur
defaults-extra-file	Le fichier spécifié par --defaults-extra-file=#
~/.my.cnf	Options spécifiques à l'utilisateur

`DATADIR` est le dossier de données de MySQL (typiquement `/usr/local/mysql/data` pour les installation binaires ou `/usr/local/var` pour une installation source). Notez que c'est ce dossier qui a été spécifié au moment de la configuration et non pas le dossier de l'option `--datadir` lorsque `mysqld` démarre ! (`--datadir` n'a aucun effet sur le serveur, car le serveur recherche les données avant de traiter les options de ligne de commande).

MySQL essaie de lire les fichiers d'options dans l'ordre dans lequel ils sont présentés ci−dessus. Si un fichier d'options n'existe pas, vous pouvez le créer avec un éditeur de texte. Si des options sont spécifiées plusieurs fois, la dernière occurrence utilisée prend la préséance sur les options spécifiées avant.

Toutes les options longues qui peuvent être donnée en ligne de commande, peuvent être mises dans un fichier d'options. Pour avoir la liste des options d'un programme, utilisez la commande `--help` .

La syntaxe de spécification dans un fichier d'option est similaire celle de ligne de commande, hormis le fait que vous omettez les deux tirets initiaux. Par exemple, `--quick` et `--host=localhost` en ligne de commande deviennent `quick` et `host=localhost` dans un fichier d'options. Pour spécifier une option de la forme `--loose-opt_name` dans un fichier d'options, écrivez la sous la forme `loose-opt_name` .

Les lignes vides du fichier d'options sont ignorées. Un fichier d'options contient des lignes ayant la forme suivante :

`#comment`
`;comment`
: Les lignes de commentaires commencent avec `'#'` ou `';'` . Depuis MySQL 4.0.14, un commentaire `'#'` peut être ouvert au milieu de la ligne.

`[group]`
: `group` est le nom du programme ou du groupe pour lequel vous souhaitez configurer des options. Après une ligne de groupe, toutes les `option` et `set-variable` s'appliqueront au groupe nommé, jusqu'à la fin du fichier d'option ou du démarrage d'un autre groupe.

`opt_name`
: Ceci est équivalent à `--opt_name` sur la ligne de commande.

`opt_name=value`
: Ceci est équivalent à `--opt_name=value` sur la ligne de commande. Dans un fichier d'options, vous pouvez mettre des espaces autour du caractère `'='` , ce qui n'est pas vrai en ligne de commande. Depuis MySQL 4.0.16, vous pouvez mettre les valeurs des options entre guillemets simples ou doubles. C'est utile lorsqu'une valeur contient un début de commentaire `'#'` ou des espaces.

`set-variable = variable=value`
: Donne à la variable programme `var_name` sa valeur. Ceci est équivalent à `--set-variable variable=value` sur la ligne de commande. Cette syntaxe doit être utilisée pour spécifier la valeur d'une variable `mysqld` . Les espaces sont autorisés autour du premier caractère `'='` mais pas autour du second. Notez que `--set-variable` est obsolète depuis MySQL 4.0, utilisez simplement `--variable=value` comme tel. Voyez Utiliser les options pour configurer les variables de programme pour plus d'informations sur la spécification des variables programme.

Les espaces initiaux et terminaux sont automatiquement effacés autour des noms d'options et de leur valeur. Vous pouvez utilisez les séquences spéciales `'\b'` , `'\t'` , `'\n'` , `'\r'` , `'\\'` et `'\s'` dans les valeurs des options pour représenter des effacement, tabulations, nouvelles lignes, retour chariot et espaces.

Sous Windows, si une valeur d'option représente un chemin de dossier, vous devez spécifier la valeur en utilisant `'/'` plutôt que `'\'` comme séparateur de dossiers. Si vous utilisez use `'\'` , vous devez le doubler `'\\'` , car `'\'` est le caractère de protection de MySQL.

Si un groupe d'options est le même que le nom d'un programme, les options de ce groupe seront réservées à ce programme.

Le groupe d'options `[client]` est lu par tous les programmes clients et pas par le serveur `mysqld` . Cela vous permet de spécifier les options qui s'appliqueront à tous les clients. Par exemple, `[client]` est le groupe parfait pour spécifier le mot de passe que vous utilisez pour vous connecter au serveur.

Mais assurez vous que le fichier est lisible et modifiable uniquement par vous−même pour que personne ne puisse découvrir votre mot de passe. Assurez vous de n'utiliser que des options qui seront reconnues par **tous** les programmes client. Les programmes qui ne comprennent pas une option vont afficher un message d'erreur lorsque vous les exécuterez.

Depuis MySQL 4.0.14, si vous voulez créer des groupes d'options qui ne doivent être lus que par une versions spécifique du serveur `mysqld` , vous pouvez le faire en utilisant des groupes avec des noms du type `[mysqld-4.0]` , `[mysqld-4.1]` , etc. Le groupe suivante indique que l'option `--new` soit utilisée avec les serveur de version 4.0.x :

```
 [mysqld-4.0]
new
```

Voici un fichier d'options globales typique :

```
 [client]
port=3306
socket=/tmp/mysql.sock

 [mysqld]
port=3306
socket=/tmp/mysql.sock
key_buffer_size=16M
max_allowed_packet=8M

 [mysqldump]
quick
```

Le fichier d'options ci−dessus utilise la syntaxe `var_name=value` pour les variables `key_buffer_size` et `max_allowed_packet` . Avant MySQL 4.0.2, vous auriez besoin d'utiliser la syntaxe `set-variable` à la place (comme présenté précédemment).

Voici un fichier d'options utilisateur classique :

```
 [client]
# Le mot de passe suivant sera envoyé par tous les clients standards MySQL
password="my_password"

 [mysql]
no-auto-rehash
set-variable = connect_timeout=2

 [mysqlhotcopy]
interactive-timeout
```

Ce fichier d'options utilise la syntaxe `set-variable` pour spécifier la variable `connect_timeout` . Depuis MySQL 4.0.2, vous pouvez aussi configurer la variable avec `connect_timeout=2` .

Si vous avez une distribution source, vous trouverez des exemples de configuration dans les fichiers nommés `my-xxxx.cnf` dans le dossier `support-files` . Si vous avez une distribution binaire, regardez dans le dossier `DIR/support-files` , où `DIR` est le chemin de l'installation MySQL (typiquement `/usr/local/mysql`). Actuellement, il y a des exemples de configuration pour des systèmes petits, moyens, grands et très grands. Vous pouvez copier l'un des fichiers `my-xxxx.cnf` dans votre dossier utilisateur (renommez le fichier en `.my.cnf`) pour le tester.

Note : sous Windows, le fichier d'options `.cnf` peut ne pas afficher son extension.

Tous les programmes MySQL qui supportent les fichiers d'options gèrent les options suivantes de ligne de commande :

`--no-defaults`
: N'utilise aucun fichier d'options.

`--print-defaults`
: Affiche le nom du programme et toutes les options qui seront lues dans les fichiers d'options.

`--defaults-file=path_name`
: Utilise uniquement le fichier d'options indiqué. `path_name` est le chemin complet pour y accéder.

`--defaults-extra-file=path_name`
: Lit ce fichier d'options après le fichier d'options globales, et avant le fichier d'options utilisateurs. `path_name` est le chemin complet pour y accéder.

Pour fonctionner correctement, toutes ces options doivent immédiatement suivre le nom de la commande en ligne, hormis `--print-defaults` qui peut être utilisée juste après `--defaults-file` et `--defaults-extra-file` .

Dans les scripts Shell, vous pouvez utiliser le programme `my_print_defaults` pour analyser les fichiers d'options. L'exemple suivant montre le résultat que `my_print_defaults` peut produire lorsqu'on lui demande d'afficher les options des groupes `[client]` et `[mysql]` :

```
shell> my_print_defaults client mysql
--port=3306
--socket=/tmp/mysql.sock
--no-auto-rehash
```

Note pour les développeurs : la gestion des fichiers est implémentée en C dans la bibliothèque cliente simplement en traitant les options qui sont trouvées, (c'est à dire, les options du groupe appropriée), et avant les options de ligne de commande. Cela fonctionne correctement avec les programmes qui utilisent la dernière option des options spécifiées plusieurs fois. Mais si vous avez un programme C ou C++ qui gère les options de cette manière mais ne lit pas les fichiers d'options, vous aurez peut être à ajouter seulement deux lignes pour lui donner cette fonctionnalité. Voyez le code source des clients MySQL standard pour voir comment faire.

De nombreux autres langages s'interfacent avec MySQL grâce à la bibliothèque C, et certains fournissent un moyen d'accéder aux fichiers d'options. Cela inclut les langages Perl et Python. Voyez la documentation de votre interface favorite pour plus de détails.

4.3.3 Utiliser les variables d'environnement pour spécifier des options

Pour spécifier des options en utilisant des variables d'environnement, utilisez la commande d'affectation de votre système. Par exemple, sous Windows ou sous NetWare, vous pouvez utiliser la variable `USER` pour spécifier votre compte utilisateur. Utilisez cette syntaxe :

```
SET USER=your_name
```

La syntaxe sous Unix dépend de votre Shell. Supposons que vous voulez spécifier le numéro de port TCP/IP en utilisant la variable `MYSQL_TCP_PORT` . La syntaxe Bourne Shell et ses variantes sont (`sh` , `bash` , `zsh` , etc.) est :

```
MYSQL_TCP_PORT=3306
```

Pour `csh` et `tcsh` , utilisez cette syntaxe :

```
setenv MYSQL_TCP_PORT 3306
```

Les commandes pour spécifier les variables d'environnement peuvent être exécutées à l'invite de commande, et prennent effet immédiatement. Ces configurations persistent jusqu'à votre déconnexion. Pour que ces configurations soient effectives lors de votre reconnexion, ajoutez les commandes appropriées dans votre fichier de démarrage. Typiquement, les fichiers de démarrage sont `AUTOEXEC.BAT` sous Windows, `.bash_profile` pour `bash` , ou `.tcshrc` pour `tcsh` . Consultez la documentation de votre interpréteur de ligne de commande pour les détails spécifiques.

La section Variables d'environnement MySQL liste toutes les variables d'environnement qui affectent le fonctionnement de MySQL.

4.3.4 Utiliser les options pour configurer des variables de programme

De nombreux programmes MySQL ont des variables internes, qui peuvent être modifiées durant l'exécution. Depuis MySQL version 4.0.2, les variables de programme peuvent être spécifiées de la même façon que toute autre option qui prend une vlaeur. Par exemple, le client `mysql` utilise la variable `max_allowed_packet` qui contrôle la taille maximale du buffer de communication. Pour spécifier la variable `max_allowed_packet` de `mysql` , à une valeur de 16Mo, utilisez l'une de ces deux commandes :

```
shell> mysql --max_allowed_packet=16777216
shell> mysql --max_allowed_packet=16M
```

La première commande spécifie la valeur en octets. La seconde spécifie la valeur en méga−octets. Les valeurs des variables peuvent être suffixées par `K` , `M` ou `G` (majuscule ou minuscule), pour indiquer une unité de kilo−octets, mega−octets ou giga−octets.

Dans un fichier d'options, la configuration de la variable se fait sans les tirets initiaux :

```
[mysql]
max_allowed_packet=16777216
```

Ou :

```
[mysql]
max_allowed_packet=16M
```

Si vous voulez, les soulignés d'un nom de variable peuvent être spécfiés comme des tirets.Avant MySQL 4.0.2, les noms de variables de programmes n'étaient pas reconnus comme des noms d'options. Au lieu de cela, il faut utiliser l'option `--set-variable` pour assigner une valeur :

```
shell> mysql --set-variable=max_allowed_packet=16777216
shell> mysql --set-variable=max_allowed_packet=16M
```

Dans un fichier d'options, omettez les tirets initiaux :

```
[mysql]
set-variable = max_allowed_packet=16777216
```

Ou :

```
[mysql]
set-variable = max_allowed_packet=16M
```

Avec `--set-variable` , les soulignés d'un nom de variable ne peuvent pas être spécifiés sous forme de tiret dans les versions de MySQL plus ancienne que la 4.0.2.L'option `--set-variable` est toujours reconnue par MySQL versions 4.0.2 et plus récent, mais elle est obsolète.

Certaines variables serveurs peuvent être configurées durant l'exécution. Voyez la section Variables dynamiques système .

5 Administration du serveur

Ce chapitre couvre les problèmes d'administration de MySQL, comme la configuration des comptes et les sauvegardes.

5.1 Scripts serveur MySQL et utilitaires

Le serveur MySQL, `mysqld` , est le programme principal qui effectue l'essentiel du travail. Le serveur est accompagné par différents scripts connexes qui effectuent des opérations de configuration lors de l'installation, ou fournissent de l'aide pour administrer le serveur.

Cette section fournit une introduction au serveur et ses utilitaires, et des informations sur les scripts de démarrage. Les informations sur la configuration du serveur lui−même sont données dans la section Configurer MySQL .

5.1.1 Présentation des scripts serveurs et des utilitaires

Tous les programmes MySQL prennent des options différentes. Toutefois, tous les programmes MySQL disposent de l'option `--help` qui vous aidera à connaître la liste complète des différentes options. Par exemple, essayez `mysql --help` .

Vous pouvez modifier toutes les valeurs par défaut des programmes en les plaçant dans le fichier de configuration. Utiliser les fichiers d'options .

Voici la liste des programmes coté serveur de MySQL :

`mysqld`
: Le démon SQL (c'est à dire, le serveur MySQL). Pour utiliser les programmes clients, ce programme doit fonctionner, car les clients viendront se connecter dessus. Configurer MySQL .

`mysqld-max`
: Une version du serveur qui inclut des fonctionnalités supplémentaires. `mysqld-max` , la version étendue du serveur `mysqld` .

`mysqld_safe`
: Un script de démarrage du serveur. `mysqld_safe` tente de démarrer `mysqld-max` s'il existe, et sinon `mysqld` . `safe_mysqld` , le script père de `mysqld` .

`mysql.server`
: Un script de démarrage du serveur. Ce script est utilisé sur les systèmes qui ont un dossier contenant des services système. Il invoque `mysqld_safe` pour démarrer le serveur MySQL. Le script de démarrage `mysql.server` .

`mysqld_multi`
: Un script de démarrage qui peut lancer ou arrêter différentes instances du serveur, installées sur le système. `mysqld_multi` , un programme pour gérer plusieurs serveurs MySQL .

`mysql_install_db`
: Crée les tables de droits MySQL, avec les droits par défaut. Il est généralement exécuté une fois, lors de la première installation de

`mysql_fix_privilege_tables`
: Ce script est utilisé après une mise à jour de MySQL, pour mettre à jour les tables de droits, et les adapter aux nouvelles versions de MySQL.

Il y a plusieurs autres utilitaires qui fonctionnent du coté du serveur :

`myisamchk`

Un utilitaire pour décrire, vérifier, optimiser et réparer les tables `MyISAM` . `myisamchk` est décrit dans Utilisation de `myisamchk` pour maintenir les tables et recouvrir les données .

`make_binary_distribution`
: Ce programme crée une version compilée de MySQL. Le résultat peut être envoyé par FTP à `/pub/mysql/Incoming` sur `support.mysql.com` pour aider les autres utilisateurs MySQL.

`mysqlbug`
: Le script de rapport de bogues de MySQL. Il peut être utilisé pour envoyer un rapport de bogues à MySQL. Vous pouvez aussi aller sur http://bugs.mysql.com/ pour remplir un formulaire en ligne.

5.1.2 mysqld−max , la version étendue du serveur mysqld

Le serveur `MySQL-Max` est une vesrion du serveur `mysqld` qui a été compilée avec des fonctionnalités supplémentaires.

La distribution à utiliser dépend de votre plate−forme :

- Sous Windows, les distributions binaires MySQL incluent le serveur standard (`mysqld.exe`) et le serveur `MySQL-Max` (`mysqld-max.exe`), alors il n'y a pas de téléchargement spécial à faire. Utilisez simplement la distribution Windows habituelle, disponible sur http://dev.mysql.com/downloads/mysql−4.0.html . Installer MySQL sous Windows .
- Sous Linux, si vous installez une distribution `RPM` , utilisez le `RPM MySQL-server` standard pour installer le serveur `mysqld` . Puis, utilisez le `RPM MySQL-Max` pour installer le serveur `mysqld-max` . Le `RPM MySQL-Max` suppose que vous avez déjà installé le serveur régulier `RPM` . Voyez Installer MySQL sous Linux pour plus d'informations sur les paquets Linux `RPM` .
- Toutes les autres distributiosn de MySQL−Max contiennent un serveur unique, appelé `mysqld` mais qui inclut les fonctionnalités supplémentaires.

Vous pouvez trouver les distributions binaires sur le site web de MySQL AB, sur http://dev.mysql.com/downloads/mysql−4.0.html .

MySQL AB compile le serveur MySQL−Max avec les options de `configure` suivantes :

−−with−server−suffix=−max
: Cette option ajoute le suffixe `-max` à la chaîne de version de `mysqld` .

−−with−innodb
: Cette option active le support du moteur `InnoDB` . MySQL−Max inclut toujours le support de `InnoDB` , mais cette option est nécessaire pour MySQL 3.23. Depuis MySQL 4, `InnoDB` est inclut par défaut dans les distributions binaires, alors il n'est pas nécessaire d'utiliser MySQL−Max pour ça.

−−with−bdb
: Cette option active le support du moteur de table Berkeley DB (`BDB`).

CFLAGS=−DUSE_SYMDIR
: Cette option active le support des liens symboliques sous Windows.

Les distributions binaires de MySQL−Max sont disponibles pour ceux qui souhaitent installer une version pré−compilée. Si vous voulez compiler MySQL−Max depuis les sources, vous pouvez le faire et choisir les fonctionnalités que vous souhaitez au moment de la compilation.

Le serveur MySQL−Max inclut le moteur de stockage BerkeleyDB (`BDB`) lorsque c'est possible, mais toutes les plate−formes ne supportent pas `BDB` . La table suivante montre quelles plate−formes permettent à MySQL−Max d'inclure `BDB` :

Système	`BDB` **Support**

AIX 4.3	N
HP−UX 11.0	N
Linux−Alpha	N
Linux−IA−64	N
Linux−Intel	Y
Mac OS X	N
NetWare	N
SCO OSR5	Y
Solaris−Intel	N
Solaris−SPARC	Y
UnixWare	Y
Windows/NT	Y

Pour connaître les moteurs de stockages que votre serveur supporte, utilisez la commande suivante :

```
mysql> SHOW ENGINES;
```

Avant MySQL 4.1.2, `SHOW ENGINES` est indisponible. Utilisez la commande suivante et vérifiez la valeur de la variable pour le moteur de table qui vous intéresse :

```
mysql> SHOW VARIABLES LIKE 'have_%';
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| have_bdb         | NO       |
| have_crypt       | YES      |
| have_innodb      | YES      |
| have_isam        | NO       |
| have_raid        | NO       |
| have_symlink     | DISABLED |
| have_openssl     | NO       |
| have_query_cache | YES      |
+------------------+----------+
```

La signification des valeurs est :

Valeur	Signification
YES	L'option est activatée et utilisable.
NO	L'option n'est pas supportée.
DISABLED	L'option est supportée mais désactivée.

La valeur `NO` signifie que le serveur a été compilé sans le support, et que la fonctionnalité ne peut pas être activée durant l'exécution.

La valeur de `DISABLED` apparait soit parce que le serveur a été lancé sans l'option qui active cette foncitonnalité, soit si toutes les options nécessaires ne sont pas dispoinbles. Dans ce dernier cas, le fichier d'erreurs `host_name.err` devrait contenir la raison indiquant pourquoi l'option a été désactivée.

Une situation dans laquelle vous pouvez voir `DISABLED` survient en version MySQL 3.23, lorsque le moteur `InnoDB` est compilé. En MySQL 3.23, vous devez fournir au moins l'option `innodb_data_file_path` à l'exécution pour configurer l'espace de tables `InnoDB` . Sans cette option, `InnoDB` se désactive. `InnoDB` en MySQL 3.23 . Vous pouvez spécifier les options de configuration opur les tables `BDB` , mais `BDB` ne se désactivera pas de lui−même si vous les oubliez. Options de démarrage `BDB` .

Vous pouvez aussi rencontrer la valeur de `DISABLED` pour `InnoDB` , `BDB` , ou `ISAM` si le serveur a été compilé pour les supporter, mais si les options de démarrage `--skip-innodb` , `--skip-bdb` ou `--skip-isam` à l'exécution.

Depuis la version 3.23, tous les serveurs MySQL supportent les tables `MyISAM` , car le moteur `MyISAM` est le moteur par défaut.

5.1.3 safe_mysqld , le script père de mysqld

`safe_mysqld` est la méthode recommandée pour démarrer un démon `mysqld` sous Unix. `safe_mysqld` ajoute des fonctionnalités de sécurité telles que le redémarrage automatique lorsqu'une erreur survient et l'enregistrement d'informations d'exécution dans un fichier de log.

Note: Avant MySQL 4.0, `mysqld_safe` s'appelait `safe_mysqld` . Pour préserver la compatibilité ascendante, la distribution binaire MySQL propose un lien symbolique de `safe_mysqld` vers `mysqld_safe` .

Par défaut, `mysqld_safe` essaie de lancer l'exécutable appelé `mysqld-max` s'il existe, ou `mysqld` sinon. Cela a des implications :

- Sous Linux, le `RPM MySQL-Max` dépend de `mysqld_safe` . Le `RPM` installe un exécutable appelé `mysqld-max` , qui fait que `mysqld_safe` va automatiquement utiliser l'exécutable installé
- Si vous installez la distribution `MySQL-Max` qui incluent un serveur appelé `mysqld-max` , puis que vous le mettez à jour avec une version non−max, `mysqld_safe` va essayer d'utiliser l'ancien serveur `mysqld-max` . Si vous faites une telle mise à jour, supprimez manuellement l'ancien serveur `mysqld-max` pour vous assurer que `mysqld_safe` utilise le nouveau `mysqld` .

Pour remplacer le comportement par défaut et spécifier explicitement le serveur que vous voulez utiliser, spécifiez l'option `--mysqld` ou `--mysqld-version` avec `mysqld_safe` .

De nombreux options de `mysqld_safe` sont identiques aux options de `mysqld` . Options de ligne de commande `mysqld` .

Toutes les options spécifiées avec `mysqld_safe` en ligne de commande sont passées à `mysqld` . Si vous voulez utiliser des options qui sont spécifiques à `mysqld_safe` et que `mysqld` ne les supporte pas, ne les spécifiez pas en ligne de commande. Au lieu de cela, listez les dans le groupe `[mysqld_safe]` du fichier d'options. Utiliser les fichiers d'options .

`mysqld_safe` lit toutes les options des groupes `[mysqld]` , `[server]` et `[mysqld_safe]` dans le fichier d'options. Pour assurer la compatibilité ascendante, il lit aussi le groupe `[safe_mysqld]` . Vous devriez renommer ces sections `[mysqld_safe]` lorsque vous passez à MySQL 4.0 ou plus récent.

`safe_mysqld` supporte les options suivantes :

`--basedir=path`
: Le chemin jusqu'à l'installation de MySQL.

`--core-file-size=#`
: Taille du fichier `core` que `mysqld` doit être capable de créer. Il est passé à `ulimit -c` .

`--datadir=path`
: Le chemin jusqu'au dossier de données.

`--defaults-extra-file=path`
: Le nom du fichier d'options à lire en plus des fichiers habituels.

`--defaults-file=path`
: Le nom d'un fichier d'options qui doit être lu à la place du fichier d'options habituel.

`--err-log=path`
: L'ancienne option `--log-error` , à utiliser avant MySQL 4.0.

`--ledir=path`
: Le chemin jusqu'au dossier contenant le dossier `mysqld` . Utilisez cette option pour indiquer explicitement le lieu du serveur.

`--log-error=path`
: Ecrit le fichier d'erreurs dans le fichier ci−dessus. Le log d'erreurs .

`--mysqld=prog_name`
: Le nom du programme serveur (dans le dossier `ledir`) que vous voulez lancer. Cette option est nécessaire si vous utilisez une distribution binaire MySQL, mais que les données sont hors du dossier d'installation.

`--mysqld-version=suffix`
: Cette option est similaire à l'option `--mysqld` , mais vous spécifiez uniquement le suffixe du nom du programme. Le nom de base sera alors `mysqld` . Par exemple, si vous utilisez `--mysqld-version=max` , `mysqld_safe` va lancer le programme `mysqld-max` dans le dossier `ledir` . Si l'argument de `--mysqld-version` est vide, `mysqld_safe` utilise `mysqld` dans le dossier `ledir` .

`--nice=priority`

Utilise le programme `nice` pour donner la priorité du serveur. Cette option a été ajoutée en MySQL 4.0.14.

`--no-defaults`
: Ne lit aucun fichier d'options.

`--open-files-limit=count`
: Le nombre de fichiers que `mysqld` ouvre au maximum. La valeur de l'option est passée à `ulimit -n` . Notez que vous devez lancer `mysqld_safe` en tant que `root` pour que cela fonctionne correctement.

`--pid-file=path`
: Le chemin jusqu'au fichier d'identifiant de processus.

`--port=port_num`
: Le numéro de port à utiliser pour attendre les connexion TCP/IP.

`--socket=path`
: Le fichier de socket Unix pour les connexions locales.

`--timezone=zone`
: Configure la variable d'environnement `TZ` . Consultez votre documentation système pour connaître le format légal des fuseaux horaires.

`--user={user_name | user_id}`
: Lance le serveur `mysqld` sous le nom d'utilisateur `user_name` ou avec l'utilisateur d'identifiant numérique ID `user_id` . (``Utilisateur'' dans ce contexte représente le compte système, et non pas les utilisateurs des tables de droits MySQL).

Le script `safe_mysqld` a été écrit pour qu'il soit capable de démarrer le serveur qui a été installé à partir des sources ou de la version binaire, même si l'installation de MySQL est légèrement exotique. Plans d'installation . `safe_mysqld` suppose que les conditions suivantes sont remplies :

- Le serveur et les bases de données sont placées dans un dossier relativement au dossier d'où `safe_mysqld` est appelé. `safe_mysqld` cherche dans les sous dossiers `bin` et `data` (pour les distributions binaires) et, `libexec` et `var` (pour les distributions sources). Cette condition doit être remplie si vous exécutez `safe_mysqld` depuis votre dossier d'installation MySQL (par exemple, `/usr/local/mysql` pour une distribution binaire).
- Si le serveur et les bases de données ne peuvent être trouvées dans le dossier de travail, `safe_mysqld` essaie de les trouver en utilisant leurs chemins absolus. Les chemin typiquement étudiés sont `/usr/local/libexec` et `/usr/local/var` . Les chemins réels sont déterminés lorsque la distribution est compilée, et `safe_mysqld` a alors aussi été généré. Ils doivent être corrects si MySQL a été installé dans un dossier standard.

Comme `safe_mysqld` essaie de trouver le serveur et les bases dans un dossier situé dans le dossier de travail, vous pouvez installer la version binaire de MySQL n'importe où, du moment que vous démarrez le script `safe_mysqld` dans le dossier d'installation de MySQL :

```
 shell> cd mysql_installation_directory
shell> bin/safe_mysqld &
```

Si `safe_mysqld` échoue, même si il est appelé depuis le dossier d'installation, vous pouvez le modifier pour qu'il reconnaisse le chemin que vous utilisez jusqu'à `mysqld` . Notez que si vous faites évoluer votre installation de MySQL, votre version de `safe_mysqld` sera écrasée, et vous devrez la rééditer.

Normalement, vous ne devez pas éditer le script `mysqld_safe` . Au lieu de cela, configurez `mysqld_safe` en utilisant les options de ligne de commande, ou les options de la section `[mysqld_safe]` du fichier d'options `my.cnf` . Dans de rares cas, il sera peut être nécessaire d'éditer `mysqld_safe` pour faire fonctionner correctement le serveur. Cependant, si vous faites cela, `mysqld_safe` risque d'être écrasé lors de la prochaine mise à jour de MySQL : faîtes en une sauvegarde avant d'installer.

Sous NetWare, `mysqld_safe` est un `NetWare Loadable Module (NLM)` qui est un port du script Unix original. Il fait ceci :

- Effectue des vérifications système et des options.

- Lance la vérification des tables `MyISAM` et `ISAM` .
- Affiche un écran de présence de MySQL.
- Lance `mysqld` , le surveille et le relance s'il s'arrête sur une erreur.
- Envoie les messages de `mysqld` dans le fichier `host_name.err` dans le dossier de données.
- Envoie les affichages de `mysqld_safe` dans le fichier `host_name.safe` dans le dossier de données.

5.1.4 Le script de démarrage mysql.server

Les distributions MySQL sous Unix incluent un script appelé `mysql.server` . Il peut être utilisé pour lancer et arrêter les services. Il est aussi utilisé par le script de démarrage de Max OS X.

`mysql.server` est placé dans le dossier `support-files` sous le dossier d'installation de MYSQL, ou dans le dossier de source.

Si vous utilisez le paquet `RPM` pour Linux (`MySQL-server-VERSION.rpm`), le script `mysql.server` sera déjà installé dans le dossier `/etc/init.d` sous le nom de `mysql` . Vous n'avez pas besoin de l'installer manuellement. Voyez Installer MySQL sous Linux pour plus d'informations sur les paquets MySQL `RPM` Linux.

Certains éditeurs fournissent des paquets `RPM` qui installent un script de démarrage sous le nom de `mysqld` .

Si vous installez MySQL depuis une distribution source, ou en utilisant une distribution binaire qui n'installe pas automatiquement le fichier `mysql.server` , vous pouvez l'installer manuellement. Les instructions sont fournies dans la section Lancer et arrêter automatiquement MYSQL .

`mysql.server` lit les options dans les sections `[mysql.server]` et `[mysqld]` du fichier de configuration. Pour la compatibilité ascendante, il lit aussi la section `[mysql_server]` , même si vous devrez la renommer en `[mysql.server]` lorsque vous commencerez à utiliser MySQL 4.0.

5.1.5 mysqld_multi , un programme pour gérer plusieurs serveurs MySQL

`mysqld_multi` sert à gérer plusieurs serveurs `mysqld` qui utilisent différentes sockets Unix et ports TCP/IP.

Le programme va rechercher les groupes nommés `[mysqld#]` dans le fichier `my.cnf` (ou le fichier appelé `--config-file=...`), où `#` peut être n'importe quel nombre positif, supérieur ou égal à 1. Ce nombre est appelé le numéro de groupe d'options. Les numéros de groupe permettent de distinguer un groupe d'options d'un autre, et sont utilisés comme argument du script `mysqld_multi` pour spécifier quel serveur vous voulez démarrer, arrêter ou examiner. Les options listées dans ces groupes doivent être les mêmes que celle que vous utiliseriez dans une section dédiée au démon `[mysqld]` . Voyez, par exemple, Lancer et arrêter automatiquement MYSQL . Cependant, pour `mysqld_multi` , vous devez vous assurer que chaque groupe contient des valeurs pour les options telles que port, socket, etc., qui seront utilisées par chaque processus `mysqld` . Pour plus d'informations sur sur les options de chaque serveur dans un environnement à serveurs multiples, voyez la section Utiliser plusieurs serveurs sur la même machine .

`mysqld_multi` est utilisé ave la syntaxe suivante :

```
shell> mysqld_multi [options] &#123;start|stop|report&#125; [GNR[,GNR]...]
```

`start` , `stop` et `report` indique le type d'opération que vous voulez faire. Vous pouvez faire une opération sur un serveur unique ou plusieurs serveurs, en fonction de la liste `GNR` qui suit le nom de l'opération. S'il n'y a pas de liste, `mysqld_multi` effectue l'opération sur tous les serveurs du fichier d'options.Chaque

`GNR` représente un numéro de groupe d'options. Vous pouvez démarrer, arrêter ou examiner n'importe quel numéro de groupe d'options, ou même plusieurs d'entre eux en même temps. Par exemple, le groupe `GNR` pour le groupe appelé `[mysqld17]` est `17` . Pour spécifier un intervalle de nombres, séparez le premier et le dernier numéro par un tiret. La valeur `GNR 10-13` représente les groupes de `[mysqld10]` à `[mysqld13]` . Les groupes multiples ou les intervalles de groupes peuvent être spécifiées en ligne de commande, séparés par virgules. Il ne doit pas y avoir d'espace blanc entre deux éléments de la liste : tout ce qui sera après un espace sera ignoré.

Cette commande lance un serveur unique, avec le groupe d'options `[mysqld17]` :

```
shell> mysqld_multi start 17
```

Cette commande arrête plusieurs serveurs, en utilisant les groupes d'options `[mysql8]` et `[mysqld10]` à `[mysqld13]` :

```
shell> mysqld_multi start 8,10-13
```

Pour afficher un exemple de configurations, utilisez cette commande :

```
shell> mysqld_multi --example
```

Les valeurs de numéro de groupe d'options peuvent être une liste de valeurs séparées par une virgule ou un tiret. Dans ce dernier cas, toutes les numéro de groupe d'options situés entre les deux numéros seront alors affectés. Sans numéro de groupe d'options spécifié, tous les numéros de groupes du fichier d'options sont affectés. Notez que vous ne devez pas avoir d'espace dans la liste des numéros de groupe d'options. Tout ce qui est placé au−delà de l'espace sera ignoré.

`mysqld_multi` supporte les options suivantes :

−−config−file=name
: Un fichier de configuration alternatif. Note : cela ne va pas modifier les options de ce programme (`[mysqld_multi]`), mais uniquement les groupes `[mysqld#]` . Sans cette option, tout sera lu dans le fichier d'options traditionnel `my.cnf` . Cette option n'affecte pas la façon avec laquelle `mysqld_multi` lit ses options, qui sont toujours prises dans le groupe `[mysqld_multi]` du fichier `my.cnf` habituel.

−−example
: Affiche un exemple de fichier de configuration.

−−help
: Affiche l'aide et quitte.

−−log=name
: Fichier de log. Le chemin complet et le nom du fichier sont nécessaires.

−−mysqladmin=prog_name
: L'exécutable `mysqladmin` à utiliser lors de l'arrêt du serveur.

−−mysqld=prog_name
: L'exécutable `mysqld` à utiliser. Notez que vous pouvez donner cette option à `safe_mysqld` . Ces options sont passées à `mysqld` . Assurez−vous que vous avez bien `mysqld` dans votre variable d'environnement `PATH` ou corrigez `safe_mysqld` .

−−no−log
: Affiche les données d'historique à l'écran plutôt que dans le fichier de log. Par défaut, le fichier de log est activé.

−−password=password
: Le mot de passe de l'utilisateur `mysqladmin` .

−−tcp−ip
: Connexion au serveur MySQL via le port TCP/IP au lieu de la socket Unix. Cela affecte l'arrêt et le rapport. Si le fichier de socket manque, le serveur peut continuer de tourner, mais il n'est plus accessible que par port TCP/IP. Par défaut, les connexions sont faites avec les sockets Unix.

−−user=user_name

L'utilisateur MySQL pour `mysqladmin` .

−−version
: Affiche le numéro de version et quitte.

Quelques notes pour `mysqld_multi` :

- Assurez−vous que l'utilisateur MySQL, qui stoppe les services `mysqld` (e.g en utilisant la commande `mysqladmin`), a les mêmes nom d'utilisateur et mot de passe pour tous les dossiers de données utilisés. Et assurez−vous que cet utilisateur a bien les droits de `SHUTDOWN` ! Si vous avez de nombreux dossiers de données et de nombreuses bases `mysql` avec différents mots de passe pour le serveur `root` MySQL, vous souhaiterez peut être créer un utilisateur commun `multi_admin` à chaque base, avec le même mot de passe (voir ci−dessous). Voici comment faire :
```
 shell> mysql -u root -S /tmp/mysql.sock -proot_password -e
"GRANT SHUTDOWN ON *.* TO multi_admin&#64;localhost IDENTIFIED BY 'multipass'"
```
  Comment le système de droits fonctionne . Vous devrez utiliser la même commande pour chaque serveur `mysqld` qui fonctionne : changez simplement la socket, `-S=...`).
- `pid-file` est très important, si vous utilisez `safe_mysqld` pour démarrer `mysqld` (e.g., `--mysqld=safe_mysqld`). Chaque `mysqld` doit avoir son propre fichier `pid-file` . L'avantage d'utiliser `safe_mysqld` au lieu de `mysqld` est que `safe_mysqld` ``surveille'' tous les processus `mysqld` et les redémarrera si un processus `mysqld` s'arrête suite à la reception d'un signal `kill -9` , ou pour toute autre raison comme une erreur de segmentation (que MySQL ne devrait jamais faire, bien sûr !). Notez bien que le script `safe_mysqld` vous imposera peut être d'être démarré depuis un dossier spécial. Cela signifie que vous devrez probablement utiliser la commande shell `cd` jusqu'à un certain dossier avant de pouvoir exécuter `mysqld_multi` . Si vous avez des problèmes pour démarrer, voyez le script `safe_mysqld` . Vérifiez notamment ces lignes :
```
 --------------------------------------------------------------
MY_PWD=`pwd`
# Check if we are starting this relative (for the binary release)
if test -d $MY_PWD/data/mysql -a -f ./share/mysql/english/errmsg.sys -a \
 -x ./bin/mysqld
--------------------------------------------------------------
```
  `safe_mysqld` , le script père de `mysqld` . Le test ci−dessus devrait fonctionner, ou bien vous rencontrerez probablement des problèmes.
- Le fichier de socket et le port TCP/IP doivent être différents pour chaque `mysqld` .
- Vous pouvez utiliser l'option `--user` de `mysqld` , mais afin de faire cela, vous devez exécuter le script `mysqld_multi` en tant que `root` Unix. Placer cette option dans le fichier de configuration ne changera rien : vous obtiendrez une alerte, si vous n'êtes pas le super utilisateur, et les démons `mysqld` seront démarrés avec vos droits Unix.
- **Important** : assurez−vous bien que le fichier de données et le fichier de `pid-file` sont accessibles en lecture et écriture (et exécution pour le dernier) à l'utilisateur Unix qui lance les processus `mysqld` . **N'utilisez pas** le compte root Unix pour cela, à moins que vous ne **sachiez** ce que vous faîtes.
- **Très important** : assurez−vous de bien comprendre la signification des options que vous passez à `mysqld` s et **pourquoi** vous avez besoin de plusieurs processus `mysqld` . Méfiez vous des pièges des serveurs multiples `mysqld` dans le même dossier de données. Utilisez des dossiers de données à moins que vous ne **sachiers** ce que vous faîtes. Démarrer plusieurs serveurs `mysqld` s dans le même dossier **ne vous donnera aucun** gain de performance dans un système threadé. Utiliser plusieurs serveurs sur la même machine .

Voici un exemple de fichier de configuration fourni par `mysqld_multi` .

```
 # This file should probably be in your home dir (~/.my.cnf) or /etc/my.cnf
# Version 2.1 by Jani Tolonen

 [mysqld_multi]
mysqld     = /usr/local/bin/safe_mysqld
mysqladmin = /usr/local/bin/mysqladmin
user       = multi_admin
password   = multipass
```

```
 [mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid-file   = /usr/local/mysql/var2/hostname.pid2
datadir    = /usr/local/mysql/var2
language   = /usr/local/share/mysql/english
user       = john

 [mysqld3]
socket     = /tmp/mysql.sock3
port       = 3308
pid-file   = /usr/local/mysql/var3/hostname.pid3
datadir    = /usr/local/mysql/var3
language   = /usr/local/share/mysql/swedish
user       = monty

 [mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid-file   = /usr/local/mysql/var4/hostname.pid4
datadir    = /usr/local/mysql/var4
language   = /usr/local/share/mysql/estonia
user       = tonu

 [mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid-file   = /usr/local/mysql/var6/hostname.pid6
datadir    = /usr/local/mysql/var6
language   = /usr/local/share/mysql/japanese
user       = jani
```

Utiliser les fichiers d'options .

5.2 Configuration de MySQL

Cette section présente la configuration du serveur MySQL :

- Options de démarrage que le serveur supporte
- Comment utiliser le mode SQL du serveur
- Les variables serveurs
- Les variables de statut du serveur

5.2.1 Options de ligne de commande de mysqld

Lorsque vous démarrez le serveur `mysqld` , vous pouvez spécifier les options de programme en utilisant les méthodes décrites dans la section Options spécifiques des programmes . Les méthodes les plus courantes sont celles du fichier d'options et celle de la ligne de commande. Cependant, dans la plupart des cas, vous devrez vous assurer que le serveur utilise toujours les mêmes options à chaque redémarrage : il est recommandé de les mettre dans le fichier d'options. Utiliser les fichiers d'options .

`mysqld` et `mysqld.server` lisent les options des groupes `mysqld` et `server` . `mysqld_safe` lit les options des groupes `mysqld` , `server` , `mysqld_safe` et `safe_mysqld` . Un serveur MySQL intégré lit généralement les options dans les groupes `server` , `embedded` et `xxxxx_SERVER` , où `xxxxx` est le nom de l'application.

`mysqld` accepte de nombreuses options de ligne de commande. Pour une liste complète, utilisez la commande `mysqld --help` . Avant MySQL 4.1.1, `--help` affiche un message d'aide complet. Depuis 4.1.1, il affiche un bref message. Pour voir la liste complète, utilisez `mysqld --verbose --help` .

La liste suivante montre les options de serveur les plus courantes. Les options supplémentaires sont détaillées ailleurs.

- Options qui affectent la sécurité : voir Options de démarrage de `mysqld` qui concernent la sécurité .
- Options liées à SSL : voir Options de ligne de commande pour SSL .
- Options de contrôle des logs binaires : voir Le log binaire des mises à jour .
- Options de réplication : voir Options de réplication .
- Options spécifiques à un moteur de table particulier : voir Options de démarrage `MyISAM` , Options de démarrage `BDB` , Options de démarrage `InnoDB` .

Vous pouvez aussi modifier les valeurs des variables système en utilisant le nom de l'option comme variable système, tel que décrit ultérieurement.

`--help` , `-?`
: Affiche l'aide courte et termine le programme. Avant MySQL 4.1.1, `--help` affiche un message d'aide complet. Depuis 4.1.1, il affiche un bref message. Pour voir la liste complète, utilisez `mysqld --verbose --help` .

`--ansi`
: Utilise la syntaxe ANSI SQL au lieu de la syntaxe MySQL. Exécuter MySQL en mode ANSI . Pour un contrôle plus précis sur le mode SQL serveur, utilisez l'option `--sql-mode` .

`--basedir=path` , `-b path`
: Chemin jusqu'au dossier d'installation. Tous les chemins sont généralement relatifs à celui−ci.

`--big-tables`
: Autorise la sauvegarde de grands résultats dans des fichiers temporaires. Cela résout le problème des erreurs " `table full` ", mais ralentit les requêtes alors que des tables en mémoire suffirait. Depuis la version version 3.23.2, MySQL est capable de résoudre automatiquement ce problème en utilisant de la mémoire pour toutes les tables temporaires de petite taille, et en passant sur le disque au besoin.

`--bind-address=IP`
: L'adresse IP à utiliser.

`--console`
: Ecrit les messages d'erreurs du log d'erreur sur la sortie standard, même si `--log-error` est spécifiée. Sous Windows, `mysqld` ne ferme pas l'écran de console si cette option est utilisée.

`--character-sets-dir=path`
: Dossier contenant les jeux de caractères. Le jeu de caractères utilisé pour les données et le stockage .

`--chroot=path`
: Met le démon `mysqld` en environnement chroot au démarrage. Recommandé pour la sécurité depuis MySQL 4.0. (MySQL 3.23 n'est pas capable de fournir un encadrement `chroot()` qui soit 100% étanche). Cela limite les commandes `LOAD DATA INFILE` et `SELECT ... INTO OUTFILE` .

`--core-file`
: Ecrire le fichier core lorsque `mysqld` s'arrête inopinément. Pour certains fichiers, vous devez aussi spécifier `--core-file-size` à `safe_mysqld` . `safe_mysqld` , le script père de `mysqld` . Notez que sur certains systèmes, comme Solaris, vous n'aurez pas de fichier de core si vous avez aussi utilisé l'option `--user` .

`--datadir=path` , `-h path`
: Chemin jusqu'au dossier de données.

`--debug[=debug_options]` , `-# [debug_options]`
: Si MySQL est configuré avec `--with-debug` , vous pouvez utiliser cette option pour obtenir un fichier de trace de ce que `mysqld` fait. Créer des fichiers de traçage .

`--default-character-set=charset`
: Spécifie le jeu de caractères par défaut. Le jeu de caractères utilisé pour les données et le stockage .

`--default-collation=collation`
: Spécifie la `collation` par défaut. Cette option est disponible depuis MySQL 4.1.1. Le jeu de caractères utilisé pour les données et le stockage .

`--default-storage-engine=type`

Cette option est un synonyme de `--default-table-type` . Cette option est disponible depuis MySQL 4.1.2.

`--default-table-type=type`

Spécifie le type de table par défaut. Types de tables MySQL .

`--delay-key-write[= OFF | ON | ALL]`

Comment l'option `DELAYED KEYS` doit être utilisée. Les écritures différées font que les buffers de clés ne sont pas écrits entre deux écriture pour les tables `MyISAM` . `OFF` désactive les écritures différées. `ON` active les écritures différées pour les tables qui ont été créées avec l'option `DELAYED KEYS` . `ALL` active les écritures différées pour les tables `MyISAM` . Cette option est disponible depuis MySQL 4.0.3. Choix des paramètres du serveur . Options de démarrage `MyISAM` .

Note : Si vous avez donné la valeur de `ALL` à cette option, vous ne devez pas utiliser les tables `MyISAM` depuis un autre programme (comme `myisamchk`) lorsque la tables utilisée. En faisant cela, vous obtiendrez une corruption d'index.

`--delay-key-write-for-all-tables`

Ancienne forme de `--delay-key-write=ALL` à utiliser sur les versions antérieures à la version 4.0.3. Depuis la version 4.0.3, utilisez `--delay-key-write` .

`--des-key-file=file_name`

Lit les clés par défaut utilisées par `DES_ENCRYPT()` et `DES_DECRYPT()` dans ce fichier.

`--enable-named-pipe`

Active le support des pipes nommés (seulement sur NT/Win2000/XP). Cette option ne s'applique qu'aux systèmes Windows NT, 2000 et XP, et peut être utilisé avec les serveurs `mysqld-nt` et `mysqld-max-nt` qui supportent les pipes nommés.

`--exit-info` , `-T`

Cette option est la combinaison d'options que vous pouvez utiliser pour le débogage du serveur `mysqld` . Si vous ne savez pas ce que ca fait exactement, ne l'utilisez pas !

`--external-locking`

Active le verrouillage système. Notez que si vous utilisez cette option sur un système pour qui `lockd` ne fonctionne pas (comme Linux), vous allez bloquer rapidement `mysqld` avec les verrous. Anciennement appelée `--enable-locking` .

Note : si vous utilisez cette option pour activer des modifications de tables `MyISAM` depuis plusieurs processus MySQL, vous devez vous assurer de trois conditions :

- Vous n'utilisez pas de cache de requête pour les requêtes qui utilisent les tables sont modifiées par un autre processus.
- Vous ne devez pas utiliser `--delay-key-write=ALL` ou `DELAY_KEY_WRITE=1` sur des tables partagées.

Le plus simple pour s'assurer de cela est de toujours utiliser l'option `--external-locking` avec `--delay-key-write=OFF --query-cache-size=0` .

(Ceci n'est pas fait par défaut, car dans de nombreuses configurations, il est pratique de faire un mélange des options ci−dessus).

`--flush`

Ecrit toutes les données sur le disque après chaque requête SQL. Normalement, MySQL fait des écritures sur le disque après chaque requête, et laisse le système d'exploitation assurer la synchronisation avec le disque. Que faire si MySQL crashe constamment .

`--init-file=file`

Lit les commandes SQL dans ce fichier au démarrage. Chaque commande doit être sur une ligne, et ne pas utiliser de commentaires.

`--language=lang_name` , `-L lang_name`

Spécifie la langue utilisée pour les messages d'erreur du client. Le chemin complet doit être utilisé. Messages d'erreurs non−anglophones .

`--log[=file]` , `-l [file]`

Enregistre les connexions et les requêtes dans ce fichier. Le log général de requêtes . Si vous ne le faites pas, `MySQL` va utiliser `host_name.log` comme nom de fichier.

`--log-bin=[file]`

Enregistre toutes les requêtes qui modifient des données dans un log. Le log binaire des mises à jour . Ce log est utilisé pour la sauvegarde et la réplication. Si vous ne le faites pas, `MySQL` va utiliser `host_name-bin` comme nom de fichier.

`--log-bin-index[=file]`
: Fichier d'index pour les noms de fichiers de log binaire. Le log binaire des mises à jour . Si vous ne le faites pas, `MySQL` va utiliser `host_name-bin.index` comme nom de fichier.

`--log-error[=file]`
: Enregistre les messages d'erreurs et les messages de démarrage dans ce fichier. Le log d'erreurs . Si vous ne le faites pas, `MySQL` va utiliser `host_name.err` comme nom de fichier.

`--log-isam[=file]`
: Enregistre toutes les modifications des tables `ISAM` / `MyISAM` dans ce fichier (uniquement nécessaire pour déboguer `ISAM` / `MyISAM`).

`--log-long-format`
: Enregistre des informations supplémentaires dans les fichiers de log (log de modifications, log binaire de modifications, log de requêtes lentes, n'importe quel log en fait). Par exemple, le nom d'utilisateur et un timestamp sont enregistrés avec la requête. Si vous utilisez `--log-slow-queries` et `--log-long-format` , alors les requêtes qui n'utilisent pas d'index seront aussi enregistrées. Notez que `--log-long-format` est obsolète depuis la version 4.1, où `--log-short-format` a été introduite (le format de log long est la configuration par défaut en version 4.1). Notez aussi que depuis la version MySQL 4.1 l'option `--log-queries-not-using-indexes` est disponible pour enregistre spécifiquement les requête qui n'utilisent pas d'index, dans le log de requêtes lentes.

`--log-queries-not-using-indexes`
: Si vous utilisez cette option avec `--log-slow-queries` , alors les requêtes qui n'utilisent pas d'index seront aussi enregistrées dans le log de requêtes lentes. Cette option est disponible depuis MySQL 4.1. Le log des requêtes lentes .

`--log-short-format`
: Enregistre moins d'information dans les fichiers de log (log de modifications, log binaire de modifications, log de requêtes lentes, n'importe quel log en fait). Par exemple, les noms d'utilisateur et un timestamp ne seront pas enregistrés avec les requêtes. Cette option a été ajoutée en MySQL 4.1.

`--log-slow-queries[=file]`
: Enregistre toutes les requêtes qui prennent plus de `long_query_time` secondes a s'exécuter. Notez que la quantité d'information enregistrée par défaut a changé en MySQL 4.1. Voyez les options `--log-long-format` et `--log-long-format` pour plus de détails. Le log des requêtes lentes .

`--log-update[=file]`
: Enregistre les modifications de données dans le fichier `file.#` où `#` est un nombre unique, s'il n'est pas fourni. Le log des mises à jour . Le log de modification est obsolète et supprimé en MySQL 5.0.0; vous devriez utiliser le log binaire à la place. (`--log-bin`). Le log binaire des mises à jour . Depuis la version 5.0.0, utilisez `--log-update` va simplement activer le log binaire. (Changements de la version5.0.0 (22 Dec 2003: Alpha)).

`--log-warnings` , `-W`
: Affiche les alertes comme `Aborted connection...` dans le fichier d'erreur `.err` . Activer cette option est recommandé, par exemple, si vous utilisez la réplication : vous obtiendrez plus d'informations sur ce qui se passe, comme les erreurs de connexion réseau, ou les reconnexions. Cette option est activée par défaut depuis MySQL 4.1.2; pour la désactiver, utilisez `--skip-log-warnings` . Erreurs de communication / connexion annulée .

  Cette option s'appelait `--warnings` avant MySQL 4.0.

`--low-priority-updates`
: Les opérations de modifications de table (`INSERT` / `DELETE` / `UPDATE`) auront une priorité inférieure aux sélections. Cela peut être aussi fait via l'attribut `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` pour baisser la priorité d'une requête, ou avec `SET LOW_PRIORITY_UPDATES=1` pour changer la priorité dans plus d'un thread. Problèmes avec le verrouillage de tables .

`--memlock`

Verrouille le processus `mysqld` en mémoire. Cela fonctionne si votre système support la fonction `mlockall()` (comme Solaris). Ceci peut être utile si vous avez des problèmes avec le système d'exploitation qui force `mysqld` à utiliser le swap sur le disque.

`--myisam-recover [=option[,option...]]]`

Cette option est la combinaison de `DEFAULT` , `BACKUP` , `FORCE` et `QUICK` . Vous pouvez aussi lui donner la valeur explicite de `""` si vous voulez désactiver cette option. Si cette option est utilisée, `mysqld` va vérifier si la table est marquée comme corrompue à l'ouverture de chaque table (cette dernière option ne fonctionne que si vous utilisez l'option `--skip-external-locking`). Si c'est le cas, `mysqld` va essayer de vérifier la table. Si la table était corrompue, `mysqld` essaie alors de la réparer.L'option suivante va affecter la manière avec la quelle la réparation s'effectue.

Option	Description
DEFAULT	Identique à ne pas donner d'option à --myisam-recover .
BACKUP	Si la table a été modifiée durant la réparation, sauver une copie du fichier table_name.MYD , sous le nom de table_name-datetime.BAK .
FORCE	Exécute une réparation même si nous allons perdre une ou plusieurs lignes dans le fichier .MYD .
QUICK	Ne vérifie pas les lignes dans la table si il n'y a pas eu d'effacement.

Avant que la table ne soit automatiquement réparée, MySQL va ajouter une note dans le fichier de log d'erreurs. Si vous voulez être capable de restaurer la plupart des erreurs sans intervention de l'utilisateur, il vaut utiliser les options `BACKUP,FORCE` . Cela va forcer la réparation de la table, même si quelques lignes sont effacées, et conserve le vieux fichier de données comme sauvegarde, pour examen ultérieur.

Cette option est disponible depuis MySQL 3.23.25.

`--new`

Depuis la version 4.0.12, l'option `--new` sert à dire au serveur d'adopter le comportement de la version 4.1 pour certains aspects, afin de simplifier la migration de 4.0 en 4.1 :

- `TIMESTAMP` est retourné sous forme de chaîne avec le format `'YYYY-MM-DD HH:MM:SS'` . Types de colonnes .

Cette option peut vous aider à voir comment vos applications vont se comporter en MySQL 4.1, sans réellement changer de version.

`--pid-file=path`

Le chemin jusqu'au fichier de `PID` utilisé par `safe_mysqld` .

`--port=port_num` , `-P port_num`

Numéro de port utilisé pour attendre les connexion TCP/IP.

`--old-protocol` , `-o`

Utilise le protocole 3.20, pour la compatibilité avec de très vieux clients. Mise à jour de Version 3.20 vers 3.21 .

`--one-thread`

Utilise uniquement un thread (pour débogage sous Linux). Cette option est disponible uniquement si le serveur est compilé avec les options de débogage. Déboguer un serveur MySQL .

`--open-files-limit=count`

Pour changer le nombre de pointeurs de fichiers disponibles pour `mysqld` . Si cette option n'est pas configurée, ou qu'elle vaut 0, alors `mysqld` va utiliser cette valeur pour réserver ce nombre de pointeurs de fichiers, à utiliser avec `setrlimit()` . Si la valeur est 0 alors `mysqld` va réserver `max_connections*5` ou `max_connections + table_cache*2` (le plus grand des deux) pointeurs de fichiers. Il est recommandé d'augmenter cette valeur si `mysqld` émet des erreurs de type `'Too many open files'` .

`--safe-mode`

Ignore certains étapes d'optimisation.

`--safe-show-database`

Avec cette option, la commande `SHOW DATABASES` retourne uniquement les bases pour lesquelles l'utilisateur a des droits. Depuis la version 4.0.2, cette option est abandonnée, et ne fait plus rien (l'option est activée par défaut) car nous avons désormais le droit de `SHOW DATABASES` . Syntaxe de `GRANT` et `REVOKE` .

`--safe-user-create`

Si cette option est activée, un utilisateur ne peut pas créer de nouveaux utilisateurs avec la commande GRANT si l'utilisateur n'a pas les droits de `INSERT` dans la table mysql.user ou dans aucune colonne de cette table.

--secure-auth
: Interdit l'identification des comptes qui ont des mots de passe ancien (avant la version 4.1). Cette option est disponible depuis MySQL 4.1.1.

--skip-bdb
: Désactive l'utilisation des tables BDB . Cela va économiser de la mémoire et accélérer le serveur un peu. N'utilisez pas cette option si vous avez besoin des tables BDB .

--skip-concurrent-insert
: Désactive la possibilité de sélectionner et insérer en même temps dans les tables MyISAM (cela n'est utile que si vous pensez que vous avez trouvé un bug dans cette fonctionnalité).

--skip-delay-key-write
: Ignore l'option DELAY_KEY_WRITE de toutes les tables. Depuis MySQL 4.0.3, vous devez utiliser --delay-key-write=OFF à la place. Choix des paramètres du serveur .

--skip-external-locking
: Ne pas utiliser le verrouillage du système. Pour utiliser les utilitaires isamchk ou myisamchk vous devez alors éteindre le système. Degré de stabilité de MySQL . Notez qu'en MySQL version 3.23 vous pouvez utiliser la commande REPAIR et CHECK pour réparer ou vérifier des tables MyISAM tables. Cette option s'appelait auparavant --skip-locking .

--skip-grant-tables
: Cette option force le serveur à ne pas utiliser le système de privilège du tout. Cela donne à tous l' **accès complet** à toutes les bases de données ! Vous pouvez demander à un serveur en exécution d'utiliser à nouveau les tables de droits en exécutant la commande mysqladmin flush-privileges ou mysqladmin reload).

--skip-host-cache
: Ne pas utiliser le cache de nom de domaine pour une résolution des IP plus rapide, mais interroger le serveur DNS à chaque connexion. Comment MySQL utilise les DNS .

--skip-innodb
: Désactive l'utilisation des tables InnoDB . Cela va économiser de la mémoire et accélérer le serveur un peu. N'utilisez pas cette option si vous avez besoin des tables InnoDB .

--skip-isam
: Désactive l'utilisation des tables ISAM . Cela va économiser de la mémoire et accélérer le serveur un peu. Depuis MySQL 4.1, ISAM est désactivé par défaut, ce qui fait que cette option n'apparaît que si le serveur a été configuré avec le support N'utilisez pas cette option si vous avez besoin des tables ISAM .

--skip-name-resolve
: Les noms d'hôtes ne sont pas résolus. Toutes les colonnes Host dans vos tables de droits doivent être des IP numériques ou le mot localhost . Comment MySQL utilise les DNS .

--skip-networking
: Ne pas attendre les connexions TCP/IP du tout. Toutes les interactions du serveur mysqld seront faites avec les sockets Unix. Cette option est particulièrement recommandée pour les systèmes qui utilisent des requêtes locales. Comment MySQL utilise les DNS .

--skip-new
: Ne pas utiliser les nouvelles routines qui sont possiblement erronées.

--skip-symlink
: C'est l'ancienne forme de --skip-symbolic-links , à utiliser avant MySQL 4.0.13.

--symbolic-links , --skip-symbolic-links
: Active ou désactive le support des liens symboliques. Cette option a différents effets sur Windows et sur Unix.
  - Sous Windows, activer les liens symboliques vous permet d'établir un lien symbolique vers une base de données, en créant un fichier directory.sym qui contient le chemin réel. Utiliser les liens symboliques pour les bases sous Windows .
  - Sous Unix, activer les liens symboliques signifie que vous pouvez mettre un fichier d'index MyISAM ou un autre fichier de données dans un autre dossier, avec les options INDEX DIRECTORY ou DATA DIRECTORY de la commande CREATE TABLE . Si

vous effacer ou renommez la table, les fichiers qui sont des liens symboliques seront aussi effacés ou renommés. Syntaxe de `CREATE TABLE` .

Cette option est disponible depuis MySQL 4.0.13.

`--skip-show-database`
: Si MySQL est configuré avec `--with-debug=full` , tous les programmes vérifieront la mémoire pour rechercher les écrasements de zone lors des allocations et libérations de mémoire. Comme ce test est lent, vous pouvez l'éviter, si vous n'avez pas besoin de tester la mémoire, en utilisant cette option.

`--skip-show-database`
: Ne pas autoriser la commande `SHOW DATABASES` , a moins que l'utilisateur n'ait les droits de `SHOW DATABASES` . Depuis la version 4.0.2, vous n'avez plus besoin de cette option, car les droits pour ce faire sont distribués avec le droit de `SHOW DATABASES` .

`--skip-stack-trace`
: Ne pas écrire les piles de traces. Cette option est pratique lorsque vous utilisez `mysqld` avec un débogueur. Sur certains systèmes, vous devez aussi utiliser cette option pour obtenir un fichier de core. Déboguer un serveur MySQL .

`--skip-thread-priority`
: Désactive les priorités des threads pour améliorer la vitesse de réponse.

`--socket=path`
: Sous Unix, le fichier de socket pour les connexions locales. (par défaut, `/tmp/mysql.sock`). Sous Windows, le nom du pipe à utiliser pour les connexions locales qui utilisent un pipe nommé (par défaut, `MySQL`).

−−sql−mode=value[,value[,value...]]
: Spécifie le mode SQL. Le mode SQL du serveur . Cette option est disponible depuis 3.23.41.

`--temp-pool`
: En utilisant cette option, vous allez réduire le jeu de noms qui sont utilisés lors de la création de fichier temporaires, plutôt qu'un nom unique à chaque fois. Ceci est un palliatif au noyau Linux qui crée plusieurs fichiers nouveaux avec des noms différents. Avec l'ancien comportement, Linux semble "perdre de la mémoire", car ils sont alloués au cache d'entrées du dossier au lieu de celui du disque.

`--transaction-isolation=level`
: Configure le niveau d'isolation des transactions. Le niveau peut être `READ-UNCOMMITTED` , `READ-COMMITTED` , `REPEATABLE-READ` ou `SERIALIZABLE` . Syntaxe de `SET TRANSACTION` .

`--tmpdir=path` , `-t path`
: Chemin vers les fichiers temporaires. Il peut s'avérer pratique si votre dossier par défaut `/tmp` réside dans une partition qui est trop petite pour absorber les tables temporaires. Depuis MySQL MySQL 4.1, cette option accepte différents chemins, qui sont utilisés en alternance. Les chemins doivent être séparés par des deux points (`':'`) sous Unix et des points−virgules (`';'`) sous Windows. Il est possible de dire à `tmpdir` de pointer sur un système de fichiers en mémoire, hormis si le serveur MySQL est un esclave. Si c'est un esclave, il faut un système de fichiers permanents (pour que la réplication des tables temporaires et des commandes `LOAD DATA INFILE`) survive a un redémarrage de la machine : un système de fichiers en mémoire `tmpdir` , qui est effacé au lancement de la machine n'est pas acceptable. Un disque est nécessaire pour `tmpdir` , dans ce contexte.

`--user={user_name | user_id}` , `-u {user_name | user_id}`
: Exécute le démon `mysqld` avec l'utilisateur `user_name` ou `userid` (numérique). (``utilisateur'' dans ce contexte fait référence à l'utilisateur du système d'exploitation, mais pas l'utilisateur MySQL, listé dans les tables de droits.)Cette option est **obligatoire** lorsque vous démarrez `mysqld` en tant que `root` . Le serveur va changer d'ID durant le lancement du serveur, pour utiliser un autre utilisateur que `root` . Problèmes généraux de sécurité .

  Depuis MySQL 3.23.56 et 4.0.12: Pour éviter des trous de sécurité si un utilisateur ajoute `--user=root` dans un fichier `my.cnf` (et donc, faisant que le serveur fonctionne en tant que utilisateur système `root`), `mysqld` utilise uniquement la première option `--user` spécifiée, et produit une alerte s'il rencontre d'autres options `--user` . Les options des fichiers `/etc/my.cnf` et `datadir/my.cnf` sont traités avant les options de ligne de commande, et il est recommandé que vous ajoutiez l'option `--user` dans le fichier `/etc/my.cnf` puis spécifiez une valeur autre que `root` . L'option de `/etc/my.cnf` peut être placée avant toute autre option `--user` , ce qui assure que le serveur fonctionnera avec l'utilisateur autre que `root` , et qu'une alerte apparaîtra si une autre option `--user` est découverte.

`--version` , `-V`
: Affiche les informations de version.

Vous pouvez assigner une valeur à une variable système en utilisant une option de la forme `--nom_de_variable=valeur` . Par exemple, `--key_buffer_size=32M` donne à la variable `key_buffer_size` la valeur de 32 Mo.Notez que lorsque vous donnez une valeur à une variable, MySQL peut corriger automatiquement la valeur pour qu'elle reste dans un intervalle donné, ou peut ajuster la valeur à la valeur possible la plus proche.

Il est aussi possible de donner des valeurs aux variables avec la syntaxe `--set-variable=var_name=value` ou `-O var_name=value` . Notez que cette syntaxe est abandonnée depuis MySQL 4.0.

Vous pouvez trouver une description complète de toutes les variables dans la section Variables sytème du serveur . La section sur le paramétrage du serveur inclut des détails sur l'optimisation. Choix des paramètres du serveur .

Vous pouvez changer la valeur de la plupart des variables système sur un serveur en fonctionnement avec la commande `SET` . Syntaxe de `SET` .

Si vous voulez limiter la valeur maximale qu'une option recevra avec la commande `SET` , vous pouvez la définir en utilisant l'option `--maximum-nom_de_variable` en ligne de commande.

5.2.2 Le mode SQL du serveur

Le serveur MySQL peut fonctionner avec différent modes SQL, et, depuis MySQL 4.1, il peut appliquer ces modes au niveau de la connexion du client. Cela permet aux applications d'adapter le comportement du serveur en fonction de leur attentes.

Le mode définit quelle syntaxe SQL MySQL doit supporter, et quels types de vérification il doit faire. Cela rend plus facile l'utilisation de MySQL dans différents environnement, et la connexion avec les autres serveurs de base de données.

Vous pouvez donner un mode SQL par défaut au démarrage de `mysqld` avec l'option `--sql-mode="modes"` . Depuis MySQL 4.1, vous pouvez aussi changer le mode après le démarrage en modifiant la variable `sql_mode` avec la commande `SET [SESSION|GLOBAL] sql_mode='modes'` . Modifier la variable `GLOBAL` affecte les opérations de tous les clients qui se connecteront. En modifiant la variable `SESSION` , cela n'affectera que le client courant. `modes` est une liste de modes différents, séparés par des virgules (`','`). Vous pouvez lire le mode courant avec la commande `SELECT @@sql_mode` . La valeur par défaut est vide (par de modes configurés).

La valeur peut aussi être vide (`--sql-mode=""`) si vous voulez la remettre à zéro.

La liste suivante présente les différents modes supportés :

`ANSI_QUOTES`
: Traite `'"'` comme un délimiteur d'identifiant (comme le caractère MySQL `'`'` ) et non comme un délimiteur de chaînes. Vous pouvez toujours utiliser `'`'` pour délimiter les identifiants en mode ANSI. Avec `ANSI_QUOTES` activée, vous ne pouvez pas utiliser les guillemets doubles pour délimiter une chaîne de caractères, car ce sera uniquement

interprété comme un identifiant. (Nouveau en MySQL 4.0.0.)

`IGNORE_SPACE`
: Permet les espaces entre le nom de la fonction et le caractère `'('` . Cela force les noms de fonctions a être traités comme des mots réservés. En conséquence, si vous voulez accéder aux bases, tables et colonnes dont le nom est un mot réservé, vous devez le mettre entre délimiteurs. Par exemple, comme la fonction `USER()` existe, le nom de la table `user` de la base `mysql` et la colonne `User` de cette table doivent être protégés :

```
SELECT "User" FROM mysql."user";
```

(Nouveau en MySQL 4.0.0.)

`NO_AUTO_VALUE_ON_ZERO`
: `NO_AUTO_VALUE_ON_ZERO` affecte la gestion des colonnes de type `AUTO_INCREMENT` . Normalement, vous générez le prochain numéro de séquence dans la colonne en insérant soit `NULL` soit `0` dedans. `NO_AUTO_VALUE_ON_ZERO` supprime ce comportement pour `0` pour que seule la valeur `NULL` génère le prochain numéro de séquence. Ce mode est utile si vous avez stocké la valeur `0` dans la colonne `AUTO_INCREMENT` de la table. Ce n'est pas recommandé. Par exemple, si vous voulez exporter une table avec `mysqldump` et que vous la rechargez, normalement MySQL va générer de nouveaux identifiants pour les lignes avec la valeur `0` , ce qui entraînera une différence avec la table originale. En activant `NO_AUTO_VALUE_ON_ZERO` avant de recharger le fichier exporter, vous éviter de problème. Depuis MySQL 4.1.1, `mysqldump` va automatiquement inclure les commandes nécessaires dans l'export, pour activer `NO_AUTO_VALUE_ON_ZERO` . (Nouveau en MySQL 4.1.1.)

`NO_DIR_IN_CREATE`
: Lors de la création d'une table, ignore les directives `INDEX DIRECTORY` et `DATA DIRECTORY` . Cette option est pratique sur un esclave de réplication. (Nouveau en MySQL 4.0.15.)

`NO_FIELD_OPTIONS`
: N'affiche pas les options spécifiques à MySQL dans le résultat de `SHOW CREATE TABLE` . Ce mode est utilisé par `mysqldump` dans un souci de portabilité. (Nouveau en MySQL 4.1.1.)

`NO_KEY_OPTIONS`
: N'affiche pas les options spécifiques à MySQL dans le résultat de `SHOW CREATE TABLE` . Ce mode est utilisé par `mysqldump` dans un souci de portabilité. (Nouveau en MySQL 4.1.1.)

`NO_TABLE_OPTIONS`
: N'affiche pas les options de tables spécifiques à MySQL (comme `ENGINE`) dans le résultat de `SHOW CREATE TABLE` . Ce mode est utilisé par `mysqldump` dans un souci de portabilité. (Nouveau en MySQL 4.1.1.)

`NO_UNSIGNED_SUBTRACTION`
: Dans les opérations de soustraction, ne marque pas le résultat `UNSIGNED` si un des opérandes est non signé. Notez que cela fait que `UNSIGNED BIGINT` n'est plus totalement utilisable dans tous les contextes. Fonctions de transtypage . (Nouveau en MySQL 4.0.2.)

`ONLY_FULL_GROUP_BY`
: N'autorise pas les requêtes dont la clause `GROUP BY` fait référence à une colonne qui n'est pas sélectionnée. (Nouveau en MySQL 4.0.0.)

`PIPES_AS_CONCAT`
: Traite `||` comme un opérateur de concaténation (identique à `CONCAT()`) au lieu d'être un synonyme de `OR` . (Nouveau en MySQL 4.0.0.)

`REAL_AS_FLOAT`
: Traite le type `REAL` comme un synonyme `FLOAT` plutôt que comme un synonyme de `DOUBLE` . (Nouveau en MySQL 4.0.0.)

Les modes suivants sont fournis comme des raccourcis pour des combinaisons de modes, issus de la liste précédente. Ils sont valables en MySQL 4.1.1.

`ANSI`
: Equivalent à `REAL_AS_FLOAT` , `PIPES_AS_CONCAT` , `ANSI_QUOTES` , `IGNORE_SPACE` , `ONLY_FULL_GROUP_BY` . Exécuter MySQL en mode ANSI .

`DB2`
: Equivalent à `PIPES_AS_CONCAT` , `ANSI_QUOTES` , `IGNORE_SPACE` , `NO_KEY_OPTIONS` , `NO_TABLE_OPTIONS` , `NO_FIELD_OPTIONS` .

`MAXDB`

Equivalent à PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS .

MSSQL
: Equivalent à PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS .

MYSQL323
: Equivalent à NO_FIELD_OPTIONS .

MYSQL40
: Equivalent à NO_FIELD_OPTIONS .

ORACLE
: Equivalent à PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS .

POSTGRESQL
: Equivalent à PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS .

5.2.3 Variables serveur système

Le serveur entretient de nombreuses variables système qui indiquent comment il est configuré. Toutes les variables ont des valeurs par défaut. Elles peuvent être configuré au lancement du serveur, avec les options de ligne de commande, ou bien durant l'exécution, avec la commande `SET` .

Depuis MySQL version 4.0.3, le serveur `mysqld` entretient deux types de variables. Les variables globales, qui affectent les opérations générales du serveur. Et les variables de session qui affectent les comportements individuels des connexions.

Lorsque `mysqld` démarre, toutes les variables globales sont initialisées à partir des arguments passés en ligne de commande et des fichiers de configuration. Vous pouvez changer ces valeurs avec la commande `SET GLOBAL` . Lorsqu'un nouveau thread est créé, les variables spécifiques aux threads sont initialisées à partir des variables globales et ne changeront pas même si vous utilisez la commande `SET GLOBAL var_name` . Pour changer une variable globale, vous devez avoir les droits de `SUPER` .

Le serveur entretient aussi un jeu de variables de session pour chaque client qui se connecte. Les variables de session du serveur sont initialisées au moment de la connexion, en utilisant les valeurs correspondantes des variables globales. Pour les variables de session qui sont dynamiques, le client peut les changer avec la commande `SET SESSION var_name` . Modifier une variable de session ne requiert aucun droit spécifique, mais le client ne modifiera le comportement du serveur que pour sa connexion, et non pour les connexions des autres.

Une modification de variable globale est visible par tous les clients qui accèdent aux variables globales. Cependant, elle n'affectera les connexions des clients que pour les nouvelles connexions. Les variables de sessions déjà en court continueront à fonctionner avec la même configuration, jusqu'à leur déconnexion. Même le client qui a émis la commande `SET GLOBAL` ne verra aucun changement.

Lorsque vous modifiez une variable avec une option de démarrage, les valeurs de variables peuvent être spécifiées avec le suffixe `K` , `M` ou `G` , pour indiquer des kilo−octets, des megaoctets ou des gigaoctets. Par exemple, pour lancer le serveur avec une taille de buffer de clé de 16 Mo, vous pouvez utiliser :

```
mysqld --key_buffer_size=16M
```

Avant MySQL 4.0, vous deviez utiliser la syntaxe suivante :

```
mysqld --set-variable=key_buffer_size=16M
```

La lettre de suffixe peut être en majuscule ou en minuscule : `16M` et `16m` sont équivalents.

Durant l'exécution, utilisez la commande `SET` pour donner de nouvelles valeurs aux variables système. Dans ce contexte, les lettres de suffixes ne pourront pas être utilisées. Leur valeur peut être calculée avec l'expression suivante :

```
mysql> SET sort_buffer_size = 10 * 1024 * 1024;
```

Pour spécifier explicitement si vous voulez modifier une variable globale ou une variable de session, utilisez les options `GLOBAL` et `SESSION` :

```
mysql> SET GLOBAL sort_buffer_size = 10 * 1024 * 1024;
mysql> SET SESSION sort_buffer_size = 10 * 1024 * 1024;
```

Si cette option est omise, la variable de session sera modifiée.Les variables qui peuvent être modifiées durant l'exécution sont listées dans la section Variables dynamiques système .

Si vous voulez restreindre le maximum possible d'une variable système, modifiée avec `SET` , vous pouvez spécifier ce maximum avec les options de la forme `--maximum-var_name` au lancement du serveur. Par exemple, pour éviter que la valeur de `query_cache_size` dépasse 32 Mo, utilisez l'option `--maximum-query_cache_size=32M` . Cette fonctionnalité est disponible depuis MySQL 4.0.2.

Vous pouvez découvrir les variables système et leur valeur avec la commande `SHOW VARIABLES` . Voyez Variables système pour plus d'informations.

```
mysql> SHOW VARIABLES;
+---------------------------------+------------------------------+
| Variable_name                   | Value                        |
+---------------------------------+------------------------------|
| back_log                        | 50                           |
| basedir                         | /usr/local/mysql             |
| bdb_cache_size                  | 8388572                      |
| bdb_home                        | /usr/local/mysql             |
| bdb_log_buffer_size             | 32768                        |
| bdb_logdir                      |                              |
| bdb_max_lock                    | 10000                        |
| bdb_shared_data                 | OFF                          |
| bdb_tmpdir                      | /tmp/                        |
| bdb_version                     | Sleepycat Software: ...      |
| binlog_cache_size               | 32768                        |
| bulk_insert_buffer_size         | 8388608                      |
| character_set                   | latin1                       |
| character_sets                  | latin1 big5 czech euc_kr     |
| concurrent_insert               | ON                           |
| connect_timeout                 | 5                            |
| convert_character_set           |                              |
| datadir                         | /usr/local/mysql/data/       |
| default_week_format             | 0                            |
| delay_key_write                 | ON                           |
| delayed_insert_limit            | 100                          |
| delayed_insert_timeout          | 300                          |
| delayed_queue_size              | 1000                         |
| flush                           | OFF                          |
| flush_time                      | 0                            |
| ft_boolean_syntax               | + -><()~*:""&|               |
| ft_max_word_len                 | 84                           |
| ft_min_word_len                 | 4                            |
| ft_query_expansion_limit        | 20                           |
| ft_stopword_file                | (built-in)                   |
| have_bdb                        | YES                          |
| have_innodb                     | YES                          |
| have_isam                       | YES                          |
| have_openssl                    | YES                          |
| have_query_cache                | YES                          |
| have_raid                       | NO                           |
| have_symlink                    | DISABLED                     |
| init_file                       |                              |
| innodb_additional_mem_pool_size | 1048576                      |
| innodb_buffer_pool_size         | 8388608                      |
| innodb_data_file_path           | ibdata1:10M:autoextend       |
| innodb_data_home_dir            |                              |
| innodb_fast_shutdown            | ON                           |
| innodb_file_io_threads          | 4                            |
| innodb_flush_log_at_trx_commit  | 1                            |
| innodb_flush_method             |                              |
| innodb_force_recovery           | 0                            |
| innodb_lock_wait_timeout        | 50                           |
| innodb_log_arch_dir             |                              |
| innodb_log_archive              | OFF                          |
| innodb_log_buffer_size          | 1048576                      |
| innodb_log_file_size            | 5242880                      |
| innodb_log_files_in_group       | 2                            |
```

```
| innodb_log_group_home_dir       | ./                          |
| innodb_mirrored_log_groups      | 1                           |
| innodb_thread_concurrency       | 8                           |
| interactive_timeout             | 28800                       |
| join_buffer_size                | 131072                      |
| key_buffer_size                 | 16773120                    |
| key_cache_age_threshold         | 300                         |
| key_cache_block_size            | 1024                        |
| key_cache_division_limit        | 100                         |
| language                        | /usr/local/mysql/share/...  |
| large_files_support             | ON                          |
| local_infile                    | ON                          |
| locked_in_memory                | OFF                         |
| log                             | OFF                         |
| log_bin                         | OFF                         |
| log_slave_updates               | OFF                         |
| log_slow_queries                | OFF                         |
| log_update                      | OFF                         |
| log_warnings                    | OFF                         |
| long_query_time                 | 10                          |
| low_priority_updates            | OFF                         |
| lower_case_table_names          | 0                           |
| max_allowed_packet              | 1047552                     |
| max_binlog_cache_size           | 4294967295                  |
| max_binlog_size                 | 1073741824                  |
| max_connect_errors              | 10                          |
| max_connections                 | 100                         |
| max_delayed_threads             | 20                          |
| max_error_count                 | 64                          |
| max_heap_table_size             | 16777216                    |
| max_join_size                   | 4294967295                  |
| max_relay_log_size              | 0                           |
| max_sort_length                 | 1024                        |
| max_tmp_tables                  | 32                          |
| max_user_connections            | 0                           |
| max_write_lock_count            | 4294967295                  |
| myisam_max_extra_sort_file_size | 268435456                   |
| myisam_max_sort_file_size       | 2147483647                  |
| myisam_recover_options          | force                       |
| myisam_repair_threads           | 1                           |
| myisam_sort_buffer_size         | 8388608                     |
| net_buffer_length               | 16384                       |
| net_read_timeout                | 30                          |
| net_retry_count                 | 10                          |
| net_write_timeout               | 60                          |
| open_files_limit                | 1024                        |
| pid_file                        | /usr/local/mysql/name.pid   |
| port                            | 3306                        |
| protocol_version                | 10                          |
| query_cache_limit               | 1048576                     |
| query_cache_size                | 0                           |
| query_cache_type                | ON                          |
| read_buffer_size                | 131072                      |
| read_rnd_buffer_size            | 262144                      |
| rpl_recovery_rank               | 0                           |
| server_id                       | 0                           |
| skip_external_locking           | ON                          |
| skip_networking                 | OFF                         |
| skip_show_database              | OFF                         |
| slave_net_timeout               | 3600                        |
| slow_launch_time                | 2                           |
| socket                          | /tmp/mysql.sock             |
| sort_buffer_size                | 2097116                     |
| sql_mode                        |                             |
| table_cache                     | 64                          |
| table_type                      | MYISAM                      |
| thread_cache_size               | 3                           |
| thread_stack                    | 131072                      |
| timezone                        | EEST                        |
| tmp_table_size                  | 33554432                    |
| tmpdir                          | /tmp/:/mnt/hd2/tmp/         |
| tx_isolation                    | READ-COMMITTED              |
| version                         | 4.0.4-beta                  |
| wait_timeout                    | 28800                       |
+---------------------------------+-----------------------------+
```

La plupart des variables système sont présentées ici. Les variables sans version sont présente depuis MySQL 3.22. Les variables système InnoDB sont listées dans Options de démarrage InnoDB .

Les valeurs pour les tailles de buffer, longueur et taille de pile sont données en octets, à moins que cela ne soit spécifié autrement.

Les informations sur le choix des valeurs de ces paramètres est disponible dans Choix des paramètres du serveur .

`ansi_mode`

Vaut `ON` si `mysqld` a été démarré en mode `--ansi` . Exécuter MySQL en mode ANSI . Cette variable a été ajoutée en MySQL 3.23.6 et supprimée en 3.23.41. Voyez la description de `sql_mode` .

`back_log`

Le nombre de connexions sortantes que MySQL peut supporter. Cette valeur entre en jeu lorsque le thread principal MySQL reçoit de très nombreuses requêtes de connexions en très peu de temps. MySQL prend un peu de temps (même si c'est très peu de temps), pour vérifier la connexion et démarrer un nouveau thread. La valeur de `back_log` indique combien de requête seront mises en attente durant ce temps. Vous devrez augmenter ce nombre si vous voulez mettre en attente plus de requêtes durant une courte période de temps.

En d'autres termes, cette valeur est la taille de la queue d'attente pour les connexions TCP/IP entrantes. Votre système d'exploitation a ses propres limites pour ce type de queue. La page du manuel Unix `listen(2)` doit contenir plus de détails. Vérifiez la documentation de votre OS pour connaître la valeur maximale de votre système. Si vous donne une valeur à `back_log` qui est plus grande que celle que votre système supporte, cela restera sans effet.

`basedir`

Le dossier d'installation de MySQL. La valeur de l'option `--basedir` .

`bdb_cache_size`

Le buffer qui est alloué pour mettre en cache des lignes et des index pour les tables `BDB` . Si vous n'utilisez pas la tables `BDB` , vous devriez démarrer `mysqld` avec l'option `--skip-bdb` pour ne pas gaspiller de mémoire. Cette variable a été ajoutée en MySQL 3.23.14.

`bdb_home`

Le dossier de base des tables `BDB` . Cette valeur doit être la même que celle de la variable `datadir` . Cette variable a été ajoutée en MySQL 3.23.14.

`bdb_log_buffer_size`

Le buffer qui est alloué pour mettre en cache des lignes et des index pour les tables `BDB` . Si vous n'utilisez pas la tables `BDB` , vous devriez démarrer `mysqld` avec l'option `--skip-bdb` pour ne pas gaspiller de mémoire. Cette variable a été ajoutée en MySQL 3.23.31.

`bdb_logdir`

Le dossier où le moteur `BDB` écrit les fichiers de log. C'est la valeur de l'option `--bdb-logdir` . Cette variable a été ajoutée en MySQL 3.23.14.

`bdb_max_lock`

Le nombre maximum de verrous (par défaut 10 000) que vous pouvez activer simultanément dans une table `BDB` . Vous devriez augmenter cette valeur si vous obtenez des erreurs du type `bdb: Lock table is out of available locks` ou `Got error 12 from ...` lorsque vous avez de longues transactions ou que `mysqld` doit examiner de nombreuses lignes pour calculer la requête.

```
 bdb: Lock table is out of available locks
Got error 12 from ...
```

Cette variable a été ajoutée en MySQL 3.23.29.

`bdb_shared_data`

Vaut `ON` si vous utilisez l'option `--bdb-shared-data` . Cette variable a été ajoutée en MySQL 3.23.29.

`bdb_tmpdir`

La valeur de l'option `--bdb-tmpdir` . Cette variable a été ajoutée en MySQL 3.23.14.

`bdb_version`

La version du moteur `BDB` . Cette variable a été ajoutée en MySQL 3.23.31.

`binlog_cache_size`

La taille du cache qui contient les requêtes SQL destinées au log binaire, durant une transaction. Un cache binaire est alloué à chaque client si le serveur supporte les moteurs transactionnel, et depuis MySQL 4.1.2, si le serveur a un log binaire activé (option `--log-bin`). Si vous utilisez souvent de grandes transactions multi−requêtes, vous devez augmenter cette valeur pour améliorer les performances. Les variables `Binlog_cache_use` et `Binlog_cache_disk_use` sont aussi utiles pour optimiser la taille de cette variable. Cette variable a été ajoutée en MySQL 3.23.29. Le log binaire des mises à jour .

`bulk_insert_buffer_size`
: `MyISAM` utilise une cache hiérarchisé pour les insertions de masses (c'est à dire `INSERT ... SELECT` , `INSERT ... VALUES (...), (...), ...` , et `LOAD DATA INFILE`). Cette variable limite la taille du cache en octets, par threads. Utiliser la valeur de 0 va désactiver cette optimisation. **Note** : ce cache est uniquement utilisé lorsque vous ajoutez des données dans une table non−vide. Par défaut, cette option vaut 8 Mo. Cette variable a été ajoutée en MySQL 4.0.3. Cette variable s'appelait `myisam_bulk_insert_tree_size` .

`character_set`
: Le jeu de caractères par défaut. Cette variable a été ajoutée en MySQL 3.23.3, puis retirée en MySQL 4.1.1 et remplacées par différentes variables `character_set_xxx` .

`character_set_client`
: Le jeu de caractères pour les commandes du client. Cette variable a été ajoutée en MySQL 4.1.1.

`character_set_connection`
: Le jeu de caractères utilisé pour les littéraux qui n'ont pas d'indication de jeu de caractères, pour certaines fonctions et pour les conversions de nombres vers une chaîne. Cette variable a été ajoutée en MySQL 4.1.1.

`character_set_database`
: Le jeu de caractères par défaut pour les bases de données. Le serveur modifie cette variable à chaque fois que la base de données par défaut change. S'il n'y a pas de base de données par défaut, cette variable prend la valeur de `character_set_server` . Cette variable a été ajoutée en MySQL 4.1.1.

`character_set_results`
: Le jeu de caractères utilisé pour retourner des résultats au client. Cette variable a été ajoutée en MySQL 4.1.1.

`character_set_server`
: Le jeu de caractères par défaut pour le serveur. Cette variable a été ajoutée en MySQL 4.1.1.

`character_set_system`
: Le jeu de caractères utilisé par le serveur pour stocker des identifiants. Cette valeur est toujours `utf8` . Cette variable a été ajoutée en MySQL 4.1.1.

`character_sets`
: Les jeux de caractères supportés. Cette variable a été ajoutée en MySQL 3.23.15.

`collation_connection`
: Cette variable a été ajoutée en MySQL 4.1.1.

`collation_database`
: La collation utilisée par la base de données par défaut. Le serveur modifie cette variable à chaque fois que la base de données par défaut change. S'il n'y a pas de base de données par défaut, cette variable prend la valeur de `collation_server` . Cette variable a été ajoutée en MySQL 4.1.1.

`collation_server`
: La collation par défaut du serveur. Cette variable a été ajoutée en MySQL 4.1.1.

`concurrent_insert`
: Si cette option vaut `ON` , MySQL va vous permettre de réaliser des commandes `INSERT` sur les tables `MyISAM` en même temps que d'autres commandes `SELECT` seront exécutées. Vous pouvez désactiver cette option en démarrant `mysqld` avec l'option `--safe` or `--skip-new` . Cette variable a été ajoutée en MySQL 3.23.7.

`connect_timeout`
: Le nombre de secondes d'attente d'un paquet de connexion avant de conclure avec une erreur `Bad handshake` .

`datadir`
: Le dossier de données de MySQL. C'est la valeur de l'option `--datadir` .

`default_week_format`
: Le mode par défaut pour la fonction `WEEK()` . Cette variable a été ajoutée en MySQL 4.0.14.

`delay_key_write`
: Les options pour les tables `MyISAM` . Elles peuvent prendre l'une des valeurs suivantes :

Option	Description
OFF	DELAYED_KEY_WRITE est ignoré.
ON	(Par défaut) MySQL va honorer l'option DELAY_KEY_WRITE de CREATE TABLE .
ALL	Toutes les nouvelles tables ouvertes sont traitées comme si elles étaient créées avec l'option DELAY_KEY_WRITE .

Si `DELAY_KEY_WRITE` est activé, cela signifie que le buffer de clé des tables ayant cette option ne seront pas écrit sur le disque dès la fin de la modification de la table, mais attendrons que la table soit écrite. Cela accélère notablement les écritures des modifications, mais il faut penser à ajouter une vérification automatique des tables au démarrage avec `--myisam-recover=BACKUP,FORCE` . Voir aussi Options de ligne de commande `mysqld` et Options de démarrage `MyISAM` .

Notez que `--external-locking` n'offre aucune protection contre les corruptions d'index pour les tables qui utilisent les écritures retardées de clés.

Cette variable a été ajoutée en MySQL 3.23.8.

`delayed_insert_limit`
: Après avoir inséré `delayed_insert_limit` lignes, le gestionnaire de `INSERT DELAYED` va vérifiez si il n'y a pas de commande `SELECT` en attente. Si c'est le cas, il va autoriser ces commandes avant de continuer.

`delayed_insert_timeout`
: Combien de temps le thread `INSERT DELAYED` doit attendre les commandes `INSERT` avant de s'achever.

`delayed_queue_size`
: Quelle taille de file (en lignes) doit être allouée pour gérer les commandes `INSERT DELAYED` . Si la file se remplit, tous les clients qui émettent des commandes `INSERT DELAYED` devront attendre un peu de place avant de pouvoir continuer.

`flush`
: Cette option vaut `ON` si vous avez démarré MySQL avec l'option `--flush` . Cette variable a été ajoutée en MySQL 3.22.9.

`flush_time`
: Si cette option a une valeur non nulle, toutes les `flush_time` secondes, toutes les tables seront fermées (pour libérez des ressources et synchroniser les index sur le disque). Nous ne recommandons cette option que sur les systèmes Windows 9x/Me, ou les systèmes qui ont très peu de ressources. Cette variable a été ajoutée en MySQL 3.22.18.

`ft_boolean_syntax`
: Liste des opérateurs supportés par `IN BOOLEAN MODE` . Cette variable a été ajoutée en MySQL 4.0.1. Opérateurs booléens de recherche en texte intégral .La valeur par défaut de cette variable est `'+ -><()~*:"" . Les règles pour modifier cette valeur sont les suivantes :`

```
La fonction de l'opérateur est déterminée par sa position dans la chaîne.
La chaîne de remplacement doit faire 14 caractères.
Chaque caractère doit être ASCII, non-alphanumérique.
Le premier ou le deuxième caractère doit être un espace.
Aucun doublon n'est autorisé, hormis les opérateurs guillemets aux positions 11 et 12. Ceux deux
caractères ne sont pas obligatoirement les mêmes, mais ils sont les deux qui peuvent l'être.
Les positions 10, 13 et 14 (qui sont par défaut ':' , ' et '|' ) sont réservées pour une utilisation
ultérieure.
```

`ft_max_word_len`
: La taille maximale d'un mot à inclure dans un index `FULLTEXT` . Cette variable a été ajoutée en MySQL 4.0.0.

Note : les index `FULLTEXT` doivent être reconstruits après chaque modification de cette variable. Utilisez `REPAIR TABLE tbl_name QUICK` .

`ft_min_word_len`
: La taille minimale d'un mot à inclure dans un index `FULLTEXT` . Cette variable a été ajoutée en MySQL 4.0.0.

Note : les index `FULLTEXT` doivent être reconstruits après chaque modification de cette variable. Utilisez `REPAIR TABLE tbl_name QUICK` .

`ft_query_expansion_limit`
: Le nombre de solutions générales à utiliser pour les extensions de recherche avec `WITH QUERY EXPANSION` . Cette variable a été ajoutée en MySQL 4.1.1.

`ft_stopword_file`
: Une fichier dans lequel lire une liste de mots interdits pour les recherches en texte plein. Tous les mots du fichier seront utilisés : les commentaires **ne sont pas** respectés. Par défaut, des listes de mots interdits internes sont utilisés, tels que

définis dans `myisam/ft_static.c`). En donnant à cette option la valeur d'une chaîne vide `""` , vous désactivez le filtre de mots interdits. **Note** : les index `FULLTEXT` doivent être reconstruits après chaque modification de cette variable. Utilisez la commande `REPAIR TABLE tbl_name QUICK` . Cette variable a été ajoutée en MySQL 4.1.0.

`group_concat_max_len`
: La taille maximale de la chaîne résultat de `GROUP_CONCAT()` . Cette variable a été ajoutée en MySQL 4.1.0.

`have_bdb`
: `YES` si `mysqld` supporte les tables `BDB` . `DISABLED` si `--skip-bdb` a été utilisé. Cette variable a été ajoutée en MySQL 3.23.30.

`have_innodb`
: `YES` si `mysqld` supporte les tables `InnoDB` . `DISABLED` si `--skip-innodb` a été utilisé. Cette variable a été ajoutée en MySQL 3.23.37.

`have_isam`
: `YES` si `mysqld` supporte les tables `ISAM` . `DISABLED` si `--skip-isam` a été utilisé. Cette variable a été ajoutée en MySQL 3.23.30.

`have_raid`
: `YES` si `mysqld` supporte les tables `RAID` . Cette variable a été ajoutée en MySQL 3.23.30.

`have_openssl`
: `YES` si `mysqld` supporte le chiffrement SSL avec le protocole de communication client / serveur. Cette variable a été ajoutée en MySQL 3.23.43.

`init_connect`
: Une chaîne a exécuter sur le serveur lors de chaque connexion. La chaîne est constituée d'une ou plusieurs commandes SQL. Pour spécifier une commande multiple, séparez les requêtes individuelles par des points−virgules. Cette variable a été ajoutée en MySQL version 4.1.2.Par exemple, chaque client commence par défaut avec le mode d'auto−validation activé. Il n'y a pas de variable globale à spécifier pour désactiver l'auto−validation, et `init_connect` peut servir à ça :

```
SET GLOBAL init_connect='SET AUTOCOMMIT=0';
```

Cette variable peut aussi être configurée en ligne de commande ou dans un fichier d'options. Pour assigner la variable comme montré dans le fichier d'options, ajoutez ces lignes là :

```
 [mysqld]
init_connect='SET AUTOCOMMIT=0'
```

Cette variable a été ajoutée en MySQL 4.1.2.

`init_file`
: Le nom du fichier spécifié avec l'option `--init-file` lorsque vous démarrez le serveur. C'est un fichier qui contient les requêtes SQL que vous voulez voir exécutées dès le démarrage. Chaque commande doit être sur une seule ligne, et ne doit pas inclure de commentaires. Cette variable a été ajoutée en MySQL 3.23.2.

`init_slave`
: Cette variable est similaire à `init_connect` , mais la chaîne doit être exécutée par l'esclave, à chaque démarrage du thread SQL. Le format de la chaîne est le même que pour la variable `init_connect` . Cette variable a été ajoutée en MySQL 4.1.2.

`innodb_xxx`
: Les variables système de `InnoDB` sont listées dans Options de démarrage `InnoDB` .

`interactive_timeout`
: Le nombre de secondes durant lequel le serveur attend une activité de la part de la connexion avant de la fermée. Un client interactif est un client qui utilise l'option `CLIENT_INTERACTIVE` avec `mysql_real_connect()` . Voir aussi `wait_timeout` .

`join_buffer_size`
: La taille du buffer qui est utilisée pour les jointures complètes (les jointures qui n'utilisent pas d'index). Ce buffer est alloué une fois pour chaque jointure entre deux tables. Augmentez cette valeur si vous voulez obtenir des jointures plus rapides, lorsque l'ajout d'index n'est pas possible. Normalement, le mieux est d'ajouter de bons index.

`key_buffer_size`
: Les blocs d'index des tables `MyISAM` et `ISAM` sont mis en buffer et partagés par tous les threads. `key_buffer_size` est la taille du buffer utilisé. Le buffer de clé est aussi appelé le cache de clé.Augmentez cette valeur pour obtenir une meilleure gestion des index (pour les lectures et écritures multiples), autant que vous le pouvez : 64 Mo sur une

machine de 256 Mo est une valeur répandue. Toutefois, si vous utilisez une valeur trop grande (par exemple, plus de 50% de votre mémoire totale), votre système risque de commencer à utiliser sa mémoire swap, et devenir très lent. N'oubliez pas que MySQL ne met pas en cache les données lues, et il faut laisser le système d'exploitation respirer.

Pour obtenir encore plus de vitesse lors de l'écriture de plusieurs lignes en même temps, utilisez `LOCK TABLES` . Syntaxe de `LOCK TABLES` et `UNLOCK TABLES` .

Vous pouvez vérifier les performances du buffer de clés avec la commande `SHOW STATUS` et en examinant les variables `Key_read_requests` , `Key_reads` , `Key_write_requests` et `Key_writes` . Syntaxe de `SET` et `SHOW` .

Le ratio `Key_reads/Key_read_requests` doit normalement être inférieur à 0.01. Le ratio `Key_writes/Key_write_requests` est généralement près de 1 si vous utilisez essentiellement des modifications et des effacements mais il peut être plus petit si vous avez des modifications qui changent plusieurs lignes en même temps, ou si vous utilisez l'option `DELAY_KEY_WRITE` .

La fraction du buffer de clé utilisée est déterminée avec la variable `key_buffer_size` en conjonction avec la variable `Key_blocks_used` et la taille de bloc de buffer. Depuis MySQL 4.1.1, la taille de bloc de buffer est disponible dans la variable serveur `key_cache_block_size` . La fraction utilisée du buffer est :

```
(Key_blocks_used * key_cache_block_size) / key_buffer_size
```

Avant MySQL 4.1.1, les blocs du cache de clé étaient de 1024 octets, ce qui fait que la fraction utilisée était :

```
(Key_blocks_used * 1024) / key_buffer_size
```

Le cache de clés `MyISAM` .

`key_cache_age_threshold`
: Cette valeur contrôle le transit des buffers d'une sous−chaîne de cache de clé vers une autre, moins prioritaire. Les valeurs les plus basses accroissent la vitesse de transit. La valeur minimale est de 100. La valeur par défaut est 300. Cette variable a été ajoutée en MySQL 4.1.1. Le cache de clés `MyISAM` .

`key_cache_block_size`
: La taille du bloc de cache de clé, en octets. La valeur par défaut est 1024. Cette variable a été ajoutée en MySQL 4.1.1. Le cache de clés `MyISAM` .

`key_cache_division_limit`
: Le point de division entre la sous−chaîne prioritaire et la seconde sous−chaîne. Cette valeur est le pourcentage du buffer à utiliser pour la sous−chaîne secondaire. Les valeurs possibles vont de 1 à 100. La valeur par défaut est 100. Cette variable a été ajoutée en MySQL 4.1.1. Le cache de clés `MyISAM` .

`language`
: La langue utilisée pour les message d'erreurs.

`large_file_support`
: Si `mysqld` a été compilé avec le support des grands fichiers. Cette variable a été ajoutée en MySQL 3.23.28.

`local_infile`
: Si `mysqld` a été configuré avec le support de `LOCAL` pour les commandes `LOAD DATA INFILE` . Cette variable a été ajoutée en MySQL 4.0.3.

`locked_in_memory`
: Si `mysqld` a été verrouillé en mémoire avec `--memlock` Cette variable a été ajoutée en MySQL 3.23.25.

`log`
: Si le log de toutes les requêtes est activé. Le log général de requêtes .

`log_bin`
: Si le log binaire est activé. Cette variable a été ajoutée en MySQL 3.23.14. Le log binaire des mises à jour .

`log_slave_updates`

Si les modifications des esclaves doivent être enregistrées. Le log binaire doit être activé pour que cette option fonctionne. Cette variable a été ajoutée en MySQL 3.23.17. Options de réplication .

`log_slow_queries`
: Indique si les requêtes lentes doivent être enregistrées. ``Lente'' est déterminé par la valeur de `long_query_time` . Cette variable a été ajoutée en MySQL 4.0.2. Le log des requêtes lentes .

`log_update`
: Si le log de modification est activé. Cette variable a été ajoutée en MySQL 3.22.18. Notez que le log binaire est préférable au log de modifications, qui n'est plus disponible en MySQL 5.0. Le log des mises à jour .

`long_query_time`
: Si une requête prend plus de `long_query_time` secondes, le compteur de requêtes lentes `Slow_queries` sera incrémenté. Si vous utilisez l'option `--log-slow-queries` , ces requêtes seront enregistrées dans un historique de requêtes lentes. Cette durée est mesurée en temps réel, et non pas en temps processus, ce qui fait que les requêtes qui seraient juste sous la limite avec un système légèrement chargé, pourrait être au dessus avec le même système, mais chargé. Le log des requêtes lentes .

`low_priority_updates`
: Si cette option vaut `1` , toutes les requêtes `INSERT` , `UPDATE` , `DELETE` et `LOCK TABLE WRITE` attendent qu'il n'y ait plus de `SELECT` ou de `LOCK TABLE READ` en attente pour cette table. Cette variable s'appelait avant `sql_low_priority_updates` . Cette variable a été ajoutée en MySQL 3.22.5.

`lower_case_table_names`
: Si cette option vaut 1, les noms de tables sont stockées en minuscules sur le disque, et les comparaisons de nom de tables seront insensibles à la casse. Depuis la version 4.0.2, cette option s'applique aussi aux noms de bases. Depuis la version 4.1.1 cette option s'applique aussi aux alias de table. Identifier Case Sensitivity .Vous **ne devez pas** mettre cette variable à 0 si vous utilisez MySQL sur un serveur qui n'a pas de sensibilité à la casse au niveau du système de fichiers (comme Windows ou Mac OS X). Nouveau en 4.0.18 : si cette variable vaut 0 est que le système de fichier n'est pas sensible à la casse, MySQL va automatiquement donner la valeur de 2 à `lower_case_table_names` .

`max_allowed_packet`
: La taille maximale d'un paquet. Le buffer de message est initialisé avec `net_buffer_length` octets, mais peut grandir jusqu'à `max_allowed_packet` octets lorsque nécessaire. Cette valeur est par défaut petit, pour intercepter les gros paquets, probablement erronés.

  Vous devez augmenter cette valeur si vous utilisez de grandes colonnes `BLOB` . Cette valeur doit être aussi grande que le plus grand `BLOB` que vous utiliserez. Le protocole limite actuellement `max_allowed_packet` à 16 Mo en MySQL 3.23 et 1 Go en MySQL 4.0.

`max_binlog_cache_size`
: Si une transaction multi−requête requiert plus que cette quantité de mémoire, vous obtiendrez une erreur `"Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage"` . Cette variable a été ajoutée en MySQL 3.23.29.

`max_binlog_size`
: Disponible depuis la version 3.23.33. Si vous écrivez dans le log binaire (de réplication) et que cela dépasse la taille de `max_binlog_size` , une erreur sera indiquée. Vous ne pouvez pas donner à `max_binlog_size` une valeur inférieure à 1024 octets, ou plus grande que 1 Go. Cette variable a été ajoutée en MySQL 3.23.33.Notez bien si vous utilisez les transactions : une transaction est écrite en une seule fois dans le log binaire, et elle ne peut pas être répartie en plusieurs fichiers. Par conséquent, si vous avez de grandes transactions, vous verrez peut être des fichiers de log plus grand que `max_binlog_size` .

  Si `max_relay_log_size` vaut 0, la valeur de `max_binlog_size` s'applique aussi aux logs de relais. `max_relay_log_size` a été ajoutée en MySQL 4.0.14.

`max_connect_errors`
: S'il y a plus que `max_connect_errors` connexion interrompues depuis un même hôte, cet hôte sera bloqué dans ses prochaines tentatives de connexions. Vous pouvez

débloquer un hôte avec la commande `FLUSH HOSTS` .

`max_connections`
: Le nombre maximal de clients simultanés accepté. En augmentant cette valeur, vous augmentez le nombre de pointeur de fichier que requiert `mysqld` . Voyez la section Comment MySQL ouvre et ferme les tables pour des commentaires sur les pointeurs de fichiers. Voyez aussi la section Erreur `Too many connections` .

`max_delayed_threads`
: Ne pas lancer plus que `max_delayed_threads` threads pour gérer les insertions `INSERT DELAYED` . Si vous essayez d'insérer des données dans une nouvelle table alors que tous les gestionnaires `INSERT DELAYED` sont utilisés, la ligne sera insérée comme si l'option `DELAYED` n'avait pas été spécifiée. Cette variable a été ajoutée en MySQL 3.23.0.

`max_error_count`
: Le nombre maximum d'erreur, alertes et messages de note à stocker pour afficher avec `SHOW ERRORS` ou `SHOW WARNINGS` . Cette variable a été ajoutée en MySQL 4.1.0.

`max_heap_table_size`
: Ne pas autoriser la création de tables de type `MEMORY` (`HEAP`) plus grande que `max_heap_table_size` . La valeur de la variable est utilisée pour calculer la valeur maximale de `MAX_ROWS` pour la table `MEMORY` . Modifier cette variable n'a pas d'effet sur les tables existantes, à moins que la table ne soit recrée, avec une commande comme `CREATE TABLE` ou `TRUNCATE TABLE` , ou encore modifiée avec `ALTER TABLE` . Cette variable a été ajoutée en MySQL 3.23.0.

`max_insert_delayed_threads`
: Cette variable est un synonyme de `max_delayed_threads` . Cette variable a été ajoutée en MySQL 4.0.19.

`max_join_size`
: Les jointures qui liront probablement plus de `max_join_size` lignes, retourneront une erreur. Utilisez cette valeur si vos utilisateurs font des jointures avec de mauvaises clauses `WHERE` , qui prennent trop de temps, et retournent des millions de lignes.

  En donnant une valeur à cette variable qui est autre que `DEFAULT` remet la valeur de `SQL_BIG_SELECTS` à `0` . Si vous modifiez à nouveau la valeur de `SQL_BIG_SELECTS` , la variable `max_join_size` sera ignorée.

  Si un résultat de requête est déjà dans le cache de requête, aucune limite de taille n'est vérifiée, car le résultat a déjà été compilé, et cela ne posera aucun problème au serveur pour l'envoyer à nouveau.

  Cette variable s'appelait précédemment `sql_max_join_size` .

`max_relay_log_size`
: Disponible depuis la version 4.0.14. Si vous écrivez des données dans un log de relais et que leur taille dépasse la valeur donnée, le log de relais passe au fichier suivant. Cette variable vous permet de placer différentes contraintes de taille sur les logs binaire et de relais. Cependant, en donnant la valeur de 0 à cette valeur, MySQL utilisera `max_binlog_size` pour les deux logs, binaire et de relais. Vous devez donner à `max_relay_log_size` la valeur de 0 ou plus de 4096, et moins que 1 Go. Par défaut, c'est 0. Cette variable a été ajoutée en MySQL 4.0.14. Détails d'implémentation de la réplication .

`max_seeks_for_key`
: La limite de recherche lors de recherche de lignes basées sur un index. L'optimiseur MySQL va supposer que lorsque vous recherchez des lignes dans une table en analysant les index, vous n'aller pas générer plus de ce nombre de recherches, indépendamment de la cardinalité de la clé. En donnant une valeur faible à cette variable (100 ?) vous pouvez forcer MySQL a préférer les scans d'index plutôt que les scans de tables. Cette variable a été ajoutée en MySQL 4.0.14.

`max_sort_length`
: Le nombre d'octets à utiliser lors du tri des colonnes de type `BLOB` et `TEXT` . Seuls les `max_sort_length` octets de chaque valeur seront utilisés pour le tri. Le reste est ignoré.

`max_tmp_tables`

Cette option ne fait encore rien. Le nombre maximal de tables temporaires qu'un client peut garder ouverte en même temps.

`max_user_connections`
: Le nombre maximum de connexions actives pour un utilisateur particulier (0 = pas de limite). Cette variable a été ajoutée en MySQL 3.23.34.

`max_write_lock_count`
: Après `max_write_lock_count` pose de verrou en écriture, autorise quelques verrous en lecture. Cette variable a été ajoutée en MySQL 3.23.7.

`myisam_data_pointer_size`
: La taille par défaut du pointeur à utiliser avec `CREATE TABLE` pour les tables `MyISAM` lorsque qu'aucune option `MAX_ROWS` n'est spécifiée. Cette variable ne peut pas être inférieure à 2 ni supérieure à 8. La valeur par défaut est de 4. Cette variable a été ajoutée en MySQL 4.1.2. Erreur `The table is full` .

`myisam_max_extra_sort_file_size`
: Si un fichier temporaire est utilisé pour créer rapidement un fichier d'index pour une table `MyISAM` est plus grand que la valeur de cette variable, alors préfère la méthode du cache de clé. C'est surtout utilisé pour forcer les grands index à utiliser la méthode plus lente du cache de clé pour créer l'index. Cette variable a été ajoutée en MySQL 3.23.37. **Note** : la valeur donnée est en megaoctets avant 4.0.3 et en octets après.

`myisam_max_sort_file_size`
: La taille maximale du fichier temporaire que MySQL est autorisé à utiliser durant la recréation des fichiers d'index (avec `REPAIR` , `ALTER TABLE` ou `LOAD DATA INFILE`). Si la taille du fichier dépasse `myisam_max_sort_file_size` , l'index sera créé avec un cache de clé (plus lent). Cette variable a été ajoutée en MySQL 3.23.37. **Note :** ce paramètre est spécifié en megaoctets avant la version 4.0.3 et en octets depuis.

`myisam_recover_options`
: La valeur de l'option `--myisam-recover` . Cette variable a été ajoutée en MySQL 3.23.36.

`myisam_repair_threads`
: Si cette valeur est plus grande que 1, les index des tables `MyISAM` durant un processus de `Repair by sorting` seront créés en parallèle : chaque index avec son propre thread. **Note :** les réparations multi−threadées sont encore en développement, et en qualité **alpha** . Cette variable a été ajoutée en MySQL 4.0.13.

`myisam_sort_buffer_size`
: Le buffer qui est alloués lors du tri d'index avec la commande `REPAIR` ou lors de la création d'index avec `CREATE INDEX` ou `ALTER TABLE` . Cette variable a été ajoutée en MySQL 3.23.16.

`named_pipe`
: Sous Windows, indique si le serveur supporte les connexions via les pipes nommés. Cette variable a été ajoutée en MySQL 3.23.50.

`net_buffer_length`
: Le buffer de communication est remis à zéro entre deux requêtes. Cela ne devrait pas être modifié, mais si vous avez très peu de mémoire, vous pouvez le remettre à la taille présumée de la requête (c'est à dire, la taille de requête envoyé par le client. Si la requête dépasse cette taille, le buffer est automatiquement agrandi jusqu'à `max_allowed_packet` octets).

`net_read_timeout`
: Nombre de secondes d'attente des dernières données, avant d'annuler la lecture. Notez que lorsque nous n'attendons pas de données d'une connexion, le délai d'expiration est donné par `write_timeout` . Voir aussi `slave_net_timeout` . Cette variable a été ajoutée en MySQL 3.23.20.

`net_retry_count`
: Si une lecture sur une port de communication est interrompu, `net_retry_count` tentatives sont faites avant d'abandonner. Cette valeur doit être particulièrement grande pour `FreeBSD` car les interruptions internes sont envoyés à tous les threads. Cette variable a été ajoutée en MySQL 3.23.7.

`net_write_timeout`
: Nombre de secondes d'attente pour qu'un bloc soit envoyé à une connexion, avant d'annuler l'écriture. Voir aussi `net_read_timeout` . Cette variable a été ajoutée en MySQL 3.23.20.

`open_files_limit`

Si `open_files_limit` ne vaut pas 0, alors `mysqld` va utiliser cette valeur pour réserver des pointeurs de fichiers à utiliser avec `setrlimit()` . Si cette valeur est 0, alors `mysqld` va réserver `max_connections*5` ou `max_connections + table_cache*2` (le plus grand des deux) pointeurs de fichiers. Vous devriez augmenter cette valeur si `mysqld` vous donne des erreurs du type `'Too many open files'` .

`pid_file`
: Le chemin vers le fichier de processus (`PID`). La valeur de l'option `--pid-file` . Cette variable a été ajoutée en MySQL 3.23.23.

`port`
: Le port de connexion sur lequel le serveur attend les connexions TCP/IP. Cette variable peut être spécifiée avec `--port` .

`protocol_version`
: La version du protocole utilisé par le serveur MySQL. Cette variable a été ajoutée en MySQL 3.23.18.

`query_alloc_block_size`
: Taille des blocs de mémoire alloués pour les objets durant l'analyse et la préparation des requêtes. Si vous avez un problème avec la fragmentation de la mémoire, cela peut être utile d'augmenter cette valeur. Cette variable a été ajoutée en MySQL 4.0.16.

`query_cache_limit`
: Ne met pas en cache les résultats qui sont plus grands que `query_cache_limit` . Par défaut, 1 Mo. Cette variable a été ajoutée en MySQL 4.0.1.

`query_cache_min_res_unit`
: La taille minimale pour les blocs alloués par le cache de requête. La valeur par défaut est de 4 ko. Des informations sur l'optimisation de cette variable sont données dans la section Configuration du cache de requêtes . Cette variable a été ajoutée en MySQL 4.1.

`query_cache_size`
: La mémoire allouée pour stocker les résultats des vieilles requêtes. Si `query_cache_size` vaut 0, le cache de requête est désactivé (par défaut). Cette variable a été ajoutée en MySQL 4.0.1.

`query_cache_type`
: Choisit le type de cache de requête. Modifier la variable `GLOBAL` modifie le cache pour tous les clients. Les clients peuvent modifier la variable de `SESSION` pour l'adapter à leur utilisation. `query_cache_type` peut prendre les valeurs numériques suivantes :

Option	Description
0 or OFF	Ne met pas en cache les résultats. Notez que cela ne va pas libérer le buffer de requête. Pour cela, il faut donner à query_cache_size la valeur de 0.
1 or ON	Met en cache tous les résultats exceptés les requêtes SELECT SQL_NO_CACHE ... .
2 or DEMAND	Met en cache uniquement les requêtes SELECT SQL_CACHE ... .

Cette variable a été ajoutée en MySQL 4.0.3.

`query_cache_wlock_invalidate`
: Normalement, lorsqu'un client pose un verrou `WRITE` sur une table `MyISAM` , les autres clients ne sont pas empêchés d'émettre des requêtes sur la table, si le résultat est déjà en cache. En donnant la valeur de 1 à cette variable, le verrou `WRITE` empêchera toutes les requêtes qui feront référence à cette table. Cela force les autres clients à attendre que le verrou se libère. Cette variable a été ajoutée en MySQL 4.0.19.

`query_prealloc_size`
: La taille du buffer persistant utilisé pour l'analyse des requêtes et leur exécution. Ce buffer n'est pas libéré entre deux requêtes. Si vous manipulez des requêtes complexes, une valeur plus grande pour `query_prealloc_size` sera plus utile pour améliorer les performances, car elle peut réduire les allocations complémentaires de mémoire durant l'exécution des requêtes. Cette variable a été ajoutée en MySQL 4.0.16.

`range_alloc_block_size`
: La taille de bloc qui est alloué lors de l'optimisation d'intervalle. Cette variable a été ajoutée en MySQL 4.0.16.

`read_buffer_size`
: Chaque thread qui fait une recherche séquentielle alloue un buffer de cette taille pour son scan. Si vous faites de nombreux scan séquentiels, vous pourriez avoir besoin d'augmenter cette valeur. Cette variable a été ajoutée en MySQL 4.0.3. Auparavant, cette variable s'appelait `record_buffer` .

`read_only`
: Lorsque cette variable vaut `ON` pour un serveur de réplication esclave, cela fait que le serveur ne permet aucune modification, hormis celles de la réplication, ou des utilisateurs ayant le droit de `SUPER` . Cela peut être pratique pour s'assurer qu'un esclave n'accepte aucune modification des clients. Cette variable a été ajoutée en MySQL 4.0.14.

`read_rnd_buffer_size`
: Lors de la lecture des lignes triées, les lignes sont lues dans un buffer, pour éviter les accès disques. En donnant à cette variable une grande valeur, vous améliorerez les performances des clauses `ORDER BY` . Cependant, ce buffer est alloué pour chaque client : il est recommandé de ne pas donner une valeur globale trop importante. Au lieu de cela, modifiez cette valeur si votre client a besoin de faire de gros tris. Cette variable a été ajoutée en MySQL 4.0.3. Auparavant, cette variable s'appelait `record_rnd_buffer` .

`safe_show_database`
: Ne montre pas les bases pour lesquelles un utilisateur n'a pas des droits de bases ou de tables. Cela peut améliorer considérablement la sécurité si vous craignez de voir les utilisateurs découvrir ce que les autres ont mis en place. Voir aussi `skip_show_database` .Cette variable a été supprimée en MySQL 4.0.5. A la place, utilisez le droit `SHOW DATABASES` pour contrôler les accès aux noms des bases de données.

`secure_auth`
: Si le serveur MySQL a été lancé avec l'option `--secure-auth` , il va empêcher les connexions des comptes qui ont un compte au format pre−version 4.1. Dans ce cas, la valeur de cette variable vaut `ON` , sinon, c'est `OFF` .Vous devriez activer cette option si vous voulez empêcher l'utilisation des mots de passe à l'ancien format (et donc, améliorer la sécurité de votre serveur). Cette variable a été ajoutée en MySQL 4.1.1.

  Le démarrage du serveur échouera avec une erreur si cette option est activée, mais que la table de droits est toujours au format pre−version 4.1.

  Lorsqu'elle est utilisée comme une option du client, le client va refuser de se connecter au serveur si le serveur requiert un mot de passe à l'ancien format.

`server_id`
: La valeur de l'option `--server-id` . Elle sert à la réplication. Cette variable a été ajoutée en MySQL 3.23.26.

`skip_external_locking`
: `skip_locking` vaut `OFF` si `mysqld` utilise le verrouillage externe. Cette variable a été ajoutée en MySQL 4.0.3. Auparavant, cette variable s'appelait `skip_locking` .

`skip_networking`
: `skip_networking` vaut ON si seules les connexions locales (via socket) sont autorisées. Sous Unix, les connexions locales utilisent un fichier de socket Unix. Sous Windows, les connexions locales utilisent les pipes nommés. Sous NetWare, seules les connexions TCP/IP sont supportées, alors ne donnez pas la valeur de `ON` à cette variable. Cette variable a été ajoutée en MySQL 3.22.23.

`skip_show_database`
: `skip_show_database` empêche les utilisateurs d'exécuter des commandes `SHOW DATABASES` si ils n'ont pas les droits de `PROCESS` . Cela peut améliorer la sécurité si vous craignez de voir les utilisateurs découvrir ce que les autres ont mis en place. Voir aussi `safe_show_database` . Cette variable a été ajoutée en MySQL 3.23.4. Depuis MySQL 4.0.2, son effet dépend aussi du droit `SHOW DATABASES` : si la variable vaut `ON` , la commande `SHOW DATABASES` n'est autorisée qu'aux comptes ayant le droit de `SHOW DATABASES` , et la commande affiche tous les noms de bases. Si la valeur est `OFF` , `SHOW DATABASES` est autorisé à tous les utilisateurs, mais il n'affichera que les noms de bases de données pour lesquelles l'utilisateur a le droit de `SHOW DATABASES` ou un droit quelconque dans la base.

`slave_net_timeout`
: Nombre de secondes d'attente de données en lecture ou écriture sur une connexion maître / esclave avant d'annuler. Cette variable a été ajoutée en MySQL 3.23.40.

`slow_launch_time`

Si la création du thread prend plus de `slow_launch_time` secondes, le compteur de threads lents `Slow_launch_threads` sera incrémenté. Cette variable a été ajoutée en MySQL 3.23.15.

`socket`
: La socket Unix utilisé par le serveur. Sous Unix, c'est le fichier de socket Unix, pour les connexions locales. Sous Windows, c'est le nom du pipe nommé, pour les connexions locales.

`sort_buffer_size`
: Chaque thread qui doit faire un tri alloue un buffer de cette taille. Augmentez cette taille pour accélérer les clauses `ORDER BY` ou `GROUP BY` . Où MySQL conserve les fichiers temporaires .

`sql_mode`
: Le mode SQL courant. Cette variable a été ajoutée en MySQL 3.23.41. Le mode SQL du serveur .

`storage_engine`
: Cette variable est un synonyme de `table_type` . Cette variable a été ajoutée en MySQL 4.1.2.

`table_cache`
: Le nombre de tables ouvertes pour tous les threads réunis. En augmentant cette valeur, vous augmentez le nombre de pointeurs de fichiers que `mysqld` utilise. Vous pouvez vérifier si vous avez besoin de plus de cache de tables en étudiant la valeur de la variable `Opened_tables` . Variables de statut du serveur . Si cette variable est grande, c'est que vous ne faites pas souvent de commandes `FLUSH TABLES` (qui force les tables à se recharger), vous devrez alors augmenter cette valeur.Pour plus d'informations sur le cache de table, voyez Comment MySQL ouvre et ferme les tables .

`table_type`
: Le type de table par défaut. Pour configurer le type de table par défaut au démarrage, utilisez `--default-table-type` . Cette variable a été ajoutée en MySQL 3.23.0. Options de ligne de commande `mysqld` .

`thread_cache_size`
: Combien de threads nous allons conserver en cache pour réutilisation. Lorsqu'un client se déconnecte, les threads du client sont mis en cache s'il n'y en a pas déjà `thread_cache_size` de conservé. Tous les nouveaux threads sont d'abord prélevé dans le cache, et uniquement lorsque le cache est vide, un nouveau thread est créé. Cette variable peut vous permettre d'améliorer les performances si vous avez de nombreuses connexions. Normalement, `thread_cache_size` ne donne pas d'amélioration notable si vous avez une bonne implémentation des threads. En examinant la différence entre les variables de statut `Connections` et `Threads_created` vous pouvez voir comment votre système de cache de threads est efficace. (Variables de statut du serveur pour plus de détails) Cette variable a été ajoutée en MySQL 3.23.16.

`thread_concurrency`
: Sous Solaris, `mysqld` va appeler `thr_setconcurrency()` avec cette valeur. `thr_setconcurrency()` permet à l'application de donner au système de threads une indication sur le nombre de threads qui seront exécutés en même temps. Cette variable a été ajoutée en MySQL 3.23.7.

`thread_stack`
: La taille de la pile pour chaque thread. De nombreuses limites détectées par `crash-me` sont dépendantes de cette valeur. La valeur par défaut est suffisamment grande pour des opérations normales. La suite de tests comparatifs de MySQL .

`timezone`
: Le fuseau horaire du serveur. Cette option prend la valeur de la variable d'environnement `TZ` lorsque `mysqld` est démarré. Elle peut aussi être modifiée avec l'argument `--timezone` de `mysqld_safe` . Cette variable a été ajoutée en MySQL 3.23.15. Les problèmes de fuseau horaire .

`tmp_table_size`
: Si une table temporaire en mémoire excède cette taille, MySQL va automatiquement la convertir en une table `MyISAM` sur le disque. Augmentez la valeur de `tmp_table_size` si vous faites un usage intensif de la clause `GROUP BY` et que vous avez beaucoup de

mémoire.

`tmpdir`
: Le dossier utilisé pour les fichiers temporaires et les tables temporaires. Depuis MySQL 4.1, cette variable peut prendre une liste de différents chemins, qui sont utilisés circulairement. Les chemins doivent être séparés par des deux points (`':'`) sous Unix et des points−virgules (`';'`) sous Windows, NetWare et OS/2.Cette fonctionnalité permet de répartir la charge en plusieurs disques. Si le serveur MySQL sert d'esclave de réplication, vous ne devez pas faire pointer `tmpdir` sur un dossier en mémoire, car il sera vidé si le serveur redémarre. Un esclave de réplication doit pouvoir reprendre ses fichiers temporaires pour que la réplication puisse redémarrer, en incluant les tables temporaires et les opérations de `LOAD DATA INFILE` . Si les fichiers du dossier temporaire sont perdus au redémarrage, la réplication s'arrêtera.

Cette variable a été ajoutée en MySQL 3.22.4.

`transaction_alloc_block_size`
: La taille de bloc d'allocation de mémoire pour le stockage des requêtes qui font partie d'une transaction, qui sera stockée dans le log binaire durant une validation. Cette variable a été ajoutée en MySQL 4.0.16.

`transaction_prealloc_size`
: Le buffer persistant pour `transaction_alloc_blocks` , qui n'est pas libéré entre deux requêtes. En rendant cet buffer ``assez grand'' pour accommoder toutes les requêtes dans une transaction classique, vous pouvez éviter de nombreux appels `malloc()` . Cette variable a été ajoutée en MySQL 4.0.16.

`tx_isolation`
: Le niveau par défaut d'isolation de transactions. Cette variable a été ajoutée en MySQL 4.0.3.

`version`
: Le numéro de version du serveur.

`wait_timeout`
: Le nombre de secondes d'attente du serveur sur une connexion non interactive avant de la refermer.

Lors du démarrage du thread, `SESSION.WAIT_TIMEOUT` est initialisé avec `GLOBAL.WAIT_TIMEOUT` ou `GLOBAL.INTERACTIVE_TIMEOUT` , suivant le type de client (tel que défini par l'option de connexion `CLIENT_INTERACTIVE`). Voir aussi `interactive_timeout` .

5.2.3.1 Variables système dynamiques

Depuis MySQL version 4.0.3, de nombreuses variables système sont dynamiques, et peuvent être modifiées durant l'exécution avec les commandes `SET GLOBAL` ou `SET SESSION` . Vous pouvez aussi sélectionner leur valeurs avec `SELECT` . Variables système .

La table suivante montre la liste complète de toutes les variables dynamiques. La dernière colonne indique si les options `GLOBAL` ou `SESSION` , ou les deux, s'appliquent.

Nom de la variable	Type de valeur	Application
autocommit	boolean	SESSION
big_tables	boolean	SESSION
binlog_cache_size	numeric	GLOBAL
bulk_insert_buffer_size	numeric	GLOBAL | SESSION
character_set_client	string	GLOBAL | SESSION
character_set_connection	string	GLOBAL | SESSION
character_set_results	string	GLOBAL | SESSION
character_set_server	string	GLOBAL | SESSION
collation_connection	string	GLOBAL | SESSION
collation_server	string	GLOBAL | SESSION
concurrent_insert	boolean	GLOBAL
connect_timeout	numeric	GLOBAL
convert_character_set	string	GLOBAL | SESSION

default_week_format	numeric	GLOBAL | SESSION
delay_key_write	OFF | ON | ALL	GLOBAL
delayed_insert_limit	numeric	GLOBAL
delayed_insert_timeout	numeric	GLOBAL
delayed_queue_size	numeric	GLOBAL
error_count	numeric	SESSION
flush	boolean	GLOBAL
flush_time	numeric	GLOBAL
foreign_key_checks	boolean	SESSION
ft_boolean_syntax	numeric	GLOBAL
group_concat_max_len	numeric	GLOBAL | SESSION
identity	numeric	SESSION
insert_id	boolean	SESSION
interactive_timeout	numeric	GLOBAL | SESSION
join_buffer_size	numeric	GLOBAL | SESSION
key_buffer_size	numeric	GLOBAL
last_insert_id	numeric	SESSION
local_infile	boolean	GLOBAL
log_warnings	boolean	GLOBAL
long_query_time	numeric	GLOBAL | SESSION
low_priority_updates	boolean	GLOBAL | SESSION
max_allowed_packet	numeric	GLOBAL | SESSION
max_binlog_cache_size	numeric	GLOBAL
max_binlog_size	numeric	GLOBAL
max_connect_errors	numeric	GLOBAL
max_connections	numeric	GLOBAL
max_delayed_threads	numeric	GLOBAL
max_error_count	numeric	GLOBAL | SESSION
max_heap_table_size	numeric	GLOBAL | SESSION
max_insert_delayed_threads	numeric	GLOBAL
max_join_size	numeric	GLOBAL | SESSION
max_relay_log_size	numeric	GLOBAL
max_seeks_for_key	numeric	GLOBAL | SESSION
max_sort_length	numeric	GLOBAL | SESSION
max_tmp_tables	numeric	GLOBAL
max_user_connections	numeric	GLOBAL
max_write_lock_count	numeric	GLOBAL
myisam_max_extra_sort_file_size	numeric	GLOBAL | SESSION
myisam_max_sort_file_size	numeric	GLOBAL | SESSION
myisam_repair_threads	numeric	GLOBAL | SESSION
myisam_sort_buffer_size	numeric	GLOBAL | SESSION
net_buffer_length	numeric	GLOBAL | SESSION
net_read_timeout	numeric	GLOBAL | SESSION
net_retry_count	numeric	GLOBAL | SESSION
net_write_timeout	numeric	GLOBAL | SESSION
query_alloc_block_size	numeric	GLOBAL | SESSION
query_cache_limit	numeric	GLOBAL
query_cache_size	numeric	GLOBAL
query_cache_type	enumeration	GLOBAL | SESSION
query_cache_wlock_invalidate	boolean	GLOBAL | SESSION
query_prealloc_size	numeric	GLOBAL | SESSION
range_alloc_block_size	numeric	GLOBAL | SESSION
read_buffer_size	numeric	GLOBAL | SESSION
read_only	numeric	GLOBAL
read_rnd_buffer_size	numeric	GLOBAL | SESSION
rpl_recovery_rank	numeric	GLOBAL
safe_show_database	boolean	GLOBAL
server_id	numeric	GLOBAL
slave_compressed_protocol	boolean	GLOBAL
slave_net_timeout	numeric	GLOBAL
slow_launch_time	numeric	GLOBAL
sort_buffer_size	numeric	GLOBAL | SESSION
sql_auto_is_null	boolean	SESSION
sql_big_selects	boolean	SESSION
sql_big_tables	boolean	SESSION
sql_buffer_result	boolean	SESSION
sql_log_bin	boolean	SESSION
sql_log_off	boolean	SESSION
sql_log_update	boolean	SESSION
sql_low_priority_updates	boolean	GLOBAL | SESSION
sql_max_join_size	numeric	GLOBAL | SESSION

sql_quote_show_create	boolean	SESSION
sql_safe_updates	boolean	SESSION
sql_select_limit	numeric	SESSION
sql_slave_skip_counter	numeric	GLOBAL
sql_warnings	boolean	SESSION
storage_engine	enumeration	GLOBAL | SESSION
table_cache	numeric	GLOBAL
table_type	enumeration	GLOBAL | SESSION
thread_cache_size	numeric	GLOBAL
timestamp	boolean	SESSION
tmp_table_size	enumeration	GLOBAL | SESSION
transaction_alloc_block_size	numeric	GLOBAL | SESSION
transaction_prealloc_size	numeric	GLOBAL | SESSION
tx_isolation	enumeration	GLOBAL | SESSION
unique_checks	boolean	SESSION
wait_timeout	numeric	GLOBAL | SESSION
warning_count	numeric	SESSION

Les variables qui sont marquées comme ``string'' prennent une valeur de chaîne de caractères. Les variables qui sont marquées comme ``numeric'' prennent un nombre. Les variables qui sont marquées comme ``boolean'' peuvent prendre 0 ou 1, `ON` ou `OFF` . Les variables qui sont marquées comme ``enumeration'' doivent normalement prendre l'une des valeurs possible de cette variable, mais elles peuvent aussi prendre le numéro de l'élément dans l'énumération. Pour les systèmes à énumération, la première énumération est 0. Cela est différent des colonnes de type `ENUM` , pour qui la première valeur est la 1.

5.2.4 Variables de statut du serveur

`SHOW STATUS` affiche des informations sur le statut du serveur (comme par exemple, `mysqladmin extended-status`). L'affichage ressemble à ce qui est affiché ci−dessous, mais les valeurs différeront sûrement de votre propre serveur.

```
mysql> SHOW STATUS;
+--------------------------+------------+
| Variable_name            | Value      |
+--------------------------+------------+
| Aborted_clients          | 0          |
| Aborted_connects         | 0          |
| Bytes_received           | 155372598  |
| Bytes_sent               | 1176560426 |
| Connections              | 30023      |
| Created_tmp_disk_tables  | 0          |
| Created_tmp_files        | 60         |
| Created_tmp_tables       | 8340       |
| Delayed_errors           | 0          |
| Delayed_insert_threads   | 0          |
| Delayed_writes           | 0          |
| Flush_commands           | 1          |
| Handler_delete           | 462604     |
| Handler_read_first       | 105881     |
| Handler_read_key         | 27820558   |
| Handler_read_next        | 390681754  |
| Handler_read_prev        | 6022500    |
| Handler_read_rnd         | 30546748   |
| Handler_read_rnd_next    | 246216530  |
| Handler_update           | 16945404   |
| Handler_write            | 60356676   |
| Key_blocks_used          | 14955      |
| Key_read_requests        | 96854827   |
| Key_reads                | 162040     |
| Key_write_requests       | 7589728    |
| Key_writes               | 3813196    |
| Max_used_connections     | 0          |
| Not_flushed_delayed_rows | 0          |
| Not_flushed_key_blocks   | 0          |
| Open_files               | 2          |
| Open_streams             | 0          |
| Open_tables              | 1          |
| Opened_tables            | 44600      |
| Qcache_free_blocks       | 36         |
| Qcache_free_memory       | 138488     |
| Qcache_hits              | 79570      |
| Qcache_inserts           | 27087      |
| Qcache_lowmem_prunes     | 3114       |
| Qcache_not_cached        | 22989      |
```

```
| Qcache_queries_in_cache  | 415        |
| Qcache_total_blocks      | 912        |
| Questions                | 2026873    |
| Select_full_join         | 0          |
| Select_full_range_join   | 0          |
| Select_range             | 99646      |
| Select_range_check       | 0          |
| Select_scan              | 30802      |
| Slave_open_temp_tables   | 0          |
| Slave_running            | OFF        |
| Slow_launch_threads      | 0          |
| Slow_queries             | 0          |
| Sort_merge_passes        | 30         |
| Sort_range               | 500        |
| Sort_rows                | 30296250   |
| Sort_scan                | 4650       |
| Table_locks_immediate    | 1920382    |
| Table_locks_waited       | 0          |
| Threads_cached           | 0          |
| Threads_connected        | 1          |
| Threads_created          | 30022      |
| Threads_running          | 1          |
| Uptime                   | 80380      |
+--------------------------+------------+
```

De nombreuses variables de statut sont remises à 0 par la commande `FLUSH STATUS` .Les variables de statut ont les significations suivantes. Les variables compteur de commande `Com_xxx` ont été ajoutées en MySQL 3.23.47. Les variables de cache de requêtes `Qcache_xxx` ont été ajoutées en MySQL 4.0.1. Sinon, les variables sans versions sont présentes depuis MySQL 3.22.

`Aborted_clients`
: Nombre de connexions annulées parce que le client est mort sans se déconnecter correctement. Erreurs de communication / connexion annulée .

`Aborted_connects`
: Nombre de tentatives de connexions au serveur MySQL qui ont échouées. Erreurs de communication / connexion annulée .

`Binlog_cache_disk_use`
: Le nombre de transactions qui ont utilisé le cache de log binaire mais qui ont dépassé la taille de `binlog_cache_size` et ont finalement utilisé un fichier temporaire pour stocker les commandes de la transaction. Cette variable a été ajoutée en MySQL 4.1.2.

`Binlog_cache_use`
: Le nombre de transactions qui ont utilisé le cache de log binaire temporaire. Cette variable a été ajoutée en MySQL 4.1.2.

`Bytes_received`
: Nombre d'octets reçu de tous les clients. Cette variable a été ajoutée en MySQL 3.23.7.

`Bytes_sent`
: Nombre d'octets envoyés à tous les clients. Cette variable a été ajoutée en MySQL 3.23.7.

`Com_xxx`
: Nombre d'exécution de chaque commande `xxx` . Il y a une variable par type de commande. Par exemple, `Com_delete` et `Com_insert` comptent respectivement les commandes `DELETE` et `INSERT` .

`Connections`
: Nombre de tentatives de connexions au serveur MySQL, réussies ou pas.

`Created_tmp_disk_tables`
: Nombre de tables temporaires implicites créées sur le disque lors d'exécutions de commandes. Cette variable a été ajoutée en MySQL 3.23.24.

`Created_tmp_files`
: Combien de fichiers temporaires `mysqld` a créé. Si `Created_tmp_disk_tables` est grand, augmentez la taille de `tmp_table_size` pour que les tables temporaires restent plus souvent en mémoire.

`Created_tmp_tables`
: Nombre de tables temporaires implicites créées en mémoire lors d'exécutions de commandes. Cette variable a été ajoutée en MySQL 3.23.28.

`Delayed_errors`
: Nombre de lignes écrites avec `INSERT DELAYED` pour lesquelles des erreurs sont survenues (probablement une erreur de doublons (`duplicate key`)).

`Delayed_insert_threads`

Nombre de gestionnaires d'insertion retardées sont en cours d'utilisation.

`Delayed_writes`
: Nombre de lignes écrites avec `INSERT DELAYED` .

`Flush_commands`
: Nombre de commandes `FLUSH` .

`Handler_commit`
: Nombre de commandes internes `COMMIT` . Cette variable a été ajoutée en MySQL 4.0.2.

`Handler_delete`
: Nombre de fois qu'une ligne a été effacées dans une table.

`Handler_read_first`
: Nombre de fois que la première ligne a été lue dans un index. Si ce chiffre est haut, c'est que le serveur fait de nombreuses recherches par analyse complète de la table, par exemple `SELECT col1 FROM foo` , en supposant que `col1` est indexé.

`Handler_read_key`
: Nombre de requête de lecture de ligne basées sur une clé. Si ce chiffre est grand, c'est une bonne indication de l'indexation correcte de vos tables.

`Handler_read_next`
: Nombre de requête de lecture de la ligne suivante en ordre. Cela sera augmenté si vous listez une colonne avec une contrainte d'intervalle. Cette valeur sera aussi incrémentée si vous effectuez un scan d'index.

`Handler_read_prev`
: Nombre de requête de lecture de la clé précédente, dans l'ordre. C'est souvent utilisé pour optimiser les clauses `ORDER BY ... DESC` . Cette variable a été ajoutée en MySQL 3.23.6.

`Handler_read_rnd_next`
: Nombre de requêtes de lecture de la prochaine ligne dans le fichier de données. Ce chiffre sera grand si vous faîtes de nombreux scans de tables. Généralement, cela indique que vos requêtes ne sont pas écrites pour profiter des index que vous avez mis en place.

`Handler_read_rnd`
: Nombre de lecture d'une ligne basée sur une position fixe. Ce chiffre sera grand si vous effectuez de nombreuses requêtes qui réclament le tri du résultat.

`Handler_rollback`
: Nombre de commandes internes `ROLLBACK` .

`Handler_update`
: Nombre de requête de modification d'une ligne dans une table.

`Handler_write`
: Nombre de requête pour insérer une ligne dans une table.

`Key_blocks_used`
: Nombre de blocs utilisés dans un cache de clé. Vous pouvez utiliser cette valeur pour déterminer l'occupation du cache de clé : voyez la discussion de `key_buffer_size` dans Variables sytème du serveur .

`Key_read_requests`
: Nombre de requêtes de lecture d'un bloc de clé dans le cache.

`Key_reads`
: Nombre de lecture physique d'un bloc de clé sur le disque. Si `Key_reads` est grand, alors votre valeur pour `key_buffer_size` est probablement trop petite. Le ratio peut être calculé avec `Key_reads` / `Key_read_requests` .

`Key_write_requests`
: Nombre de requêtes d'écriture d'un bloc de clé dans le cache.

`Key_writes`
: Nombre d'écriture physiques de bloc de clé sur le disque.

`Max_used_connections`
: Nombre maximum de connexions utilisées simultanément.

`Not_flushed_delayed_rows`
: Nombre de lignes en attente d'écriture dans les listes `INSERT DELAY` .

`Not_flushed_key_blocks`
: Nombre de blocs de clés dans le cache de clés, qui ont été modifiées, mais pas encore écrites sur le disque.

`Open_files`
: Nombre de fichiers ouverts.

`Open_streams`
: Nombre de flux ouverts (utilisés généralement pour les logs).

`Open_tables`
: Nombre de tables ouvertes.

`Opened_tables`

Nombre de tables qui ont été ouvertes. Si `Opened_tables` est grand, votre valeur pour `table_cache` est probablement trop petite.

`Qcache_free_blocks`
: Le nombre de blocs de mémoire libre dans le cache de requête.

`Qcache_free_memory`
: La quantité de mémoire libre dans le cache de requête.

`Qcache_hits`
: Le nombre de sollicitations du cache.

`Qcache_inserts`
: Le nombre de requêtes ajoutées dans le cache.

`Qcache_lowmem_prunes`
: Le nombre de requêtes qui ont été effacées du cache, pour libérer de la place.

`Qcache_not_cached`
: Le nombre de requêtes non−cachées (elles ne peuvent pas être mises en cache, ou à cause de `query_cache_type`).

`Qcache_queries_in_cache`
: Le nombre de requêtes enregistrées dans le cache.

`Qcache_total_blocks`
: Le nombre total de blocs dans le cache de requêtes.

`Questions`
: Nombre de requêtes envoyées au serveur.

`Rpl_status`
: Statut de la réplication sans erreur (réservé pour utilisation ultérieure).

`Select_full_join`
: Nombre de jointures sans clé (si cette variable vaut 0, vous devriez vérifier soigneusement les index de vos tables). Cette variable a été ajoutée en MySQL 3.23.25.

`Select_full_range_join`
: Nombre de jointures où une recherche d'intervalle a été utilisée. Cette variable a été ajoutée en MySQL 3.23.25.

`Select_range_check`
: Nombre de jointures sans clé, où l'utilisation de clé a été vérifiée après chaque ligne (si cette variable vaut 0, vous devriez vérifier soigneusement les index de vos tables). Cette variable a été ajoutée en MySQL 3.23.25.

`Select_range`
: Nombre de jointures où une recherche d'intervalle a été utilisée sur la première table. (Ce n'est généralement pas important, même si cette valeur est importante). Cette variable a été ajoutée en MySQL 3.23.25.

`Select_scan`
: Nombre de jointures où la première table a été totalement analysée. Cette variable a été ajoutée en MySQL 3.23.25.

`Slave_open_temp_tables`
: Nombre de tables temporaires actuellement utilisée par le thread esclave. Cette variable a été ajoutée en MySQL 3.23.29.

`Slave_running`
: Cette variable vaut `ON` si ce serveur est un esclave connecté au maître. Cette variable a été ajoutée en MySQL 3.23.16.

`Slow_launch_threads`
: Nombre de threads qui ont pris plus de `slow_launch_time` secondes pour être créés. Cette variable a été ajoutée en MySQL 3.23.15.

`Slow_queries`
: Nombre de requêtes qui ont pris plus de `long_query_time` pour s'exécuter. Le log des requêtes lentes .

`Sort_merge_passes`
: Nombre de passes que l'algorithme de tri a du faire. Si cette valeur est grande, vous devriez vérifier la taille de `sort_buffer` .

`Sort_range`
: Nombre de tris qui ont été fait sur des intervalles.

`Sort_rows`
: Nombre de lignes triées.

`Sort_scan`
: Nombre de tris qui ont été fait en analysant la table.

`ssl_xxx`
: Variables utilisées par SSL; Réservée pour utilisation ultérieure. Ces variables ont été ajoutées en MySQL 4.0.0.

`Table_locks_immediate`

Nombre de fois que la table a reçu immédiatement un verrou. Disponible depuis 3.23.33. Cette variable a été ajoutée en MySQL 3.23.33.

`Table_locks_waited`
: Nombre de fois qu'une table n'a pu recevoir de verrou immédiatement, et qu'il a fallu attendre. Si ce chiffre est haut, vous avez des problèmes de performance, et vous devriez optimiser vos requêtes, couper vos tables en deux, ou utiliser la réplication. Disponible depuis la version 3.23.33. Cette variable a été ajoutée en MySQL 3.23.33.

`Threads_cached`
: Nombre de threads dans le cache de thread. Cette variable a été ajoutée en MySQL 3.23.17.

`Threads_connected`
: Nombre de connexions actuellement ouvertes.

`Threads_created`
: Nombre de threads créés pour gérer les connexions. Si `Threads_created` est grand, vous pouvez augmenter la valeur de

`Threads_running`
: Nombre de threads qui ne dorment pas.

`Uptime`
: Durée de vie du serveur, en secondes depuis le redémarrage.

5.3 Sécurité générale du serveur

Cette section décrit certaines règles générales de sécurité a bien connaître pour rendre votre installation MySQL plus sécuritaire contre des attaques ou des erreurs de manipulations. Pour des informations sur le contrôle d'accès à MySQL, voyez Sécurité générale et accès à MySQL .

5.3.1 Guide de sécurité

Tous ceux qui utilisent MySQL avec un serveur connecté à Internet doivent lire cette section, pour éviter les erreurs les plus communes.

En parlant de sécurité, nous devons insister sur la nécessiter de protéger tout le serveur, et non pas juste MySQL, contre tous les types d'attaques : surveillance des communications, usurpation, ré−exécution et dénis de service. Nous ne pouvons pas couvrir tous les aspects de tolérance aux fautes et de disponibilité ici.

MySQL dispose d'un système de sécurité basé sur des listes de contrôle d'accès (`Access Control Lists` , or `ACL`) pour toutes les connexions, requêtes et opérations que l'utilisateur peut faire. Il y a aussi le support des connexions SSL entre le client et le serveur MySQL. De nombreux concepts présentés ici ne sont pas spécifiques à MySQL : le même concept s'applique à de nombreuses applications.

Lorsque vous utilisez MySQL, suivez ces règles aussi souvent que possible :

- **Ne donnez jamais à personne (sauf aux comptes MySQL `root`) accès à la table `user` de la base `mysql` !** C'est primordial. **Le mot de passe chiffré est le vrai mot de passe de MySQL.** Toute personne qui connaît le mot de passe de la table `user` et qui a accès à l'hôte associé **peut facilement se connecter sous le nom de cet utilisateur** .
- Apprenez à fond le système de droits MySQL. Les commandes `GRANT` et `REVOKE` sont utilisées pour contrôler les accès à MySQL. Ne donnez pas plus de droits que nécessaire. Ne donnez jamais de droits à tous les serveurs hôtes.Liste de vérification :
    - Essayez la commande en ligne `mysql -u root` . Si vous pouvez vous connecter, sans donner de mot de passe, vous avez un problème. Toute personne peut se connecter au serveur comme utilisateur `root` avec le maximum de droits! Passez en revue les

instructions d'installation de MySQL, en insistant sur les passages où le mot de passe `root` est configuré. Sécurisation des comptes MySQL initiaux .
  - ♦ Utilisez la commande `SHOW GRANTS` et vérifiez qui a accès à quoi. Puis, utilisez la commande `REVOKE` pour retirer les droits inutiles.
- Ne stockez jamais de mot de passe en clair dans votre base de données. Si votre serveur est compromis, le pirate aura alors la liste complète des mots de passe, et pourra les utiliser. A la place, utilisez `MD5()` , `SHA1()` ou une autre fonction de signature injective.
- Ne choisissez pas vos mots de passe dans un dictionnaire. Il y a des programmes spéciaux pour les rechercher. Même des mots de passe tels que `` `xfish98` '' est très faible. Par contre, `` `duag98` '' est bien mieux : il contient aussi le mot ``fish'' mais décalé d'une touche sur un clavier `QWERTY` . Une autre méthode de génération consiste à prendre la première lettre de chaque mot d'une phrase : ``Maupa'' est issu de ``Marie a un petit agneau.'' C'est facile à retenir, mais difficile à devenir pour un attaquant.
- Investissez dans un coupe−feu. Cela protège de 50% de tous les types d'attaque et vulnérabilité. Placez MySQL derrière le coupe−feu, ou dans une zone démilitarisée (`DMZ`).Liste de vérification :
  - ♦ Essayez de scanner vos portes depuis l'Internet, avec des outils comme `nmap` . MySQL utilise le port 3306 par défaut. Ce port ne doit pas être accessible à tous les serveurs. Une autre méthode simple pour vérifier si le port MySQL est ouvert ou non, est d'essayez la commande suivante depuis une machine distante, où `server_host` est le serveur qui héberge MySQL :

    ```
    shell> telnet server_host 3306
    ```

    Si vous obtenez une connexion et des caractères binaires, le port est ouvert, et il devrait être fermé par votre routeur ou votre coupe−feu, à moins d'avoir une bonne raison pour le garder ouvert. Si `telnet` attend, ou que la connexion est refusée, tout va bien : le port est bloqué.
- Ne faîtes confiance à aucune donnée entrée par les utilisateurs de votre application. Ils peuvent déjouer vos filtres en entrant des séquences spéciales via les formulaires Web, les URL ou tout autre point d'entrée de l'application. Assurez vous que votre application reste sûre si un utilisateur entre une chaîne telle que `` `; DROP DATABASE mysql;` ''. C'est un exemple extrêmement simple, mais il dévoilera un trou de sécurité important. Il engendrera aussi des pertes de données si un pirate, utilisant cette technique, vous attaque.Une erreur courante est de ne protéger que les chaînes de caractères. N'oubliez pas de protéger aussi les valeurs numériques. Si une application génère une requête telle que `SELECT * FROM table WHERE ID=234` où l'utilisateur fournit le `234` , alors ce dernier peut proposer la valeur `234 OR 1=1` pour conduire à la requête `SELECT * FROM table WHERE ID=234 OR 1=1` . Par conséquent, le serveur va lire toutes les lignes de la table. Cela va diffuser toutes les lignes de votre application, et générer un trafic excessif. Pour vous prémunir contre ce type d'attaque, ajoutez toujours des guillemets autour des constantes numériques : `SELECT * FROM table WHERE ID='234'` . Si un utilisateur entre des informations supplémentaires, elles seront intégrées dans la chaîne. Dans un contexte numérique, MySQL supprimera automatiquement les caractères incompréhensibles.

  Parfois, les gens pensent que si une base de données contient des informations publiques, elle n'a pas besoin d'être défendue. C'est faux. Même si vous pouvez accéder à toutes les lignes de la table, il faut toujours se prémunir contre les dénis de service (par exemple, en utilisant la technique ci−dessus pour générer un trafic excessif). Sinon, votre serveur sera inutilisable.

  Liste de vérification :

  - ♦ Essayez d'entrer des caractères `'` et `"` dans tous vos formulaires Web. Si vous obtenez une erreur MySQL, étudiez immédiatement le problème.
  - ♦ Essayez de modifier une URL dynamique en ajoutant les séquences `%22` (`"`), `%23` (`#`) et `%27` (`'`).

  - Essayez de modifier les types de données des URL dynamiques de numériques en textuels, avec les caractères cités ci−dessus. Votre application doit être sécurisée contre ce type d'attaque.
  - Essayez d'entrer des caractères, des espaces et d'autres symboles spéciaux, autre que des nombres, dans un champ numérique. Votre application devrait supprimer tous ces caractères avant de les passer à MySQL, ou générer une erreur. Passer à MySQL des valeurs non vérifiées est très dangereux.
  - Vérifiez la taille des chaînes avant de les passer à MySQL.
  - Essayez de faire connecter votre application en utilisant un autre nom que celui qui est utilisé pour les tâches d'administration. Ne donnez pas à votre application des droits dont elle n'a pas besoin.
- De nombreuses interfaces de programmation disposent de moyens pour protéger les valeurs. Correctement utilisés, ils évitent aux utilisateurs de l'application de faire passer des caractères qui auront un effet différent de celui attendu :
  - MySQL C API : Utilisez la fonction `mysql_real_escape_string()` .
  - MySQL++ : Utilisez les options `escape` et `quote` dans le flux de requête.
  - PHP : Utilisez la fonction `mysql_escape_string()` , qui est basée sur la fonction C du même nom. Avant PHP 4.0.3, utilisez `addslashes()` .
  - Perl `DBI` : Utilisez la méthode `quote()` ou utilisez les variables de requête.
  - Java JDBC : Utilisez un objet `PreparedStatement` ou utilisez les variables de requête.

  Les autres interfaces ont des fonctionnalités similaires.
- Ne transmettez pas de données déchiffrées sur Internet. Cette information est accessible à tout ceux qui ont le temps et la capacité d'intercepter et d'utiliser ces mots de passe. Utilisez plutôt un protocole sécurisé comme SSL ou SSH. MySQL supporte les connexions SSL depuis la version 4.0.0. SSH peut être utilisé pour créer un tunnel chiffré et compressé de communication.
- Apprenez à utiliser les programmes `tcpdump` et `strings` . Dans la plupart des cas, vous pouvez vérifier si un flux MySQL est chiffré avec la commande suivante :

  ```
  shell> tcpdump −l −i eth0 −w − src or dst port 3306 | strings
  ```

  (Cette commande fonctionne sous Linux, et devrait être adaptée facilement dans les autres systèmes.) Attention : si vous ne voyez pas de données en clair, cela ne signifie pas toujours que les informations sont chiffrées. Si vous avez besoin de haute sécurité, consultez un expert.

5.3.2 Protéger MySQL contre les attaques

Lorsque vous vous connectez à MySQL, vous devriez avoir besoin d'un mot de passe. Ce mot de passe n'est pas transmis en texte clair sur le réseau. La gestion des mots de passe durant l'identification des utilisateurs a été améliorée en MySQL 4.1.1 pour être très sécurisée. Si vous utilisez une vieille version de MySQL, ou si vous utilisez toujours les mots de passe de type pre−4.1.1, l'algorithme de chiffrement n'est pas très fort, quelques efforts permettront à un pirate d'obtenir votre mot de passe s'il est capable de surveiller le trafic entre le client et le serveur. (Voyez Chiffrement des mots de passe en MySQL 4.1 pour une discussion des différentes méthodes de gestions des mots de passe). Si la connexion entre le client et le serveur utilise des réseaux non fiables, il est alors recommandé d'utiliser un tunnel SSH.

Toutes les autres informations sont transférées comme du texte clair, et quiconque surveille la connexion pourra les lire. Si vous souhaitez relever ce niveau de sécurité, il est recommandé d'utiliser le protocole compressé (avec les versions de MySQL 3.22 et plus récentes), pour compliquer considérablement le problème. Pour rendre la communication encore plus sûre, vous pouvez aussi utiliser `ssh` . Vous trouverez une version `Open Source` du client `ssh` sur le site http://www.openssh.org/ , et une version commerciale du client `ssh` sur le site de http://www.ssh.com/ . Avec eux, vous pouvez mettre en place une connexion TCP/IP chiffrée entre le serveur et le client

MySQL.

Si vous utilisez MySQL 4.0, vous pouvez aussi utiliser le support OpenSSL interne. Utilisation des connexions sécurisées .

Pour rendre le système MySQL encore plus sûr, nous vous recommandons de suivre les suggestions suivantes :

- Utilisez des mots de passe pour tous les utilisateurs MySQL. N'oubliez pas que tout le monde peut se connecter avec un nom d'utilisateur quelconque, simplement avec l'option `mysql -u autre_utilisateur nom_de_base` , si `autre_utilisateur` n'a pas de mot de passe. C'est un comportement classique pour les applications client/serveur que le client spécifie son nom d'utilisateur. Il sera plus difficile à un attaquant de pénétrer dans votre serveur si tous les comptes ont un mot de passe.

  Vous pouvez modifier les mots de passe de tous les utilisateurs en modifiant le script `mysql_install_db` avant de l'exécuter, ou vous pouvez modifier seulement le mot de passe du `root` MySQL comme ceci :

  ```
  shell> mysql -u root
  mysql> UPDATE mysql.user SET Password=PASSWORD('nouveau_mot_de_passe')
      -> WHERE User='root';
  mysql> FLUSH PRIVILEGES;
  ```
- N'exécutez jamais le démon MySQL avec l'utilisateur Unix `root` . C'est très dangereux, car tout personne ayant le droit de `FILE` pour créer des fichiers au nom du `root` (par exemple, `~root/.bashrc`). Pour éviter cela, `mysqld` refusera de s'exécuter au nom de `root` à moins que soit précisé l'option `--user=root` .

  `mysqld` peut être exécuté avec un utilisateur ordinaire sans droits particuliers. Vous pouvez aussi créer un utilisateur Unix `mysql` pour rendre cela encore plus sûr. Si vous exécutez `mysqld` sous un autre utilisateur Unix, vous n'avez pas à changer le mot de passe `root` dans la table `user` , car les noms d'utilisateurs MySQL n'ont rien à voir avec les noms d'utilisateurs Unix. Pour démarrer `mysqld` sous un autre nom d'utilisateur Unix, ajoutez la ligne `user` , qui spécifie le nom de l'utilisateur, dans le fichier d'options de `[mysqld] /etc/my.cnf` ou dans le fichier `my.cnf` présent dans le dossier de données du serveur. Par exemple :

  ```
  [mysqld]
  user=mysql
  ```

  Cette ligne va forcer le serveur à démarrer en tant qu'utilisateur `mysql` , même si vous démarrez le serveur manuellement ou avec les scripts `safe_mysqld` , ou `mysql.server` . Pour plus de détails, voyez Comment faire tourner MySQL en tant qu'utilisateur normal .

  Exécuter `mysql` sous un autre compte Unix que `root` ne signifie pas que vous devez changer le nom de `root` dans la table `user` . Les comptes utilisateurs de MySQL n'ont rien à voir avec ceux du compte Unix.
- N'autorisez pas l'utilisation de liens symboliques pour les tables. Cette fonctionnalité peut être désactivée avec l'option `--skip-symbolic-links` . C'est particulièrement important si vous utilisez `mysqld` comme `root` , car tout utilisateur a alors le droit d'écrire des données sur le disque, n'importe où sur le système!! Utiliser les liens symboliques pour les tables sous Unix .
- Vérifiez que l'utilisateur Unix qui exécute `mysqld` est le seul utilisateur avec les droits de lecture et écriture dans le dossier de base de données.
- Ne donnez pas le droit de `PROCESS` à tous les utilisateurs. La liste fournie par `mysqladmin processlist` affiche le texte des requêtes actuellement exécutées, ce qui permet à toute personne pouvant exécuter cette commande de lire des valeurs qui seraient en clair, comme : `UPDATE user SET password=PASSWORD('not_secure')` .

`mysqld` réserve une connexion supplémentaire pour les utilisateurs qui ont le droit de `PROCESS` , afin que le `root` MySQL puisse toujours se connecter et vérifier que tout fonctionne bien, même s'il ne reste plus de connexions libres pour les autres utilisateurs.

Le droit `SUPER` peut être utilisé pour fermer des connexions clients, changer les variables systèmes et contrôler la réplication.

- Ne donnez pas le droit de `FILE` à tous les utilisateurs. Tout utilisateur qui possède ce droit peut écrire un fichier n'importe où sur le serveur, avec les droits hérités du démon `mysqld` ! Pour rendre cela plus sécuritaire, tous les fichiers générés par `SELECT ... INTO OUTFILE` sont lisibles par tous, mais personne ne peut les modifier. Le droit de `FILE` peut aussi être utilisé pour lire n'importe quel fichier accessible en lecture au démon qui fait tourner MySQL. Il devient donc possible, suivant les configurations, d'utiliser la commande `LOAD DATA` sur le fichier `/etc/passwd` pour tout mettre en table, et ensuite le relire avec la commande `SELECT` .
- Si vous ne faites pas confiance à votre DNS, vos pouvez simplement utiliser des adresses IP au lieu des noms d'hôtes. Dans ce cas, soyez très prudents lors de la création de droits qui utilisent des caractères joker.
- Si vous voulez restreindre le nombre de connexions d'un utilisateur, vous pouvez le faire en utilisant la variable `max_user_connections` de `mysqld` . La commande `GRANT` dispose aussi d'option de contrôle des ressources, pour limiter l'utilisation du serveur par un compte utilisateur.

5.3.3 Options de démarrage qui concernent la sécurité

Les options suivantes de `mysqld` affectent la sécurité :

−−local−infile[=(0|1)]
: Si vous utilisez `--local-infile=0` alors vous ne pourrez pas utiliser `LOAD DATA LOCAL INFILE` . Problèmes de sécurité avec `LOAD DATA LOCAL` .

−−safe−show−database
: Avec cette option, la commande `SHOW DATABASES` ne retourne que les bases pour lesquelles l'utilisateur courant a des droits. Depuis la verison 4.0.2, cette option est abandonnée, et ne sert plus à rien (elle est activée par défaut), car désormais, il y a le droit de `SHOW DATABASES` . Syntaxe de `GRANT` et `REVOKE` .

−−safe−user−create
: Si cette option est activée, tout utilisateur ne peut créer d'autres utilisateurs avec les droits de `GRANT` , s'il ne dispose pas des droits d'insertion dans la table `mysql.user` . Si vous voulez donner un accès à un utilisateur pour qu'il puisse créer des utilisateurs avec les droits dont il dispose, vous pouvez lui donner les droits suivants :

```
mysql> GRANT INSERT(user) ON mysql.user TO 'user'&#64;'hostname';
```

: Cela va s'assurer que l'utilisateur ne peut par modifier une colonne directement, mais qu'il peut exécuter la commande `GRANT` sur d'autres utilisateurs.

−−secure−auth
: Interdit l'identification pour les comptes qui ont d'anciens mot de passe (avant la version 4.1). Cette option est valable depuis MySQL 4.1.1.

−−skip−grant−tables
: Cette option force le serveur à ne pas utiliser les tables de droits. Cette option donne donc **tous les droits** à tout le monde sur le serveur ! Vous pouvez forcer un serveur en fonctionnement à reprendre les tables de droits en exécutant la commande `mysqladmin flush-privileges` ou `mysqladmin reload` .)

−−skip−name−resolve
: Les noms d'hôtes ne sont pas résolus. Toutes les valeurs de la colonne `Host` dans les tables de droits doivent être des adresses IP, ou bien `localhost` .

−−skip−networking
: Ne pas accepter les connexions TCP/IP venant du réseau. Toutes les connexions au

serveur `mysqld` doivent être faîtes avec les sockets Unix. Cette option n'existe pas pour les versions antérieures à la 3.23.27, avec les MIT−pthread, car les sockets Unix n'étaient pas supportés par les `MIT-pthreads` à cette époque.

−−skip−show−database
: Ne pas autoriser la commande `SHOW DATABASES` , à moins que l'utilisateur n'ait les droits de `SHOW DATABASES` . Depuis la version 4.0.2, vous n'avez plus besoin de cette option, car les accès sont désormais donnés spécifiquement avec le droit `SHOW DATABASES` .

5.3.4 Problèmes de sécurité avec LOAD DATA LOCAL

La commande `LOAD DATA` peut lire des données sur le serveur hôte, ou bien charger un fichier sur le client, avec l'option `LOCAL` .

Il existe deux problèmes particuliers pour le support de cette commande :

- Comme la lecture du fichier est réalisée depuis le serveur, il est possible théoriquement de créer un serveur MySQL modifié qui pourrait lire n'importe quel fichier de la machine cliente, qui serait accessible à l'utilisateur.
- Dans un environnement web, où les clients se connectent depuis un serveur web, un utilisateur peut se servir de la commande `LOAD DATA LOCAL` pour lire les fichiers qui sont sur le serveur web, et auquel ce dernier a accès (en supposant qu'un utilisateur peut exécuter n'importe quelle commande sur le serveur).

Pour traiter ces problèmes, nous avons changé la gestion des commandes `LOAD DATA LOCAL` depuis MySQL version 3.23.49 et MySQL version 4.0.2 (4.0.13 pour Windows) :

- Par défaut, tous les clients MySQL et les bibliothèques des distributions binaires sont compilées avec l'option `--enable-local-infile` , pour être compatible avec MySQL 3.23.48 et plus ancien.
- Si vous ne configurez pas MySQL avec l'option `--enable-local-infile` , alors `LOAD DATA LOCAL` sera désactivé par tous les clients, à moins que l'option `mysql_options(... MYSQL_OPT_LOCAL_INFILE, 0)` soit activée dans le client. `mysql_options()` .
- Pour le client en ligne de commande `mysql` , `LOAD DATA LOCAL` peut être activé en spécifiant l'option `--local-infile[=1]` , ou désactivé avec `--local-infile=0` .
- Vous pouvez désactiver toutes les commandes `LOAD DATA LOCAL` du serveur MySQL en démarrant `mysqld` avec `--local-infile=0` . Similairement, pour `mysqlimport` , les options `--local` et `-L` active le chargement distant de fichiers. Dans ce cas, il faut que le serveur accepte aussi cette configuration pour que l'opération fonctionne.
- Au cas où `LOAD DATA LOCAL INFILE` est désactivé sur le serveur ou le client, vous obtiendrez le message d'erreur (1148) :
  ```
  The used command is not allowed with this MySQL version
  ```

5.4 Règles de sécurité et droits d'accès au serveur MySQL

MySQL est pourvu d'un système avancé mais non standard de droits. Cette section décrit son fonctionnement.

5.4.1 Rôle du système de privilèges

La fonction première du système de privilèges de MySQL est d'authentifier les utilisateurs se connectant à partir d'un hôte donné, et de leur associer des privilèges sur une base de données comme `SELECT` , `INSERT` , `UPDATE` et `DELETE` .

Les fonctionnalités additionnelles permettent d'avoir un utilisateur anonyme et de contrôler les privilèges pour les fonctions spécifiques à MySQL comme `LOAD DATA INFILE` et les opérations administratives.

5.4.2 Comment fonctionne le système de droits

Le système de droits de MySQL s'assure que les utilisateurs font exactement ce qu'ils sont supposés pouvoir faire dans la base. Lorsque vous vous connectez au serveur, vous identité est déterminée par **l'hôte d'où vous vous connectez** et **le nom d'utilisateur que vous spécifiez** . Le système donne les droits en fonction de votre identité et de **ce que vous voulez faire** .

MySQL considère votre nom d'hôte et d'utilisateur pour vous identifier, car il n'y pas que peu de raisons de supposer que le même nom d'utilisateur appartient à la même personne, quelque soit son point de connexion sur Internet. Par exemple, l'utilisateur `joe` qui se connecte depuis `office.com` n'est pas forcément la même personne que `joe` qui se connecte depuis `elsewhere.com` . MySQL gère cela en vous aidant à distinguer les différents utilisateurs et hôtes qui ont le même nom : vous pourriez donner des droits à `joe` lorsqu'il utilise sa connexion depuis `office.com` , et un autre jeu de droits lorsqu'il se connecte depuis `elsewhere.com` .

Le contrôle d'accès de MySQL se fait en deux étapes :

- Etape 1 : Le serveur vérifie que vous êtes autorisé à vous connecter.
- Etape 2 : En supposant que vous pouvez vous connecter, le serveur vérifie chaque requête que vous soumettez, pour vérifier si vous avez les droits suffisants pour l'exécuter. Par exemple, si vous sélectionnez des droits dans une table, ou effacez une table, le serveur s'assure que vous avez les droits de `SELECT` pour cette table, ou les droits de `DROP` , respectivement.

Si vos droits ont changé (par vous−mêmes ou bien par un administrateur), durant votre connexion, ces changements ne prendront peut être effets qu'à la prochaine requête. Voyez la section Quand est−ce que les droits prennent effets pour plus détails.

Le serveur stocker les droits dans des tables de droits, situées dans la base `mysql` . Le serveur lit le contenu de ces tables en mémoire lorsqu'il démarre, et les relit dans différentes circonstances, détaillées dans Quand est−ce que les droits prennent effets . Le contrôle d'accès se fait par rapport aux tables en mémoire.

Normalement, vous manipulez le contenu des tables indirectement, via les commandes `GRANT` et `REVOKE` pour configurer des comptes et des droits. Syntaxe de `GRANT` et `REVOKE` . La discussion de cette section décrit la structure des tables de droits, et comment elle interagit avec les clients.

Le serveur utilise les tables `user` , `db` et `host` dans la base `mysql` durant les deux étapes. Les champs de cette table sont les suivants :

Table name	**utilisateur**	**base**	**hôte**
Scope fields	Host	Host	Host
	User	Db	Db
	Password	User	
Privilege fields	Select_priv	Select_priv	Select_priv
	Insert_priv	Insert_priv	Insert_priv
	Update_priv	Update_priv	Update_priv
	Delete_priv	Delete_priv	Delete_priv
	Index_priv	Index_priv	Index_priv

Alter_priv	Alter_priv	Alter_priv
Create_priv	Create_priv	Create_priv
Drop_priv	Drop_priv	Drop_priv
Grant_priv	Grant_priv	Grant_priv
References_priv	References_priv	References_priv
Reload_priv		
Shutdown_priv		
Process_priv		
File_priv		
Show_db_priv		
Super_priv		
Create_tmp_table_priv	Create_tmp_table_priv	Create_tmp_table_priv
Lock_tables_priv	Lock_tables_priv	Lock_tables_priv
Execute_priv		
Repl_slave_priv		
Repl_client_priv		
ssl_type		
ssl_cypher		
x509_issuer		
x509_cubject		
max_questions		
max_updates		
max_connections		

Lors de la seconde étape du contrôle d'accès (vérification de la requête), le serveur peut, suivant la requête, consulter aussi les tables tables_priv et columns_priv . Les champs de ces tables sont :

Nom de la table	**tables_priv**	**columns_priv**
Champ	Host	Host
	Db	Db
	User	User
	Table_name	Table_name
		Column_name
Droit	Table_priv	Column_priv
	Column_priv	
Autre champ	Timestamp	Timestamp
	Grantor	

Chaque table de droit contient des champs d'identification et des champs de droits.

- Les champs d'identification déterminent quels utilisateurs correspondent à cette ligne dans la table. Par exemple, une ligne dans la table user avec les valeurs dans les colonnes Host et User de 'thomas.loc.gov' et 'bob' servira à identifier les connexions qui sont faites par l'utilisateur bob depuis l'hôte thomas.loc.gov . De même, une ligne dans la table db avec les valeurs des colonnes Host , User et Db de 'thomas.loc.gov' , 'bob' et 'reports' sera utilisée lorsque l'utilisateur bob se connecte depuis l'hôte thomas.loc.gov pour accéder à la base reports . Les tables tables_priv et columns_priv contiennent en plus des champs indiquant les tables et combinaisons tables et colonnes auxquelles les lignes s'appliquent.
- Les champs de droits indiquent si le droit est donné, c'est à dire si l'opération indiquée peut être exécuté. Le serveur combine les informations dans différentes tables pour former une description complète de l'utilisateur. Les règles utilisées sont décrites dans Contrôle d'accès, étape 2 : vérification des requêtes .

Les champs d'identification sont des chaînes, déclarées comme suit. La valeur par défaut de chacun des champs est la chaîne vide.

Nom de la colonne	**Type**
Host	CHAR(60)
User	CHAR(16)
Password	CHAR(16)
Db	CHAR(64)
Table_name	CHAR(60)
Column_name	CHAR(60)

Avant MySQL 3.23, la colonne Db valait CHAR(32) dans certaines tables, et CHAR(60) dans d'autres.

Pour vérifier les accès, la comparaison sur les valeurs de la colonne `Host` sont sensibles à la casse. `User` , `Password` , `Db` et `Table_name` sont insensibles. Les valeurs de `Column_name` sont insensibles depuis MySQL 3.22.12.

Dans les tables `user` , `db` et `host` , tous les champs de droits sont déclarés avec le type `ENUM('N','Y')` : il peuvent prendre tous les valeurs de `'N'` (non) ou `'Y'` (oui, YES), et la valeur par défaut est `'N'` .

Dans les tables `tables_priv` et `columns_priv` , les champs de droits sont déclarés comme des champs de type `SET` :

Nom de la table	Nom du champs	Valeurs possibles
tables_priv	Table_priv	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'
tables_priv	Column_priv	'Select', 'Insert', 'Update', 'References'
columns_priv	Column_priv	'Select', 'Insert', 'Update', 'References'

En bref, le serveur utilise les tables de droits comme ceci :

- La table `user` détermine si le serveur accepte ou rejette la connexion. Pour les connexions acceptées, tous les privilèges donnés dans la table `user` indiquent des privilèges globaux. Ces droits d'appliquent à **toutes** les bases du serveur.
- Les champs d'identification de la table `db` déterminent quels utilisateurs peuvent accéder à quelles bases, depuis quel hôte. Les champs de droits indiquent alors les opérations permises. Les droits s'appliquent alors à **toutes** les bases sur le serveur.
- La table `host` est utilisée comme extension de la table `db` lorsque vous voulez qu'une ligne de la table `db` s'applique à plusieurs hôtes. Par exemple, si vous voulez qu'un utilisateur soit capable d'utiliser une base depuis plusieurs hôtes dans votre réseau, laissez la colonne `Host` vide dans la table `db` , Ce mécanisme est décrit en détails dans Contrôle d'accès, étape 2 : vérification des requêtes .
- Les tables `tables_priv` et `columns_priv` sont similaires à la table `db` , mais sont plus atomiques : elle s'appliquent au niveau des tables et des colonnes, plutôt qu'au niveau des bases.

Notez que les droits d'administration tels que (`RELOAD` , `SHUTDOWN` , etc...) ne sont spécifiés que dans la table `user` . En effet, ces opérations sont des opérations au niveau serveur, et ne sont pas liées à une base de données, ce qui fait qu'il n'y a pas de raison de les lier avec les autres tables. En fait `user` doit être consulté pour déterminer les autorisations d'administration.

Le droit de `FILE` est spécifié par la table `user` . Ce n'est pas un droit d'administration, mais votre capacité à lire ou écrire des fichiers sur le serveur hôte et dépendant de la base à laquelle vous accédez.

Le serveur `mysqld` lit le contenu des tables de droits une fois, au démarrage. Lorsqu'il y a des modifications dans les tables, elles prennent effet tel qu'indiqué dans Quand est−ce que les droits prennent effets .

Lorsque vous modifiez le contenu des tables de droits, c'est une bonne idée que de s'assurer que vous avez bien configuré les droits qui vous intéressent. Un moyen de vérifier les droits pour un compte est d'utiliser la commande `SHOW GRANTS` . Par exemple, pour déterminer les droits qui sont donnés à un compte avec les valeurs `Host` et `User` de `pc84.example.com` et `bob` , utilisez cette commande :

```
mysql> SHOW GRANTS FOR 'bob'&#64;'pc84.example.com';
```

Un outil de diagnostique pratique est le script `mysqlaccess` , que Yves Carlier a fourni à la distribution MySQL. Appelez `mysqlaccess` avec l'option the `--help` pour comprendre comment il fonctionne. Notez que `mysqlaccess` ne vérifie les accès que pour les tables `user` , `db` et `host` . Il n'utilise pas les tables de droit de niveau table ou colonne.

Pour plus d'aide au diagnostique pour les problèmes de droits, voyez la section Causes possibles de l'erreur `Access denied` . Pour des conseils généraux sur la sécurité, voyez la section Problèmes généraux de sécurité .

5.4.3 Droits fournis par MySQL

Les droits des utilisateurs sont stockés dans les tables `user` , `db` , `host` , `tables_priv` et `columns_priv` de la base `mysql` (c'est−à−dire, la base nommée `mysql`). Le serveur MySQL lit ces tables au démarrage, et dans les circonstances indiquées dans la section Quand est−ce que les droits prennent effets .

Les noms utilisés dans ce manuel font référence aux droits fournis par MySQL version 4.0.2, tel que présentés dans la table ci−dessous, avec le nom de la colonne associée au droit, dans la table de droits, et dans le contexte d'application. Plus d'informations sur la signification de chaque droit sont disponibles à Syntaxe de `GRANT` et `REVOKE` .

Droit	**Colonne**	**Contexte**
ALTER	Alter_priv	tables
DELETE	Delete_priv	tables
INDEX	Index_priv	tables
INSERT	Insert_priv	tables
SELECT	Select_priv	tables
UPDATE	Update_priv	tables
CREATE	Create_priv	bases de données, tables ou index
DROP	Drop_priv	bases de données ou tables
GRANT	Grant_priv	bases de données ou tables
REFERENCES	References_priv	bases de données ou tables
CREATE TEMPORARY TABLES	Create_tmp_table_priv	administration du serveur
EXECUTE	Execute_priv	administration du serveur
FILE	File_priv	accès aux fichiers du serveur
LOCK TABLES	Lock_tables_priv	administration du serveur
PROCESS	Process_priv	administration du serveur
RELOAD	Reload_priv	administration du serveur
REPLICATION CLIENT	Repl_client_priv	administration du serveur
REPLICATION SLAVE	Repl_slave_priv	administration du serveur
SHOW DATABASES	Show_db_priv	administration du serveur
SHUTDOWN	Shutdown_priv	administration du serveur
SUPER	Super_priv	administration du serveur

Les droits de `SELECT` , `INSERT` , `UPDATE` et `DELETE` vous permettent de faire des opérations sur les lignes qui existent, dans une table existante d'une base.

La commande `SELECT` requiert le droit de `SELECT` uniquement si des lignes sont lues dans une une table. Vous pouvez exéctuer une commande `SELECT` même sans aucun droit d'accès à une base de données dans le serveur. Par exemple, vous pourriez utiliser le client `mysql` comme une simple calculatrice :

```
 mysql> SELECT 1+1;
mysql> SELECT PI()*2;
```

Le droit de `INDEX` vous donne le droit de créer et détruire des index de table.

Le droit de `ALTER` vous donne le droit de modifier une table avec la commande `ALTER TABLE` .

Les droits de `CREATE` et `DROP` vous permettent de créer de nouvelles tables et bases de données, et de les supprimer.

Notez que si vous donnez le droit de `DROP` pour la base de données `mysql` à un utilisateur, cet utilisateur pourra détruire la base qui contient les droits d'accès du serveur !

Le droit de `GRANT` vous permet de donner les droits que vous possédez à d'autres utilisateurs.

Le droit de FILE vous donne la possibilité de lire et écrire des fichiers sur le serveur avec les commandes LOAD DATA INFILE et SELECT ... INTO OUTFILE . Tout utilisateur qui possède ce droit peut donc lire ou écrire dans n'importe quel fichier à l'intérieur duquel le serveur MySQL peut lire ou écrire.

Les autres droits sont utilisés pour les opérations administratives qui sont exécutées par l'utilitaire mysqladmin . La table ci−dessous montre quelle commande est associée à mysqladmin avec un de ces droits :

Droit	**Commande autorisée**
RELOAD	reload , refresh , flush-privileges , flush-hosts , flush-logs et flush-tables
SHUTDOWN	shutdown
PROCESS	processlist
SUPER	kill

La commande reload indique au serveur de relire les tables de droits. La commande refresh vide les tables de la mémoire, écrit les données et ferme le fichier de log. flush-privileges est un synonyme de reload . Les autres commandes flush-* effectuent des fonctions similaires à la commande refresh mais sont plus limitées dans leur application, et sont préférables dans certains contextes. Par exemple, si vous souhaitez simplement vider les tampons dans le fichier de log, utilisez flush-logs , qui est un meilleur choix que refresh .

La commande shutdown éteint le serveur.

La commande processlist affiche les informations sur les threads qui s'exécutent sur le serveur. La commande kill termine un des threads du serveur. Vous pouvez toujours afficher et terminer vos propres threads, mais vous aurez besoin des droits de PROCESS pour afficher les threads, et le droit de SUPER pour terminer ceux qui ont été démarrés par d'autres utilisateurs. Syntaxe de KILL .

C'est une bonne idée en général, de ne donner les droits de Grant qu'aux utilisateurs qui en ont besoin, et vous devriez être particulièrement vigilant pour donner certains droits :

- Le droit de GRANT permet aux utilisateurs de donner leurs droits à d'autres utilisateurs. Deux utilisateurs avec des droits différents et celui de GRANT pourront combiner leurs droits respectifs pour gagner un autre niveau d'utilisation du serveur.
- Le droit de ALTER peut être utilisé pour tromper le système en renommant les tables.
- Le droit de FILE peut servir à lire des fichiers accessibles à tous sur le serveur, et les placer dans une base de données. Le contenu pourra alors être lu et manipulé avec SELECT . Cela inclus le contenu de toutes les bases actuellement hébergées sur le serveur !
- Le droit de SHUTDOWN peut conduire au dénis de service, en arrêtant le serveur.
- Le droit de PROCESS permet de voir en texte clair les commandes qui s'exécutent actuellement, et notamment les changements de mot de passe.
- Le droit de SUPER peut être utilisé pour terminer les connexions ou modifier le mode opératoire du serveur.
- Les droits sur la base de données mysql peuvent être utilisés pour changer des mots de passe ou des droits dans la table des droits (Les mots de passe sont stockés chiffrés, ce qui évite que les intrus ne les lisent). S'ils accèdent à un mot de passe dans la table mysql.user , ils pourront l'utiliser pour se connecter au serveur avec cet utilisateur (avec des droits suffisants, le même utilisateur pourra alors remplacer un mot de passe par un autre).

Il y a des choses qui ne sont pas possibles avec le système de droits de MySQL :

- Vous ne pouvez pas explicitement interdire l'accès à un utilisateur spécifique. C'est à dire, vous ne pouvez pas explicitement décrire un utilisateur et lui refuser la connexion.
- Vous ne pouvez pas spécifier qu'un utilisateur a les droits de créer et de supprimer des tables dans une base, mais qu'il n'a pas les droits pour créer et supprimer cette base.

5.4.4 Se connecter au serveur MySQL

Les clients MySQL requièrent généralement que vous spécifiez les paramètres de connexion pour vous connecter au serveur MySQL :

- l'hôte que vous voulez utiliser
- votre nom d'utilisateur
- votre mot de passe

Par exemple, le client `mysql` peut être démarré comme ceci :

```
shell> mysql [-h nom_d_hote] [-u nom_d_utilisateur] [-pvotre_mot_de_passe]
```

Les formes alternatives des options `-h` , `-u` et `-p` sont `--host=host_name` , `--user=user_name` et `--password=your_pass` . Notez qu'il n'y a **aucun espace** entre l'option `-p` ou `--password=` et le mot de passe qui le suit.

Si vous utilisez les options `-p` et `--password` mais que vous ne spécifiez pas de mot de passe, le client vous le demandera interactivement. Le mot de passe ne sera alors pas affiché. C'est la méthode la plus sécuritaire. Sinon, n'importe quel utilisateur du système pourra voir le mot de passe de la ligne de commande en exécutant une commande telle que `ps auxww` . Sécuriser votre mot de passe .

`mysql` utilise des valeurs par défaut pour chacun des paramètres qui manquent en ligne de commande :

- Le nom d'hôte par défaut est `localhost` .
- Le nom d'utilisateur par défaut est votre nom d'utilisateur de système Unix.
- Aucun mot de passe n'est transmis si `-p` manque.

Par exemple, pour un utilisateur Unix `joe` , les commandes suivantes sont équivalentes :

```
 shell> mysql -h localhost -u joe
shell> mysql -h localhost
shell> mysql -u joe
shell> mysql
```

Les autres clients MySQL se comportent de manière similaire.Sous Unix, vous pouvez spécifier différentes valeurs par défaut qui seront utilisées lorsque vous établirez la connexion, de manière à ce que vous n'ayez pas à entrer ces informations en ligne de commande lorsque vous invoquez un programme client. Cela peut se faire de plusieurs façons :

- Vous pouvez spécifier les informations de connexion dans la section `[client]` du fichier de configuration `.my.cnf` de votre dossier personnel. La section qui vous intéresse ressemble à ceci :

  ```
   [client]
  host=nom_d_hote
  user=nom_d'utilisateur
  password=votre_mot_de_passe
  ```

  Les fichiers d'options sont présentés dans la section Utiliser les fichiers d'options .
- Vous pouvez spécifier les paramètres de connexion avec les variables d'environnement. L'hôte peut être spécifié à `mysql` avec la variable `MYSQL_HOST` . L'utilisateur MySQL peut être spécifié avec la variable `USER` (uniquement pour Windows). Le mot de passe peut être spécifié avec `MYSQL_PWD` : mais ceci est peu sécuritaire voyez Sécuriser votre mot de passe . Voyez aussi la prochaine section Variables d'environnement MySQL .

5.4.5 Contrôle d'accès, étape 1 : Vérification de la connexion

Lorsque vous tentez de vous connecter au serveur MySQL, le serveur accepte ou rejette la connexion en fonction de votre identité et du mot de passe que vous fournissez. Si le mot de passe ne correspond pas à celui qui est en base, le serveur vous interdit complètement l'accès. Sinon, le serveur accepte votre connexion et passe à l'étape 2, et la gestion de commandes.

Votre identité est basée sur trois informations :

- L'hôte depuis lequel vous vous connectez
- Votre nom d'utilisateur MySQL

La vérification d'identité est réalisée avec les trois colonnes de la table `user` (`Host` , `User` et `Password`). Le serveur accepte la connexion uniquement si une entrée dans la table `user` correspond à votre hôte, et que vous fournissez le mot de passe qui correspond.

Les valeurs de la table `user` peuvent être paramétrées comme ceci :

- Une valeur de la colonne `Host` peut être un nom d'hôte, une adresse IP numérique, ou encore `'localhost'` , qui représente l'hôte local.
- Vous pouvez utiliser les caractères jokers `'%'` et `'_'` dans le champ `Host` . Ces caractères ont la même valeur que pour les opérations de recherches avec l'opérateur `LIKE` . Par exemple, une valeur `Host` de `'%'` remplace n'importe quel nom d'hôte, alors que la valeur `'%.mysql.com'` représente tous les hôtes du domaine `mysql.com` .
- Depuis MySQL version 3.23, les valeurs de `Host` spécifiées sous la forme d'IP numériques peuvent être complétées avec le masque de réseau qui indique combien de bits d'adresse sont utilisés. Par exemple :
  ```
  mysql> GRANT ALL PRIVILEGES ON db.*
     -> TO david&#64;'192.58.197.0/255.255.255.0';
  ```
  Cela permet à toute personne se connectant depuis une adresse IP qui satisfait la contrainte suivante :
  ```
  user_ip & netmask = host_ip
  ```
  C'est à dire, pour la commande `GRANT` ci−dessus :
  ```
  client_ip & 255.255.255.0 = 192.58.197.0
  ```
  Les adresses IP qui satisfont cette condition et qui peuvent se connecter au serveur MySQL sont dans l'intervalle `192.58.197.0` à `192.58.197.255` .
- Une valeur vide pour la colonne `Host` indique que les droits doivent être gérés avec les entrées de la table `host` qui correspond à l'hôte se connectant. Vous trouverez plus d'informations à ce sujet dans le chapitre Contrôle d'accès, étape 2 : vérification des requêtes .

  Une valeur vide dans la colonne `Host` des autres tables de droits revient à `'%'` .

Comme vous pouvez utiliser des jokers dans les valeurs IP de `Host` (par exemple, `'144.155.166.%'` pour tous les hôtes d'un sous−réseau), il est possible d'exploiter cette possibilité en appelant un hôte `144.155.166.ailleurs.com` . Pour contrer ce type d'attaque, MySQL bloque les noms de domaines qui commence par des chiffres et des points. Par conséquent, si vous avez un hôte nommé `1.2.foo.com` , il ne sera jamais accepté par la colonne `Host` des tables de droits. Un caractère joker d'adresse IP peut remplacer uniquement des nombres d'IP, et pas un nom d'hôte.

Dans la colonne `User` , les caractères joker ne sont pas autorisés, mais vous pouvez laisser cette valeur vide, qui acceptera tous les noms. Si la table `user` contient une connexion avec un nom d'utilisateur vide, l'utilisateur est considéré comme anonyme. Cela signifie que le nom d'utilisateur vide est utilisé pour les prochaines vérifications d'accès pour la durée de la connexion.

Le champ `Password` peut être vide. Cela ne signifie pas que n'importe quel mot de passe est valable, mais que l'utilisateur peut se connecter sans fournir de mot de passe.

Les valeurs non vides du champ `Password` représentent des valeurs du mot de passe chiffrées. MySQL ne stocke pas les mots de passe en clair, à la vue de tous. Au contraire, le mot de passe fourni pas l'utilisateur qui tente de se connecter est chiffré (avec la fonction `PASSWORD()`). Le mot de passe ainsi chiffré est alors utilisé entre le client et le serveur pour vérifier s'il est valable. Cela évite que des mots de passe en clair circulent entre le client et le serveur, sur la connexion. Notez que du point de vue de MySQL, le mot de passe chiffré est le vrai mot de passe, ce qui fait que vous ne devez en aucun cas le donner à un tiers. En particulier, ne donnez pas accès en lecture aux utilisateurs normaux aux tables d'administration dans la base `mysql` ! A partir de sa version 4.1, MySQL utilise un mécanisme différent pour les logins, mots de passes qui est sécurisé même si les paquets TCP/IP sont sniffés et/ou que la base de données mysql est capturée.

Depuis la version 4.1, MySQL emploie une identification forte qui protège mieux les mots de passe durant le processus de connexion. Cette méthode est sécuritaire, même si les paquets TCP/IP sont surveillés pour que la base de données `mysql` est capturée. Le chiffrement est présenté dans la section Chiffrement des mots de passe en MySQL 4.1 .

Les exemples ci−dessous illustrent comment différentes variantes de `Host` et `User` dans la table `user` s'appliquent aux connexion entrantes :

Host **value**	User **value**	**Connexions autorisées**
'thomas.loc.gov'	'fred'	fred , se connectant depuis thomas.loc.gov
'thomas.loc.gov'	''	N'importe quel utilisateur, se connectant depuis thomas.loc.gov
'%'	'fred'	fred , se connectant depuis n'importe quel hôte
'%'	''	N'importe quel utilisateur, se connectant depuis n'importe quel hôte
'%.loc.gov'	'fred'	fred , se connectant depuis n'importe quel hôte dans le domaine loc.gov
'x.y.%'	'fred'	fred , se connectant depuis x.y.net , x.y.com , x.y.edu , etc. (Ceci n'est probablement pas très utilisé)
'144.155.166.177'	'fred'	fred , se connectant depuis l'hôte d'IP 144.155.166.177
'144.155.166.%'	'fred'	fred , se connectant depuis un hôte d'IP dans la classe C 144.155.166
'144.155.166.0/255.255.255.0'	'fred'	Identique à l'exemple précédent

Comme vous pouvez utiliser des caractères jokers dans les adresses IP de la colonne `Host` (par exemple, `'144.155.166.%'` pour identifier tout un sous−réseau), il est possible d'exploiter cette fonctionnalité en nommant un hôte `144.155.166.kekpart.com` . Pour contrer de telles tentatives, MySQL interdit les caractères jokers avec les noms d'hôtes qui commencent par des chiffres ou des points. Par exemple, si vous avez un nom d'hôte tel que `1.2.foo.com` , il ne sera jamais trouvé dans la colonne `Host` des tables de droits. Seule une adresse IP numérique peut être comparée avec un masque à caractère joker.

Une connexion entrante peut être identifiée par plusieurs entrées dans la table `user` . MySQL résout ce problème comme ceci :

- Lorsque le serveur lit la table `user` en mémoire, il trie les lignes.
- Lorsqu'un client tente de se connecter, le serveur lit les lignes dans l'ordre.
- Le serveur utilise la première ligne qui correspond au nom du client et à son nom d'utilisateur.

Supposons que votre table `user` ressemble à ceci :

```
 +-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

Lorsque le serveur lit cette table, il ordonne les lignes depuis les valeurs les plus spécialisées de la colonne `Host` jusqu'aux plus générales (`'%'` dans la colonne `Host` signifie ``tous les hôtes'' et elle est la moins spécifique). Les entrées identiques dans la colonne `Host` sont ordonnées en fonction de la spécificité des valeurs de la colonne `User` (une entrée vide dans la colonne `User` signifie ``n'importe quel utilisateur'' et est spécifique). Le résultat de ce tri donne quelque chose comme ceci :

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

Lorsqu'une connexion est en cours de mise en place, le serveur regarde dans cette liste, et utilisera la première entrée trouvée. Pour une connexion depuis l'hôte `localhost` avec le nom d'utilisateur `jeffrey` , les entrées `'localhost'` dans la colonne `Host` sont trouvées en premier. Parmi celles−la, la ligne avec un utilisateur vide satisfait les deux contraintes sur le nom et l'hôte. `'%'/'jeffrey'` pourrait avoir fonctionné, mais comme ce n'est pas le premier rencontré, il n'est pas utilisé.

Voici un autre exemple. Supposons que la table `user` ressemble à ceci :

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| %              | jeffrey  | ...
| thomas.loc.gov |          | ...
+----------------+----------+-
```

La table triée ressemble à ceci :

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| thomas.loc.gov |          | ...
| %              | jeffrey  | ...
+----------------+----------+-
```

Une connexion depuis l'hôte `thomas.loc.gov` avec `jeffrey` satisfait les conditions de la première ligne, tandis qu'une connexion depuis `whitehouse.gov` avec `jeffrey` satisfait la seconde ligne.Une erreur commune est de penser que pour un utilisateur donné, toutes les entrées qui utilisent explicitement ce nom seront utilisées en premier lorsque la connexion est en cours d'établissement. Ceci est tout simplement faux. L'exemple précédent illustre cette situation, car la connexion depuis l'hôte `thomas.loc.gov` avec `jeffrey` est la première ligne qui est trouvée, alors que la ligne contenant `'jeffrey'` dans la colonne `User` est ignorée, car il n'y a pas de nom d'utilisateur.

Si vous arrivez à vous connecter au serveur, mais que les droits ne sont pas ce que vous attendez, vous vous êtes probablement identifié avec un autre compte. Pour savoir quel compte le serveur utilise lors de votre identification, utilisez la fonction `CURRENT_USER()` . Elle retourne la valeur au format `user_name@host_name` qui indique les valeurs des colonne `User` et `Host` de la table `user` qui vous est affectée. Supposons que `jeffrey` se connecte et utilise la requête suivante :

```
mysql> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| &#64;localhost     |
+----------------+
```

Le résultat affiché ci−dessus indique que la ligne de la table `user` est l'utilisateur `User` vide. En d'autres termes, le serveur traite `jeffrey` comme un utilisateur anonyme.

La fonction `CURRENT_USER()` est disponible depuis MySQL 4.0.6. Fonctions d'informations . Une autre piste à explorer : imprimez le contenu de la table `user` et triez la manuellement pour voir quelle est la première ligne qui est utilisée.

5.4.6 Contrôle d'accès, étape 2 : Vérification de la requête

Une fois que vous avez établi la connexion, le serveur passe à l'étape 2. Pour chaque requête qui est fournie avec la connexion, le serveur vérifie si vous avez les droits suffisants pour exécuter une commande, en fonction du type de commande. C'est à ce moment que les colonnes de droits des tables d'administration entrent en scène. Ces droits peuvent provenir de la table `user` , `db` , `host` , `tables_priv` ou `columns_priv` . Les tables d'administration sont manipulées avec les commandes `GRANT` et `REVOKE` . (Vous pouvez aussi vous reporter à la section Comment le système de droits fonctionne qui liste les champs présents dans chaque table d'administration).

La table d'administration `user` donne les droits aux utilisateurs au niveau global, c'est à dire que ces droits s'appliquent quelle que soit la base de données courante. Par exemple, si la table `user` vous donne le droit d'effacement , `DELETE` , vous pouvez effacer des données dans n'importe quelle base de ce serveur. En d'autres termes, les droits stockés dans la table `user` sont des droits de super utilisateur. Il est recommandé de ne donner des droits via la table `user` uniquement aux super utilisateurs, ou aux administrateurs de bases. Pour les autres utilisateurs, il vaut mieux laisser les droits dans la table `user` à `'N'` et donner des droits au niveau des bases uniquement, avec les tables `db` et `host` .

Les tables `db` et `host` donnent des droits au niveau des bases. Les droits peuvent être spécifiés dans ces tables comme ceci :

- Les caractères `'%'` et `'_'` peuvent être utilisés dans la colonne `Host` et `Db` des deux tables. Si vous souhaitez utiliser le caractère `'_'` comme nom de base, utiliser la séquence `'\_'` dans la commande `GRANT` .
- La valeur `'%'` dans la colonne `Host` de la table `db` signifie ``tous les hôtes''. Une valeur vide dans la colonne `Host` de la table `db` signifie ``consulte la table `host` pour plus de détails''.
- La valeur `'%'` ou vide dans la colonne `Host` de la table `host` signifie ``tous les hôtes''.
- La valeur `'%'` ou vide dans la colonne `Db` des deux tables signifie ``toutes les bases de données''.
- Un utilisateur vide dans la colonne `User` de l'une des deux tables identifie l'utilisateur anonyme.

Les tables `db` et `host` sont lues et triées par le serveur au démarrage (en même temps que la table `user` . La table `db` est triée suivant les valeurs des colonnes `Host` , `Db` et `User` , et la table `host` est triée en fonction des valeurs des colonnes `Host` et `Db` . Comme pour la table `user` , le tri place les entrées les plus spécifiques au début, et les plus générales à la fin. Lorsque le serveur recherche une ligne, il utilise la première qu'il trouve.

Les tables `tables_priv` et `columns_priv` spécifient les droits au niveau des tables et des colonnes. Les valeurs des droits dans ces tables peuvent être spécifiés avec les caractères spéciaux suivants :

- Les caractères `'%'` et `'_'` peuvent être utilisés dans la colonne `Host` des deux tables.
- La valeur `'%'` dans la colonne `Host` des deux tables signifie ``tous les hôtes''.
- Les colonnes `Db` , `Table_name` et `Column_name` ne peuvent pas contenir de valeur vide ou de caractères jokers, dans les deux tables.

Les tables `tables_priv` et `columns_priv` sont triées en fonction des colonnes `Host` , `Db` et `User` . Ce tri est similaire à celui du tri de la table `db` , même si le tri est bien plus simple, car seul le champ `Host` peut contenir des caractères jokers.

Le processus de vérification est décrit ci−dessous. Si vous êtes familier avec le code source de contrôle d'accès, vous noterez que la description diffère légèrement de l'algorithme utilisé. La description est équivalente à ce que fait en réalité le code. La différence permet une meilleure approche pédagogique.

Pour les requêtes d'administration comme `SHUTDOWN` , `RELOAD` , etc., le serveur vérifie uniquement l'entrée dans la table `user` , car c'est la seule table qui spécifie des droits d'administration. Le droit est donné si la ligne utilisée dans la connexion courante dans la table `user` donne le droit, et sinon, ce droit est interdit. Par exemple, si vous souhaitez exécuter la commande `mysqladmin shutdown` mais que votre ligne dans la table `user` ne vous en donne pas le droit (`SHUTDOWN`), vous n'aurez pas le droit sans même vérifier les tables `db` ou `host` : ces tables ne contiennent pas de colonne `Shutdown_priv` , ce qui évite qu'on en ait besoin.

Pour les requêtes exploitant une base de données, comme `INSERT` , `UPDATE` , etc., le serveur vérifie d'abord les droits globaux de l'utilisateur (droits de super utilisateur), en regardant dans la table `user` . Si la ligne utilisée dans cette table donne droit à cette opération, le droit est donné. Si les droits globaux dans `user` sont insuffisants, le serveur déterminera les droits spécifiques à la base avec les tables `db` et `host` :

- Le serveur recherche dans la table `db` des informations en se basant sur les colonnes `Host` , `Db` et `User` . Les champs `Host` et `User` sont comparés avec les valeurs de l'hôte et de l'utilisateur qui sont connectés. Le champ `Db` est comparé avec le nom de la base de données que l'utilisateur souhaite utiliser. S'il n'existe pas de ligne qui corresponde à `Host` et `User` , l'accès est interdit.
- S'il existe une ligne dans la table `db` et que la valeur de la colonne `Host` n'est pas vide, cette ligne définit les droits de l'utilisateur.
- Si dans la ligne de la table `db` , la colonne `Host` est vide, cela signifie que la table `host` spécifie quels hôtes doivent être autorisés dans la base. Dans ce cas, une autre recherche est faite dans la table `host` pour trouver une ligne avec les colonnes `Host` et `Db` . Si aucune ligne de la table `host` n'est trouvée, l'accès est interdit. S'il y a une ligne, les droits de l'utilisateur sont calculés comme l'intersection (**NON PAS** l'union !) des droits dans les tables `db` et `host` , c'est−à−dire que les droits doivent être marqués `'Y'` dans les deux tables (de cette façon, vous pouvez donner des droits généraux dans la table `db` puis les restreindre sélectivement en fonction des hôtes, en utilisant la table `host` .

Après avoir déterminé les droits spécifiques à l'utilisateur pour une base grâce aux tables `db` et `host` , le serveur les ajoute aux droits globaux, donnés par la table `user` . Si le résultat autorise la commande demandée, l'accès est donné. Sinon, le serveur vérifie les droits au niveau de la table et de la colonne dans les tables `tables_priv` et `columns_priv` , et les ajoute aux droits déjà acquis. Les droits sont alors donnés ou révoqués en fonction de ces résultats.

Exprimée en termes booléens, la description précédente du calcul des droits peut être résumé comme ceci :

```
 droits globaux
OR (droits de base AND droits d'hôte)
OR droits de table
OR droits de colonne
```

Il n'est peut−être pas évident pourquoi, si les droits globaux issus de la table `user` sont initialement insuffisants pour l'opération demandée, le serveur ajoute ces droits à ceux de base, table ou colonne ? La raison est que la requête peut demander l'application de plusieurs droits. Par exemple, si vous exécutez une commande `INSERT ... SELECT` , vous aurez besoin des droits de `INSERT` et de `SELECT` . Vos droits peuvent être tels que la table `user` donne un droit, mais que la table `db` en donne un autre. Dans ce cas, vous aurez les droits nécessaires pour faire une opération, mais le serveur ne peut le déduire d'une seule table : les droits de plusieurs tables doivent être combinés pour arriver à la bonne conclusion.

La table `host` sert à gérer une liste d'hôtes reconnus et sécuritaires.

Chez TcX, la table `host` contient une liste de toutes les machines du réseau local. Ces machines reçoivent tous les droits.

Vous pouvez aussi utiliser la table `host` pour spécifier les hôtes qui **ne sont pas** sécuritaires. Supposons que la machine `public.votre.domaine` t est placée dans une zone publique que vous considérez comme peu sûre. Vous pouvez autoriser l'accès de toutes les machines, hormis celle−ci, grâce à la table `host` configurée comme ceci :

```
+--------------------+----+-
| Host               | Db | ...
+--------------------+----+-
| public.your.domain | %  | ... (tous les droits à 'N')
| %.your.domain      | %  | ... (tous les droits à 'Y')
+--------------------+----+-
```

Naturellement, vous devriez toujours tester vos requêtes dans la table de droits, en utilisant l'utilitaire `mysqlaccess` pour vous assurer que vous disposez des droits nécessaires pour réaliser cette opération.

5.4.7 Quand les modifications de privilèges prennent−ils effets ?

Lorsque `mysqld` est lancé, toutes les tables de droits sont lues, et sont utilisées.

Les modifications aux tables de droits que vous faites avec `GRANT` , `REVOKE` et `SET PASSWORD` sont immédiatement prises en compte par le serveur.

Si vous modifiez les tables de droits manuellement (avec `INSERT` , `UPDATE` , etc...), vous devez exécuter la commande `FLUSH PRIVILEGES` ou la commande `mysqladmin flush-privileges` , ou encore `mysqladmin reload` pour dire au serveur de relire les tables de droits. Sinon, vos modifications n'auront **aucun effet** jusqu'au redémarrge du serveur. Si vous modifiez les tables de droits manuellement, mais que vous oubliez de recharger les droits, vous vous demanderez sûrement pourquoi vos modifications n'ont pas d'effet.

Lorsque le serveur remarque que les tables de droits ont été modifiées, les connexions existantes avec les clients sont modifiées comme ceci :

- Les droits de table et colonnes prennent effet à la prochaine requête du client.
- Les droits de bases prennent effet à la prochaine commande `USE nom_de_base` .
- Les droits globaux et les modifications de droits prennent effets lors de la prochaine connexion.

5.4.8 Causes des erreurs Access denied

Si vous rencontrez des erreurs `Access denied` quand vous essayez de vous connecter au serveur MySQL, la liste suivante indique quelques actions à entreprendre pour corriger le problème :

- Assurez vous que le serveur fonctionne. S'il ne fonctionne pas, vous ne pourrez pas vous y connecter. Par exemple, si vous tentez de vous connecter au serveur, et que vous recevez un message comme celui−ci, c'est peut−être que le serveur ne fonctionne pas :

  ```
  shell> mysql
  ERROR 2003: Can't connect to MySQL server on 'host_name' (111)
  shell> mysql
  ERROR 2002: Can't connect to local MySQL server through socket
  '/tmp/mysql.sock' (111)
  ```

  Il se peut aussi que le serveur fonctionne, mais que vous essayez de vous connecter en utilisant un port TCP/IP, un pipe nommé ou un fichier de socket Unix qui n'est pas celui que le serveur utilise. Pour corriger cela, lorsque vous utilisez un client, spécifiez l'option `--port` pour indiquer le bon port, et l'option `--socket` pour indiquer le bon fichier de socket Unix ou le pipe nommé Windows.

- Les tables de droits doivent être correctement configurée pour que le serveur les utilise lors de l'identification. Les installations Windows qui utilisent une distribution binaire ou les installations binaires Unix `RPM` initialisent automatiquement la base `mysql` contenant les tables de droits. Pour les autres types d'installation, vous devez initialiser les tables de droits manuellement, avec le script `mysql_install_db` . Procédure de post−installation sur Unix .

  Un moyen de déterminer si vous avez besoin d'initialiser les tables de droits est de regarder dans le dossier `mysql` dans le dossier de données. Le dossier de données s'appelle `data` ou `var` et est situé dans le dossier d'installation de MySQL. Assurez vous que vous avez un fichier appelé `user.MYD` dans le dossier `mysql` . Si vous ne le trouvez pas, exécutez le script `mysql_install_db` . Après exécution de ce script, et redémarrage du serveur, testez les premiers droits avec la commande :

  ```
  shell> mysql -u root test
  ```

  Le serveur doit vous laisser vous connecter sans erreur.
- Après une installation toute fraîche, vous devez vous connecter au serveur et créer les utilisateurs en réglant leurs permissions d'accès :

  ```
  shell> mysql -u root mysql
  ```

  Le serveur devrait vous laisser vous connecter car l'utilisateur `root` de MySQL n'a pas de mot de passe initial. Ceci est aussi une faille de sécurité, et donc, vous devez choisir un mot de passe pour l'utilisateur `root` en même tant que les autres utilisateurs MySQL. Pour des instructions sur la configuration des mots de passe initiaux, voyez la section Sécurisation des comptes MySQL initiaux .
- Si vous avez mis à jour une version de MySQL avec une nouvelle versions, avez−vous utilisé le script `mysql_fix_privilege_tables` ? Si ce n'est pas le cas, faîtes−le. La structure des tables de droits change occasionnellement, lorsque de nouvelles fonctionnalités sont ajoutées : après une mise à jour, assurez−vous que vos tables ont la bonne structure. Pour des instructions, voyez Mise à jour des tables de droits .
- Si un programme client reçoit l'erreur suivante lorsqu'il essaie de se connecter, cela signifie que le serveur attend un mot de passe dans un nouveau format, alors que le client fournit un ancien format :

  ```
  shell> mysql
  Client does not support authentication protocol requested
  by server; consider upgrading MySQL client
  ```

  Pour des informations sur comment traiter ce type de situations, voyez Chiffrement des mots de passe en MySQL 4.1 et `Client does not support authentication protocol` .
- Si vous essayez de vous connecter en tant que `root` et que vous recevez l'erreur suivante, cela signifie que vous n'avez pas d'entrée dans la table `user` avec une valeur `'root'` dans la colonne `User` et que `mysqld` ne peut pas résoudre le nom d'hôte du client :

  ```
  Access denied for user: ''&#64;'unknown' to database mysql
  ```

  Dans ce cas, vous devez relancer le serveur avec l'option `--skip-grant-tables` , et éditer votre fichier `/etc/hosts` ou `\windows\hosts` pour ajouter une ligne vous votre hôte.
- N'oubliez pas que les clients utilisent les paramètres de connexions placés dans les fichiers d'options ou les variables d'environnement. Si un client semble envoyer des paramètres de connexions invalides, lorsque vous n'en spécifiez aucun, vérifiez votre environnement, et les options appropriées. Par exemple, si vous recevez l'erreur `Access denied` avec un client utilisé sans option, assurez vous que vous n'avez pas spécifié un ancien mot de passe dans vos anciens fichiers d'options.

  Vous pouvez supprimer l'utilisation des fichiers d'options d'un client en utilisant l'option `--no-defaults` . Par exemple :

  ```
  shell> mysqladmin --no-defaults -u root version
  ```

Le fichier d'option que les clients utilisent sont listés dans la section Utiliser les fichiers d'options . Les variables d'environnement sont listées dans Variables d'environnement MySQL .

* Si vous obtenez une erreur qui ressemble à celle−ci :

```
shell> mysqladmin −u root −pxxxx ver
Access denied for user: 'root'&#64;'localhost' (Using password: YES)
```

Cela signifie que vous utilisez un mot de passe erroné.

Si l'erreur précédente survient lorsque vous n'avez pas spécifié de mot de passe, cela signifie que vous n'avez pas spécifié de mot de passe dans un fichier d'options. Essayez l'option `−−no−defaults` telle décrit ci−dessus.

Pour des informations sur les changements de mot de passe, voyez Mise en place d'un mot de passe .

Si vous avez oublié le mot de passe root, vous pouvez redémarrer `mysqld` avec `−−skip−grant−tables` pour changer le mot de passe. Comment modifier un mot de passe Root oublié .
* Si vous n'arrivez pas à faire fonctionner votre mot de passe, souvenez−vous que vous devez utiliser la fonction `PASSWORD()` si vous le changez avec les commandes `INSERT` , `UPDATE` , ou `SET PASSWORD` . L'utilisation de la fonction `PASSWORD()` n'est pas nécessaire si vous spécifiez le mot de passe en utilisant la commande `GRANT ... INDENTIFIED BY` ou la commande `mysqladmin password` . Mise en place d'un mot de passe .

```
mysql> SET PASSWORD FOR 'abe'&#64;'host_name' = 'eagle';
```

A la place, utilisez cette commande :

```
mysql> SET PASSWORD FOR 'abe'&#64;'host_name' = PASSWORD('eagle');
```

La fonction `PASSWORD()` n'est pas nécessaire si vous spécifiez un mot de passe avec la commande `GRANT` ou la commande en ligne `mysqladmin password` , qui utilisent automatiquement `PASSWORD()` pour chiffrer le mot de passe.
* `localhost` est un synonyme de votre nom d'hôte local, et est aussi l'hôte par défaut auquel le client essaye de se connecter si vous n'en spécifiez pas un explicitement. Toutefois, les connexions à `localhost` ne fonctionnent pas si vous utilisez une version antérieure à la 3.23.27 qui utilise les `MIT−pthreads` .Pour contourner ce problème sur de tels systèmes, vous devez utiliser l'option `−−host` pour nommer l'hôte du serveur explicitement. Cela créera une connexion TCP/IP vers le serveur `mysqld` . Dans ce cas, vous devez avoir votre vrai nom d'hôte dans les entrées de la table `user` du serveur hôte. (Cela est vrai même si vous utilisez un programme client sur la même machine que le serveur.)
* Si vous obtenez une erreur `Access denied` lorsque vous essayez de vous connecter à la base de données avec `mysql −u nom_utilisateur nom_base` , vous pouvez avoir un problème dans la table `user` . Vérifiez le en vous exécutant `mysql −u root mysql` et entrant la commande SQL suivante :

```
mysql> SELECT * FROM user;
```

Le résultat devrait comprendre une entrée avec les colonnes `Host` et `User` correspondante au nom d'hôte de votre ordinateur et votre nom d'utilisateur MySQL.
* Le message d'erreur `Access denied` vous dira en tant que qui vous essayez de vous identifier, l'hôte à partir duquel vous voulez le faire, et si vous utilisez ou pas un mot de passe. Normalement, vous devez avoir une entrée dans la table `user` qui correspondent au nom d'hôte et nom d'utilisateur donnés dans le message d'erreur. Par exemple, si vous obtenez une erreur qui contient `Using password: NO` , cela signifie que vous avez essayé de vous connecter sans mot de passe.
* Si vous obtenez l'erreur suivante en essayant de vous connecter à partir d'un hôte différent de celui sur lequel est placé le serveur, c'est qu'il n'y a pas d'enregistrement dans la table `user` qui correspond à cet hôte :

```
Host ... is not allowed to connect to this MySQL server
```

Vous pouvez corriger ce problème en configurant un compte avec la combinaison hôte / nom d'utilisateur que vous utilisez lors de la connexion.Si vous ne connaissez ni l'IP ni le nom d'hôte à partir duquel vous essayez de vous connecter, vous devez créer une entrée avec '`%`' dans la colonne `Host` dans la table `user` et redémarrer `mysqld` avec l'option `−−log` sur la machine serveur. Après avoir essayé à nouveau de vous connecter à partir de la machine cliente, les informations contenues dans le log de MySQL vous apprendront comment vous vous êtes vraiment connectés. (Remplacez alors l'entrée de la table `user` contenant `'%'` avec le nom d'hôte qui apparaît dans le log. Sinon, vous aurez un système non−sécurisé.)

Une autre raison pour cette erreur sous Linux est que vous utilisez une version binaire de MySQL qui est compilée avec une version de `glibc` différente de celle que vous utilisez. Dans ce cas, vous devez soit mettre à jour votre système d'exploitation et sa bibliothèque `glibc` , soit télécharger les sources de MySQL et les compiler vous−même. Un `RPM` de sources est normalement facile à compiler et installer, cela ne devrait donc pas vous poser de gros problèmes.

- Si vous obtenez une erreur où le nom d'hôte est absent ou que celui−ci est une adresse IP alors que vous avez bien entré le nom d'hôte :

```
 shell> mysqladmin −u root −pxxxx −h some−hostname ver
Access denied for user: 'root&#64;' (Using password: YES)
```

  Cela signifie que MySQL a rencontré des erreurs lors de la résolution de l'IP du nom d'hôte. Dans ce cas, vous pouvez exécuter `mysqladmin flush−hosts` pour vider le cache interne des DNS. Comment MySQL utilise les DNS .Les autres solutions sont :
  - Essayez de trouver le problème avec votre serveur DNS et corrigez le.
  - Spécifiez les IP à la place des noms d'hôtes dans les tables de droits de MySQL.
  - Ajoutez une ligne pour le nom de votre machine dans `/etc/hosts` .
  - Démarrez `mysqld` avec `−−skip−name−resolve` .
  - Démarrez `mysqld` avec `−−skip−host−cache` .
  - Sous Unix, si vous utilisez le serveur et le client sur la même machine, connectez vous à `localhost` . Les connexions Unix à `localhost` utilisent une socket Unix plutôt que TCP/IP.
  - Sous Windows, si vous exécutez le serveur et le client sur la même machine, et que le serveur supporte les pipes nommés, connectez vous à l'hôte `.` (point). Les connexions à `.` utilisent les pipes nommés plutôt que TCP/IP.
- Si `mysql −u root test` fonctionne mais que `mysql −h votre_hote −u root test` provoque une erreur `Access denied` , il se peut que vous ayez entré de mauvaises informations pour votre nom d'hôte dans la table `user` . Un problème commun ici est que la valeur `Host` dans la table `user` spécifie un nom d'hôte non−qualifié, mais que vos routines système de résolution de noms retournent un nom de domaine pleinement qualifié (ou vice−versa). Par exemple, si vous avez une entrée avec l'hôte `'tcx'` dans la table `user` , mais que vos DNS disent à MySQL que votre nom d'hôte est `'tcx.subnet.se'` , l'entrée ne fonctionnera pas. Essayez d'ajouter une entrée dans la table `user` qui contient votre adresse IP en tant que valeur de la colonne `Host` . (Une alternative est d'ajouter une entrée dans la table `user` avec une valeur de `Host` qui contient un caractère spécial, par exemple, `'tcx.%'` . Toutefois, l'utilisation des noms d'hôtes se terminant par `'%'` est **non−sécurisé** et **n'est pas** recommandé !)
- Si `mysql −u utilisateur test` fonctionne mais que `mysql −u utilisateur autre_base` ne fonctionne pas, vous n'avez pas d'entrée pour `autre_base` listée dans la table `db` .
- Si `mysql −u utilisateur nom_base` fonctionne à partir du serveur, mais que `mysql −h nom_hote −u utilisateur nom_base` ne fonctionne pas à partir d'une autre machine, cette machine n'est pas listée dans la table `user` ou `db` .
- Si vous n'arrivez pas à trouver pourquoi vous obtenez l'erreur `Access denied` , effacez toutes les entrées de la table `user` dont la valeur du champ `Host` contiennent des caractères spéciaux (entrées contenant `'%'` ou `'_'`). Une erreur commune est d'insérer une nouvelle entrée avec `Host` = `'%'` et `User` = `'un utilisateur'` , en pensant que cela vous permettra de spécifier `localhost` pour vous connecter à partir de la même machine. La raison pour laquelle cela ne fonctionnera pas est que les droits par défaut incluent une entrée avec `Host` = `'localhost'` et `User` = `''` . Puisque cette entrée possède une valeur de `Host` égale à `'localhost'` , qui est plus spécifique que `'%'` , elle est utilisée de préférence à la nouvelle entrée lors de la connexion à partir de `localhost` ! La procédure correcte est d'insérer une seconde entrée avec `Host` = `'localhost'` et `User` = `'un_utilisateur'` , ou de supprimer l'entrée avec `Host` = `'localhost'` et `User` = `''` .

- Si vous avez l'erreur suivante, vous avez peut−être un problème avec la table `db` ou `host` :

```
Access to database denied
```

  Si l'entrée sélectionnée dans la table `db` possède un champ `Host` vide, assurez−vous qu'il y a au moins une entrée correspondante dans la table `host` spécifiant les hôtes auxquels l'entrée dans la table `db` s'applique.
- Si vous obtenez l'erreur lors de l'utilisation des commandes SQL `SELECT ... INTO OUTFILE` ou `LOAD DATA INFILE` , votre entrée dans la table `user` ne possède probablement pas les droits de `FILE` .
- Si vous apportez des modifications aux tables de droits directement (en utilisant une requête `INSERT` ou `UPDATE`) et que vos changements semblent ignorés, souvenez vous que vous devez exécuter une requête `FLUSH PRIVILEGES` ou la commande `mysqladmin flush-privileges` pour demander au serveur de lire à nouveau les tables de droits. Sinon, vos changements ne seront pris en compte qu'au prochain démarrage du serveur. Souvenez−vous qu'après avoir choisi le mot de passe `root` avec une commande `UPDATE` , vous n'aurez pas à le spécifier avant de recharger les privilèges, car le serveur ne sait pas que vous l'avez modifié !
- Si vos droits changent en milieu de session, c'est peut être qu'un administrateur MySQL a changé les droits. En rechargeant les tables de droits, il a modifié aussi les connexions existantes, comme indiqué dans Quand est−ce que les droits prennent effets .
- Si vous avez des problèmes d'accès avec un programme Perl, PHP, Python, ou ODBC, essayez de vous connecter au serveur avec `mysql -u utilisateur nom_base` ou `mysql -u utilisateur -pvotre_passe nom_base` . Si vous pouvez vous connecter en utilisant le client `mysql` , c'est que le problème vient de votre programme et non des droits MySQL. (Notez qu'il n'y a pas d'espace entre `-p` et le mot de passe; vous pouvez aussi utiliser la syntaxe `--password=votre_passe` pour spécifier le mot de passe. Si vous utilisez l'option `-p` toute seule, MySQL vous demandera le mot de passe.)
- Pour les tests, démarrez le démon `mysqld` avec l'option `--skip-grant-tables` . Vous pourrez alors changer les tables de droits MySQL puis utiliser le script `mysqlaccess` pour vérifier si vos changements ont l'effet désiré. Lorsque vous êtes satisfait de vos modifications, exécutez `mysqladmin flush-privileges` pour dire au serveur `mysqld` de commencer à utiliser les nouvelles tables de droits. Recharger les tables de droits écrase l'option `--skip-grant-tables` . Cela vous permet de dire au serveur de commencer à prendre en considération les droits sans avoir à le couper et le redémarrer.
- Si rien ne fonctionne, démarrez le démon `mysqld` avec l'option de débogage (par exemple, `--debug=d,general,query`). Cela affichera l'hôte et les informations de l'utilisateur pour chaque tentative de connexion. Les informations à propos de chaque commande exécutée seront aussi affichées. Créer des fichiers de traçage .
- Si vous avez d'autres problèmes avec les tables de droits de MySQL et que vous sentez que vous devez envoyer le problème à la liste de diffusion, fournissez toujours le contenu de vos tables de droits. Vous pouvez obtenir les données avec la commande `mysqldump mysql` . Comme toujours, postez votre problème à l'aide du script `mysqlbug` . Comment rapporter des bogues ou problèmes . Dans certains cas, vous aurez besoin de redémarrer `mysqld` avec `--skip-grant-tables` pour pouvoir exécuter `mysqldump` .

5.4.9 Hashage de mots de passe en MySQL 4.1

Les comptes utilisateurs de MySQL sont stockés dans la table `user` de la base `mysql` . Chaque compte MySQL a un mot de passe, même si ce qui est stocké dans la colonne `Password` de la table `user` n'est pas la version texte du mot de passe, mais un hash calculé à partir du mot de passe. La transformation est faîte avec la fonction `PASSWORD()` .

MySQL utilise les mots de passe en deux phases, lors de la communication client/serveur :

- Premièrement, lorsqu'un client tente de se connecter au serveur, il y a une identification initial au cours de laquelle le client doit présenter un mot de passe dont la valeur hashée est la même que celle qui est présente dans la table d'utilisateur, pour le compte que le client veut utiliser.

- Ensuite, après la connexion du client, il peut modifier ou changer le mot de passe pour les utilisateurs du serveur (s'il a les droits nécessaires pour cela). Le client peut faire cela avec la fonction `PASSWORD()` , pour générer un autre mot de passe, ou en utilisant les commandes `GRANT` ou `SET PASSWORD` .

En d'autres termes, le serveur **utilise** les valeurs hashées durant la phase d'identification, lorsque le client tente de se connecter. Le serveur **génère** des valeurs hash, si un client appelle la fonction `PASSWORD()` ou utilise les commandes `GRANT` ou `SET PASSWORD` .

Le mécanisme de modifications des mots de passe a été modifié en MySQL 4.1, pour apporter une sécurité accrue et réduire le risque de vol de mots de passe. Cependant, ce nouveau mécanisme ne peut être compris que de la version 4.1, et des clients MySQL 4.1, ce qui pose des problèmes de compatibilité. Un client 4.1 peut se connecter sur un serveur pre−4.1, car le client comprend les deux méthodes de hashage, ancienne et nouvelle. Cependant, un client pre−4.1 qui tente se de connecter à un serveur 4.1 aura des problèmes. Par exemple, si un client `mysql` 4.0 essaie de se connecter au serveur 4.1, il va recevoir l'erreur suivante :

```
 shell> mysql -h localhost -u root
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

La discussion suivante décrit les différences entre les mécanismes de mots de paase, et ce que vous devez faire pour mettre à jour votre serveur en version 4.1, tout en conservant la compatibilité avec les clients pre−4.1.

Note : Cette discussion compare les comportements des versions 4.1 avec les versions d'avant (dites pre−4.1), mais le comportement 4.1 ne commence en réalité qu'avec la version 4.1.1. MySQL 4.1.0 est une version ``marginale'' car elle a un mécanisme légèrement différent de celui qui est implémenté en versions 4.1.1 et plus récent. Les différences entre les versions 4.1.0 et les versions plus récentes sont décrites ultérieurement.

Avant MySQL 4.1, les hashes calculés par `PASSWORD()` étaient longs de 16 octets. Des hashes ressemblait à ceci :

```
 mysql> SELECT PASSWORD('mypass');
+--------------------+
| PASSWORD('mypass') |
+--------------------+
| 6f8c114b58f2ce9e   |
+--------------------+
```

La colonne `Password` de la table `user` , dans laquelle les hash de mot de passse sont stockés, faisait 16 octets de long, avant MySQL 4.1.Depuis MySQL 4.1, la fonction `PASSWORD()` a été modifiée, pour produire une valeur de 41 octets, comme ceci :

```
 mysql> SELECT PASSWORD('mypass');
+-----------------------------------------------+
| PASSWORD('mypass')                            |
+-----------------------------------------------+
| *43c8aa34cdc98eddd3de1fe9a9c2c2a9f92bb2098d75 |
+-----------------------------------------------+
```

La colonne `Password` de la table `user` a été aggrandie pour faire désormais 41 octets de long :

- Si vous faites une nouvelle installation de MySQL 4.1, la colonne `Password` fera automatiquement 41 octets.
- Si vous mettez à jour une ancienne installation, il est recommandé d'utiliser le script `mysql_fix_privilege_tables` pour mettre à jour la taille de la colonne `Password` , de 16 à 41 octets. Le script ne modifie pas les valeurs elles−mêmes, qui restent à 16 octets de long.

Une colonne `Password` élargie peut stocker les mots de passe dans les deux formats, ancien et nouveau. Le format d'un hash de mot de passe peut être déterminer de deux manières :

- La différence principale et évidente est la taille : 16 octets et 41 octets.
- La seconde différence est que les hashs au nouveau format commencent par le caractère `'*'` , alors que l'ancien format ne le fait pas.

Plus le hash du mot de passe est long, meilleure sont ses caractéristiques de chiffrement, et l'identification des client, basée sur des hash longs, est plus sécuritaire que l'ancienne méthode, dont les hashs sont plus courts.

La différence de taille entre les mots de passe est utile lors de l'utilisation des mots de passe, pour l'identification, et lors de la génération des hashs pour la modification des mots de passe, sur le client.

La façon de traiter le hash de mot de passe durant la phase d'identification diffère, en fonction de la taille de la colonne `Password` :

- Si la colonne est étroite, l'identification par hash court sera utilisée.
- Si la colonne est large, elle peut contenir des hashs longs ou courts, et le serveur peut utiliser l'un ou l'autre des formats :
    - ♦ Les clients pre−4.1 peuvent se connecter, car ils connaissent l'ancien mécanisme de hashing, et ils peuvent s'identifier pour les comptes qui ont des mots de passe court.
    - ♦ Les clients 4.1 peuvent s'identifier pour les comptes qui ont des hash longs ou courts.

Pour les comptes à hash court, l'identification est un peut plus sécuritaire pour les clients 4.1 que pour les anciens clients. En terme de sécurité, le gradient de sécurité du plus faible au meilleur est :

- Les clients pre−4.1 s'identifiant avec un hash court
- Les clients 4.1 s'identifiant avec un hash court
- Les clients 4.1 s'identifiant avec un hash long

La méthode de génération des hashs de mots de passe pour les clients connectés est aussi affectée par la taille de la colonne `Password` , et par l'option `--old-passwords` . Un serveur 4.1 génère des hashs longs sous certaines conditions : La colonne `Password` doit être assez grande pour acceuillir un hash de mot de passe long, et l'option `--old-passwords` doit être inactive. Ces conditions s'appliquent comme suit :

- La colonne `Password` doit être assez grande pour acceuillir des hashs de mot de passe longs (41 octets). Si la colonne n'a pas été mise à jour, et qu'elle a toujours la taille de 16 octets, le serveur le remarque, et générera des hashs de mots de passe courts lorsque le client va modifier son mot de passe avec `PASSWORD()` , `GRANT` ou `SET PASSWORD` . Ce comportement survient si vous avez mis à jour le serveur en version 4.1, mais omis d'utiliser le script `mysql_fix_privilege_tables` pour élargir la colonne `Password` .
- Si la colonne `Password` est suffisamment grande, elle peut stocker un mot de passe long ou court. Dans ce cas, `PASSWORD()` , `GRANT` et `SET PASSWORD` vont générer des hashs longs, à moins que le serveur n'ait été lancé avec l'option `--old-passwords` . Cette option force le serveur à utiliser les hashs courts.

Le but de l'option `--old-passwords` est d'assurer la compatibilité ascendante avec les clients pre−4.1 clients dans certaines circonstances où le serveur aurait généré des hashs longs. Cela n'affecte pas l'identification, puisque les clients 4.1 peuvent continuer à utiliser les comptes avec des hashs longs, mias cela empêche la création de hash longs dans la table `user` , lors de la modification de mots de passe. Si cela arrive, le compte me pourra plus être utilisé avec les clients pre−4.1. Sans l'option `--old-passwords` le scénario suivant est possible :

- Un ancient client se connecter sur un compte, avec un hash court.
- Le client change le mot de passe. Sans l'option `--old-passwords` , cela conduit à la création d'un hash long.
- Lors de la prochaine connexion, le client pre−4.1 ne peut plus se connecter, car le compte requiert désormais le nouveau mécanisme d'identification. Une fois que le hash long est dans la table `user` , seuls les clients 4.1 peuvent l'utiliser, car les clients pre−4.1 ne le comprennent pas.

Ce scénario montre combien il est dangeureux d'utiliser un serveur 4.1 sans l'option `--old-passwords` si vous devez supporter des clients pre−4.1. En utilisant l'option `--old-passwords` sur le serveur, les opérations de modification de mots de passe ne génèrent pas de hashs longs, et les utilisateurs ne se barreront pas l'accès par inadvertence.

L'inconvénient de l'option `--old-passwords` est que tous les hashs que vous allez créer seront des hashs courts, même pour les clients 4.1. Par conséquent, vous perdez la sécurité améliorée que les hashs longs apportent. Si vous voulez créer un compte avec un hash long (par exemple, pour un client 4.1), il faudra le faire avec un serveur qui n'utilise pas l'option `--old-passwords` .

Les scénarios suivants sont possibles avec un serveur 4.1 :

Scenario 1 : Colonne `Password` courte dans la table `user`

- Seuls, les hashs courts peuvent être stockés dans la colonne `Password` .
- Le serveur utilise uniquement les hashs courts pour les identifications.
- Pour les clients connectés, la génération de mot de passe avec `PASSWORD()` , `GRANT` ou `SET PASSWORD` utilise les mots de passe courts uniquement. Toute modification de compte entraine la création d'un hash court.
- L'option `--old-passwords` peut être utilisée, mais est superflue, car la colonne `Password` courte impose la manipulation de hashs courts de toutes manières.

Scenario 2 : colonne `Password` longue dans la table `user` ; serveur sans l'option `--old-passwords`

- les hashs courts et longs peuvent être stockés dans la colonne `Password` .
- Les clients 4.1 peuvent s'identifier sur leur compte avec des hashs courts ou longs.
- Les clients pre−4.1 peuvent s'identifier sur leur compte avec des hashs courts.
- Pour les clients connectés, la génération de mot de passe avec `PASSWORD()` , `GRANT` ou `SET PASSWORD` utilise les mots de passe longs uniquement. Toute modification de compte entraine la création d'un hash long.

Comme indiqué précédemment, le danger de ce scénario est qu'il est possible que les clients pre−4.1 se voient l'accès au serveur barré. Toutes les modifications du compte avec `GRANT` , `SET PASSWORD` et `PASSWORD()` conduisent à un hash long, qui empêrchera les clients pre−4.1 d'utiliser ce compte.

Pour régler ce problème, vous pouvez modifier le mot de passe d'une manière spéciale. Par exemple, normalement, vous pouvez utiliser la commande `SET PASSWORD` comme ceci pour modifier un mot de passe :

```
mysql> SET PASSWORD FOR
    -> 'some_user'&#64;'some_host' = PASSWORD('mypass');
```

Pour changer le mot de passe avec un hash court, utilisez la fonction `OLD_PASSWORD()` :

```
mysql> SET PASSWORD FOR
    -> 'some_user'&#64;'some_host' = OLD_PASSWORD('mypass');
```

`OLD_PASSWORD()` est pratique pour les situations où vous voulez explicitement générer un hash court.

Scenario 3 : colonne `Password` longue dans la table `user` ; serveur avec l'option `--old-passwords`

- les hashs courts et longs peuvent être stockés dans la colonne `Password` .
- Les clients 4.1 peuvent s'identifier sur leur compte avec des hashs courts ou longs. Notez qu'il n'est alors possible de créer des hashs longs si le serveur utilise `--old-passwords`).
- Les clients pre−4.1 peuvent s'identifier sur leur compte avec des hashs courts.
- Pour les clients connectés, la génération de mot de passe avec `PASSWORD()` , `GRANT` ou `SET PASSWORD` utilise les mots de passe courts uniquement. Toute modification de compte entraine la création d'un hash long.

Dans ce scénario, vous ne pouvez plus créer de compte avec un hash long, car `--old-passwords` l'empêche. De même, si vous créez un compte avec un hash long sur un serveur qui utilise l'option `--old-passwords` , la modification du mot de passe tant que `--old-passwords` est active, aura pour effet de réduire la taille du hash, et vous perdre en sécurité.

Les inconvénients de ces scénario sont les suivants :

Scenario 1) vous ne pouvez pas tirer partie des hashs long et de leur sécurité accrue.

Scenario 2) Les comptes avec des mots de passe courts sont inaccessibles aux clients pre−4.1 si vous modifiez leur mot de passe sans utiliser la fonction `OLD_PASSWORD()` .

Scenario 3) `--old-passwords` empêche les comptes avec des hashs courts d'être barrés, mais les opérations de modifications de mots de passe créeront des hashs courts, et vous ne pourrez pas les modifier tant que à `--old-passwords` est effective.

5.4.9.1 Implications des modifications de mot de passe pour les applications

Une mise à jour en version MySQL 4.1 peut se révéler un problème de compatibilité pour les applications qui utilisent `PASSWORD()` pour générer leurs propres mots de passe. Les applications ne devraient pas faire cela, car `PASSWORD()` doit être réservé pour gérer les mots de passe de MySQL. Mais certaines applications utilisent `PASSWORD()` pour leurs propres objectifs.

Si vous passez en version 4.1 et lancez le serveur dans certaines conditions, où il va générer des hashs de mots de passe longs, l'application qui utilise `PASSWORD()` va sûrement planter. Notre recommandation est d'utiliser les fonctions de chiffrement `SHA1()` ou `MD5()` pour produire des signatures. Si ce n'est pas possible, vous pouvez utiliser `OLD_PASSWORD()` , qui est fournie pour générer des hashs courts, dans l'ancien format (mais notez que `OLD_PASSWORD()` pourrait être abandonné un jour aussi.

Si le serveur fonctionne dans des conditions où il génère des hashs courts, `OLD_PASSWORD()` est disponible comme alias de `PASSWORD()` .

5.4.9.2 Gestion des mots de passe en MySQL 4.1.0

Le hashing de mot de passe de MySQL 4.1.0 diffère de celui de la version 4.1.1 et plus récents. Les différences avec la version 4.1.0 sont :

- Les mots de passe sont stockés sur 45 octets plutôt que 41.
- La fonction `PASSWORD()` n'est pas répétable. C'est à dire, à partir du même argument `X` , des appels successifs à `PASSWORD(X)` génèreront différents résultats.

Ces différences rendent l'identification de la version 4.1.0 incompatible avec les versions suivantes. Si vous avez mis à jour MySQL 4.1.0, il est recommandé de passer à une version plus récente aussitôt que possible. Après cela, réassignez les mots de passe de la table `user` pour qu'ils soient compatibles avec le format 41 octets.

5.5 Gestion des comptes utilisateurs de MySQL

Cette section décrit comment configurer des comptes clients pour un serveur `MySQL` . Elle traite des points suivants :

- La signification des nom de compte et mots de passes, tels qu'utilisés par `MySQL` , et quelle différence il y avec ceux de votre système d'exploitation.
- Comment configurer de nouveaux comptes.
- Comment modifier les mots de passe.
- Des conseils de gestion des mots de passe.
- Comment configurer des connexions sécurisées avec SSL

5.5.1 Nom d'utilisateurs MySQL et mots de passe

Il y a de nombreuses différences entre les utilisations des noms et mots de passe sous MySQL, et celles qui sont faites sous Unix ou Windows :

- Les noms d'utilisateurs, tels qu'utilisés pour le processus d'identification sous MySQL, n'ont rien à voir avec les noms d'utilisateurs Unix ou Windows. La plupart des clients utilisent par défaut leur mot de passe Unix, mais c'est surtout parce que c'est pratique. Les programmes clients permettent d'utiliser des noms d'utilisateurs différents avec les options `−u` et `−−user` . Cela signifie que vous ne pouvez pas rendre une base de données sécuritaire sans donner de mots de passe à tous les clients. Tout le monde peut essayer de se connecter au serveur sous n'importe quel nom, et il sera possible de se connecter si un nom d'utilisateur n'a pas de mot de passe.
- Les noms d'utilisateurs MySQL peuvent avoir jusqu'à 16 caractères ; les noms d'utilisateurs Unix sont généralement limités à 8 caractères.
- Les mots de passe MySQL n'ont aucun rapport avec le passeport Unix. Il n'y a pas nécessairement de connexion entre le mot de passe que vous utilisez pour vous connecter sur la machine Unix et celui que vous utilisez pour accéder au serveur MySQL.
- MySQL chiffre les mots de passe avec un algorithme différent de celui qui est utilisé par Unix. Reportez−vous aux descriptions des fonctions `PASSWORD()` et `ENCRYPT()` dans Fonctions de chiffrement . Notez que même si le mot de passe est enregistré 'brouillé', connaître votre mot de passe 'brouillé' est suffisant pour se connecter au serveur MySQL.

Lorsque vous installez MySQL, la table de droit contient quelques comptes initiaux. Ces comptes ont des noms et droits qui sont décrits dans Sécurisation des comptes MySQL initiaux , qui montre aussi comment donner des mots de passe. Après cela, vous pouvez créer, modifier ou supprimer normalement des comptes MySQL avec les commandes `GRANT` et `REVOKE` . Syntaxe de `GRANT` et `REVOKE` .

Lorsque vous vous connectez à un serveur MySQL avec un client en ligne de commande, vous devez spécifier le mot de passe avec l'option `−−password=mot-de-passe` . Connexion au serveur MySQL .

```
mysql −−user=monty −−password=devine nom_base
```

ou la version courte :

```
shell> mysql -u monty -pdevine nom_base
```

Il **ne doit pas** y avoir d'espace entre l'option `-p` et le mot de passe suivant. Les commandes incluent la valeur d'un mot de passe en ligne de commande, ce qui n'est pas sécuritaire. Sécuriser votre mot de passe .

Pour éviter cela, spécifiez l'option `--password` ou `-p` sans aucune valeur :

```
shell> mysql --user=monty --password nom_base
shell> mysql -u monty -p nom_base
```

Le client va alors afficher une invite, et vous demander d'y saisir le mot de passe. Dans les exemples, `nom_base` **n'est pas** interprété comme un mot de passe, car il est séparé de l'option précédente par un espace.

Sur certains systèmes, l'appel que MySQL utilise pour demander le mot de passe va limiter automatiquement le mot de passe à 8 caractères. C'est un problème avec la bibliothèque système, et non pas avec MySQL. En interne, MySQL n'a pas de limite pour la taille du mot de passe. Pour contourner le problème, modifiez la taille du mot de passe pour qu'il fasse 8 caractères ou mois, ou placez votre mot de passe dans un fichier d'options.

5.5.2 Ajouter de nouveaux utilisateurs à MySQL

Vous pouvez ajouter des utilisateurs de deux façons différentes :

- en utilisant la commande `GRANT`
- manipulant la table des droits de MySQL directement

La méthode préférée consiste à utiliser la commande `GRANT` , car elle est plus concise et qu'il y a moins de risques d'erreur. Sa syntaxe est présentée dans la section Syntaxe de `GRANT` et `REVOKE` .

Il y a aussi beaucoup de programmes utilitaires comme `phpmyadmin` qui peuvent être utilisés pour créer et administrer les utilisateurs.

Les exemples suivants montrent comment utiliser le client `mysql` pour créer de nouveaux utilisateurs. Ces exemples supposent que les privilèges sont attribués en accord avec les valeurs par défaut discutées dans la section Sécurisation des comptes MySQL initiaux . Cela signifie que pour effectuer des changements, vous devez être sur la même machine où `mysqld` tourne, vous devez vous connecter en tant qu'utilisateur MySQL `root` , et l'utilisateur `root` doit avoir le droit `INSERT` sur la base `mysql` et le droit d'administration `RELOAD` . Si vous avez changé le mot de passe de l'utilisateur `root` , vous devez le spécifier dans les commandes `mysql` ci−dessous.

D'abord, utilisez le programme client `mysql` pour vous connecter au serveur MySQL en tant qu'utilisateur `root` :

```
shell> mysql --user=root mysql
```

Vous pouvez ajouter de nouveaux utilisateurs en utilisant des commandes `GRANT` :

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'&#64;'localhost'
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'&#64;'%'
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO 'admin'&#64;'localhost';
mysql> GRANT USAGE ON *.* TO 'dummy'&#64;'localhost';
```

Ces commandes `GRANT` ajoutent trois nouveaux utilisateurs :

- Deux comptes de super−utilisateur qui utilisent le mot de passe `'un_mot_de_passe'` . Les deux comptes ont tous les droits sur le serveur. Un des comptes, `'monty'@'localhost'` , peut être utilisé depuis la machine locale. L'autre depuis n'importe quel autre serveur : `'monty'@'%'` . Notez que nous devons exécuter une commande `GRANT` pour `'monty'@'localhost'` et `'monty'@'%'` . Si nous n'ajoutons pas l'entrée avec `localhost` , l'entrée concernant l'utilisateur anonyme pour `localhost` qui est créée par `mysql_install_db` prendra précédence lors de la connexion à partir de l'hôte local, car elle a une entrée plus spécifique pour la valeur du champ `Host` et de plus, elle vient en premier dans l'ordre de tri de la table `user` . Les tris de la table `user` sont présentés dans la section Contrôle d'accès, étape 1 : vérification de la connexion .
- Un utilisateur `admin` qui peut se connecter depuis `localhost` sans mot de passe et qui a les droits administratifs `RELOAD` et `PROCESS` . Cela permet à cet utilisateur d'exécuter les commandes `mysqladmin reload` , `mysqladmin refresh` , et `mysqladmin flush-*` , ainsi que `mysqladmin processlist` . Aucun droit lié aux bases de données n'est donné. Ils peuvent l'être plus tard en utilisant d'autres instructions `GRANT` .
- Un utilisateur `dummy` qui peut se connecter sans mot de passe, mais seulement à partir de l'hôte local. Les droits globaux sont tous à `'N'` : le type de droit `USAGE` vous permet de créer un utilisateur démuni de privilège. Il est supposé que vous lui assignerez les droits spécifiques aux bases de données plus tard.

Vous pouvez ajouter les mêmes droits d'accès aux utilisateurs en utilisant directement des requêtes `INSERT` puis en demandant au serveur de recharger les tables de droits :

```
 shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('un_mot_de_passe'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('un_mot_de_passe'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user SET Host='localhost',User='admin',
    ->           Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
    ->                  VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Selon votre version de MySQL, vous pouvez avoir un nombre différent de valeurs `'Y'` plus haut (les versions antérieures à la 3.22.11 possèdent moins de colonnes de privilèges). Pour l'utilisateur `admin` , la syntaxe d' `INSERT` étendue la plus lisible disponible depuis la version 3.22.11 est utilisée.

Notez que pour ajouter un super−utilisateur, vous avez juste besoin de créer une entrée dans la table `user` avec tous les champs de droits à `'Y'` . Aucune entrée n'est requise dans les tables `db` et `host` .

Les colonnes de privilèges de la table `user` n'étaient pas renseignées explicitement dans la dernière requête `INSERT` (pour l'utilisateur `dummy`), ses colonnes prennent donc la valeur par défaut, `'N'` . C'est la même chose que ce que fait `GRANT USAGE` .

L'exemple suivant ajoute un utilisateur `custom` qui peut se connecter à partir des hôtes `localhost` , `server.domain` , et `whitehouse.gov` . Il ne pourra accéder à la base de données `bankaccount` qu'à partir de `localhost` , à la base `expenses` qu'à partir de `whitehouse.gov` , et à la base `customer` à partir des trois hôtes. Il utilisera le mot de passe `stupid` pour les trois hôtes.

Pour configurer les privilèges de cet utilisateur en utilisant des commandes `GRANT` , exécutez ce qui suit :

```
 shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON bankaccount.*
    ->     TO custom&#64;localhost
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON expenses.*
    ->     TO custom&#64;whitehouse.gov
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON customer.*
    ->     TO custom&#64;'%'
    ->     IDENTIFIED BY 'stupid';
```

Les trois comptes peuvent être utilisés comme suit :

- Le premier compte a accès à la base `bankaccount` , mais uniquement depuis l'hôte local.
- Le second compte peut accéder à la base `expenses` , mais uniquement depuis l'hôte `whitehouse.gov` .
- Le troisième compte peut accéder à la base `customer` , mais uniquement depuis l'hôte `server.domain` .

Pour régler les permissions d'accès en modifiant directement les tables de droits, exécutez ces commandes (notez l'appel à `FLUSH PRIVILEGES` à la fin) :

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('localhost','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('server.domain','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('whitehouse.gov','custom',PASSWORD('stupid'));
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('whitehouse.gov','expenses','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES('%','customer','custom','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

Les trois premières requêtes `INSERT` ajoute les entrées dans la table `user` qui permettent l'utilisateur `custom` à se connecter à partir de plusieurs hôtes avec le mot de passe donné, mais ne lui donnent aucun droit (tous les droits sont mis à la valeur par défaut qui est `'N'`). Les trois requêtes `INSERT` suivantes ajoutent des entrées dans la table `db` qui autorisent `custom` à utiliser les bases de données `bankaccount` , `expenses` , et `customer` , mais seulement s'il y accède à partir de l'hôte spécifié. Comme d'habitude, lorsque les tables de droits sont modifiées directement, on doit demander au serveur des les recharger (avec `FLUSH PRIVILEGES`) pour que les changements soient pris en compte.Si vous voulez donner un accès spécifique à un utilisateur à partir de n'importe quelle machine d'un domaine donné, vous pouvez utiliser la commande `GRANT` , en utilisant `'%'` comme joker dans le nom de l'hôte :

```
mysql> GRANT ...
    ->     ON *.*
    ->     TO monutilisateur&#64;"%.mondomaine.com"
    ->     IDENTIFIED BY 'monmotdepasse';
```

Pour faire la même chose en modifiant directement la table de droits, faites :

```
mysql> INSERT INTO user VALUES ('%.mondomaine.com', 'monutilisateur',
    ->             PASSWORD('monmotdepasse'),...);
mysql> FLUSH PRIVILEGES;
```

5.5.3 Supprimer un compte utilisateur de MySQL

Pour supprimer un compte, utilisez la commande `DROP USER` , qui a été ajoutée en MySQL 4.1.1. Pour les anciennes versions de MySQL, utilisez la commande `DELETE` . La suppression de compte est décrite dans la section Syntaxe de `DROP USER` .

5.5.4 Limiter les ressources utilisateurs

Jusqu'à la version 4.0.2, la seule méthode possible pour limiter l'utilisation des ressources serveurs MySQL était de configurer la variable de démarrage `max_user_connections` avec une valeur non nulle. De même, le nombre de connexions simultanées pouvaient être limitées pour un compte, mais pas les opérations réalisables, une fois l'utilisateur connecté. Ces deux types de contrôles ont importants pour les administrateurs systèmes et les fournisseurs de services.

Depuis MySQL 4.0.2, il est possible de limiter certaines ressources accessibles à un utilisateur possible :

- Nombre de requête par heure : Toutes les commandes qu'un utilisateur peut exécuter.
- Nombre de modifications par heure : Toute commande qui implique la modification d'une table ou d'une base.
- Nombre de connexions réalisées par heure : Le nombre de nouvelles connexions par heure.

Toute commande que le client émet compte pour la limiter de requêtes. Seules les commandes qui modifient les tables ou bases comptant pour la limite de modifications.

Pour utiliser cette fonctionnalité, la table `user` de la base `mysql` doit contenir les colonnes nécessaires pour stocker les limites. Les limites doivent être stockées dans les colonnes `max_questions` , `max_updates` et `max_connections` . Si votre table `user` ne dispose pas de ces colonnes, elle doit être mise à jour. Voyez Mise à jour des tables de droits .

Par défaut, les utilisateurs ne sont pas limités dans l'utilisation des ressources ci−dessus, à moins que des limites ne leur soient imposées. Ces limites peuvent être configurées uniquement via la commande `GRANT (*.*)` , avec cette syntaxe :

Pour spécifier des limites de ressources avec la commande `GRANT` , utilisez la clause `WITH` pour chaque ressource que vous voulez limiter. Par exemple, pour créer un nouveau compte avec un accès à la base `customer` , mais sans abuser, utilisez ceci :

```
mysql> GRANT ALL ON customer.* TO 'francis'&#64;'localhost'
   ->     IDENTIFIED BY 'frank'
   ->     WITH MAX_QUERIES_PER_HOUR 20
   ->          MAX_UPDATES_PER_HOUR 10
   ->          MAX_CONNECTIONS_PER_HOUR 5;
```

Les types de limite n'ont pas besoin d'être tous appelés avec `WITH` , mais ils peuvent être appelés dans n'importe quel ordre. La valeur de chaque limite doit être un entier représentant le nombre autorisé par heure. Si la commande `GRANT` n'a pas de clause `WITH` , les limites valent alors 0, c'est à dire qu'il n'y a pas de limite.

Pour configurer et changer les limites d'un compte existant, utilisez la commande `GRANT USAGE` au niveau global, avec `ON *.*` . La commande suivante modifie la limite de requêtes du compte `francis` à 100 :

```
mysql> GRANT USAGE ON *.* TO 'francis'&#64;'localhost'
   ->     WITH MAX_QUERIES_PER_HOUR 100;
```

Cette commande ne touche pas aux droits du compte : elle ne modifie que les valeurs des limites.

Pour supprimer une limite existante, donnez lui la valeur de 0. Par exemple, pour supprimer la limite de connexions de `francis` , utilisez cette commande :

```
mysql> GRANT USAGE ON *.* TO 'francis'&#64;'localhost'
   ->     WITH MAX_CONNECTIONS_PER_HOUR 0;
```

Le compteur d'utilisation des ressources se met en marche dès que la limite n'est pas nulle.

Durant le fonctionnement du serveur, les ressources utilisées sont comptées. Si le compte atteint la limite de connexions dans un intervalle d'une heure, les connexions suivantes sont rejetées, jusqu'à la fin de l'heure. Similairement, si le compte atteint la limite de requête dans un intervalle d'une heure, les requêtes suivantes sont sont rejetées, jusqu'à la fin de l'heure. Dans tous les cas, un message approprié est affiché.

Le compte de ressource est fait par compte, et non pas client. Par exemple, si votre compte a une limite de requêtes de 50, vous ne pouvez pas augmenter votre limite à 100 en vous connectant deux fois. Les requêtes issues des deux connexions seront alors comptées ensemble.

Le compte courant d'utilisation peut être remis à zéro, globalement, ou individuellement :

- Pour remettre à zéro les compteurs pour tous les comptes, faites un `FLUSH USER_RESOURCES` . Les comptes sont remis à zéro au moment du re−chargement des tables de droits : par exemple, avec la commande `FLUSH PRIVILEGES` ou la commande `mysqladmin reload` .
- Les comptes individuels peuvent être remis à zéro en donnant de nouvelles limites ou changeant les droits. Pour cela, utilisez `GRANT USAGE` tel que décrit précédemment, en donnant la même limite que celle qui est configurée.

5.5.5 Configurer les mots de passe

Les mots de passe peuvent être assigné en ligne de commande avec l'utilitaire `mysqladmin` :

```
shell> mysqladmin −u user_name −h host_name password "newpwd"
```

Le compte qui est remis à zéro par cette commande est celui pour lequel une ligne de la table `user` qui correspond à la valeur `user_name` dans la colonne `User` et l'hôte client **d'où vous vous connectez** dans la colonne `Host` .Un autre moyen pour assigner un mot de passe à un compte est d'utiliser la commande `SET PASSWORD` :

```
mysql> SET PASSWORD FOR 'jeffrey'&#64;'%' = PASSWORD('biscuit');
```

Seuls les utilisateurs `root` ayant des accès en écriture à la base `mysql` peuvent changer les mots de passe des autres utilisateurs. Si vous n'êtes pas connectés en tant qu'utilisateur anonyme, vous pouvez modifier votre propre mot de passe en omettant la clause `FOR` :

```
mysql> SET PASSWORD = PASSWORD('biscuit');
```

Vous pouvez aussi utiliser la commande `GRANT USAGE` au niveau global (`ON *.*`) pour assigner un mot de passe sans affecter les droits du compte :

```
mysql> GRANT USAGE ON *.* TO 'jeffrey'&#64;'%' IDENTIFIED BY 'biscuit';
```

Même s'il est généralement préférable d'assigner un mot de passe en utilisant une des méthodes précédentes, vous pouvez aussi modifier la table `user` directement :

- Pour établir un mot de passe lors de la création d'un compte, fournissez une valeur à la colonne `Password` :

  ```
   shell> mysql −u root mysql
  mysql> INSERT INTO user (Host,User,Password)
      −> VALUES('%','jeffrey','biscuit');
  mysql> FLUSH PRIVILEGES;
  ```
- Pour changer le mot de passe d'un compte existant, utilisez la commande `UPDATE` pour modifier la valeur de la colonne `Password` :

  ```
   shell> mysql −u root mysql
  mysql> UPDATE user SET Password = PASSWORD('bagel')
      −> WHERE Host = '%' AND User = 'francis';
  mysql> FLUSH PRIVILEGES;
  ```

Lorsque vous assignez un mot de passe à un compte avec `SET PASSWORD` , `INSERT` , ou `UPDATE` , vous devez utiliser la fonction `PASSWORD()` pour le chiffrer. La seule exception est que vous n'avez pas besoin d'utiliser `PASSWORD()` si le mot de passe est vide. `PASSWORD()` est nécessaire car la table `user` stocke les mots de passe sous forme chiffrée, et non en texte clair. Si vous oubliez cela, vous risquez d'avoir des mots de passe de la forme :

```
 shell> mysql −u root mysql
mysql> INSERT INTO user (Host,User,Password)
    −> VALUES('%','jeffrey','biscuit');
mysql> FLUSH PRIVILEGES;
```

Le résultat est que la valeur `'biscuit'` est stockée dans la colonne de mot de passe de la table `user`. Lorsque l'utilisateur `jeffrey` tente de se connecter au serveur avec ce mot de passe, le client `mysql` compare ce mot de passe chiffré avec sa version en clair stockée dans la table `user`. Cependant, la version stockée est la valeur littérale de `'biscuit'`, et la comparaison échoue, le serveur rejette la connexion :

```
shell> mysql -u jeffrey -pbiscuit test
Access denied
```

Si vous modifiez les mots de passe en utilisant la commande `GRANT ... IDENTIFIED BY` ou la commande `mysqladmin password`, la fonction `PASSWORD()` n'est pas nécessaire. Ces commandes assureront le chiffrement de votre mot de passe pour vous, ce qui vous permet de spécifier le mot de passe de `'biscuit'` comme ceci :

Note : `PASSWORD()` n'effectue pas le chiffrement du mot de passe de la même façon qu'Unix. Noms d'utilisateurs MySQL et mots de passe .

5.5.6 Garder vos mots de passe en lieu sûr

Il est recommandé de ne pas placer votre mot de passe là où il risque d'être découvert par d'autres personnes. Les méthode que vous utiliserez pour spécifier votre mot de passe lors de la connexion avec le client sont listées ici, avec les risques liés à chaque méthode :

- Utilisez l'option `-p` ou `--password` (sans la valeur du `mot de passe`). Dans ce cas, le programme client va solliciter la saisie du mot de passe depuis le terminal :
  ```
  shell> mysql -u user_name -p
  Enter password: ********
  ```
  Cette méthode est pratique mais peu sûre, car le mot de passe est visible par les programmes système tels que `ps` qui peuvent être appelé par les autres utilisateurs. Les clients MySQL remplacent généralement les arguments de la ligne de commande par des zéros durant leur initialisation, mais il y a un court instant où la valeur est visible.
- Utilisez les options `-p` ou `--password` sans valeur de mot de passe. Dans ce cas, le client va solliciter explicitement le mot de passe du terminal :
  ```
  shell> mysql -u francis
  Enter password: ********
  ```
  Les caractères `'*'` représentent votre mot de passe. Le mot de passe n'est pas affiché en clair lors de votre saisie.

  Cette méthode est bien plus sûre pour saisir votre mot de passe qu'en le spécifiant directement en ligne de commande, car il n'est pas visible des autres utilisateurs. Cependant, cette méthode n'est possible qu'avec les programmes que vous utilisez en mode interactif. Si vous voulez invoquer le client depuis un script qui s'exécute de manière non interactive, il n'y aura pas d'opportunité pour saisir ce mot de passe dans le terminal. Sur certains systèmes, vous pourriez même voir la première ligne de votre script lue et interprétée comme votre mot de passe, incorrectement.
- Stockez votre mot de passe dans le fichier de configuration. Par exemple, vous pouvez lister votre mot de passe dans la section `[client]` du fichier `.my.cnf` dans votre dossier personnel :
  ```
  [client]
  password=mot_de_passe
  ```
  Si vous stockez ce mot de passe dans le fichier `.my.cnf`, le fichier ne doit pas être lisible par le groupe ou par les autres utilisateurs, ou encore accessible en écriture : seul le propriétaire de ce fichier doit avoir ces droits. Assurez−vous les droits d'accès au fichiers sont `400` ou `600`. Par exemple :
  ```
  shell> chmod 600 .my.cnf
  ```
  Utiliser les fichiers d'options présente les options pour plus de détail.s

- Vous pouvez stocker votre mot de passe dans la variable d'environnement `MYSQL_PWD` , mais cette méthode doit être considérée comme extrêmement peu sûre, et doit être évitée autant que possible. Certaines versions de la commande en ligne `ps` incluent une option pour afficher les variables d'environnement des processus : votre mot de passe sera alors facilement accessible, et en texte clair, si vous configurez la commande `MYSQL_PWD` . Même sur les systèmes sans une telle version de la commande `ps` , il est peu recommandé de supposer que les variables d'environnement sont inaccessibles par une méthode quelconque. Variables d'environnement MySQL .

En conclusion, la méthode la plus sûre est encore de laisser le client vous demander le mot de passe, ou de le spécifier dans le fichier de configuration.

5.5.7 Utilisation des connexions sécurisées

5.5.7.1 Introduction aux connexions sécurisées

Disponible depuis la version 4.0.0, MySQL supporte les connexions sécurisées. Pour comprendre comment MySQL utilise SSL, il est nécessaire de comprendre les concepts SSL et X509 de base. Ceux qui les connaissent, peuvent aisément sauter ce chapitre.

Par défaut, MySQL utilise une connexion en clair entre le client et le serveur. Cela signifie qu'une personne peut surveiller votre trafic, et lire les données échangées. Cette personne pourrait aussi modifier les données qui transitent entre le client et le serveur. Parfois, vous aurez besoin d'échanger des informations sur un réseau public, mais en sécurisant ces informations. Dans ce cas, utiliser une connexion sans protection est inacceptable.

SSL est un protocole qui utilise différents algorithmes de chiffrement pour s'assurer que les données qui transitent par un réseau public peuvent être considérées comme fiables. Ce protocole dispose de méthodes pour s'assurer que les données n'ont pas été modifiées, ce que soit par une altération, une perte ou une répétition des données. SSL inclut aussi des algorithmes pour reconnaître et fournit des outils de vérifications d'identité, pris en charge par le standard X509.

Le chiffrement est une méthode pour rendre des données illisibles. En fait, les pratiques actuelles requièrent d'autres éléments de sécurité issus des algorithmes de chiffrement. Ils doivent savoir résister à de nombreux types d'attaque, comme la modification de l'ordre des messages ou les répétitions inopinées.

X509 est un standard qui rend possible l'identification d'une personne sur l'internet. Il est particulièrement utilisé pour les applications e−commerce. En termes simples, il doit y avoir une entreprise (appelée l'``autorité de certification'') qui assigne un certificat électronique à toute personne qui en a besoin. Ces certificats utilisent un chiffrement assymétrique qui exploitent deux clés de chiffrement, une clé publique et une clé privée. Le propriétaire d'un certificat peut prouver son identité en montrant son certificat à l'autre partie. Un certificat est constitué de la clé publique du propriétaire. Toute donnée qui est chiffrée avec cette clé publique doit être déchiffrée avec la clé secrète correspondante, qui est détenue par le propriétaire du certificat.

MySQL n'utilise pas les connexions chiffrées par défaut, car cela ralentit considérablement le protocole de communication. Toute fonctionnalité supplémentaire requiert du travail supplémentaire de la part du serveur, et chiffrer des données est une tâche particulièrement coûteuse, qui peut ralentir considérablement les tâches principales de MySQL. Par défaut, MySQL est paramétré pour être aussi rapide que possible.

Si vous avez besoin de plus d'informations sur SSL, X509 ou le chiffrement, utilisez votre moteur de recherche préféré sur Internet, et utilisez ces mots clés pour avoir plus de détails.

5.5.7.2 Pré requis aux connexions sécurisées

Pour utiliser les connexions SSL entre le serveur MySQL et les clients, vous devez avoir le support de OpenSSL et votre version de MySQL doit être 4.0.0 ou plus récente.

Pour faire fonctionner les connexions sécurisées avec MySQL, vous devez disposer de ceci :

- Installation de la bibliothèque d'OpenSSL. Nous avons testé MySQL avec OpenSSL 0.9.6. http://www.openssl.org/ .
- Lorsque vous configurez MySQL, utilisez le script `configure` avec les options `--with-vio` et `--with-openssl` .
- Assurez vous que vous avez une version de la table `mysql.user` à jour. Ceci est nécessaire si vos tables de droits proviennent d'un version de MySQL antérieure à la version 4.0.0. La procédure de mise à jour est décrite dans Mise à jour des tables de droits .
- Vous pouvez vérifier que vous posséder un serveur `mysqld` qui supporte OpenSSL en examinant le résultat de la commande `SHOW VARIABLES LIKE 'have_openssl'` :

```
 mysql> SHOW VARIABLES LIKE 'have_openssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
+---------------+-------+
```

  elle doit retourner `YES` .

5.5.7.3 Configurer les certificats SSL pour MySQL

Voici un exemple de configuration de certificats SSL pour MySQL :

```
 DIR=`pwd`/openssl
PRIV=$DIR/private

 mkdir $DIR $PRIV $DIR/newcerts
cp /usr/share/ssl/openssl.cnf $DIR
replace ./demoCA $DIR -- $DIR/openssl.cnf

 # Créez les dossiers nécessaires : $database, $serial et $new_certs_dir
 # optionnel

 touch $DIR/index.txt
echo "01" > $DIR/serial

 #
# Génération du cerificat d'autorité (CA)
#

 openssl req -new -x509 -keyout $PRIV/cakey.pem -out $DIR/cacert.pem \
    -config $DIR/openssl.cnf

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ................++++++
# .........++++++
# writing new private key to '/home/monty/openssl/private/cakey.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
```

```
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL admin
# Email Address []:

 #
# Création des clé et requêtes serveur
#
openssl req -new -keyout $DIR/server-key.pem -out \
    $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ..++++++
# ..........++++++
# writing new private key to '/home/monty/openssl/server-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL server
# Email Address []:
#
 # Please enter the following 'extra' attributes
# to be sent with your certificate request
# A challenge password []:
# An optional company name []:

 #
# Supprimez la passe-phrase de la clé (optionnel)
#

 openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem

 #
# Signez le certificat serveur
#
openssl ca  -policy policy_anything -out $DIR/server-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/server-req.pem

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL admin'
# Certificate is to be certified until Sep 13 14:22:46 2003 GMT (365 days)
# Sign the certificate? [y/n]:y
#
 #
 # 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

 #
# Créez les clé et requêtes client
#
openssl req -new -keyout $DIR/client-key.pem -out \
    $DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# .....................................++++++
# .............................................++++++
# writing new private key to '/home/monty/openssl/client-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be incorporated
# into your certificate request.
# What you are about to enter is what is called a Distinguished Name or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
```

```
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL user
# Email Address []:
#
 # Please enter the following 'extra' attributes
# to be sent with your certificate request
# A challenge password []:
# An optional company name []:

 #
# Supprimez la passe-phrase de la clé (optionnel)
#
openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem

 #
# Signez le cerficat client
#

 openssl ca  -policy policy_anything -out $DIR/client-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/client-req.pem

 # Exemple de résultat :
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL user'
# Certificate is to be certified until Sep 13 16:45:17 2003 GMT (365 days)
# Sign the certificate? [y/n]:y
#
 #
 # 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

 #
# Créez le fichier my.cnf que vous pourrez utiliser pour tester les différents certificats
#

 cnf=""
cnf="$cnf [client]"
cnf="$cnf ssl-ca=$DIR/cacert.pem"
cnf="$cnf ssl-cert=$DIR/client-cert.pem"
cnf="$cnf ssl-key=$DIR/client-key.pem"
cnf="$cnf [mysqld]"
cnf="$cnf ssl-ca=$DIR/cacert.pem"
cnf="$cnf ssl-cert=$DIR/server-cert.pem"
cnf="$cnf ssl-key=$DIR/server-key.pem"
echo $cnf | replace " " '
' > $DIR/my.cnf

 #
# To test MySQL

 mysqld --defaults-file=$DIR/my.cnf &

 mysql --defaults-file=$DIR/my.cnf
```

Pour tester les connexions SSL, lancez le serveur comme ceci, où `$DIR` est le dossier où le fichier de configuration `my.cnf` est situé :

```
shell> mysqld --defaults-file=$DIR/my.cnf &
```

Puis, lancez le programme client en utilisant le même fichier d'options :

```
shell> mysql --defaults-file=$DIR/my.cnf
```

Si vous avez une distribution source MySQL, vous pouvez aussi tester votre configuration en modifiant le fichier `my.cnf` précédent, pour utiliser les certificats et fichiers de clé `SSL` de la distribution.

5.5.7.4 Options de GRANT avec SSL

MySQL peut vérifier les certificats X509 en plus de la combinaisons habituelle de nom d'utilisateur et mot de passe. Toutes les options habituelles sont toujours nécessaires (nom d'utilisateur, masque d'adresse IP, nom de base de données, nom de table). Syntaxe de `GRANT` et `REVOKE` .

Voici différentes possibilités pour limiter les connexions :

- Sans aucune option SSL ou X509, toutes les connexions chiffrées ou non chiffrées sont autorisées si le nom d'utilisateur et le mot de passe sont valides.
- L'option `REQUIRE SSL` requiert que les connexions soient chiffrées avec SSL. Notez que cette option peut être omise si il n'y a pas de ligne ACL qui autorise une connexion sans SSL.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret" REQUIRE SSL;
```

- `REQUIRE X509` impose au client d'avoir un certificat valide, mais le certificat lui même est de peu d'importance. La seule restriction est qu'il doit être possible de vérifier la signature avec une des autorités de certification.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret" REQUIRE X509;
```

- `REQUIRE ISSUER "issuer"` restreint les tentatives de connexion : le client doit se présenter avec un certificat X509 valide, émis par l' autorité de certification `"issuer"` . Utiliser un certificat X509 implique obligatoirement des chiffrements, donc l'option `SSL` est sous entendue.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE ISSUER "C=FI, ST=Some-State, L=Helsinki,
    "> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu&#64;mysql.com";
```

- `REQUIRE SUBJECT "subject"` impose au client d'avoir un certificat X509 valide, avec le sujet `"subject"` . Si le client présente un certificat valide, mais que le `"subject"` est différent, la connexion est refusée.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn,
    "> O=MySQL demo client certificate,
    "> CN=Tonu Samuel/Email=tonu&#64;mysql.com";
```

- `REQUIRE CIPHER "cipher"` est utilisé pour s'assurer que les chiffrements sont suffisamment robuste, et que la bonne longueur de clé est utilisée. SSL lui même peut être faible si des algorithmes sont utilisés avec des clés courtes. En utilisant cette option, il est possible d'imposer la méthode de chiffrement avec la connexion.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE CIPHER "EDH-RSA-DES-CBC3-SHA";
```

Les options `SUBJECT` , `ISSUER` et `CIPHER` peuvent être combinées avec la clause `REQUIRE` comme ceci :

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root&#64;localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn,
    "> O=MySQL demo client certificate,
    "> CN=Tonu Samuel/Email=tonu&#64;mysql.com"
    -> AND ISSUER "C=FI, ST=Some-State, L=Helsinki,
    "> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu&#64;mysql.com"
    -> AND CIPHER "EDH-RSA-DES-CBC3-SHA";
```

Depuis MySQL 4.0.4, le mot clé `AND` est optionnel, entre les options `REQUIRE` .L'ordre de ces options n'a pas d'importance, mais aucune option ne peut être spécifiée deux fois.

5.5.7.5 Options SSL en ligne de commande

La table suivante liste les différentes options que vous pouvez utiliser avec SSL, les fichiers de certificats et de clés. Ces options sont disponibles depuis MySQL 4.0. Elles peuvent être spécifiées en ligne de commande ou bien dans le fichier d'options.

`--ssl`

Pour le serveur, indique que le serveur autorise les connexions SSL. Pour le client, permet au logiciel de se connecter au serveur en utilisant le protocole SSL. Cette option seule n'est pas suffisante pour que la connexion soit sécurisée par SSL. Vous devez aussi spécifier les options `--ssl-ca` , `--ssl-cert` et `--ssl-key` .Cette option est plus souvent utilisée que sa forme contraire, pour indiquer que SSL **ne doit pas** être utilisé. Pour cela, spécifiez l'option sous la forme `--skip-ssl` ou `--ssl=0` .

Notez que cette option ne **requiert** pas de connexion SSL. Par exemple, si le serveur ou le client sont compilés sans le support SSL, une connexion sans chiffrement sera utilisée.

Une méthode pour s'assurer que la connexion sera bien sécurisée par SSL est de créer un compte sur le serveur, avec l'option `REQUIRE SSL` dans la commande `GRANT` . Utilisez alors ce compte pour vous connecter au serveur, avec le serveur et le client qui disposent du support SSL.

Vous pouvez utiliser cette option pour indiquer que la connexion ne doit pas être SSL. Pour faire cela, spécifiez l'option `--skip-ssl` ou `--ssl=0` .

`--ssl-ca=file_name`
: Le chemin jusqu'au fichier avec une liste des autorités de certifications SSL connues.

`--ssl-capath=directory_name`
: Le chemin jusqu'au dossier qui contient les certificats SSL au format `PEM` .

`--ssl-cert=file_name`
: Le nom du fichier de certificat SSL à utiliser pour établir une connexion sécurisée.

`--ssl-cipher=cipher_list`
: Une liste de chiffrements autorisées, à utiliser avec SSL. `cipher_list` a le même format que la commande `openssl ciphers` .Exemple : `--ssl-cipher=ALL:-AES:-EXP`

`--ssl-key=file_name`
: Le nom du fichier de la clé SSL a utiliser pour établir une connexion sécurisée.

5.5.7.6 Connexion à MySQL à distance avec Windows et SSH

Voici une note pour connecter un serveur MySQL avec une connexion sécurisée grâce à SSH (de David Carlson dcarlson@mplcomm.com) :

- Installez un client SSH pour votre machine Windows. En tant qu'utilisateur, le meilleur que je connaisse est celui de `SecureCRT` de http://www.vandyke.com/ . Une autre option est `f-secure` de http://www.f−secure.com/ . Vous pouvez aussi en trouver d'autres de gratuit avec `Google` à http://directory.google.com/Top/Computers/Security/Products_and_Tools/Cryptography/SSH/Clients/Wi .
- Démarrez votre client SSH Windows. Spécifiez `Host_Name = yourmysqlserver_URL_or_IP` . Spécifiez `userid=your_userid` pour vous loger dans votre serveur (probablement avec un mot de passe et un nom d'utilisateur différent).
- Configurez le `forward` de port. Faites soit un `forward` distant (spécifiez `local_port: 3306` , `remote_host: yourmysqlservername_or_ip` , `remote_port: 3306`) soit un forward local (spécifiez `port: 3306` , `host: localhost` , `remote port: 3306`).
- Sauvez le tout, sinon vous devrez le refaire la prochaine fois.
- Connectez vous à votre serveur avec la session SSH que vous venez de créer.
- Sous votre machine Windows, démarrez une application ODBC (comme `Access`).
- Créez un nouveau fichier dans Windows et reliez le avec MySQL en utilisant le pilote ODBC de la même façon que vous le feriez habituellement, hormis le fait que vous devrez taper `localhost` comme hôte serveur au lieu de `yourmysqlservername` .

Vous avez maintenant une connexion ODBC avec un serveur MySQL distant, et sécurisée avec SSH.

5.6 Prévention des désastres et restauration

Cette section présente comment créer une sauvegarde de base de données, et comment assurer la maintenance des tables. La syntaxe des commandes SQL est décrite dans la section Référence de langage d'administration de la base de données .

5.6.1 Sauvegardes de base de données

Comme les tables MySQL sont stockées sous forme de fichiers, il est facile d'en faire une sauvegarde. Pour avoir une sauvegarde consistante, faites un `LOCK TABLES` sur les tables concernées suivi d'un `FLUSH TABLES` pour celles−ci. Voyez Syntaxe de `LOCK TABLES` et `UNLOCK TABLES` et Syntaxe de `FLUSH` . Vous n'avez besoin que d'un verrou en lecture; cela permet aux autre threads de continuer à effectuer des requêtes sur les tables dont vous faites la copie des fichiers dans le dossier des bases de données. `FLUSH TABLE` est requise pour s'assurer que toutes les pages d'index actifs soient écrits sur le disque avant de commencer la sauvegarde.

Si vous voulez faire une sauvegarde d'une table avec SQL, vous pouvez utiliser `SELECT INTO OUTFILE` ou `BACKUP TABLE` . Voyez Syntaxe de `SELECT` et Syntaxe de `BACKUP TABLE` .

Une autre façon de sauvegarder une base de données est d'utiliser l'utilitaire `mysqldump` ou le `script mysqlhotcopy` . Voyez `mysqldump` , exporter les structures de tables et les données et `mysqlhotcopy` , copier les bases et tables MySQL .

- Effectuez une sauvegarde complète de votre base de données :
  ```
  shell> mysqldump --tab=/chemin/vers/un/dossier --opt --all
  ```
  ou
  ```
  shell> mysqlhotcopy base /chemin/vers/un/dossier
  ```
  Vous pouvez aussi copier tout simplement tous les fichiers de tables (les fichiers `*.frm` , `*.MYD` , et `*.MYI`) du moment que le serveur ne met rien à jour. Le script `mysqlhotcopy` utilise cette méthode.
- Arrêtez `mysqld` si il est en marche, puis démarrez le avec l'option `--log-update[=nom_fichier]` . Le log des mises à jour . Le ou les fichiers de log fournissent les informations dont vous avez besoin pour répliquer les modifications de la base de données qui sont subséquents au moment où vous avez exécuté `mysqldump` .

Si votre serveur MySQL est un esclave, quelque soit la sauvegarde que vous utilisez, lorsque vous sauvez vos données sur votre esclave, vous devez aussi sauver les fichiers `master.info` et `relay-log.info` , qui sont nécessaires pour relancer la réplication après la restauration des données de l'esclave. Si votre esclave doit traiter des commandes `LOAD DATA INFILE` , vous devez aussi sauver les fichiers nommés `SQL_LOAD-*` , qui sont dans le dossier spécifié par `--slave-load-tmpdir` . Ce dossier vaut par défaut la valeur de la variable `tmpdir` , si elle n'est pas spécifiée. L'esclave aura besoin de ces fichiers pour relancer la réplication d'une opération `LOAD DATA INFILE` interrompue.

Si vous avez besoin de restaurer quelque chose, essayez d'abord de restaurer vos tables avec `REPAIR TABLE` ou `myisamchk -r` en premier. Cela devrait fonctionner dans 99.9% des cas. Si `myisamchk` ne réussi pas, essayez la procédure suivante (cela ne fonctionnera que si vous avez démarré MySQL avec `--log-update` , Le log binaire des mises à jour) :

- Restaurez la sauvegarde originale de `mysqldump` .
- Exécutez la commande suivante pour remettre en marche les mises à jour dans le log binaire :
  ```
  shell> mysqlbinlog hostname-bin.[0-9]* | mysql
  ```
  Dans votre cas, vous voudrez peut−être n'exécuter que certains logs binaires, depuis certaines positions : par exemple, depuis la date de la sauvegarde que vous avez restauré, hormis quelques requêtes problématiques. Voyez L'utilitaire binaire `mysqlbinlog` pour plus d'informations sur l'utilitaire `mysqlbinlog` , et comment l'utiliser.

  Si vous utilisez le journal des mises à jour (qui a été supprimé en MySQL 5.0.0) vous pouvez utiliser :

  ```
  shell> ls -1 -t -r hostname.[0-9]* | xargs cat | mysql
  ```

`ls` est utilisée pour avoir tous les fichiers de mise à jour dans le bon ordre.

Vous pouvez aussi faire des sauvegardes sélectives de fichiers individuels :

- Exportez la table avec `SELECT * INTO OUTFILE 'nom_fichier' FROM nom_de_table`
- Restaurez avec `LOAD DATA INFILE 'nom_fichier' REPLACE ...` . Pour éviter les lignes dupliquées, vous aurez besoin d'une `PRIMARY KEY` ou une clef `UNIQUE` dans la table. Le mot clef `REPLACE` fait que les anciens enregistrements sont remplacés par les nouveaux lorsque l'un d'eux duplique un ancien sur une valeur de clef unique.

Si vous obtenez des problèmes de performances sur votre système, vous pouvez les contourner en mettant en place une réplication et faisant les copies sur l'esclave au lieu du maître. Introduction .

Si vous utilisez un système de fichiers Veritas , vous pourrez faire :

- A partir d'un client (ou de Perl), exécutez : `FLUSH TABLES WITH READ LOCK` .
- A partir d'un autre Shell, exécutez : `mount vxfs snapshot` .
- Depuis le premier client, exécutez : `UNLOCK TABLES` .
- Copiez les fichiers à partir de la sauvegarde.
- Démontez snapshot.

5.6.2 Utilisation de myisamchk pour la maintenance des tables et leur recouvrement

La section suivante discute de l'utilisation de `myisamchk` pour vérifier et réparer les tables `MyISAM` (les tables avec les fichiers `.MYI` et `.MYD`). Les mêmes concepts s'appliquent à `isamchk` pour vérifier et réparer les tables `ISAM` (les tables avec les fichiers `.ISM` et `.ISD`). Types de tables MySQL .

Vous pouvez utiliser `myisamchk` pour obtenir des informations sur les tables de votre base de données, pour analyser, réparer ou optimiser ces tables. Les sections suivantes décrivent comment appeler `myisamchk` (y compris les options), comment mettre en place une politique d'entretien, et comment utiliser `myisamchk` pour effectuer différentes opérations.

Même si la réparation d'une table avec `myisamchk` est sécuritaire, il est toujours préférable de faire une sauvegarde **avant** la réparation, ou toute autre opération de maintenance qui pourrait faire de nombreuses modifications dans la table.

Les opérations `myisamchk` qui affectent les index peuvent causer la recompilation des index `FULLTEXT` avec des paramètres qui ne sont pas les paramètres courants du serveur. Pour éviter cela, voyez la section Options générales pour `myisamchk` .

Dans de nombreux cas, vous pouvez trouver plus simple de faire l'entretien des tables avec des requêtes SQL qu'avec `myisamchk` :

- Pour vérifier ou réparer les tables `MyISAM` , utilisez `CHECK TABLE` ou `REPAIR TABLE` .
- Pour optimiser les tables `MyISAM` , utilisez `OPTIMIZE TABLE` .
- Pour analyser les tables `MyISAM` , utilisez `ANALYZE TABLE` .

Ces commandes ont été ajoutées dans différentes versions, mais sont toutes disponibles depuis MySQL 3.23.14. Voyez Syntaxe de `ANALYZE TABLE` , Syntaxe de `CHECK TABLE` , Syntaxe de `OPTIMIZE TABLE` , et Syntaxe de `REPAIR TABLE` .

Les commandes peuvent être utilisées directement, ou via le client mysqlcheck , qui fournit une interface en ligne de commande.

Un avantage de ces commandes par rapport à myisamchk est que le serveur se charge de tout. Avec myisamchk , vous devez vous assurer que le serveur ne va pas utiliser les tables en même temps que vous. Sinon, il va y avoir des interférences entre myisamchk et le serveur.

5.6.2.1 Syntaxe de l'utilitaire myisamchk

myisamchk s'exécute avec une commande de la forme :

```
shell> myisamchk [options] tbl_name
```

Les options spécifient ce que vous voulez que myisamchk fasse. Elles sont décrites dans ce chapitre. Vous pouvez aussi obtenir une liste d'options en invoquant le programme avec myisamchk --help . Sans option, myisamchk va simplement vérifier les tables. Pour obtenir plus d'information ou pour demander à myisamchk de prendre des mesures correctives, il faut ajouter l'une des options listées ici.

tbl_name est la table que vous voulez réparer ou vérifier. Si vous exécutez myisamchk autre part que dans le dossier de données, vous devez spécifier le chemin jusqu'au fichier, car sinon, myisamchk n'aura aucune idée d'où chercher les données dans votre base. En fait, myisamchk ne se préoccupe pas du fait que le fichier que vous utilisez est dans le dossier de base ou pas : vous pouvez copier le fichier à réparer dans un autre dossier, et y faire les opérations d'entretien.

Vous pouvez spécifier plusieurs noms de tables à myisamchk si vous le voulez. Vous pouvez aussi spécifier un nom sous la forme d'un fichier d'index (avec l'option .MYI), qui vous permettra de spécifier toutes les tables dans un dossier en utilisant le schéma *.MYI . Par exemple, si vous êtes dans le dossier de données, vous pouvez spécifier toutes les tables dans le dossier comme ceci :

```
shell> myisamchk *.MYI
```

Si vous n'êtes pas dans le dossier de données, et que vous souhaitez vérifier toutes les tables, vous devez ajouter le chemin jusqu'au dossier :

```
shell> myisamchk /path/to/database_dir/*.MYI
```

Vous pouvez même vérifier toutes les tables de toutes les bases avec le chemin suivant :

```
shell> myisamchk /path/to/datadir/*/*.MYI
```

La méthode recommandée pour vérifier rapidement toutes les tables est :

```
 myisamchk --silent --fast /path/to/datadir/*/*.MYI
isamchk --silent /path/to/datadir/*/*.ISM
```

Si vous voulez vérifier toutes les tables et réparer celles qui sont corrompues, vous pouvez utiliser la ligne suivante :

```
 myisamchk --silent --force --fast --update-state -O key_buffer=64M \
          -O sort_buffer=64M -O read_buffer=1M -O write_buffer=1M \
          /path/to/datadir/*/*.MYI
isamchk --silent --force -O key_buffer=64M -O sort_buffer=64M \
        -O read_buffer=1M -O write_buffer=1M /path/to/datadir/*/*.ISM
```

Ces commandes ci−dessus supposent que vous avez plus de 64 Mo de libres. Pour plus d'informations sur l'allocation de mémoire avec myisamchk , voyez la section Utilisation de la mémoire avec myisamchk .Notez que si vous obtenez une erreur comme celle−ci :

```
myisamchk: warning: 1 clients is using or hasn't closed the table properly
```

Cela signifie que vous essayez de vérifier une table qui a été modifiée par un autre programme (comme le serveur mysqld) qui n'a pas encore refermé le fichier de table, ou que le fichier n'a pas été correctement refermé.Si mysqld fonctionne, vous devez forcer la fermeture correcte des fichiers de tables avec la commande FLUSH TABLES , et vous assurer que personne n'utilise les tables durant vos opérations avec myisamchk . En MySQL version 3.23, la meilleure méthode pour éviter ce problème est d'utiliser la commande CHECK TABLE au lieu de myisamchk pour vérifier les tables.

5.6.2.2 Options générales de myisamchk

Les options décrites dans cette section peuvent être utilisées pour toutes les maintenances de tables effectuée `myisamchk` . Les sections suivant celles−ci décrivent les options spécifiques à certaines opérations, comme la vérification et la réparation.

`myisamchk` supporte les options suivantes :

−−help, −?
: Affiche le message d'aide, et termine le programme.

−−debug=debug_options, −# debug_options
: Affiche le log de débogage. La chaîne `debug_options` vaut souvent : `'d:t:o,filename'` .

−−silent, −s
: Mode silencieux. Affiche uniquement les erreurs. Vous pouvez utiliser deux fois `-s` (`-ss`) pour que `myisamchk` soit très silencieux.

−−verbose, −v
: Mode détaillé. Affiche plus d'informations. Vous pouvez combiner ce mode avec les options `-d` et `-e` . Utilisez `-v` plusieurs fois, (`-vv` , `-vvv`) pour plus de détails encore.

−−version, −V
: Affiche la version et quitte.

−−wait, −w
: Au lieu de s'arrêter avec une erreur si la table est verrouillé, le programme attend que la table soit libérée avant de continuer. Notez que si vous utilisez `mysqld` avec l'option `--skip-external-locking` , la table peut ne peut être verrouillée que par une autre commande `myisamchk` .

Vous pouvez aussi configurer les variables suivantes avec la syntaxe `--var_name=value` :

Variable	Valeur par défaut
decode_bits	9
ft_max_word_len	dépend de la version
ft_min_word_len	4
ft_stopword_file	liste par défaut
key_buffer_size	523264
myisam_block_size	1024
read_buffer_size	262136
sort_buffer_size	2097144
sort_key_blocks	16
write_buffer_size	262136

Il est aussi possible de configurer les variables avec les syntaxes `--set-variable=var_name=value` et `-O var_name=value` . Toutefois, cette syntaxe est obsolète depuis MySQL 4.0.

Les variables `myisamchk` possibles et leur valeur par défaut sont affichées par `myisamchk --help` :

`sort_buffer_size` sert lors de la réparation des index par tri des clés, qui est le mode utilisé par l'option `--recover` .

`key_buffer_size` sert lorsque vous vérifiez une table avec l'option `--extend-check` ou lorsque les clés sont réparées par insertion de lignes dans la table (comme lors des insertions normales). La réparation par buffer de clés est utilisée dans ces situations :

- Vous utiliez l'option `--safe-recover` .
- Les fichiers temporaires utilisés pour trier les clés seraient deux fois plus gros que lors de la création directe du fichier. C'est souvent le cas lorsque vous avez de grandes clés pour les colonnes `CHAR` , `VARCHAR` et `TEXT` , car l'opération de trie a besoin de stocker la clé complète. Si vous avez beaucoup d'espace temporaire, vous pouvez forcer `myisamchk` à réparer en triant, en utilisant l'option `--sort-recover` .

La répartion par buffer de clé prend beaucoup moins d'espace disque, mais est bien plus lente.

Si vous voulez une réparation plus rapide, donnez à `key_buffer_size` et `sort_buffer_size` des valeurs représentant 25% de votre mémoire. Vous pouvez leur donner de grandes valeurs, car une seule des deux variables est utilisée.

`myisam_block_size` est la taille des blocs d'index. Elle est disponible depuis MySQL 4.0.0.

Les variables `ft_min_word_len` et `ft_max_word_len` sont disponibles depuis MySQL 4.0.0. `ft_stopword_file` est disponible depuis MySQL 4.0.19.

`ft_min_word_len` et `ft_max_word_len` indique la taille minimum et maximum pour les index `FULLTEXT` . `ft_stopword_file` est le nom du fichier de mots ignorés. Ils doivent toujours être configurés.

Si vous utilisez `myisamchk` pour faire une opération qui modifie les index de tables (comme la répartion ou l'analyse), les index `FULLTEXT` sont reconstruit en utilisant les valeurs par défaut pour les tailles minimales et maximales, et pour le fichier de mots ignorés. Cela peut conduire à l'échec de requêtes.

Le problème survient lorsque ces paramètre ne sont connus que par le serveur. Elles ne sont pas stockées dans le fichier d'index `MyISAM` . Pour éviter ce problème si vous avez modifié la taille des mots ou le fichier de mots ignorés dans le serveur, pensez à spécifier les mêmes valeurs avec les options `ft_min_word_len` , `ft_max_word_len` et `ft_stopword_file` de `myisamchk` . Par exemple, si vous avez configuré une taille minimale de 3, vous pourrez réparer la table avec `myisamchk` comme ceci :

```
shell> myisamchk --recover --ft_min_word_len=3 tbl_name.MYI
```

Pour vous assurer que `myisamchk` et le serveur utilisent les mêmes valeurs pour les paramètres des index en texte plein, vous pouvez placez ces valeurs dans les groupes `[mysqld]` et `[myisamchk]` du fichier d'options :

```
 [mysqld]
ft_min_word_len=3

 [myisamchk]
ft_min_word_len=3
```

Une alternative à l'utilisation de `myisamchk` est les commandes `REPAIR TABLE` , `ANALYZE TABLE` , `OPTIMIZE TABLE` et `ALTER TABLE` . Ces commandes sont effectuées par le serveur, qui sait comment traiter les paramétrages des index.

5.6.2.3 Options de vérifications pour myisamchk

`-c` , `--check`
: Vérifie les erreurs d'une table. Ceci est l'opération par défaut de `myisamchk` si vous ne lui donnez aucune autre option.

`-e` , `--extend-check`
: Vérifie la table minutieusement (ce qui est un peu lent si vous avez des index). Cette option ne doit être utilisée que pour les cas extrêmes. Normalement, `myisamchk` ou `myisamchk --medium-check` devrait, dans la plupart des cas, être capable de trouver s'il y a des erreurs dans la table.

  Si vous utilisez `--extended-check` et que vous avez beaucoup de mémoire, vous devez augmenter de beaucoup la valeur de `key_buffer_size` !

`-F` , `--fast`
: Ne vérifie que les tables qui n'ont pas été fermées proprement.

`-C` , `--check-only-changed`

Ne vérifie que les tables qui ont changé depuis la dernière vérification.

`-f` , `--force`
: Redémarrez `myisamchk` avec `-r` (répare) sur la table, si `myisamchk` trouve une erreur dans la table.

`-i` , `--information`
: Affiche des statistiques à propos de la table vérifiée.

`-m` , `--medium-check`
: Plus rapide que `--extended-check` , mais ne trouve que 99.99% des erreurs. Devrait, cependant, être bon pour la plupart des cas.

`-U` , `--update-state`
: Enregistre le fichier `.MYI` lorsque la table à été vérifiée ou a été corrompue. Cela devrait être utilisé pour tirer tous les avantages de l'option `--check-only-changed` , mais vous ne devez pas utiliser cette option si le serveur `mysqld` utilise cette table et que vous utilisez `mysqld` avec `--skip-external-locking` .

`-T` , `--read-only`
: Ne marque pas la table comme vérifiée. C'est pratique si vous utilisez `myisamchk` pour vérifier une table issue d'une autre application qui n'utilise pas les verrous. (comme `mysqld --skip-external-locking`).

5.6.2.4 Options de réparation de myisamchk

Les options suivantes sont utilisées avec `myisamchk` et l'option de réparation `-r` ou `-o` :

`--backup` , `-B`
: Fait une sauvegarde du fichier `.MYD` , sous le nom `filename-time.BAK`

`--character-sets-dir=path`
: Dossier qui contient les jeux de caractères. Le jeu de caractères utilisé pour les données et le stockage .

`--correct-checksum`
: Somme de contrôle correcte pour la table.

`--data-file-length=#` , `-D #`
: Taille maximale du fichier de données (lors de la recréation du fichier de données, et qu'il est complet).

`--extend-check` , `-e`
: Essaie de retrouver toutes les lignes possibles du fichier de données. Normalement, cette option va aussi découvrir beaucoup de lignes erronées. N'utilisez pas cette option si vous n'êtes pas totalement désespérés.

`--force` , `-f`
: Ecrase les anciens fichiers temporaires (`table_name.TMD`) au lieu d'annuler.

`--keys-used=#` , `-k #`
: Si vous utilisez les tables `ISAM` , indique au gestionnaire de table `ISAM` qu'il doit uniquement modifier les `#` premiers index. Si vous utilisez le gestionnaire de table `MyISAM` , cette option indique quelles clés utiliser, et chaque bit binaire représente une clé (la première clé est le bit 0). Cela permet de réaliser des insertions plus rapides. Les index désactivés pourront être réactivés avec l'option `myisamchk -r` .

`--no-symlinks` , `-l`
: Ne pas suivre les lignes symboliques. Normalement, `myisamchk` répare les tables qu'un lien symbolique représente. Cette option n'existe pas en MySQL 4.0, car MySQL 4.0 ne va pas supprimer les liens symboliques durant la réparation.

`--parallel-recover` , `-p`
: Utilise la même technique que `-r` et `-n` , mais crée les clés avec des threads différents, en parallèle. Cette option a été ajoutée en MySQL 4.0.2. **Ceci est du code alpha. Utilisez le à vos risques et périls!**

`--quick` , `-q`

Réparation rapide, sans modifier le fichier de données. Il est possible d'ajouter l'option `-q` pour forcer `myisamchk` à modifier le fichier original en cas de clés doublons.

`--recover` , `-r`

Peut réparer presque tout, sauf les clés uniques qui ne le sont plus (ce qui est extrêmement rare avec les tables `ISAM` / `MyISAM`). Si vous voulez restaurer un table, c'est l'option à utiliser en premier. Si `myisamchk` indique que la table ne peut pas être corrigée avec l'option `-r` , vous pouvez alors passer à l'option `-o` . Notez que dans le cas rarissime où `-r` , le fichier de données est toujours intact. Si vous avez beaucoup de mémoire, vous pouvez augmenter la taille du buffer `sort_buffer_size` !

`--safe-recover` , `-o`

Utilise une ancienne méthode de restauration (lit toutes les lignes dans l'ordre, et modifie l'arbre d'index conformément pour les lignes trouvées). C'est une méthode qui est beaucoup plus lente que l'option `-r` , mais elle est capable de traiter certaines situations exceptionnelles que `-r` ne pourrait pas traiter. Cette méthode utilise aussi moins d'espace disque que `-r` . Normalement, vous devriez commencer à réparer avec l'option `-r` , et uniquement sur l'échec de cette option, passer à `-o` .Si vous avez beaucoup de mémoire, vous devriez augmenter la taille du buffer de clé ! `key_buffer_size` !

`--set-character-set=name`

Change le jeu de caractères utilisé par l'index.

`--sort-recover` , `-n`

Force `myisamchk` à utiliser le tri pour résoudre les clés, même si le fichier temporaire doit être énorme.

`--tmpdir=path` , `-t path`

Chemin pour stocker les fichiers temporaires. Si cette option n'est pas fournie, `myisamchk` va utiliser la variable d'environnement `TMPDIR` pour cela. Depuis MySQL 4.1, `tmpdir` peut prendre une liste de chemins différents, qui seront utilisés successivement, pour les fichiers temporaires. Le caractère de séparation des différents chemins est le deux−points sous Unix (`':'`) et le point−virgule (`';'`) sous Windows, NetWare et OS/2.

`--unpack` , `-u`

Décompresse des données compressées avec `myisampack` .

5.6.2.5 Autres options de myisamchk

Les autres actions que `myisamchk` peut réaliser, en dehors de vérifier et réparer une table sont :

`-a` , `--analyze`

Analyser la distribution des clés. Cela améliore les performances des jointures en permettant à l'optimiseur de jointure de mieux choisir l'ordre d'utilisation des clés. `myisamchk --describe --verbose table_name'` ou `SHOW KEYS` dans MySQL.

`-d` , `--description`

Affiche des informations sur la table.

`-A` , `--set-auto-increment[=value]`

Force `AUTO_INCREMENT` à commencer avec une valeur supérieure. Si aucune valeur n'est fournie, la prochaine valeur de la colonne `AUTO_INCREMENT` sera la plus grande valeur de la colonne +1.

`-S` , `--sort-index`

Trie les blocs de l'arbre d'index dans l'ordre haut / bas. Cela va optimiser les recherches, et les scans de tables par clés.

`-R` , `--sort-records=#`

Trie les lignes en fonction de l'index. Cela rassemble vos données, et peut accélérer les lectures de lignes par intervalle avec `SELECT` et `ORDER BY` sur cet index (ce tri peut être très lent la première fois). Pour connaître les numéros d'index de tables, utilisez la commande `SHOW INDEX` , qui affiche les index dans le même ordre que `myisamchk` ne les voit. Les index sont numérotés à partir de 1.

5.6.2.6 Utilisation de la mémoire par myisamchk

L'espace mémoire est très important quand vous utilisez `myisamchk` . `myisamchk` n'utilise pas plus de mémoire que ce que vous spécifiez avec les options `-O` . Si vous pensez utiliser `myisamchk` sur des fichiers très grands, vous devez d'abord décider la quantité de mémoire que vous souhaitez utiliser. Avec des valeurs plus grandes, vous pouvez accélérer `myisamchk` . Par exemple, si vous avez plus de 32 Mo de RAM, vous pourriez utiliser les options suivantes (en plus des autres options que vous pourriez spécifier) :

```
shell> myisamchk -O sort=16M -O key=16M -O read=1M -O write=1M ...
```

Utiliser `-O sort=16M` sera probablement suffisant pour la plupart des cas.

Soyez conscient que `myisamchk` utilise des fichiers temporaires dans le dossier `TMPDIR` . Si `TMPDIR` est un fichier en mémoire, vous pourriez facilement rencontrer des erreurs de mémoire. Si cela arrive, choisissez une autre valeur pour `TMPDIR` , avec plus d'espace disque, et redémarrez `myisamchk` .

Lors de la réparation, `myisamchk` va aussi avoir besoin d'espace disque :

- Doublez la taille du fichier de données (l'original plus une copie). Cet espace n'est pas nécessaire si vous faites des réparations de type `--quick` , car dans ce cas, seul le fichier d'index sera recréé. Cet espace est nécessaire sur le même disque que l'original !
- De l'espace pour le nouveau fichier d'index qui remplacera l'ancien. L'ancien fichier d'index est réduit dès le démarrage, ce qui vous permet généralement d'ignorer cet espace. Cet espace est nécessaire sur le même disque que l'original !
- Lorsque vous utilisez les options `--recover` ou `--sort-recover` (mais pas lorsque vous utilisez `--safe-recover`), vous aurez besoin d'espace pour le buffer de tri :
  ```
  (plus_grande_cle + taille_du_pointeur_de_ligne)*nombre_de_lignes * 2
  ```
  Vous pouvez vérifier la taille des clés et la taille du pointeur de ligne avec la commande `myisamchk -dv table` . Cet espace est alloué sur le disque temporaire (spécifié par `TMPDIR` par `--tmpdir=#`).

Si vous avez des problèmes avec l'espace disque durant la réparation, vous pouvez utiliser l'option `--safe-recover` au lieu de `--recover` .

5.6.2.7 Utiliser myisamchk pour restaurer une table

Si vous utilisez `mysqld` avec l'option `--skip-external-locking` (qui est la configuration par défaut pour certains systèmes, comme Linux), vous ne pouvez pas utiliser `myisamchk` pour vérifier une table, lorsque `mysqld` utilise aussi la table. Si vous pouvez être sûr que personne n'utilise cette table via `mysqld` lorsque vous utilisez `myisamchk` , vous n'aurez qu'à utiliser la commande `mysqladmin flush-tables` avant de commencer à vérifier les tables. Si vous ne pouvez pas garantir cette condition, vous devez alors éteindre le serveur `mysqld` pour vérifier les tables. Si vous exécutez `myisamchk` alors que `mysqld` modifie la table, vous pourriez obtenir un diagnostic de corruption de la table, alors que ce n'est pas le cas.

Si vous n'utilisez pas l'option `--skip-external-locking` , vous pouvez vous servir de `myisamchk` pour vérifier les tables à tout moment. Pendant que vous le faites, les autres clients qui tentent de modifier la table devront attendre que `myisamchk` ait fini.

Si vous utilisez `myisamchk` pour réparer ou optimiser les tables, vous **devez** toujours vous assurer que `mysqld` n'utilise pas cette table (ce qui s'applique aussi si vous utilisez `--skip-external-locking`). Si vous n'éteignez pas le serveur `mysqld` , vous devez au moins utiliser `mysqladmin flush-tables` avant de lancer `myisamchk` . Vos tables **peuvent être corrompues** si le serveur et `myisamchk` travaillent dans une même table simultanément.

Ce chapitre décrit comment vérifier et gérer les corruptions de données dans les bases MySQL. Si vos tables sont fréquemment corrompues, vous devriez commencer par en rechercher la raison ! Que faire si MySQL crashe constamment .

La section sur les tables `MyISAM` contient différentes raisons pour lesquelles une table peut être corrompue. Problèmes avec les tables `MyISAM` .

Lorsque vous effectuez une restauration de table, il est important que chaque table `tbl_name` dans une base corresponde aux trois fichiers dans le dossier de base, du dossier de données :

Fichier	Utilisation
tbl_name.frm	Définition de la table
tbl_name.MYD	Fichier de données
tbl_name.MYI	Fichier d'index

Chacun de ces trois fichiers est sujet à des corruptions diverses, mais les problèmes surviennent généralement dans les fichiers de données ou d'index.

`myisamchk` fonctionne en créant une copie du fichier `.MYD` (les données), ligne par ligne. Il termine sa réparation en supprimant l'ancien fichier `.MYD` et en renommant le nouveau à la place de l'ancien. Si vous utilisez l'option `--quick` , `myisamchk` ne crée pas de fichier temporaire `.MYD` mais suppose plutôt que le fichier `.MYD` est correct et il génère simplement un nouveau fichier d'index sans toucher au fichier `.MYD` . C'est une méthode sécuritaire, car `myisamchk` va automatiquement détecter si le fichier `.MYD` est corrompu, et annulera alors la réparation si c'est le cas. Vous pouvez aussi ajouter deux options `--quick` à `myisamchk` . Dans ce cas, `myisamchk` ne s'interrompt pas sur certaines erreurs (comme des clés doublons), et essaie de résoudre ce problème en modifiant le fichier `.MYD` . Normalement, l'utilisation de deux options `--quick` n'est utile que si vous n'avez pas trop d'espace disque pour réaliser la réparation. Dans ce cas, vous devez au moins faire une copie de sauvegarde avant d'utiliser `myisamchk` .

5.6.2.8 Comment vérifier la cohérence d'une table

Pour vérifier les tables de type MyISAM, utilisez les commandes suivantes :

myisamchk nom_de_table
: Cette commande trouvera 99.99% de toutes les erreurs. Ce qu'elle ne peut pas découvrir comme erreurs, sont celles qui impliquent **uniquement** le fichier de données (ce qui est très inhabituel). Si vous voulez vérifier une table, vous devriez utiliser l'utilitaire `myisamchk` sans les options ou avec les options `-s` ou `--silent` .

myisamchk −m nom_de_table
: Cette commande trouvera 99.999% de toutes les erreurs. Elle vérifie toutes les entrées dans le fichier d'index, puis lit toutes les lignes. Elle calcule une somme de contrôle pour toutes les clés et les lignes, et vérifie que les deux se correspondent dans l'arbre d'index.

myisamchk −e nom_de_table
: Cette commande fait une vérification complète et exhaustive de toutes les données (`-e` signifie ``extended check''). Elle fait une lecture de contrôle de chaque ligne, pour vérifier qu'elle correspond bien aux index. Cette commande va prendre un long

moment sur les grosses tables. `myisamchk` va normalement s'arrêter dès qu'il trouve une erreur. Si vous voulez obtenir plus d'information sur cette erreur, vous pouvez utiliser l'option `--verbose` (ou `-v`). Cela fera que `myisamchk` va continuer à travailler et accumuler jusqu'à 20 erreurs. En utilisation normale, l'utilisation de cet utilitaire sans options est suffisante.

myisamchk −e −i nom_de_table

Comme les commandes précédentes, mais l'option `-i` indique à `myisamchk` qu'il doit afficher des informations statistiques.

5.6.2.9 Comment réparer des tables

Dans la section présente, nous allons uniquement parler de l'utilitaire `myisamchk` sur les tables `MyISAM` (extensions `.MYI` et `.MYD`). Si vous utilisez les tables `ISAM` (extensions `.ISM` et `.ISD`), vous devriez vous servir de `isamchk` à la place.

Depuis MySQL version 3.23.14, vous pouvez réparer les tables MyISAM avec la commande SQL `REPAIR TABLE` . Syntaxe de `REPAIR TABLE` .

Les symptômes de corruption de tables sont des requêtes qui s'interrompent inopinément :

- `tbl_name.frm locked against change` : `tbl_name.frm` est verrouillée en écriture
- `Can't find file tbl_name.MYI (Errcode: ###)` : Impossible de trouver le fichier `tbl_name.MYI` (Errcode: ###)
- `Unexpected end of file` : Fin de fichier inattendue
- `Record file is crashed` : Fichier de données crashé
- `Got error ### from table handler` : Reception de l'erreur ### de la part du gestionnaire de table

Pour obtenir plus d'informations sur l'erreur, vous pouvez exécuter la commande `perror ###` . Voici les erreurs les plus courantes :

```
 shell> perror 126 127 132 134 135 136 141 144 145
126 = Index file is crashed / Wrong file format : le fichier d'index est corrompu / le format du fichier est incorrect.
127 = Record-file is crashed : le fichier de données est corrompu.
132 = Old database file / ce fichier provient d'une vieille base de données.
134 = Record was already deleted (or record file crashed) / La ligne était déjà effacée.
135 = No more room in record file / Plus de place dans le fichier de données.
136 = No more room in index file / Plus de place dans le fichier d'index.
141 = Duplicate unique key or constraint on write or update / Doublon pour une clé unique trouvé durant la lecture ou l'écriture.
144 = Table is crashed and last repair failed / la table est corrompue et la dernière réparation a échoué.
145 = Table was marked as crashed and should be repaired / La table a été marquée comme corrompue et doit être réparée.
```

Notez que l'erreur 135, "no more room in record file", n'est pas une erreur qui sera facile à corriger. Dans ce cas, vous devez utiliser la commande suivante :

```
ALTER TABLE table MAX_ROWS=xxx AVG_ROW_LENGTH=yyy;
```

Dans d'autres cas, vous devrez réparer vos tables. `myisamchk` peut généralement détecter et corriger la plupart des erreurs.

Le processus de réparation se déroule en 4 étapes décrites ici. Avant de vous lancer, vous devriez vous placer dans le dossier de données et vérifier les permissions des fichiers de données. Assurez−vous qu'ils sont bien lisibles par l'utilisateur Unix que MySQL utilise (et à vous aussi, car vous aurez besoin d'accéder à ces fichiers durant la vérification. Si vous devez corriger ces fichiers, vous aurez aussi besoin des droits d'écriture.

Si vous utilisez MySQL version 3.23.16 et plus récent, vous pouvez (et vous devriez) utiliser les commandes `CHECK` et `REPAIR` pour réparer vos tables `MyISAM` . Voyez Syntaxe de `CHECK TABLE` et Syntaxe de `REPAIR TABLE` .

La section du manuel sur l'entretien des tables inclut la présentation des options des utilitaires `isamchk` / `myisamchk` : Utilisation de `myisamchk` pour maintenir les tables et recouvrir les données .

La section suivante est destinée aux cas où les commandes ci−dessus ont échoué ou que vous voulez exploiter les fonctionnalités avancées que `isamchk` / `myisamchk` proposent.

Si vous allez réparer une table en ligne de commande, il est recommandé d'arrêter le serveur `mysqld` . Notez que lorsque vous exécutez une commande `mysqladmin shutdown` sur un serveur distant, le serveur `mysqld` sera encore opérationnel pendant un instant après que `mysqladmin` ait terminé, jusqu'à ce que toutes les requêtes et toutes les clés aient été écrites sur le disque.

Etape 1 : Vérifier vos tables

Exécutez la commande `myisamchk *.MYI` ou `myisamchk -e *.MYI` si vous avez plus de temps. Utilisez `-s` (silencieux) pour supprimer les informations peu pertinentes.

Si le serveur `mysqld` a terminé, vous devriez utiliser l'option −−update pour indiquer à `myisamchk` d'enregistrer la vérification des tables ('checked').

Vous n'aurez à réparer que les tables pour lesquelles `myisamchk` vous annonce une erreur. Pour de telles tables, passez à l'étape 2.

Si vous obtenez des erreurs étranges lors de la vérification, (comme, l'erreur `out of memory`), ou si `myisamchk` crashe, passez à l'étape 3.

Etape 2 : réparation simple et facile

Note : Si vous voulez réparer très rapidement, vous devriez ajouter `-O sort_buffer=# -O key_buffer=#` (où # vaut environ le quart de la mémoire du serveur), à toutes les commandes `isamchk/myisamchk` .

Premièrement, essayez `myisamchk -r -q tbl_name` (`-r -q` signifie ``mode de réparation rapide''). Cette commande va tenter de réparer le fichier d'index sans toucher au fichier de données. Si le fichier de données contient toutess les données qu'il est sensé contenir, et que les points d'ancrage pour les effacements sont corrects, cette commande doit réussir, et la table sera alors réparée. Passez alors à la table suivante. Sinon, suivez la procédure suivante :

- Faites une copie de sauvegarde de votre fichier de données.
- Utilisez la commande `myisamchk -r tbl_name` (`-r` signifie ``mode de réparation''). Cette commande va supprimer les lignes invalides et effacer ces lignes du fichier de données, puis reconstruire le fichier d'index.
- Si l'instruction précédente a échoué, utilisez `myisamchk --safe-recover tbl_name` . Le mode restauration sécuritaire utilise une vieille méthode de réparation qui peut gérer certains cas rares, mais elle est bien plus lente.

Si vous obtenez des erreurs étranges lors de la répaaration (comme des erreurs de type `out of memory`), ou si `myisamchk` crashe, passez à l'étape 3.

Etape 3 : Réparations difficiles

Nous ne devriez atteindre cette étape que si les 16 premiers ko du fichier d'index sont détruits, ou qu'il contient des données erronées, ou si le fichier d'index manque. Dans ce cas, il est nécessaire de créer un nouveau fichier d'index. Faites ceci :

- Déplacez le fichier de données dans une archive sûre.

* Utilisez le fichier description de la table pour créer de nouveaux fichiers de données et d'index vides.

```
 shell> mysql db_name
mysql> SET AUTOCOMMIT=1;
mysql> TRUNCATE TABLE table_name;
mysql> quit
```

  Si votre version SQL ne dispose pas de `TRUNCATE TABLE` , utilisez la commande `DELETE FROM table_name` .
* Copiez l'ancien fichier de données à la place du nouveau fichier de données (ne faites pas un simple déplacement de fichier. Utilisez une copie, au cas où un problème surviendrait).

Retournez à l'étape 2. `myisamchk -r -q` doit alors fonctionner (et ceci ne doit pas être une boucle infinie).

Depuis `MySQL` 4.0.2, vous pouvez aussi utiliser `REPAIR ... USE_FRM` qui effectue toute cette opération automatiquement.

Etape 4 : Réparation très difficiles

Vous ne devriez atteindre cette étape que si votre fichier de description `.frm` a aussi crashé. Cela ne devrait jamais arriver, car le fichier de description n'est jamais modifié une fois que la table est créée.

* Restaurez le fichier de description avec une sauvegarde, et retournez à l'étape 3. Vous pouvez aussi restaurer le fichier d'index et retourner à l'étape 2. Dans ce dernier cas, vous pouvez démarrer avec l'option `myisamchk -r` .
* Si vous n'avez pas de sauvegarde, mais que vous savez exactement comment la table a été créée, vous pouvez créer une telle table dans une autre base. Supprimez alors le nouveau fichier de données, puis déplacez les fichiers de description `.frm` et d'index `.MYI` dans votre base de données crashée. Cela vous donnera un nouveau fichier d'index et de description, mais laisse intact le fichier de données `.MYD` . Retournez à l'étape 2 et essayez de reconstruire le fichier d'index.

5.6.2.10 Optimisation de table

Pour réorganiser les lignes fragmentées et éliminer l'espace perdu par les effacements et les modifications de lignes, vous pouvez exécuter l'utilitaire `myisamchk` en mode de restauration :

```
shell> myisamchk -r tbl_name
```

Vous pouvez optimiser une table de la même façon que vous le faîtes avec la commande SQL `OPTIMIZE TABLE` . `OPTIMIZE TABLE` effectue une réparation de la table, et une analyse des index, puis trie l'arbre d'index pour accélérer les recherches de clés. L'utilisation de la commande réduit aussi les interférences entre le serveur et l'utilitaire car c'est le serveur lui−même qui fait le travail. Syntaxe de `OPTIMIZE TABLE` .

`myisamchk` dispose aussi d'un grand nombre d'options que vous pouvez utiliser pour améliorer les performances de la table :

* `-S` , `--sort-index`
* `-R index_num` , `--sort-records=index_num`
* `-a` , `--analyze`

Pour une description complète de ces options, voyez Syntaxe d'invocation de `myisamchk` .

5.6.3 Mettre en place un régime d'entretien de MySQL

C'est une bonne idée que d'effectuer des vérifications des tables régulièrement, plutôt que d'attendre qu'un problème survienne. Pour faire ces vérifications, vous pouvez utiliser la commande `myisamchk -s` . L'option `-s` (raccourci pour `--silent`) fait que `myisamchk` s'exécute en mode silencieux, et n'affiche que les messages d'erreurs. Voyez les sections Syntaxe de `CHECK TABLE` et Syntaxe de `REPAIR TABLE` .

C'est aussi une bonne idée que de vérifier les tables lorsque le serveur démarre. Par exemple, à chaque fois qu'une machine redémarre au milieu d'une modification de table, vous devrez faire une vérification de toutes les tables qui pourraient être affectées : c'est une ``table supposément corrompue''. Vous pouvez ajouter un test à `safe_mysqld` pour qu'il exécute `myisamchk` , afin de vérifier toutes les tables qui ont été modifiées dans les 24 dernières heures, si il reste un vieux fichier `.pid` (identifiant de processus) après un redémarrage : le fichier `.pid` est créé par le serveur `mysqld` lorsqu'il démarre, et il est supprimé lorsque le serveur s'arrête dans des conditions normales. La présence d'un fichier `.pid` au démarrage indique que le serveur s'est arrêté anormalement.

Un test encore meilleur serait de vérifier toutes les tables dont la date de modification est plus récente que celle du fichier `.pid` .

Vous devriez aussi vérifiez vos tables régulièrement durant les opérations normales. Chez MySQL AB, nous utilisons une tâche en `cron` pour vérifier toutes nos tables importantes au moins une fois par semaine, avec une ligne comme celle−ci dans le fichier `crontab` :

```
35 0 * * 0 /path/to/myisamchk --fast --silent /path/to/datadir/*/*.MYI
```

Cela nous affiche les informations sur les tables qui ont été corrompues, de façon à ce que nous puissions les examiner et les réparer.Comme nous n'avons jamais eu de table qui se soit corrompue inopinément (des tables qui se corrompent pour d'autres raisons que des problèmes matériels) depuis quelques années (ce qui est véridique), une fois par semaine est un bon rythme pour nous.

Nous recommandons que vous commenciez par exécuter la commande `myisamchk -s` chaque nuit, sur toutes les tables qui ont été modifiées dans les 24 dernières heures, jusqu'à ce que vous preniez confiance en MySQL.

Normalement, vous n'avez pas à maintenir autant les tables MySQL. Si vous changez les tables avec un format de ligne dynamique (les tables avec des colonnes `VARCHAR` , `BLOB` ou `TEXT`) ou que vous avez des tables avec de nombreuses lignes effacées, vous pouvez envisager de faire des défragmentations du fichier, pour récupérer cet espace. Une fois par mois est un bon rythme.

Vous pouvez faire cela avec la commande SQL `OPTIMIZE TABLE` sur les tables en question, ou bien, si vous avez éteint le serveur `mysqld` , faites :

```
shell> myisamchk -r -s --sort-index -O sort_buffer_size=16M */*.MYI
```

Pour les tables `ISAM` , la commande est similaire à :

```
shell> isamchk -r -s --sort-index -O sort_buffer_size=16M */*.MYI
```

5.6.4 Obtenir des informations sur une table

Pour obtenir la description d'une table ou des statistiques à son sujet, utiliser les commandes affichées ici. Nous allons expliquer certains de leurs détails ultérieurement.

* myisamchk −d nom_de_table Exécute `myisamchk` en ``mode description'' pour produire une description de votre table. Si vous démarrez le serveur MySQL en utilisant l'option `--skip-external-locking` , `myisamchk` va rapporter une erreur si la table est modifiée durant l'exécution de la commande. Cependant, comme `myisamchk` ne modifie pas les tables, durant le mode description, il n'y a pas de risque de perte de données.
* myisamchk −d −v nom_de_table Pour produire plus d'informations durant l'exécution de `myisamchk` , ajoutez l'option `-v` pour indiquer qu'elle doit fonctionner en mode détaillé.
* myisamchk −eis nom_de_table Affiche les informations les plus importantes pour une table. C'est une commande lente, car elle doit lire toute la table.
* myisamchk −eiv nom_de_table C'est l'équivalent de `-eis` , mais qui vous indique ce qui se passe.

Exemple d'affichage résultant de `myisamchk -d` :

```
 MyISAM file:     company.MYI
Record format:   Fixed length
Data records:    1403698  Deleted blocks:         0
Recordlength:    226

 table description:
Key Start Len Index   Type
1   2     8   unique  double
2   15    10  multip. text packed stripped
3   219   8   multip. double
4   63    10  multip. text packed stripped
5   167   2   multip. unsigned short
6   177   4   multip. unsigned long
7   155   4   multip. text
8   138   4   multip. unsigned long
9   177   4   multip. unsigned long
    193   1           text
```

Exemple d'affichage résultant de `myisamchk -d -v` :

```
 MyISAM file:         company
Record format:       Fixed length
File-version:        1
Creation time:       1999-10-30 12:12:51
Recover time:        1999-10-31 19:13:01
Status:              checked
Data records:           1403698  Deleted blocks:              0
Datafile parts:         1403698  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     3
Max datafile length: 3791650815  Max keyfile length: 4294967294
Recordlength:               226

 table description:
Key Start Len Index   Type                 Rec/key     Root Blocksize
1   2     8   unique  double                     1 15845376      1024
2   15    10  multip. text packed stripped       2 25062400      1024
3   219   8   multip. double                    73 40907776      1024
4   63    10  multip. text packed stripped       5 48097280      1024
5   167   2   multip. unsigned short          4840 55200768      1024
6   177   4   multip. unsigned long           1346 65145856      1024
7   155   4   multip. text                    4995 75090944      1024
8   138   4   multip. unsigned long             87 85036032      1024
9   177   4   multip. unsigned long            178 96481280      1024
    193   1           text
```

Exemple d'affichage résultant de `myisamchk -eis` :

```
 Checking MyISAM file: company
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:  98%  Packed:   17%

 Records:          1403698    M.recordlength:     226
Packed:             0%
Recordspace used:     100%   Empty space:          0%
Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

 User time 1626.51, System time 232.36
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 627, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
```

```
Voluntary context switches 639, Involuntary context switches 28966
```

Exemple d'affichage résultant de `myisamchk -eiv` :

```
 Checking MyISAM file: company
Data records: 1403698   Deleted blocks:       0
- check file-size
- check delete-chain
block_size 1024:
index  1:
index  2:
index  3:
index  4:
index  5:
index  6:
index  7:
index  8:
index  9:
No recordlinks
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 2
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
- check data record references index: 3
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
- check data record references index: 5
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 6
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 7
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 8
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 9
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:   9%  Packed:   17%

 - check records and index references
[LOTS OF ROW NUMBERS DELETED]

 Records:          1403698    M.recordlength:     226   Packed:             0%
Recordspace used:     100%   Empty space:          0%  Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

 User time 1639.63, System time 251.61
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 10580, Swaps 0
Blocks in 4 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 10604, Involuntary context switches 122798
```

Voici les tailles des fichiers de données et d'index utilisés dans les tables précédentes :

```
 -rw-rw-r--   1 monty    tcx     317235748 Jan 12 17:30 company.MYD
-rw-rw-r--   1 davida   tcx      96482304 Jan 12 18:35 company.MYM
```

Des détails sur les types d'informations retournés par `myisamchk` sont listés ici. Le ``keyfile'' est le fichier d'index. ``Record'' et ``row'' sont synonymes de ligne :

- `ISAM file` Nom du fichier d'index ISAM.
- `Isam-version` Version du format ISAM. Actuellement, c'est toujours 2.
- `Creation time` Date de création du fichier de données.
- `Recover time` Date de dernière reconstruction du fichier de données ou d'index.
- `Data records` Combien de lignes sont stockées dans la table.
- `Deleted blocks` Combien de blocs effacés occupent toujours de l'espace. Vous pouvez optimiser la table pour récupérer cet espace. Optimisation de table .
- `Data file: Parts` Pour les tables au format de ligne dynamique, ceci indique combien de blocs de données sont présents. Pour une table optimisée sans lignes fragmentées, la valeur doit être égale à `Data records` .
- `Deleted data` Combien d'octets de données effacées et non réutilisées sont présents dans la table. Vous pouvez optimiser la table pour récupérer cet espace. Optimisation de table .
- `Data file pointer` La taille du pointeur de fichier de données, en octets. C'est généralement 2, 3, 4, ou 5 octets. La plupart des tables peuvent se gérer avec 2 octets, mais ceci ne peut être contrôlé par MySQL actuellement. Pour les tables à format de ligne fixe, c'est une adresse de ligne. Pour les tables dynamiques, c'est une adresse d'octet.
- `Keyfile pointer` La taille du pointeur de fichier d'index, en octets. C'est généralement 1, 2 ou 3 octets. La plupart des tables supportent 2 octets, mais cela est calculé automatiquement par

MySQL. C'est toujours une adresse de bloc.

- `Max datafile length` Taille maximale du fichier de données, en octets.
- `Max keyfile length` Taille maximale du fichier d'index, en octets.
- `Recordlength` Taille occupée par chaque ligne, en octets.
- `Record format` Le format utilisé pour stocker les lignes de la table. Les exemples ci−dessus utilisaient `Fixed length` . Les autres valeurs possibles sont `Compressed` et `Packed` .
- `table description` Une liste de toutes les clés de la table. Pour chaque clé, des informations de bas niveau sont présentées :
    - `Key` Le numéro d'index.
    - `Start` Où, dans la ligne, l'index débute.
    - `Len` Taille de cette partie d'index. Pour les nombres compactés, c'est toujours la taille maximale de la colonne. Pour les chaînes, c'est plus petit que la taille maximale de la colonne index, car vous pouvez indexer un préfixe de la chaîne.
    - `Index unique` et `multip.` (multiple). Indique si une valeur peut exister plusieurs fois dans cet index.
    - `Type` De quel type de données cet index est. C'est un type de données ISAM avec les options `packed` , `stripped` ou `empty` .
    - `Root` Adresse du premier bloc d'index.
    - `Blocksize` La taille de chaque bloc d'index. Par défaut, c'est 1024, mais cette valeur peut être modifiée lors de la compilation.
    - `Rec/key` C'est une valeur statistique, utilisée par l'optimiseur. Il indique combien de lignes sont disponibles par valeur de cette clé. Une clé unique aura toujours une valeur de 1. Cela peut être modifié une fois que la table est chargée (ou modifiée de façon majeure), avec la commande `myisamchk -a` . Si ce n'est pas mis à jour, une valeur par défaut de 30 est utilisée.

  Dans le premier exemple ci−dessus, la neuvième clé est une clé multi−partie, avec deux parties.
- `Keyblocks used` Quel pourcentage des blocs de clé est utilisé. Comme les tables utilisées dans les exemples ont tout juste été réorganisées avec `myisamchk` , ces valeurs sont très grandes (très proches du maximum théorique).
- `Packed` MySQL essaie de compacter les clés ayant un préfixe commun. Cela ne peut être utilisé que pour les colonnes de type `CHAR` / `VARCHAR` / `DECIMAL` . Pour les longues chaînes comme des noms, cette technique va significativement réduire l'espace utilisé. Dans le troisième exemple ci−dessus, la quatrième clé fait 10 caractères de long et a une réduction de 60 % dans l'espace utilisé effectivement.
- `Max levels` La profondeur du `B-tree` . Les grandes tables avec de longues clés peuvent obtenir de grandes valeurs.
- `Records` Combien de lignes sont enregistrées dans la table.
- `M.recordlength` La taille moyenne d'une ligne. Pour les tables avec un format de ligne statique, c'est la taille de chaque ligne.
- `Packed` MySQL efface les espaces à la fin des chaînes. `Packed` indique le pourcentage d'économie d'espace réalisé.
- `Recordspace used` Quel est le pourcentage d'utilisation du fichier de données.
- `Empty space` Quel est le pourcentage d'utilisation du fichier d'index.
- `Blocks/Record` Le nombre moyen de blocs par enregistrements (c'est à dire, de combien de liens une ligne fragmentées est constituée). C'est toujours 1.0 pour les tables à format de ligne statique. Cette valeur doit être aussi proche que possible de 1.0. Si elle grossit trop, vous pouvez réorganiser la table avec `myisamchk` . Optimisation de table .
- `Recordblocks` Combien de blocs sont utilisés. Pour les tables à format de ligne fixe, c'est le même nombre que le nombre de lignes.
- `Deleteblocks` Combien de blocs (liens) sont effacés.
- `Recorddata` Combien d'octets sont utilisés dans le fichier.
- `Deleted data` Combien d'octets dans le fichier de données sont effacés (inutilisés).
- `Lost space` Si une ligne est modifiée, et réduite en taille, de l'espace est perdu. Ce chiffre est la

somme de ces espaces perdus, en octets.
- `Linkdata` Lorsque le format de ligne dynamique est utilisé, les fragments de lignes sont liés avec des pointeurs de (4 à 7 octets chacun). `Linkdata` est la somme du stockage utilisé par ces pointeurs.

Si une table a été compressée avec `myisampack` , `myisamchk -d` affiche des informations supplémentaires à propos de chaque colonne. Voir `myisampack` , le générateur de tables MySQL compressées , pour un exemple de ces informations, et une description de leur signification.

5.7 Localisation MySQL et utilisation internationale

5.7.1 Le jeu de caractères utilisé pour les données et le stockage

Par défaut, MySQL utilise le jeu de caractères ISO−8859−1 (Latin1) avec tri en accord au Suédois/Finnois. C'est le jeu de caractère le mieux adapté pour les USA et l'Europe de l'ouest.

Tous les binaires standards MySQL sont compilés avec `--with-extra-charsets=complex` . Cela ajoutera du code à tous les programmes standards pour qu'ils puissent gérer `latin1` et tous les jeux de caractères multi−octets compris dans le binaire. Les autres jeux de caractères seront chargés à partir d'un fichier de définition de jeu si besoin.

Le jeu de caractères détermine quels caractères sont autorisés dans les noms et comment s'effectuent les tris dans les clauses `ORDER BY` et `GROUP BY` de la commande `SELECT` .

Vous pouvez changer le jeu de caractères avec l'option de démarrage du serveur `--default-character-set` . Les jeux de caractères disponibles dépendent des options `--with-charset=charset` et `--with-extra-charsets= list-of-charset | complex | all` de `configure` , et des fichiers de configuration de jeux de caractères situés dans `SHAREDIR/charsets/Index` . Les options habituelles de `configure` .

Depuis MySQL 4.1.1, vous pouvez aussi changer la collation du jeu de caractères avec l'option `--default-collation` lorsque le serveur démarre. La collation doit être valide pour le jeu de caractères par défaut. Utilisez la commande `SHOW COLLATION` pour déterminer quelles collations sont disponibles pour chaque jeu de caractères. Les options habituelles de `configure` .

Si vous changez le jeu de caractères lors de l'utilisation de MySQL (ce qui pourra aussi changer l'ordre de tri), vous devez exécuter `myisamchk -r -q --set-character-set=charset` sur toutes les tables. Sinon, vos index pourront ne pas être ordonnés correctement.

Lorsqu'un client se connecte à un serveur MySQL, le serveur envoie le jeu de caractères utilisé par défaut au client. Le client changera de jeu de caractères pour cette connexion.

Vous devez utiliser `mysql_real_escape_string()` pour protéger les chaînes pour une requête SQL. `mysql_real_escape_string()` est identique à l'ancienne fonction `mysql_escape_string()` , excepté qu'elle prend le gestionnaire de connexion `MYSQL` en tant que premier paramètre.

Si le client est compilé avec d'autres chemins que ceux où le serveur est installé et que la personne qui a configuré MySQL n'a pas inclut tous les jeux de caractères dans le binaire MySQL, vous devez indiquer au client où il peut trouver les jeux de caractères additionnels dont il aura besoin si le serveur utilise un autre jeu de caractères que le client.

On peut le spécifier en plaçant dans un fichier d'options MySQL :

```
[client]
character-sets-dir=/usr/local/mysql/share/mysql/charsets
```

où le chemin pointe vers le répertoire où les jeux de caractères dynamiques de MySQL sont stockés.On peut forcer le client à utiliser un jeu de caractères spécifique en précisant :

```
[client]
default-character-set=character-set-name
```

mais on n'en a normalement jamais besoin.

5.7.1.1 Jeu de caractères allemand

Pour obtenir l'ordre de tri Allemand, vous devez démarrer `mysqld` avec `--default-character-set=latin1_de` . Cela vous donnera les caractéristiques différentes.

- Lors du tri et de la comparaison des chaînes, les remplacements suivants sont faits dans la chaîne avant d'effectuer la comparaison :

```
ä  ->  ae
ö  ->  oe
ü  ->  ue
ß  ->  ss
```

- Tous les caractères accentués sont convertis en leur majuscule non−accentuée. Toutes les lettres sont transformées en majuscules.
- Lors de la comparaison des chaînes de caractères avec `LIKE` , la conversion un vers deux caractères n'est pas effectuée. Toutes les lettres sont transformées en majuscules. Les accents sont supprimés de toutes les lettres, à l'exception de : `Ü` , `ü` , `Ö` , `ö` , `Ä` et `ä` .

5.7.2 Langue des messages d'erreurs

`mysqld` peut émettre des messages d'erreurs dans les langues suivantes : Tchèque, Danois, Néerlandais, Anglais (par défaut), Estonien, Français, Allemand, Grec, Hongrois, Italien, Japonais, Coréen, Norvégien, Norwegian−ny, Polonais, Portugais, Roumain, Russe, Slovaque, Espagnol et Suédois.

Pour démarrer `mysqld` avec une langue particulière, utilisez soit l'option `--language=lang` , soit `-L lang` . Par exemple :

```
shell> mysqld --language=french
```

ou :

```
shell> mysqld --language=/usr/local/share/french
```

Notez que tout les noms de langue sont spécifiés en minuscule.Les fichiers de langue sont situés (par défaut) dans `Array` .

Pour modifier le fichier de messages d'erreurs, vous devez éditer le fichier `errmsg.txt` et exécuter la commande suivante pour générer le fichier `errmsg.sys` :

```
shell> comp_err errmsg.txt errmsg.sys
```

Si vous changez de version de MySQL, pensez à modifier le nouveau fichier `errmsg.txt` .

5.7.3 Ajouter un nouveau jeu de caractères

Cette section présente la procédure à suivre pour ajouter un autre jeu de caractères à MySQL. Vous devez avoir une distribution source pour suivre ces instructions.

Pour choisir la procédure adaptée, il faut savoir si le jeu de caractères est simple ou complexe :

- Si le jeu de caractères n'a pas besoin d'utiliser des routines de collations de chaînes spéciales pour le tri et n'a pas besoin du support des jeux de caractères multi−octets, il est simple.
- S'il a besoin de l'une de ces deux fonctionnalités, il est complexe.

Par exemple, `latin1` et `danish` sont des jeux de caractères simples tandis que `big5` et `czech` sont complexes.

Dans la section suivante, nous supposons que vous nommez votre jeu de caractères `MONJEU` .

Pour un jeu de caractères simple, effectuez ce qui suit :

- Ajoutez `MONJEU` à la fin du fichier `sql/share/charsets/Index` Assignez−lui un nombre unique.
- Créez le fichier `sql/share/charsets/MONJEU.conf` . (Vous pouvez vous inspirer de `sql/share/charsets/latin1.conf` .)La syntaxe pour le fichier est très simple :
    - Les commentaires commencent avec le caractère `'#'` et se terminent à la fin de la ligne.
    - Les mots sont séparés par un nombre changeant d'espaces blancs.
    - Lors de la définition d'un jeu de caractères, chaque mot doit être un nombre au format hexadécimal.
    - Le tableau `ctype` prends les 257 premiers mots. Les tableaux `to_lower[]` , `to_upper[]` et `sort_order[]` prennent chacun 256 mots après cela.

  Les tableaux de définition des caractères .
- Ajoutez le nom du jeu de caractères aux listes `CHARSETS_AVAILABLE` et `COMPILED_CHARSETS` dans `configure.in` .
- Reconfigurez, recompilez et testez.

Pour un jeu de caractères complexe faites ce qui suit :

- Créez le fichier `strings/ctype-MONJEU.c` dans la distribution des sources MySQL.
- Ajoutez `MONJEU` à la fin du fichier `sql/share/charsets/Index` . Assignez−lui un nombre unique.
- Regardez un des fichiers `ctype-*.c` existant pour voir ce qui doit être défini, par exemple, `strings/ctype-big5.c` . Notez que les tableaux dans votre fichier doivent avoir des noms tels que `ctype_MONJEU` , `to_lower_MONJEU` , etc. Cela correspond aux tableaux dans les jeux de caractères simples. Les tableaux de définition des caractères . Pour un jeu de caractère complexe
- Au début du fichier, placez un commentaire spécial comme celui−ci :

```
/*
* This comment is parsed by configure to create ctype.c,
* so don't change it unless you know what you are doing.
*
* .configure. number_MONJEU=MYNUMBER
* .configure. strxfrm_multiply_MONJEU=N
* .configure. mbmaxlen_MONJEU=N
*/
```

  Le programme `configure` utilise ce commentaire pour inclure automatiquement le jeu de caractères dans la bibliothèque MySQL.Les lignes `strxfrm_multiply` et `mbmaxlen` seront expliquées dans les sections suivantes. Ne les incluez que si vous avez besoin des fonctions d'assemblage des chaînes ou des fonctions de jeu de caractères multi−octets, respectivement.
- Vous devez alors créer les fonctions suivantes :
    - `my_strncoll_MONJEU()`
    - `my_strcoll_MONJEU()`
    - `my_strxfrm_MONJEU()`
    - `my_like_range_MONJEU()`

  Support d'assemblage de chaînes .
- Ajoutez le nom du jeu de caractères aux listes `CHARSETS_AVAILABLE` et `COMPILED_CHARSETS` dans `configure.in` .
- Reconfigurez, recompilez et testez.

Le fichier `sql/share/charsets/README` fournit plus d'instructions.

Si vous voulez qu'un jeu de caractères soit ajouté dans la distribution MySQL, envoyez un patch aux listes `internes` . Les listes de diffusion MySQL .

5.7.4 Le tableau de définition des caractères

`to_lower[]` et `to_upper[]` sont de simples tableaux qui contiennent les caractères minuscules et majuscules correspondant à chaque membre du jeu de caractère. Par exemple :

```
 to_lower['A'] doit contenir 'a'
to_upper['a'] doit contenir 'A'
```

`sort_order[]` est une carte indiquant comment les caractères doivent être ordonnés pour les comparaisons et les tris. Pour beaucoup de jeux de caractères, c'est la même chose que `to_upper[]` (ce qui signifie que le tri sera insensible à la casse). MySQL triera les caractères en se basant sur la valeur de `sort_order[caractère]` . Pour des règles de tri plus compliquées, voyez la discussion suivante sur l'assemblage des chaînes. Support d'assemblage de chaînes . `ctype[]` est un tableau de valeurs de bit, avec un élément par caractère. (Notez que `to_lower[]` , `to_upper[]` , et `sort_order[]` sont indexés par la valeur du caractère, mais que `ctype[]` est indexé par la valeur du caractère + 1. C'est une vielle habitude pour pouvoir gérer `EOF` .)

Vous pouvez trouver les définitions de bitmask suivantes dans `m_ctype.h` :

```
 #define _U      01      /* Majuscule */
#define _L      02      /* Minuscule */
#define _N      04      /* Numérique (nombre) */
#define _S      010     /* Caractère d'espacement */
#define _P      020     /* Ponctuation */
#define _C      040     /* Caractère de contrôle */
#define _B      0100    /* Blanc */
#define _X      0200    /* nombre hexadecimal */
```

L'entrée `ctype[]` de chaque caractère doit être l'union des valeurs de masque de bits qui décrivent le caractère. Par exemple, `'A'` est un caractère majuscule (`_U`) autant qu'une valeur hexadécimale (`_X`), et donc `ctype['A'+1]` doit contenir la valeur :

```
 _U + _X = 01 + 0200 = 0201
```

5.7.5 Support d'assemblage des chaînes

Si les règles de tri de votre langue sont trop complexes pour être gérées par le simple tableau `sort_order[]` , vous devez utiliser les fonctions d'assemblage de chaînes.

Jusqu'à présent, la meilleure documentation traitant de ce sujet est présente dans les jeux de caractères implémentés eux−mêmes. Regardez les jeux de caractères `big5` , `czech` , `gbk` , `sjis` , et `tis160` pour des exemples.

Vous devez spécifier la valeur de `strxfrm_multiply_MYSET=N` dans le commentaire spécial au début du fichier. `N` doit être le rationnel maximal vers lequel la chaîne pourra croître durant `my_strxfrm_MYSET` (cela doit être un entier positif).

5.7.6 Support des caractères multi−octets

Si vous voulez ajouter le support de jeu de caractères incluant des caractères multi−octets, vous devez utiliser les fonctions de caractères multi−octets.

Jusqu'à présent, la meilleure documentation traitant de ce sujet est présente dans les jeux de caractères implémentés eux−mêmes. Regardez les jeux de caractères `euc_kr` , `gb2312` , `gbk` , `sjis` , et `ujis` pour des exemples. Ils sont implémentés dans les fichiers `ctype-'charset'.c` dans le dossier `strings` .

Vous devez spécifier la valeur de `mbmaxlen_MYSET=N` dans le commentaire spécial en haut du fichier source. `N` doit être la taille en octet du caractère le plus large dans le jeu.

5.7.7 Problèmes avec les jeux de caractères

Si vous essayez d'utiliser un jeu de caractères qui n'est pas compilé dans votre exécutable, vous pouvez rencontrer différents problèmes :

- Votre programme a un chemin faux en ce qui concerne l'endroit où sont stockés les jeux de caractères. (Par défaut `/usr/local/mysql/share/mysql/charsets`). Cela peut être réparé en utilisant l'option `--character-sets-dir` du programme en question.
- Le jeu de caractères est un jeu de caractères multi−octets qui ne peut être chargé dynamiquement. Dans ce cas, vous devez recompiler le programme en incluant le support du jeu de caractère.
- Le jeu de caractères est un jeu de caractères dynamique, mais vous n'avez pas de fichier de configuration lui étant associé. Dans ce cas, vous devez installer le fichier de configuration du jeu de caractères à partir d'une nouvelle distribution MySQL.
- Votre fichier `Index` ne contient pas le nom du jeu de caractères.

  ```
  ERROR 1105: File '/usr/local/share/mysql/charsets/?.conf' not found
  (Errcode: 2)
  ```

  Dans ce cas, vous devez soit obtenir un nouveau fichier `Index` ou ajouter à la main le nom du jeu de caractères manquant.

Pour les tables `MyISAM` , vous pouvez vérifier le nom du jeu de caractères et son nombre associé d'une table avec `myisamchk -dvv nom_de_table` .

5.8 Les fichiers de log de MySQL

MySQL a plusieurs fichiers de log qui peuvent vous aider à savoir ce qui se passe à l'intérieur de `mysqld` :

Fichier	Description
Le log d'erreurs	Problèmes rencontrés lors du démarrage, de l'exécution ou du stoppage de mysqld .
Le log ISAM	Garde une trace des changements liés au tables ISAM . Utilisé uniquement pour déboguer le code ISAM .
Le log de requêtes	Connexions établies et requêtes exécutées.
Le log de mises à jour	Désapprouvé : Enregistre toutes les commandes qui changent les données.
Le log binaire	Enregistre toutes les commandes qui changent quelque chose. Utilisé pour la réplication.
Le log des requêtes lentes	Enregistre toutes les requêtes qui ont pris plus de long_query_time à s'exécuter ou celles qui n'ont pas utilisé d'index.

Par défaut, tous les fichiers de log peuvent être trouvés dans le dossier de données de `mysqld` . Vous pouvez forcer `mysqld` à rouvrir les fichiers de log (ou dans quelques cas à passer à un nouveau log) en exécutant `FLUSH LOGS` . Syntaxe de `FLUSH` .

Si vous utilisez les fonctionnalités de réplications de MySQL, les serveurs esclaves entretiennent un fichier de log supplémentaire, appelé log de relais. Réplication avec MySQL .

5.8.1 Le log d'erreurs

Le fichier d'erreurs contient les informations indiquant quand `mysqld` a été lancé et arrêté, ainsi que les erreurs critiques qui sont survenues lorsque le serveur fonctionnait.

Si `mysqld` s'arrête inopinément, et que `mysqld_safe` doit le relancer, `mysqld_safe` va écrire un message `restarted mysqld` dans le log d'erreurs. Si `mysqld` remarque qu'une table a besoin d'être réparée ou analysée, il écrit aussi un message dans le log d'erreurs.

Sur certains systèmes d'exploitation, le log d'erreur contient automatiquement une pile de trace si `mysqld` . La trace peut être utilisée pour déterminer quand `mysqld` . Utilisation d'un traçage de la pile .

Depuis MySQL 4.0.10, vous pouvez spécifier où `mysqld` stocke le fichier d'erreurs avec l'option `--log-error[=file_name]` . Si aucune valeur `file_name` n'est donnée, `mysqld` utilise le nom `host_name.err` et écrit le fichier dans le dossier de données. (Avant MySQL 4.0.10, le nom de fichier d'erreurs Windows était `mysql.err` .) Si vous exécutez `FLUSH LOGS` , le log d'erreur est renommé avec le suffixe `-old` et `mysqld` crée un nouveau fichier de log vide.

Dans les anciennes versions de MySQL sous Unix, le log d'erreur était géré par `mysqld_safe` qui redirigeait les erreurs vers `host_name.err` . Vous pouvez modifier le nom du fichier en spécifiant l'option `--err-log=filename` de `mysqld_safe` .

Si vous ne spécifiez pas `--log-error` , ou, sous Windows, si vous utilisez l'option `--console` , les erreurs sont écrites dans la sortie standard `stderr` . C'est généralement le terminal qui a lancé MySQL.

Sous Windows, les erreurs sont toujours écrites dans le fichier `.err` si `--console` n'est pas donné.

5.8.2 Le log général de requêtes

Si vous voulez savoir ce qui se passe à l'intérieur de `mysqld` , vous devez le démarrer avec `--log[=fichier]` . Cela aura pour effet d'écrire toutes les connexions et les requêtes dans le fichier de log (pas défaut nommé `'hostname'.log`). Ce log peut être très utile quand vous suspectez une erreur dans un client et voulez savoir exactement ce que `mysqld` pense que le client lui a envoyé.

Les anciennes versions du script `mysql.server` (de MySQL 3.23.4 à 3.23.8) passent à `safe_mysqld` une option `--log` (active le log général de requêtes). Si vous avez besoin de meilleurs performances lorsque vous démarrez MySQL dans un environnement de production, vous pouvez supprimer l'option `--log` de `mysql.server` ou la changer en `--log-bin` . Le log binaire des mises à jour .

`mysqld` écrit le scommandes dans le log de requêtes, dans l'ordre où il les recçoit. Cela peut être différent de l'ordre dans lequel elles sont exécutées. Cela est différent du log de modifications et du log binaire, qui sont toujours écrits après exécution, mais avant la libération des verrous.

Les redémarrage de serveur et les écritures de logs ne génèrent pas un nouveau fichier de log de requêtes (même si l'écriture des logs ferme puis ouvre à nouveau le fichier). Sous Unix, vous pouvez renommer le fichier et en créer un nouveau avec les commandes suivantes :

```
 shell> mv hostname.log hostname-old.log
shell> mysqladmin flush-logs
shell> cp hostname-old.log to-backup-directory
shell> rm hostname-old.log
```

Sous Windows, vous ne pouvez pas renommer le fichier de logs, alors que le serveur l'a ouvert. Vous devez arrêter le serveur, puis renommer le log. Puis, redémarrez le le serveur pour créer un nouveau log.

5.8.3 Le log de modification

Note : le log de modifications a été remplacé par le log binaire. Le log binaire des mises à jour . Avec ce nouveau log, vous pouvez faire tout ce que vous faisiez avec le log de modifications. **Le log de modifications n'est plus disponible depuis MySQL 5.0.0** .

Lors l'option `--log-update[=file_name]` est utilisée au démarrage, `mysqld` écrit un fichier de log contenant toutes les commandes SQL qui modifie les données. Si aucun fichier n'est spécifié, il prendra la valeur par défaut du nom de l'hôte. Si un fichier est spécifié mais qu'aucun chemin n'est indiqué, le fichier sera écrit dans le dossier de données. Si le fichier `file_name` n'a pas d'extension, `mysqld` va créer un fichier de log avec ce nom : `file_name.###` , où `###` est un nombre qui s'incrémente à chaque vois que vous exécutez la commande `mysqladmin refresh` , `mysqladmin flush-logs` , `FLUSH LOGS` ou que vous redémarrez le serveur.

Note : pour que la technique ci−dessus fonctionne, vous ne devez pas créer de fichiers avec le nom du fichier de log + une extension, qui pourrait être considérée comme un nombre, dans le dossier qui contient les log de modifications.

L'enregistrement dans le log de modification est fait juste après l'achèvement de la requête, mais avant la levée des verrous, et les validations. Cela garantit que la requête sera enregistrée.

Si vous voulez modifier une base grâce aux fichier de log de modification, vous pouvez utiliser la commande suivante (en supposant que vos fichiers de log de modification porte le nom de `file_name.###`) :

```
shell> ls −1 −t −r file_name.[0−9]* | xargs cat | mysql
```

`ls` est utilisé pour obtenir toute la liste des fichiers de logs du dossier.

Ceci peut être utile si vous devez repartir d'un fichier de sauvegarde après un crash, et que vous souhaitez re−exécuter les modifications qui ont eu lieu depuis la sauvegarde.

5.8.4 Le log binaire

Le log binaire a remplacé l'ancien log de modifications, qui ne sera plus disponible à partir de MySQL version 5.0. Le log binaire contient toutes les informations du log de modifications, dans un format plus efficace, et compatible avec les transactions.

Le log binaire, comme le log de modifications, contient toutes les requêtes qui modifient les données. Ainsi, une commande `UPDATE` ou `DELETE` avec une clause `WHERE` qui ne trouve aucune ligne ne sera pas écrite dans le log. Les commandes `UPDATE` qui donnent à une colonne sa valeur courante sont même évitées.

Le log binaire contient aussi des informations sur le temps d'exécution de la requête dans la base. Il ne contient que des commandes qui modifient des données. Si vous voulez avoir toutes les commandes (par exemple, si vous identifiez un problème de requête, vous devez utiliser le log de requête général. Le log général de requêtes .

Le but principal de ce log est de pouvoir reprendre les modifications de la base durant les opérations de restaurations, car le log binaire contiendra les requêtes qui ont eu lieu après une sauvegarde.

Le log binaire est aussi utilisé lorsque de la réplication d'un maître par un esclave. Réplication avec MySQL .

L'utilisation du log binaire ralentit le serveur d'environ 1%. Cependant, les avantages du log binaire durant les opérations de restauration et pour la réplication sont généralement plus intéressants.

Lorsque l'option de démarrage `--log-bin[=file_name]` est utilisée, `mysqld` écrit un fichier de log contenant toutes les commandes SQL qui modifient les données. Si aucun nom de fichier n'est donné, le nom de la machine hôte est utilisé, suivi de `-bin` . Si un nom est donné, mais qu'il ne contient pas de chemin, le fichier sera écrit dans le dossier de données.

Si vous fournissez une extension à `--log-bin=filename.extension` , l'extension sera automatiquement supprimée.

`mysqld` va ajouter une extension au nom du fichier de log binaire qui est un nombre automatiquement incrémenté chaque fois que vous exécutez `mysqladmin refresh` , `mysqladmin flush-logs` , `FLUSH LOGS` ou redémarrez le serveur. Un nouveau fichier de log sera automatiquement créé lorsque le fichier en cours atteint la taille de `max_binlog_size` . Un fichier de log binaire peut être plus grand que `max_binlog_size` si vous utilisez de grandes transactions : une transaction est écrite dans le log binaire d'un seul coup, et n'est jamais répartie entre plusieurs fichiers.

Pour être capable de faire la différence entre les fichiers de logs binaire utilisés, `mysqld` crée aussi un fichier d'index de logs, qui porte le même nom que le fichier de log, mais avec l'extension `'.index'` . Vous pouvez changer le nom du fichier de log avec l'option `--log-bin-index=[file_name]` . N'éditez pas manuellement ce fichier durant l'exécution de `mysqld` ; cela va induire `mysqld` en erreur.

Vous pouvez effacer tous les fichiers de log avec la commande `RESET MASTER` , ou seulement certains d'entre eux avec `PURGE MASTER LOGS` . Voyez Syntaxe de `RESET` et Commandes SQL pour contrôler les maîtres .

Vous pouvez utiliser les options suivantes avec `mysqld` pour modifier ce qui est enregistré dans le fichier de log :

`binlog-do-db=database_name`
: Indique au maître qu'il doit enregistrer les modifications si la base courante (c'est à dire, celle qui est sélectionnée par `USE`) est `db_name` . Toutes les autres bases de données qui ne sont pas explicitement mentionnées sont ignorées. Si vous utilisez cette option, assurez vous que vous ne faites des modifications que dans la base courante.Un exemple qui ne fonctionnera pas comme on pourrait l'attendre : Si le serveur est lancé avec l'option `binlog-do-db=sales` , et que vous utilisez `USE prices; UPDATE sales.january SET amount=amount+1000;` , cette commande ne sera pas écrite dans le fichier de log binaire.

`binlog-ignore-db=database_name`
: Indique au maître qu'il doit ne doit pas enregistrer les modifications si la base courante (c'est à dire, celle qui est sélectionnée par `USE`) est `db_name` . Si vous utilisez cette option, assurez vous que vous ne faites des modifications que dans la base courante.Un exemple qui ne fonctionnera pas comme on pourrait l'attendre : Si le serveur est lancé avec l'option `binlog-ignore-db=sales` , et que vous utilisez `USE prices; UPDATE sales.january SET amount=amount+1000;` , cette commande sera écrite dans le fichier de log binaire.

Pour ignorer ou forcer plusieurs bases, spécifiez l'option plusieurs fois, une fois par base.Les règles suivante sont utilisées dans l'ordre suivant, pour décider si la requête doit aller dans le log binaire ou pas :

* Y a−t−il des règles `binlog-do-db` ou `binlog-ignore-db` ?
    * Non : écrit la requête dans le log binaire, et quitte.
    * Oui : aller à l'étape suivante.
* Il y a des règles (`binlog-do-db` ou `binlog-ignore-db` ou les deux). Y a t il une base de données courante (une base sélectionnée avec la commande `USE`)?
    * Non : **N'écrit pas** la requête, et quitte.
    * Oui : aller à l'étape suivante.
* Il y a une base de données courante. Y a−t−il des règles `binlog-do-db` ?
    * Oui : Est−ce que la base de données courante vérifie une des règles `binlog-do-db` ?
        * Oui : écrit la requête dans le log binaire, et quitte.
        * Non : **N'écrit pas** la requête, et quitte.
    * Non : aller à l'étape suivante.
* Il y a des règles `binlog-ignore-db` . Est−ce que la base de données courante vérifie une des règles `binlog-ignore-db` ?
    * Oui : N'écrit pas la requête, et quitte.
    * Non : écrit la requête dans le log binaire, et quitte.

Par exemple, un esclave qui fonctionne avec l'option `binlog-do-db=sales` ne va pas écrire dans le log binaire les commandes qui concernent d'autres bases que `sales` (en d'autres termes, l'option `binlog-do-db` peut être considéré comme ``ignore les autres bases'').

Si vous utilisez la réplication, vous ne devez pas effacer les anciens log binaires jusqu'à ce que vous soyez sûrs que les esclaves n'en auront plus besoin. Une façon de faire cela est d'utiliser la commande `mysqladmin flush-logs` une fois par jour, et d'effacer les fichiers de log qui ont plus de trois jours. Vous pouvez les supprimer manuellement, ou utilisez de préférence la commande `PURGE MASTER LOGS TO` (Commandes de réplication) qui va aussi modifier le fichier de log binaires pour vous depuis MySQL 4.1.

Un client avec le droit de `SUPER` peut désactiver le log binaire pour ses commandes avec `SET SQL_LOG_BIN=0` . Syntaxe de `SET` .

Vous pouvez examiner le fichier de log binaire avec la commande `mysqlbinlog` . Par exemple, vous pouvez mettre à jour le serveur MySQL depuis la ligne de commande comme ceci :

```
shell> mysqlbinlog log-file | mysql -h server_name
```

Vous pouvez aussi utiliser le programme L'utilitaire binaire `mysqlbinlog` pour lire le fichier de log binaire directement dans le serveur MySQL.

Si vous utilisez les transactions, vous devez utiliser le fichier de log binaire pour les sauvegardes, plutôt que le vieux fichier de log de modifications.

L'enregistrement dans le fichier de log binaire est fait immédiatement après l'achèvement de la requête, mais avant la libération des verrous ou la validation de la requête. Cela garantit que les requêtes seront enregistrées dans l'ordre d'exécution.

Les modifications dans les tables non transactionnelles sont enregistrées dans le fichier de log binaire immédiatement après exécution. Pour les tables transactionnelles comme `BDB` ou `InnoDB` , toutes les modifications (`UPDATE` , `DELETE` ou `INSERT`) qui modifient les tables sont mises en cache jusqu'à ce qu'une commande `COMMIT` ne les envoie au serveur. A ce moment, `mysqld` écrit la totalité de la transaction dans le log binaire, avant d'appliquer la commande `COMMIT` . Tous les threads vont, au démarrage, allouer un buffer de la taille de `binlog_cache_size` octets pour enregistrer les requêtes. Si la requête est plus grande que ce buffer, le thread va ouvrir un fichier temporaire pour écrire la transaction. Le fichier temporaire sera supprimé dès que le thread se termine.

L'option `max_binlog_cache_size` (par défaut 4Go) peut être utilisé pour limiter la taille utilisée pour mettre en cache une transaction multi−requête. Si la transaction est plus grande que que cette taille, elle sera annulée.

Si vous utilisez les log de modification ou binaire, les insertions concurrentes seront converties en insertions normales lors de l'utilisation de `CREATE ... SELECT` ou `INSERT ... SELECT` . Cela garantit que vous pourrez recréer une copie exacte de la table en appliquant les même commandes sauvegardées.

Le format de log binaire est différent entre les versions 3.23, 4.0 et 5.0.0. Ces changements de formats sont nécessaires pour améliorer la réplication. MySQL 4.1 a le même format de log binaire que 4.0. Compatibilité de la réplication entre les versions de MySQL .

5.8.5 Le log des requêtes lentes

Lorsqu'il est démarré avec l'option `--log-slow-queries[=file_name]` , `mysqld` va écrire dans un fichier les requêtes SQL qui vont mettre plus de `long_query_time` secondes à s'exécuter. Le temps d'acquisition d'un verrou n'est pas compté.

Les requêtes lentes sont enregistrées après l'achèvement de l'exécution de la requête, et libération du verrou. Cela peut être différent de l'ordre dans lequel les commandes sont exécutées.

Si aucun nom de fichier n'est donné, le fichier de log prendra par défaut le nom de la machine, suffixé avec `-slow.log` . Si un nom de fichier est donné, mais qu'il manque le chemin, le fichier sera écrit dans le dossier de données.

Le log de requêtes lentes peut être utilisé pour repérer les requêtes qui prennent longtemps à s'exécuter, et donc, qui sont candidates à l'optimisation. Avec un grand fichier de log, cela peut devenir difficile. Vous pouvez alors passer le fichier de log à `mysqldumpslow` pour obtenir un sommaire des requêtes dans ce fichier.

Si vous utilisez l'option `--log-long-format` alors les requêtes qui n'utilisent pas d'index sont aussi enregistrées. Utiliser les options de ligne de commande .

5.8.6 Entretien des fichiers de log

Le serveur MySQL peut créer un grand nombre de fichiers de logs différents, qui permettent de suivre ce qui se passe. Fichiers de log . Vous devez toutefois nettoyer régulièrement ces fichiers, pour être sûr que les logs ne prennent pas tout le disque de la machine.

Lorsque vous utilisez MySQL avec des fichiers de log, vous voudrez, de temps en temps, supprimer ou sauvegarder les fichiers, et demander à MySQL d'utiliser de nouveaux fichiers. Sauvegardes de base de données .

Sous une installation Linux (`Redhat`), vous pouvez utiliser le script `mysql-log-rotate` pour cela. Si vous avez installé MySQL depuis une distribution `RPM` , le script doit avoir été installé automatiquement. Notez que vous devez être prudent avec cette commande si vous utilisez les logs pour la réplication.

Sur d'autres systèmes, vous devez installer un court script par vous même, qui sera exécuté via le démon `cron` .

Vous pouvez forcer MySQL à utiliser de nouveaux fichiers de log en utilisant la commande `mysqladmin flush-logs` ou avec la commande SQL `FLUSH LOGS` . Si vous utilisez MySQL version 3.21, vous devez utiliser `mysqladmin refresh` .

Les commandes ci−dessus effectue les tâche suivantes :

- Si le log standard (`--log`) ou le log de requêtes lentes (`--log-slow-queries`) est utilisé, la commande ferme et rouvre le fichier de log (`mysql.log` et `` `hostname`-slow.log `` par défaut).
- Si le log de modifications est utilisé (`--log-update`), la commande ferme le log de modification et ouvre un nouveau fichier, avec un nouveau numéro de séquence plus grand.

Si vous utilisez uniquement le log de modification, vous pour simplement vider les logs sur le disque, et sauver l'ancien fichier de modification dans une sauvegarde. Si vous utilisez le log normal, vous pouvez faire ceci :

```
 shell> cd mysql-data-directory
shell> mv mysql.log mysql.old
shell> mysqladmin flush-logs
```

puis faire une sauvegarde du fichier, et le supprimer (`mysql.old`).

5.9 Faire fonctionner plusieurs serveurs MySQL sur la même machine

Il y a des situations où vous souhaiterez avoir plusieurs serveurs MySQL sur la même machine. Par exemple, si vous voulez tester une nouvelle version du serveur avec votre configuration de production sans perturber votre installation de production. Ou bien, vous êtes un fournisseur de services Internet, et vous voulez fournir des installations distinctes pour des clients différents.

Si vous voulez exécuter plusieurs serveurs MySQL, le plus simple et de compiler les serveurs avec différents ports TCP/IP et fichiers de sockets pour qu'ils ne soient pas tous à l'écoute du même port ou de la même socket. Voir Utiliser les options de ligne de commande et Utiliser les fichiers d'options .

Au minimum, les options suivantes doivent être différentes sur chaque serveur :

`--port=port_num`
: `--port` contrôle le numéro de port des connexions TCP/IP.

`--socket=path`
: `--socket` contrôle le chemin de la socket sous Unix, et le nom du pipe nommé sous Windows. Sous Windows, il est nécessaire de spécifier un pipe distinct pour les serveurs qui supportent les connexions par pipes nommés.

`--shared-memory-base-name=name`
: Cette option ne sert actuellement que sous Windows. Elle désigne la portion de mémoire partagée, utilisée par Windows pour mettre aux clients de se connecter via la mémoire partagée. Cette option est nouvelle en MySQL 4.1.

`--pid-file=path`
: Cette option ne sert que sous Unix. Elle indique le nom du fichier dans lequel le serveur écrit l'identifiant de processus.

`--port` contrôle le numéro de port des connexions TCP/IP. `--socket` contrôle le chemin du fichier de socket sous Unix et le nom du pipe sous Windows. Il est nécessaire d'indiquer des noms de pipe différents sous Windows, uniquement si le serveur supporte les pipes nommés. `--shared-memory-base-name` désigne le nom du segment de mémoire partagée utilisé par un serveur Windows pour permettre à ses clients de se connecter via la mémoire partagée. `--pid-file` indique le nom du fichier sur lequel le serveur Unix écrit le numéro de processus.Si vous utilisez les options suivantes, elles doivent être différentes sur chaque serveur :

- `--log=path`
- `--log-bin=path`
- `--log-update=path`
- `--log-error=path`
- `--log-isam=path`
- `--bdb-logdir=path`

Si vous voulez plus de performances, vous pouvez aussi spécifier les options suivantes différemment pour chaque serveur, pour répartir la charge entre plusieurs disques physiques :

- `--tmpdir=path`
- `--bdb-tmpdir=path`

Avoir plusieurs dossiers temporaires comme ci−dessus est aussi recommandé car il est plus facile pour vous de savoir quel serveur MySQL aura créé quel fichier temporaire.

Généralement, chaque serveur doit aussi utiliser des options différentes pour les dossiers de données, qui est spécifié avec l'option `--datadir=path` .

Attention : normalement, vous ne devez pas avoir deux serveurs qui modifient en même temps les données dans les mêmes bases. Si votre OS ne supporte pas le verrouillage sans échec, cela peut vous mener à de déplaisantes surprises !

Cette mise en garde contre le partage de données entre deux serveur s'applique aussi aux environnements NFS. Permettre à plusieurs serveurs MYSQL d'accéder aux même données via NFS est une **mauvaise idée** !

- Le problème principale est que NFS devient rapidement une limitation en termes de vitesse. Il n'a pas été conçu pour cela.
- Un autre risque avec NFS est que vous devez vous assurer que deux serveurs n'interfèrent pas les uns avec les autres. Généralement, le verrouillage de fichier NFS est fait avec `lockd` , mais pour le moment, aucune plate−forme ne sera 100% fiable avec cette technique.

Simplifie vous la vie : évitez de partager le même dossier de données entre plusieurs serveurs, via NFS. Une solution plus intelligente est d'avoir un serveur avec plusieurs processeurs, et un système d'exploitation bien optimisé pour les threads.

Si vous avez plusieurs installations de serveurs MySQL a plusieurs endroits, vous pouvez spécifier le dossier de données pour chaque serveur avec l'option `--basedir=path` , pour que chaque serveur utilise des données différentes, des fichiers de log différents et un PID différent. Les valeurs par défaut de ces options sont relatives au dossier d'installation. Dans ce cas, la seule chose qui reste à adapter est la socket `--socket` et le `--port` . Par exemple, supposons que vous installez différentes versions de MySQL en utilisant la distribution binaire `.tar` . Elles vont être installées en différents dossiers, et vous pouvez lancer le serveur à chaque fois, avec le script `./bin/mysqld_safe` de chaque dossier respectif. `mysqld_safe` va déterminer la bonne option `--basedir` à passer à `mysqld` , et vous devrez spécifier `--socket` et `--port` à `mysqld_safe` .

Comme présenté dans les prochaines sections, il est possible de démarrer d'autres serveurs en modifiant des variables d'environnement ou en spécifiant les bonnes options de ligne de commande. Cependant, si vous devez lancer des serveurs de manière récurrente, il sera plus pratique de stocker ces valeurs dans le fichier de configuration.

5.9.1 Utiliser plusieurs serveurs MySQL un serveur Windows

Vous pouvez utiliser plusieurs serveur MySQL sur Windows, en lançant manuellement chacun d'entre eux en ligne de commande, avec ses paramètres de fonctionnement appropriés. Sur les systèmes Windows NT, vous pouvez aussi installer plusieurs serveurs comme services Windows, et les faire fonctionner de cette façon. Des instructions générales pour exécuter MySQL depuis la ligne de commande ou comme services sont données dans la section Installer MySQL sous Windows . Cette section décrit comme lancer chaque serveur, avec différentes valeurs de démarrage, qui doivent être uniques pour chaque serveur, à commencer par le dossier de données. Ces options sont décrites dans la section Utiliser plusieurs serveurs sur la même machine .

5.9.1.1 Lancer plusieurs serveurs depuis la console

Pour lancer plusieurs serveurs manuellement depuis la ligne de commande, vous pouvez spécifier les options appropriés en ligne de commande, ou dans un fichier d'options. Il est plus pratique de le faire dans un fichier d'option, mais il est nécessaire que chaque serveur utilise bien un fichier d'options différent. Pour cela, créez un fichier pour chaque serveur, et donner le nom du fichier au serveur avec l'option `--defaults-file` .

Supposez que vous voulez utiliser `mysqld` sur le port 3307 avec un dossier de données situé dans `C:\mydata1` , et `mysqld-max` sur le port 3308 avec un dossier de données situé dans `C:\mydata2` . Pour cela, créez deux fichiers d'options. Par exemple créez un fichier appelé `C:\my-opts1.cnf` qui ressemble à celui−ci :

```
[mysqld]
datadir = C:/mydata1
port = 3307
```

Créez un fichier appelé `C:\my-opts2.cnf` qui ressemble à celui−ci :

```
[mysqld]
datadir = C:/mydata2
port = 3308
```

Puis, lancez chaque serveur avec ses propres options :

```
shell> mysqld --defaults-file=C:\my-opts1.cnf
shell> mysqld-max --defaults-file=C:\my-opts2.cnf
```

(Sur NT, les serveurs vont se lancer en tâche de fond, et vous devrez ouvrir deux consoles pour lancer les deux commandes séparées).Pour éteindre ces deux serveurs, vous devez vos connecter au bon numéro de port :

```
shell> mysqladmin --port=3307 shutdown
shell> mysqladmin --port=3308 shutdown
```

Les serveurs configurés comme décrit ci−dessus permettent aux clients de se connecter via un réseau TCP/IP. Si vous voulez aussi utiliser les pipes nommés, utilisez les serveurs `mysqld-nt` ou `mysqld-max-nt` et spécifiez les options qui permettent d'activer les pipes nommés et leur nom. Chaque serveur qui supporte les pipes nommés doit avoir un nom de pipe unique. Par exemple, le fichier `C:\my-opts1.cnf` peut être écrit comme ceci :

```
[mysqld]
datadir = C:/mydata1
port = 3307
enable-named-pipe
socket = mypipe1
```

Puis, lancez le serveur comme ceci :

```
shell> mysqld-nt --defaults-file=C:\my-opts1.cnf
```

`C:\my-opts2.cnf` doit être modifié similairement pour l'utiliser avec le second serveur.

5.9.1.2 Lancer plusieurs serveurs comme services Windows

Sur les systèmes NT, MySQL peut fonctionner comme un service Windows. Les procédures pour installer, contrôler et supprimer un service MySQL sont décrites dans la section Lancer MySQL comme un service Windows .

Depuis MySQL 4.0.2, vous pouvez installer plusieurs serveurs sous forme de plusieurs services. Dans ce cas, vous devez vous assurer que chaque serveur utilise un nom de service différente, en plus de tous les autres paramètres qui doivent être uniques pour chaque instance.

Pour les instructions suivantes, supposez que vous vouez utiliser le serveur `mysqld-nt` avec différentes versions de MySQL, qui sont installées dans les dossiers `C:\mysql-4.0.8` et `C:\mysql-4.0.17` , respectivement. Cela peut être le cas si vous utilisez la version 4.0.8 comme serveur de production, mais que vous voulez tester la version 4.0.17 avant de l'utiliser.

Les règles suivantes sont applicables lors de l'installation d'un service MYSQL avec l'option `--install` :

- Si vous ne spécifiez aucun nom de service, le serveur utilise le nom de service par défaut de `MySQL` et le serveur lit les options dans le groupe `[mysqld]` , du groupe de fichiers d'options standard.
- Si vous spécifiez un nom de service après l'option `--install` , le serveur va ignorer le group d'options `[mysqld]` et lire les options dans le groupe qui a le même nom que le service. Les options seront lues dans le fichier d'options standard.
- Si vous spécifiez `--defaults-file` après un nom de service, le serveur va ignorer le fichier d'options standard, et lire les options dans le groupe `[mysqld]` du fichier ainsi nommé.

Note : avant MySQL 4.0.17, seul le serveur utilisant le nom de service par défaut (`MySQL`) ou un service installé explicitement avec le nom de `mysqld` lira le groupe d'options `[mysqld]` dans le fichier d'options. Depuis 4.0.17, tous les serveurs lisent le groupe `[mysqld]` s'ils lisent dans le fichier d'options standard, même si ils ont été installé avec un autre nom de service. Cela permet d'utiliser le groupe `[mysqld]` pour des options qui doivent être utilisées par tous les services MySQL, et un groupe d'options pour chaque service sera utilisé individuellement par chaque service.

En se basant sur les informations précédentes, vous avez plusieurs moyens pour installer des services Windows multiples. Les instructions suivantes décrivent certaines situations. Avant de tous les essayer, assurez vous de bien éteindre et supprimer tous les services MySQL existant.

- **Approche 1 :** Spécifiez les options de tous les services dans un fichier d'options. Pour cela, utilisez différent noms de services pour chaque serveur. Supposons que vous vouliez utiliser `mysqld-nt` 4.0.8 sous le nom de `mysqld1` et le `mysqld-nt` 4.0.17 sous le nom de `mysqld2` . Dans ce cas, vous pouvez utiliser le groupe `[mysqld1]` pour le serveur version 4.0.8 et le groupe `[mysqld2]` pour le serveur version 4.0.17. Par exemple, vous pourriez configurer votre fichier d'options `C:\my.cnf` ainsi :

```
 # options for mysqld1 service
[mysqld1]
basedir = C:/mysql-4.0.8
port = 3307
enable-named-pipe
socket = mypipe1

 # options for mysqld2 service
[mysqld2]
basedir = C:/mysql-4.0.17
port = 3308
enable-named-pipe
socket = mypipe2
```

  Installez les services comme ceci, en utilisant les noms de chemin complet pour vous assurer que Windows trouve les bons programmes pour chaque service :

```
 shell> C:\mysql-4.0.8\bin\mysqld-nt --install mysqld1
shell> C:\mysql-4.0.17\bin\mysqld-nt --install mysqld2
```

  Pour lancer les services, utilisez le gestionnaire de services, ou bien utilisez la commande `NET START` avec les bons noms de services :

```
 shell> NET START mysqld1
shell> NET START mysqld2
```

Pour lancer les services, utilisez le gestionnaire de services, ou bien utilisez la commande `NET STOP` avec les bons noms de services :

```
shell> NET STOP mysqld1
shell> NET STOP mysqld2
```

- **Approche 2 :** Spécifiez les options de chaque serveur dans un fichier séparé, et utilisez l'option `--defaults-file` lorsque vous installez le service pour dire au serveur quel fichier utiliser. Dans ce cas, chaque fichier doit contenir les options du groupe `[mysqld]` .

  Avec cette approche, pour spécifier les options du serveur `mysqld-nt` 4.0.8, il faut créer un fichier `C:\my-opts1.cnf` qui ressemble à ceci :

```
[mysqld]
basedir = C:/mysql-4.0.8
port = 3307
enable-named-pipe
socket = mypipe1
```

  Pour la version 4.0.17 de `mysqld-nt` , créez un fichier `C:\my-opts2.cnf` qui ressemble à ceci :

```
[mysqld]
basedir = C:/mysql-4.0.17
port = 3308
enable-named-pipe
socket = mypipe2
```

  Installez les services comme suit (entrez chaque commande comme une seule ligne) :

```
shell> C:\mysql-4.0.8\bin\mysqld-nt --install mysqld1
          --defaults-file=C:\my-opts1.cnf
shell> C:\mysql-4.0.17\bin\mysqld-nt --install mysqld2
          --defaults-file=C:\my-opts2.cnf
```

  Pour utiliser l'option `--defaults-file` lors de l'installation du serveur MySQL comme service, vous devez la faire précéder du nom de service.

  Après avoir installé les services, lancez et arrêtez les services de la même façon que dans l'exemple précédent.

Pour supprimer plusieurs services, utilisez la commande `mysqld --remove` pour chacun d'entre eux, en spécifiant le nom du service, suivi de l'option `--remove` si le service a un nom qui n'est pas le nom par défaut.

5.9.2 Utiliser plusieurs serveurs sous Unix

Le plus simple pour utiliser plusieurs serveurs sous Unix, est de le compiler avec différents ports TCP/IP et sockets pour que chacun puisse utiliser une interface réseau différente. De plus, en compilant le serveur dans différents dossier de base, cela conduit automatiquement à la configuration de différents dossiers de données, fichiers de logs, et PID pour chaque serveur.

Supposons que le serveur existant est configuré avec le numéro de port (3306) et le fichier de socket par défaut (`/tmp/mysql.sock`). Pour configurer un nouveau serveur en ayant des paramètres opératoires différents, vous pouvez utiliser le script de configuration `configure` avec les options suivantes :

```
shell> ./configure --with-tcp-port=port_number \
            --with-unix-socket-path=file_name \
            --prefix=/usr/local/mysql-4.0.17
```

Ici, `port_number` et `file_name` doivent être différents des valeurs par défaut de numéro de port et de chemin. La valeur `--prefix` doit spécifier un dossier d'installation différent de celui dans lequel le serveur existant est installé.

Si vous avez un serveur MySQL qui écoute sur un port donné, vous pouvez utiliser la commande suivante pour connaître ses caractéristiques, y compris son dossier de base et son fichier de socket :

```
shell> mysqladmin --host=host_name --port=port_number variables
```

Avec les informations affichées par la commande, vous pouvez savoir quelles valeurs **ne doivent pas être utilisées** lors de la configuration du nouveau serveur.

Notez que si vous spécifiez `` `localhost` '' comme nom d'hôte, `mysqladmin` va utiliser par défaut une socket Unix plutôt que TCP/IP. En MySQL 4.1, vous pouvez explicitement spécifier le protocole de connexion avec l'option `--protocol={TCP | SOCKET | PIPE | MEMORY}` .

Vous n'avez pas à compiler un nouveau serveur MySQL pour le lancer avec un numéro de port et une socket différente. Il est aussi possible de spécifier ces valeurs au moment du démarrage. Une méthode pour faire cela est d'utiliser les options de ligne de commande :

```
shell> /path/to/mysqld_safe --socket=file_name --port=port_number
```

Pour utiliser un dossier de données différent, utilisez l'option `--datadir=path` à `mysqld_safe` .Un autre moyen pour arriver au même résultat est d'utiliser les variables d'environnement pour spécifier le nom de la socket et le numéro de port.

```
shell> MYSQL_UNIX_PORT=/tmp/mysqld-new.sock
shell> MYSQL_TCP_PORT=3307
shell> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
shell> scripts/mysql_install_db
shell> bin/mysqld_safe &
```

C'est une méthode rapide pour lancer un second serveur pour le tester. Le plus agréable de cette méthode est que les variables d'environnement vont être adoptées par les logiciels clients que vous invoquerez avec le même Shell. Par conséquent, les connexions seront automatiquement dirigées vers le nouveau serveur. Variables d'environnement MySQL inclut une liste des variables d'environnement que vous pouvez utiliser pour affecter `mysqld` .

Pour les scripts de lancement automatique, votre script de démarrage qui est exécuté au démarrage doit utiliser la commande suivante avec les options appropriées pour chaque serveur :

```
mysqld_safe --defaults-file=path-to-option-file
```

Chaque fichier d'options doit contenir les valeurs spécifique du serveur.

Sous Unix, le script `mysqld_multi` est une autre méthode pour lancer plusieurs serveurs. `mysqld_multi` , un programme pour gérer plusieurs serveurs MySQL .

5.9.3 Utiliser les clients dans un environnement multi−serveur

Lorsque vous voulez connecter un client à un serveur MySQL qui écoute sur différentes interfaces réseau que celles que vous utilisez sur votre client, vous devez utiliser les méthodes suivantes :

- Lancez le client avec les options `--host=host_name --port=port_number` pour vous connecter via TCP/IP sur un hôte distant, ou avec `--host=localhost --socket=file_name` pour vous connecter localement, via les sockets Unix, ou un pipe nommé Windows.
- Depuis MySQL 4.1, lancez le programme avec l'option `--protocol=tcp` pour vous connecter via TCP/IP, `--protocol=socket` pour vous connecter via les socket Unix, `--protocol=pipe` pour vous connecter via un pipe nommé, ou `--protocol=memory` pour vous connecter via la mémoire partagée. Pour TCP/IP, vous aurez peut être besoin d'indiquer les options d'hôte `--host` et de port `--port` . Pour les autres types de connexion, vous aurez peut être besoin de spécifier l'option `--socket` pour indiquer la socket ou le pipe nommé, ou encore `--shared-memory-base-name` pour

spécifier la mémoire partagée.

- Sous Unix, configurez les variables d'environnement `MYSQL_UNIX_PORT` et `MYSQL_TCP_PORT` pour qu'elles pointent sur la socket Unix et le port TCP/IP que vous voulez, avant de lancer le client. Si vous utilisez normalement une socket ou un port spécifique, vous pouvez placer des commandes pour configurer ces variables dans votre script `.login` , afin que vous les ayez à chaque connexion. Variables d'environnement MySQL .
- Spécifiez la socket par défaut et le port TCP/IP dans le groupe d'options `[client]` du fichier d'options. Par exemple, vous pouvez utiliser `C:\my.cnf` sur Windows, ou `.my.cnf` dans votre dossier racine sous Unix. Utiliser les fichiers d'options .
- Dans un programme C, vous pouvez spécifier le port ou la socket dans l'appel à `mysql_real_connect()` . Vous pouvez aussi faire que le programme lise des fichiers d'options en utilisant la fonction `mysql_options()` . Description des fonctions C .
- Si vous utilisez le module `DBD::mysql` , vous pourrez lire les options dans les fichiers d'options MySQL. Par exemple :

```
 $dsn = "DBI:mysql:test;mysql_read_default_group=client;"
        . "mysql_read_default_file=/usr/local/mysql/data/my.cnf";
$dbh = DBI->connect($dsn, $user, $password);
```

  Interface Perl .

5.10 Cache de requêtes MySQL

Depuis la version 4.0.1, le `MySQL server` bénéficie d'un `cache de requêtes` . En fait, le cache sauvegrade le texte d'une requête `SELECT` avec le résultat qui a été envoyé au client. Si une requête identique est appelée par la suite, le serveur retournera le résultat à partir du cache plutôt que d'analyser puis exécuter la requête à nouveau.

Le cache de requêtes est extrêmement utile dans un environnement où les tables ne changent pas souvent, et que vous avez de nombreuses requêtes identiques. C'est la situation classique des serveurs Web, qui génèrent beaucoup de pages dynamiques à partir du même contenu.

Note: The query cache does not return stale data. When tables are modified, any relevant entries in the query cache are flushed.

Note : Le cache de requêtes ne retourne pas de données périmées. A chaques fois que les données sont modifiées, les entrées correspondantes dans le cache sont effacées.

Voici quelques performances du cache de requêtes. (Ces résultats on été générés en utilisant la suite benchmark MySQL sur un Linux Alpha 2 x 500 MHz avec 2GB RAM et un cache de requêtes de 64MB) :

- Si toutes les requêtes que vous effectuez sont simples (comme séléctionner un champ d'une table n'en contenant qu'un) mais diffèrent d'une manière que toutes les requêtes ne peuvent être cachées, le gain lors de l'utilisation du cache est de 13%. Cela peut être considéré comme le pire des cas. En réalité, les requêtes sont plus compliquées que notre exemple le gain est donc plus petit.
- Les recherches sur une colonne dans une table n'en contenant qu'une sont 238% plus rapides. Cela peut être considéré comme le gain minimal à attendre pour une requête cachée.

Si vous ne voulez pas utiliser le cache de requêtes paramétrez `query_cache_size` à zéro. En désactivant le cache de requête, il n'y a aucune surcharge apparente. (le cache de requêtes peut être désactivé à l'aide de l'option de configuration `--without-query-cache`)

5.10.1 Comment fonctionne le cache de requêtes

Cette section décrit le fonctionnement du cache de requêtes lorsque ce dernier est opérationnel. La section Configuration du cache de requêtes décrit comment contrôler ce cache, qu'il soit opérationnel ou pas.

Les requêtes sont comparées avant analyse, alors les deux requêtes suivantes seront considérées comme différentes pour le cache de requête :

```
SELECT * FROM tbl_name
Select * from tbl_name
```

Les requêtes doivent être les mêmes (caractère à caractère) pour être considérées comme identiques. Les requêtes qui utilisent différentes bases de données, différents protocoles ou différents jeux de caractères sont alors considérées comme différentes, et mises en cache différemment.Si un résultat de requête a été retourné depuis le cache de requête, alors la variable `Com_select` ne sera pas incrémenté, mais `Qcache_hits` le sera. Status et maintenance du cache de requêtes .

Si une table change (`INSERT` , `UPDATE` , `DELETE` , `TRUNCATE` , `ALTER` ou `DROP TABLE|DATABASE`), alors toutes les requêtes mises en cache qui utilisaient cette table deviennent obsolètes et en sont retirées.

Les tables transactionnelles `InnoDB` qui ont été modifiées seront rendues obsolètes lorsqu'un `COMMIT` sera exécuté.

En MySQL 4.0, le cache de requêtes est désactivé dans les transactions : elles ne retourne pas le résultats. Depuis MySQL 4.1.1, le cache de requête fonctionne aussi dans les transactions avec les tables `InnoDB` : le serveur utilise le numéro de version de table pour détecter si le contenu est à jour ou non.

Avant MySQL 5.0, une requête qui commence avec un commentaire peut être mise en cache, mais ne sera jamais lue depuis le cache. Ce problème est résolu en MySQL 5.0.

Le cache de requête fonctionne pour les requêtes `SELECT SQL_CALC_FOUND_ROWS ...` et `SELECT FOUND_ROWS()` . Les requêtes `FOUND_ROWS()` retournent la valeur correcte même si la requête précédent a aussi été lue dans le cache, car le nombre de lignes lues est conservé dans le cache.

Une requête ne peut être mise en cache si elle contient l'une des fonctions suivantes :

BENCHMARK()	CONNECTION_ID()	CURDATE()
CURRENT_DATE()	CURRENT_TIME()	CURRENT_TIMESTAMP()
CURTIME()	DATABASE()	ENCRYPT() avec un paramètre
FOUND_ROWS()	GET_LOCK()	LAST_INSERT_ID()
LOAD_FILE()	MASTER_POS_WAIT()	NOW()
RAND()	RELEASE_LOCK()	SYSDATE()
UNIX_TIMESTAMP() sans paramètre	USER()	

Une requête ne sera pas mise en cache dans ces conditions :

- Elle contient des fonctions définies par l'utilisateur : `UDF` .
- Elle contient des variables utilisateur.
- Elle fait référence à des tables de la base `mysql` .
- Elle est de l'une des formes suivantes :

  ```
  SELECT ... IN SHARE MODE
  SELECT ... INTO OUTFILE ...
  SELECT ... INTO DUMPFILE ...
  SELECT * FROM ... WHERE autoincrement_col IS NULL
  ```

  La dernière forme n'est pas mise en cache, car elle est utilisée comme palliatfi pour ODBC, afin d'obtenir la dernière valeur insérée. Comment obtenir la valeur d'une colonne `AUTO_INCREMENT` avec ODBC .

- Elle utilise une table TEMPORARY .
- Elle n'utilise pas de tables.
- L'utilisateur a un droit de niveau colonne pour l'une des tables impliquée.
- Avant la lecture de la requête dans le cache de requête, MySQL vérifie que l'utilisateur a les droits SELECT pour toutes les bases de données impliquées. Si ce n'est pas le cas, le résultat n'est pas utilisé.

5.10.2 Options relatives au cache de requêtes dans un SELECT

Il y a deux options relatives au cache de requêtes qui peuvent être utilisées dans une requête SELECT :

SQL_CACHE
: Le résultat de la requête est en cache si la valeur de la variable système query_cache_type est à ON ou DEMAND .

SQL_NO_CACHE
: Le résultat de la requête n'est pas mis en cache.

Exemples:

```
 SELECT SQL_CACHE id, name FROM customer;
SELECT SQL_NO_CACHE id, name FROM customer;
```

5.10.3 Configuration du cache de requêtes

La variable système serveur have_query_cache indique si le cache est actif :

```
 mysql> SHOW VARIABLES LIKE 'have_query_cache';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| have_query_cache | YES   |
+------------------+-------+
```

Le cache de requête ajoute plusieurs variables système MySQL liées à mysqld qui peuvent être spécifiées dans un fichier de configuration, en ligne de commande lors du démarrage de mysqld . Les variables systèmes liées au cache sont toutes préfixées avec query_cache_ . Elles sont décrites brièvement dans la section Variables sytème du serveur , avec des informations complémentaires.Pour configurer la taille du cache de requêtes, modifiez la variable système query_cache_size . En lui donnant la valeur de 0, vous le désactivez. La taille par défaut du cache est de 0 : le cache est désactivé par défaut.

Si le cache de requête est actif, la variable query_cache_type modifie son comportement. Cette variable peut prendre les valeurs suivantes :

- La valeur de 0 ou OFF empêche la mise en cache ou la lecture de résultats en cache.
- La valeur de 1 ou ON permet le cache, sauf pour les commandes qui commencent par SELECT SQL_NO_CACHE .
- La valeur de 2 ou DEMAND impose la mise en cache de toutes les requêtes, même celles qui commencent par SELECT SQL_CACHE .

Modifier la valeur GLOBAL de query_cache_type détermine le comportement du cache pour tous les clients qui se connecteront après la modification. Les clients individuels peuvent modifier le comportement du cache pour leur connexion avec l'option SESSION de query_cache_type . Par exemple, un client peut désactiver le cache de requête pour ses propres requêtes avec :

```
 mysql> SET SESSION query_cache_type = OFF;
```

Pour contrôler la taille maximale des résultats de requêtes qui peuvent être mis en cache, il faut modifier la valeur de la variable `query_cache_limit` . La valeur par défaut de 1Mo.Le résultat d'une requête (les données envoyées au client) sont stockées dans le cache durant la lecture. Par conséquent, les données ne sont pas manipulées en un seul gros morceau. Le cache de requête alloue des blocs à la demande, pour stocker les données, et dès qu'un bloc est rempli, un autre est alloué. Comme l'allocation de mémoire est une opération coûteuse (en temps), le cache de requêtes crée des blocs avec une taille minimale de `query_cache_min_res_unit` , jusqu'à la taille des données à mettre en cache. Suivant le type de requêtes exécutées, vous pourrez adapter la valeur de la variable `query_cache_min_res_unit` :

- La valeur par défaut de `query_cache_min_res_unit` est 4Ko. Cela doit être adapté la plupart des situations.
- Si vous avez beaucoup de requêtes avec de petits résultats, la taille par défaut sera un peu grande, et conduit à une fragmentation inutile de la mémoire, indiquée par un grand nombre de blocs libres. La fragmentation va forcer le cache à effacer d'anciennes requêtes pour libérer de la place. Dans ce cas, réduisez la valeur de `query_cache_min_res_unit` . Le nombre de blocs libres et de requêtes supprimées pour libérer de la place sont stockées dans les variables `Qcache_free_blocks` et `Qcache_lowmem_prunes` .
- Si la plupart de vos requêtes ont de grands résultats (vérifiez les variables `Qcache_total_blocks` et `Qcache_queries_in_cache`), vous pouvez augmenter la valeur de `query_cache_min_res_unit` . Cependant, soyez prudent de ne pas aggrandir trop la valeur (voir point précédent).

`query_cache_min_res_unit` a été ajoutée en MySQL 4.1.

5.10.4 Statut du cache de requêtes et maintenance

Vous pouvez vérifier que vous avez le cache de requête sur MySQL avec la commande suivante :

```
 mysql> SHOW VARIABLES LIKE 'have_query_cache';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| have_query_cache | YES   |
+------------------+-------+
1 row in set (0.00 sec)
```

Avec la commande `FLUSH QUERY CACHE` , vous pouvez défragmenter le cache de requêtes pour mieux en utiliser la mémoire. Cette commande n'effacera aucune requête du cache.

La commande `RESET QUERY CACHE` efface tous les résultats de requêtes du cache. `FLUSH TABLES` aussi.

Vous pouvez visualiser les performances du cache de requêtes avec `SHOW STATUS` :

```
 mysql> SHOW STATUS LIKE 'Qcache%';
+-------------------------+--------+
| Variable_name           | Value  |
+-------------------------+--------+
| Qcache_free_blocks      | 36     |
| Qcache_free_memory      | 138488 |
| Qcache_hits             | 79570  |
| Qcache_inserts          | 27087  |
| Qcache_lowmem_prunes    | 3114   |
| Qcache_not_cached       | 22989  |
| Qcache_queries_in_cache | 415    |
| Qcache_total_blocks     | 912    |
+-------------------------+--------+
```

La description de chaque variable est présentée dans la section Variables de statut du serveur . Certaines utilisations sont présentées ici.Le nombre total de commandes `SELECT` vaut :

```
   Com_select
+ Qcache_hits
+ requêtes avec une erreur
```

La valeur de `Com_select` est :

```
   Qcache_inserts
+ Qcache_not_cached
+ erreurs de droits d'accès ou de colonnes
```

Le cache de requêtes utilise des blocs de longueur variable, ce qui fait que `Qcache_total_blocks` et `Qcache_free_blocks` peuvent indiquer une fragmentation de la mémoire du cache. Après un appel à `FLUSH QUERY CACHE` un seul (grand) bloc libre subsiste.

Note : Chaque requête a besoin au minimum de deux blocs (un pour le texte de la requête et un autre, ou plus, pour le résultat). De même, chaque table utilisée par une requête a besoin d'un bloc, mais si deux ou plusieurs requêtes utilisent la même table, seul un bloc a besoin d'être alloué.

Vous pouvez aussi utiliser la variable `Qcache_lowmem_prunes` pour ajuster la taille du cache de requêtes. Elle compte le nombre de requêtes qui ont été supprimées du cache pour libérer de la place pour les nouvelles requêtes. Le cache de requêtes utilise une stratégie du type `la plus anciennement utilisée` (`least recently used ou LRU`) pour décider de quelle requête doit être retirée. Des informations de paramétrage sont données dans la section Configuration du cache de requêtes .

6 Réplication de MySQL

Les possibilités de réplication, permettant à un serveur d'être recopié à l'identique sur un autre serveur, ont été introduites en MySQL 3.23.15. Cette section décrit les différentes fonctionnalités de la réplication MySQL. Elle sert de référence pour les options disponibles avec la réplication. Vous y trouverez une introduction à la réplication.

Vers la fin, vous y trouverez les questions et problèmes les plus fréquents, avec leur solution.

Pour une description de la syntaxe des commandes de réplication, voyez Commandes de réplication .

Nous vous suggérons de visiter notre site web http://www.mysql.com/ souvent pour y lire les mises à jour de cette section. La réplication est constamment améliorée, et nous modifions souvent le manuel.

6.1 Introduction à la réplication

Depuis la version 3.23.15, MySQL supporte la réplication unidreictionnelle interne. Un serveur sert de maître, et les autres serveurs servent d'esclaves. Le serveur entretient des logs binaires de toutes les modifications qui surviennent. Il entretient aussi un fichier d'index des fichiers de logs binaires, pour garder la trace de la rotation des logs. Chaque esclave, après connexion réussie au serveur maître, indique au maître le point qu'il avait atteint depuis la fin de la dernière réplication, puis rattrappe les dernières modifications qui ont eu lieu, puis se met en attente des prochains événements en provenance du maître.

Un esclave peut aussi servir de maître à son tour, pour réaliser une chaîne de réplication.

Notez que lorsque vous utilisez la réplication, toutes les modifications de tables sont répliquées, et doivent intervenir sur le serveur maître. Sinon, vous devez être prudents dans vos interventions, pour ne pas créer de conflits entre les modifications de tables sur le maître et celles qui interviennent sur l'esclave.

La réplication unidirectionnelle permet de renforcer la robustesse, la vitesse et l'administration du serveur :

- La robustesse est augmentée par la configuration maître/esclave. Dans le cas où un problème survient sur le maître, vous pouvez utiliser un esclave comme serveur de secours.
- L'accélération provient de la répartition de la charge de traitement des requêtes clients entre le maître et les esclaves, permettant un meilleur temps de réponse. Les requêtes `SELECT` peuvent être envoyées aux esclaves pour réduire la charge du maître. Les requêtes de modifications des données sont envoyées au maître, qui les transmettra aux esclaves. Cette stratégie de répartition de charge est efficace si les lectures sont plus nombreuses que les écritures, ce qui est la situation la plus courante.
- Un autre avantage de la réplication est que vous pouvez faire des sauvegardes non−bloquantes de vos données sur l'esclave et non plus sur le serveur principal : ce dernier n'est pas perturbé. Sauvegardes de base de données .

6.2 Présentation de l'implémentation de la réplication

La réplication MySQL est basée sur le fait que le serveur va garder la trace de toutes les évolutions de vos bases (modifications, effacements, etc.) dans un fichier de log binaire et les esclaves vont lire les requêtes du maître dans ce fichier de log, pour pouvoir exécuter les mêmes requêtes sur leurs copies. Le log binaire des mises à jour .

Il est **très important** de comprendre que le fichier de log binaire est simplement un enregistrement des modifications depuis un point fixe dans le temps (le moment où vous activez le log binaire). Tous les esclaves que vous activez auront besoin de la copie des données qui existaient au moment du démarrage du log. Si vous démarrez vos esclaves sur sans qu'ils ne disposent des données identiques à celles du maître **au moment du démarrage du log binaire** , votre réplication va échouer.

Depuis la version 4.0.0, vous pouvez utiliser la commande `LOAD DATA FROM MASTER` pour configurer un esclave. Soyez bien conscient qu'actuellement, `LOAD DATA FROM MASTER` ne fonctionne que si toutes les tables du maître sont du type `MyISAM` , et qu'il est possible d'obtenir un verrou de lecture global, pour qu'aucune lecture ne se fasse durant le transfert des tables depuis le maître. Cette limitation est de nature temporaire, et elle est dûe au fait que nous n'avons pas encore programmé un système de sauvegarde des tables sans verrou. La limitation sera supprimée dans la future version 4.0 une fois que nous aurons programmé le système de sauvegarde, qui permettra à `LOAD DATA FROM MASTER` de fonctionner sans bloquer le maître.

Etant donné la limitation ci−dessus, nous vous recommandons actuellement d'utiliser la commande `LOAD DATA FROM MASTER` uniquement si le jeu de données du maître est petit, ou si un verrou prolongé sur le maître est acceptable. Suivant la vitesse de lecture de `LOAD DATA FROM MASTER` en fonction des systèmes, une règle de base indique que le transfert se fera au rythme de 1 Mo par seconde. Vous pourrez ainsi obtenir une estimation du temps qu'il vous faudra pour transférer les données, si le maître et l'esclave sont connectés sur un réseau de 100 MBit/s, avec des configurations à base de Pentium 700 MHz. Bien sur, votre cas particulier pourra varier en fonction de votre système : la règle ci−dessus vous donnera une première évaluation du temps à attendre.

Une fois que l'esclave est correctement configuré, et qu'il fonctionne, il va simplement se connecter au maître et attendre des requêtes de modifications. Si le maître est indisponible ou que l'esclave perd la connexion avec le maître, il va essayer de se reconnecter toutes les `master-connect-retry` secondes jusqu'à ce qu'il soit capable d'établir la communication, et de recommencer à appliquer les modifications.

Chaque esclave garde la trace du point où il en était rendu. Le serveur maître n'a pas de notions du nombre d'esclave qui se connectent, ou qui sont à jour à un moment donné.

6.3 Détails d'implémentation de la réplication

Les capacités de réplication de MySQL sont implémentées à l'aide de trois threads : un thread sur le maître et deux sur l'esclave. Lorsque la commande `START SLAVE` est envoyée, l'esclave crée un thread d'I/O (Entrée/Sortie). Le thread d'I/O se connecte au maître et lit les commandes qui ont été stockées dans le log binaire. Le maître crée un thread pour envoyer le contenu des logs binaire à l'esclave. Ce thread peut être identifié comme le thread `Binlog Dump` dans le résultat de la commande `SHOW PROCESSLIST` . Le thread esclave I/O lit ce que le thread maître `Binlog Dump` lui envoie, et le stocke dans un fichier local à l'esclave. Le troisième thread SQL lit ces commandes et les exécute.

Dans la description précédente, il y a trois threads par esclave. Pour un maître avec de nombreux esclaves, il crée un thread par esclave simultanément connecté, et chaque esclave a son propre thread I/O et SQL.

Pour les versions de MySQL avant 4.0.2, la réplication implique uniquement deux threads : un sur le maître et un sur l'esclave. Les threads I/O et SQL sont combinés en un seul thread, et il n'y a pas de log de relais.

L'avantage d'utiliser deux threads est que la lecture et l'exécution des requêtes sont découplées. La tâche de lecture n'est pas ralentie par l'exécution. Par exemple, si l'esclave n'a pas fonctionné depuis un bon moment, le thread d'I/O peut lire rapidement le contenu de toutes les commandes à appliquer, même si le thread SQL met du temps à les concrétiser. Si l'esclave s'arrête avant que toutes les commandes n'ait été exécutées, le thread d'I/O aura au moins lu les commandes, et elles sont désormais locales. Cela permettra au maître de purger ces lignes, si les autres esclaves n'en ont pas besoin non plus.

La commande `SHOW PROCESSLIST` affiche des informations qui vous indiquent ce qui se passe sur le maître et sur l'esclave, concernant la réplication.

L'exemple ci−dessous montre les trois threads dans le résultat de `SHOW PROCESSLIST` . Le format qui est présenté est celui de `SHOW PROCESSLIST` pour MySQL version 4.0.15, où le contenu de la colonne `State` a été changé pour être plus significatif.

Sur le serveur maître, le résultat de `SHOW PROCESSLIST` ressemble à ceci :

```
 mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 2
   User: root
   Host: localhost:32931
     db: NULL
Command: Binlog Dump
   Time: 94
  State: Has sent all binlog to slave; waiting for binlog to
         be updated
   Info: NULL
```

Ici, le thread 2 est le thread de réplication pour un esclave connecté. L'information indique que toutes les requêtes ont été envoyées à l'esclave, et que le maître attend de nouvelles instructions.

Sur le serveur esclave, le résultat de `SHOW PROCESSLIST` ressemble à ceci :

```
 mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 10
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 11
  State: Waiting for master to send event
   Info: NULL
*************************** 2. row ***************************
     Id: 11
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 11
  State: Has read all relay log; waiting for the slave I/O
         thread to update it
   Info: NULL
```

Cette information indique que le thread 10 est le thread d'I/O, en communication avec le serveur, et le thread 11 est le thread SQL, qui traite les commandes stockées dans le log de relais. Actuellement, les deux threads sont oisifs, et attendent des instructions.

Notez que la valeur de la colonne `Time` vous indique le retard de l'esclave par rapport au maître. FAQ de la réplication .

6.3.1 Etat de réplication du maître

La liste suivante montre les états les plus courants que vous verrez dans la colonne `State` pour le thread maître `Binlog Dump` . Si vous ne voyez pas le thread `Binlog Dump` sur le maître, la réplication ne fonctionne pas. C'est à dire qu'aucun esclave n'est connecté.

Envoi de log binaire à l'esclave (`Sending binlog event to slave` *)*
: Le log binaire contient les événements, qui sont les commandes de modifications des tables, ainsi que d'autres informations supplémentaire.s Le thread a lu un événement dans le log binaire, et il l'envoie à l'esclave.

Fini de lire le log binaire. Passe au fichier de log suivant (`Finished reading one binlog; switching to next binlog` *)*
: Le thread a fini de lire le fichier de log binaire, et il en ouvre un nouveau, pour l'envoyer à l'esclave.

A envoyé tous les logs binaire à l'esclave; attente de nouveau événements dans le log binaire (`Has sent all binlog to slave; waiting for binlog to be updated` *)*
: Le thread a lu toutes les commandes de modification dans le log binaire, et les a envoyé à l'esclave. Il est inactif, attend de nouveaux événements dans le log binaire pour reprendre ses activités.

Attente de la finalisation (`Waiting to finalize termination` *)*
: Un état très bref qui survient lorsque le thread s'arrête.

6.3.2 Etats du thread esclave d'E/S

La liste suivante montre les états les plus courants que vous verrez dans la colonne `State` pour le thread esclave d'entrée/sortie. Depuis 4.1.1, cette information apparaît aussi dans la colonne `Slave_IO_State` affichée par la commande `SHOW SLAVE STATUS` . Cela signifie que vous pouvez avoir une bonne idée de ce qui se passe juste avec `SHOW SLAVE STATUS` .

Connexion au maître (`Connecting to master` *)*
: Le thread tente de se connecter au maître

Vérification de la version du maître (`Checking master version` *)*
: Un état très bref qui survient juste après la connexion au maître.

Enregistrement de l'esclave auprès du maître (`Registering slave on master` *)*
: Un état très bref qui survient juste après la connexion au maître.

Demande de l'export du log binaire (`Requesting binlog dump` *)*
: Un état très bref qui survient juste après la connexion au maître. Le thread envoie une requête au maître pour obtenir le contenu des logs binaires, en indiquant le fichier de log et la position de démarrage.

Attente de reconnexion avec un échec de demande de log binaire (`Waiting to reconnect after a failed binlog dump request` *)*
: Si la demande de log binaire a échoué (à cause d'une déconnexion), le thread passe dans cet état durant sa mise en sommeil, et essaie de se reconnecter périodiquement. L'intervalle entre deux tentative est spécifié avec l'option `--master-connect-retry` .

Reconnexion avec un échec de demande de log binaire (`Reconnecting after a failed binlog dump request` *)*
: Le thread tente de se reconnecter au maître.

Attente d'informations de la part du maître (`Waiting for master to send event` *)*

Le thread est connecté au maître, et il attend les événement du log binaire. Cela peut durer longtemps sur le maître est inactif. Si l'attente de prolonge au−delà de `slave_read_timeout` secondes, un dépassement de délai survient. A ce moment, le thread considère que la connexion est perdue, et il va se reconnecter.

Ajoute un événement au log de relais (`Queueing master event to the relay log` *)*
: Le thread a lu un événement, et il le copie dans le log de relais, pour que le thread SQL puisse le lire.

Attente de reconnexion après un échec de lecture d'événement (`Waiting to reconnect after a failed master event read` *)*
: Une erreur est survenue durant la lecture, à cause d'une déconnexion. Le thread est en sommeil pour `master-connect-retry` secondes avant de tenter de se reconnecter.

Reconnexion après un échec de lecture d'événement (`Reconnecting after a failed master event read` *)*
: Le thread tente de se reconnecter au maître. Lorsque la reconnexion est faite, l'état deviendra `Waiting for master to send event` .

Attente d'espace pour le log de relais auprès du thread SQL (`Waiting for the slave SQL thread to free enough relay log space` *)*
: Si vous utilisez une valeur `relay_log_space_limit` non nulle, et que le log de relais a atteint sa taille maximale, le thread d'E/S va attendre que le thread SQL ait libéré suffisamment d'espace en traitant les requêtes pour qu'il puisse effacer un des fichiers de logs.

Attente du mutex de l'esclave (`Waiting for slave mutex on exit` *)*
: Un état très bref qui survient juste à l'extinction.

6.3.3 Etats des esclaves de réplication

La liste suivante présente les différents types les plus courants de valeurs de la colonne `State` pour un esclave SQL :

Lecture d'un événement dans le log de relais (`Reading event from the relay log` *)*
: Le thread a lu un événement dans le log de relais, et il le traite.

A lu tous les logs de relais; attente de l'esclave d'E/S (`Has read all relay log; waiting for the slave I/O thread to update it` *)*
: Le thread a traité tous les événements dans le log de relais et attend que le thread écrive de nouveaux événements dans le log de relais.

Attente du mutex de l'esclave pour terminer (`Waiting for slave mutex on exit` *)*
: Un état très bref qui survient lorsque le thread s'arrête.

La colonne `State` du thread d'E/S peut aussi afficher une commande. Cela indique que le thread a lu un événement dans le log de relais, a extrait la commande et est en train de l'exécuter.

6.3.4 Fichiers de relais et de statut de la réplication

Par défaut, les logs de relais sont nommés en utilisant des noms de la forme `host_name-relay-bin.nnn` , où `host_name` est le nom de l'hôte serveur esclave, et `nnn` est un numéro de séquence. Les fichiers de log de relais successifs sont créés en utilisant une séquence de nombre commençant à `001` . L'esclave garder la trace des logs avec un fichier d'index. Le nom du fichier d'index des logs de relais est `host_name-relay-bin.index` . Par défaut, ces fichiers sont créés dans le dossier de données de l'esclave. Les noms par défaut peuvent être remplacés grâce aux options `--relay-log` et `--relay-log-index` du serveur. Options de réplication .

Les logs de relais ont le même format que les logs binaires, et ils peuvent être lus avec `mysqlbinlog` . Un log de relais est automatiquement effacé par le thread SQL aussitôt qu'il n'en a plus besoin :

c'est à dire aussitôt qu'il en a exécuté les commandes. Il n'y a pas de commande pour effacer les logs de relais, car le thread SQL se charge de le faire. Toutefois, depuis MySQL 4.0.14, la commande `FLUSH LOGS` effectue la rotation des logs de relais, qui influence leur effacement par le thread SQL.

Un nouveau log de relais est créé dans les conditions suivantes :

* La première fois qu'un thread d'I/O démarre après le démarrage du serveur. Avec MySQL 5.0, un nouveau log de relais sera créé chaque fois que le thread d'I/O démarre, et pas seulement la première fois.
* Une commande `FLUSH LOGS` ou `mysqladmin flush-logs` est émise (MySQL 4.0.14 et plus récent uniquement).
* La taille du log de relais courant est trop grosse. ``trop grosse'' signifie :
    * `max_relay_log_size` , si `max_relay_log_size` > 0
    * `max_binlog_size` , si `max_relay_log_size` = 0 ou si MySQL est plus ancien que la version 4.0.14

Un serveur de réplication esclave crée deux autres petits fichiers dans le dossier de données. Ces fichiers sont appelés `master.info` et `relay-log.info` par défaut. Ils contiennent des informations comme celles affichées par la commande `SHOW SLAVE STATUS` (Commandes SQL pour contrôler les esclaves pour une description de cette commande). En tant que fichier disques, ils survivent à l'extinction de l'esclave. Au prochain démarrage de l'esclave, ce dernier peut lire ces fichiers pour savoir où il en était du traitement des événements du maître et de leur lecture.

Le fichier `master.info` est modifié par le thread d'I/O. Avant la version 4.1, la correspondance entre les lignes du fichier et les colonnes affichées par `SHOW SLAVE STATUS` est la suivante :

Ligne	Description
1	Master_Log_File
2	Read_Master_Log_Pos
3	Master_Host
4	Master_User
5	Mot de passe (pas affiché par SHOW SLAVE STATUS)
6	Master_Port
7	Connect_Retry

Depuis MySQL 4.1, le fichier inclus un compteur de ligne et des informations sur les options SSL :

Line	Description
1	Nombre de lignes dans le fichier
2	Master_Log_File
3	Read_Master_Log_Pos
4	Master_Host
5	Master_User
5	Mot de passe (pas affiché par SHOW SLAVE STATUS)
7	Master_Port
8	Connect_Retry
9	Master_SSL_Allowed
10	Master_SSL_CA_File
11	Master_SSL_CA_Path
12	Master_SSL_Cert
13	Master_SSL_Cipher
14	Master_SSL_Key

Le fichier `relay-log.info` est modifié par le thread SQL. La correspondance entre les lignes du fichier et les colonnes affichées par `SHOW SLAVE STATUS` est la suivante :

Ligne	Description
1	Relay_Log_File
2	Relay_Log_Pos
3	Relay_Master_Log_File
4	Exec_Master_Log_Pos

Lorsque vous sauvegardez les données de votre esclave, vous devriez aussi sauver ces deux fichiers, ainsi que les logs de relais. Ils ont nécessaires pour reprendre la réplication après une restauration de la base. Si vous perdez les logs de relais mais avez encore le fichier `relay-log.info` , vous pouvez l'étudier pour déterminer ce que le thread SQL a traité des logs binaires du maître.

Puis, vous pouvez utiliser `CHANGE MASTER TO` avec les options `MASTER_RELAY_LOG` et `MASTER_RELAY_POS` pour dire au thread d'I/O de relire les logs depuis ce point. Cela impose que ces logs sont toujours disponibles sur le serveur.

Si votre esclave doit répliquer une commande `LOAD DATA INFILE` , vous devriez aussi sauver les fichiers `SQL_LOAD-*` qui existent dans le dossier que l'esclave utilise à cette fin. L'esclave aura besoin de ces fichiers pour reprendre la réplication des commandes `LOAD DATA INFILE` . Le chemin du dossier est spécifié avec l'option `--slave-load-tmpdir` . Sa valeur par défaut est `tmpdir` .

6.4 Comment mettre en place la réplication

Voici les instructions pour mettre en place la réplication sur votre serveur MySQL. Nous supposons que vous voulez répliquer toutes vos bases, et que vous ne l'avez jamais configuré auparavant. Vous aurez besoin d'éteindre brièvement le serveur principal pour suivre toutes les instructions.

La procédure est écrite pour configurer un esclave seul, mais elle peut être répétée pour configurer plusieurs esclaves.

Si cette méthode n'est pas la plus simple pour configurer un esclave, ce n'est pas la seule. Par exemple, si vous avez déjà une sauvegarde des données du maître, et que le maître a déjà un identifiant de serveur, et le log binaire activé, vous pouvez configurer l'esclave sans éteindre le serveur et sans bloquer les mises à jours. Pour plus de détails, voyez FAQ de la réplication .

Si vous voulez administrer une architecture de réplication MySQL, nous vous suggérons de commencer par étudier, tester et expérimenter toutes les commandes mentionnées dans les chapitres Commandes SQL pour contrôler les maîtres et Commandes SQL pour contrôler les esclaves . Vous devriez aussi vous familiariser avec les options de démarrage décrites dans la section Options de réplication .

- Assurez vous que vous avez une version récente de MySQL installée comme maître et comme esclave. Assurez vous que ces versions sont compatibles entre elles, conformément à la table présentée dans la section Mettre à jour une architecture de réplication .

  Ne nous rapportez pas de bugs tant que vous n'avez pas vérifié que le problème persiste dans la dernière version de MySQL.
- Créez un utilisateur MySQL spécial pour la réplication sur le maître, avec les droits de `FILE` (dans les versions plus anciennes que la versions 4.0.2) ou le droit de `REPLICATION SLAVE` pour les nouvelles versions. Vous devez aussi lui donner les droits de connexion depuis tous les esclaves. Si l'utilisateur ne fait que de la réplication (ce qui est recommandé), vous n'avez pas à lui donner d'autres droits.

  Le nom d'hôte du compte doit être tel que chaque serveur esclave peut l'utiliser pour se connecter au maître. Par exemple, pour créer un utilisateur appelé `repl` qui peut accéder au maître, vous pourriez utiliser une commande comme :

  ```
  mysql> GRANT REPLICATION SLAVE ON *.* TO repl&#64;'%' IDENTIFIED BY '<password>';
  ```

  Pour les versions de MySQL antérieure à la 4.0.2, utilisez cette commande :

  ```
  mysql> GRANT FILE ON *.* TO repl&#64;'%' IDENTIFIED BY '<password>';
  ```
  Si vous envisagez d'utiliser `LOAD TABLE FROM MASTER` ou `LOAD DATA FROM MASTER` sur l'esclave, vous devez donner les droits supplémentaires suivants :
  - ♦ Donnez le droit de `SUPER` et `RELOAD` .

    - Donnez le droit de `SELECT` pour toutes les tables que vous voulez charger. Toutes les tables maîtres dans lesquelles l'esclave ne pourra pas utiliser `SELECT` seront ignorées par `LOAD DATA FROM MASTER` .
- Si vous utiliez des tables MyISAM, déchargez toutes les tables et blocs en utilisant la commande `FLUSH TABLES WITH READ LOCK` .

```
mysql> FLUSH TABLES WITH READ LOCK;
```

puis faire une sauvegarde des données de votre maître.Le plus simple pour cela (sous Unix) et d'utiliser la commande `tar` pour produire une archive de votre dossier de données total. Le dossier de données dépend de votre installation.

```
shell> tar −cvf /tmp/mysql−snapshot.tar .
```

Si vous voulez que vos archives incluent seulement une base de données appelée `cette_base` , utilisez cette commande :

```
shell> tar −cvf /tmp/mysql−snapshot.tar ./cette_base
```

Puis copiez le fichier d'archive dans le dossier `/tmp` sur le serveur esclave. Sur cette machine, placez vous dans le dossier de données du serveur et décompressez l'archive locale avvec cette commande :

```
shell> tar −xvf /tmp/mysql−snapshot.tar
```

Il n'est pas besoin de répliquer la base `mysql` . Si c'est le cas, vous pouvez l'exclure de votre archive. Vous n'avez pas besoin d'inclure les fichiers de log dans l'archive, ou les fichiers `master.info` ou `relay−log.info` .Lorsque le verrou de lecture a été posé par `FLUSH TABLES WITH READ LOCK` et est en action, lisez les valeurs courantes du fichie de log et de son offset sur le maître :

```
mysql > SHOW MASTER STATUS;
+−−−−−−−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−−−−−+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+−−−−−−−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−−−−−+
| mysql−bin.003 | 73       | test,bar     | foo,manual,mysql |
+−−−−−−−−−−−−−−−+−−−−−−−−−−+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−−−−−+
1 row in set (0.06 sec)
```

La colonne `File` montre le nom du fichier de log, et la colonne `Position` affiche l'offset. Dans l'exemple ci−dessus, le nom du fichier de log est `mysql−bin.003` et son offset est 73. Notez ces valeurs. Vous en aurez besoin pour configurer l'esclave.Une fois que vous avez pris une sauvegarde et enregistré le nom de fichier, et son offset, vous pouvez réactiver l'activité sur votre maître :

```
mysql> UNLOCK TABLES;
```

Si vous utilisez des tables `InnoDB` , l'outil idéal est `InnoDB` Hot Backup, qui est disponible pour ceux qui achètent des licences commerciales MySQL, du support ou l'outil lui−même. Il fait un sauvegarde cohérente du maître, enregistre le nom du fichier de log binaire et son offset, pour que cette archive soit directement utilisée par l'esclave plus tard. Pour plus d'informations sur cet outil, voyez http://www.innodb.com/order.php .Sans `Hot Backup` , le mieux pour faire une sauvegarde rapide d'une base `InnoDB` est d'arrêter le serveur, puis de copier les fichiers de données `InnoDB` , leurs logs, et leur fichier de définition (`.frm`). Pour enregistrer le fichier de log courant et son offset, vous devez utiliser les commandes suivantes lors de l'extinction du serveur :

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SHOW MASTER STATUS;
```

Et ensuite, enregistrer le nom du fichier et son offset, lu dans le résultat de la commande `SHOW MASTER STATUS` présentée précédemment. Une fois que vous avez ces informations, éteignez le serveur **sans déverrouiller** les tables, pour vous assurer qu'il va bien s'arrêter dans l'état que vous avez noté :

```
shell> mysqladmin −uroot shutdown
```

Une alternative, valable pour les deux types de tables MyISAM et `InnoDB` , est de prendre un export SQL du maître, au lieu d'une copie binaire. Pour cela, vous pouvez utiliser l'utilitaire `mysqldump −−master−data` sur votre maître, puis exécuter les commandes SQL sur votre esclave. Toutefois, c'est plus lent que de faire une copie binaire.

Si le maître fonctionnait sans l'option `−−log−bin` , le nom du fichier de log et l'offset seront vides, lorsqu'ils sont demandé à `SHOW MASTER STATUS` et `mysqldump` sera vide aussi. Dans ce cas, utilisez la chaîne vide (`''`)comme nom de fichier de log, et la valeur 4 comme offset.

- Dans le fichier `my.cnf` du maître, ajoutez les options `log-bin` et `server-id=unique number` , où `master_id` doit être un entier positif entre 1 et 2^32 −{} 1, à la section `[mysqld]` et redémarrez le serveur. Il est très important que l'identifiant des esclaves soient différents de celui du maître. Pensez à `server-id` comme à une valeur comparable à une adresse IP : elle identifie de manière unique un serveur dans la communauté des réplicateurs.

```
[mysqld]
log-bin
server-id=1
```

  Si ces options ne sont pas présentes, ajoutez−les, et redémarrez le serveur.
- Arrêtez le serveur qui va servir d'esclave, et ajoutez les lignes suivantes dans son fichier `my.cnf` :

```
[mysqld]
server-id=slave_id
```

  La valeur de `slave_id` , comme la valeur de `master_id` , doit être un entier, entre 1 et 2^32 −{} 1. De plus, il est très important que l'identifiant de l'esclave soit différent de celui du maître. Par exemple :

```
[mysqld]
server-id=2
```

  Si vous configurez plusieurs esclaves, chacun d'entre eux doit avoir une valeur `server-id` distincte de celle du maître et des autres esclaves. Pensez aux `server-id` comme étant des adresses IP : ces identifiants repèrent de manière unique un esclave dans la communauté de réplication.

  Si vous ne spécifiez pas de valeur pour `server-id` , il prendra la valeur de 1 si vous n'avez pas défini de valeur pour `master-host` , sinon, il prendra la valeur de 2. Notez que dans le cas où vous omettez `server-id` , un maître refusera laconnexion à tous les esclaves. Par conséquent, omettre `server-id` est uniquement valable pour des opérations de sauvegarde avec log binaire.
- Copiez la sauvegarde des données dans vos esclaves. Assurez vous que les droits sur ces données sont corrects. L'utilisateur qui fait fonctionner MySQL doit avoir les droits d'écriture et de lecture sur ces fichiers, tout comme le maître l'avait.

  Si vous avez fait une sauvegarde avec `mysqldump` , lancez d'abord les esclaves (voir prochaine étape).
- Redémarrez les esclaves. S'il était déjà configuré pour la réplication, lancez l'esclave avec l'option `--skip-slave-start` . Vous pouvez uassi lancer l'esclave avec l'option `--log-warnings` . De cette manière, vous aurez plus de détails sur les problèmes que l'esclave rencontrera (problèmes réseau, d'identification, etc.)
- Si vous avez fait une sauvegarde du maître avec l'utilitaire `mysqldump` , chargez l'export avec la commande suivante :

```
shell> mysql -u root -p < dump_file.sql
```

- Exécutez la commande sur l'esclave, en remplaçant les valeurs entre crochets `<>` par les valeurs que vous aviez lu sur le maître, ou qui sont valables pour votre système :

```
mysql> CHANGE MASTER TO
    ->     MASTER_HOST='<master host name>',
    ->     MASTER_USER='<replication user name>',
    ->     MASTER_PASSWORD='<replication password>',
    ->     MASTER_LOG_FILE='<recorded log file name>',
    ->     MASTER_LOG_POS=<recorded log offset>;
```

  La table suivante vous donne les tailles maximales de ces variables :

MASTER_HOST	60
MASTER_USER	16
MASTER_PASSWORD	32
MASTER_LOG_FILE	255

- Lancez les threads esclaves.

```
mysql> START SLAVE;
```

Après avoir suivi les instructions ci−dessus, les esclaves doivent se connecter au maître, et rattraper les modifications qui ont eu lieu depuis la sauvegarde des données.

Si vous avez oublié de spécifier un `server-id` pour un esclave, vous allez obtenir l'erreur suivante dans le fichier d'erreur :

```
 Warning: one should set server_id to a non-0 value if master_host is set.
The server will not act as a slave.
```

Si vous avez oublié de le faire pour le maître, les esclaves ne pourront pas se connecter avec le maître.Si un esclave n'est pas capable de faire la réplication pour une raison quelconque, vous allez trouvez le message d'erreur dans le fichier de log d'erreurs de l'esclave.

Une fois qu'un esclave a activé la réplication, vous trouverez deux fichiers dans son dossier de données : `master.info` et `relay-log.info` . L'esclave utilise ces deux fichiers pour savoir où il en est des logs du maître. **Ne supprimer pas et n'éditez pas** ces fichiers, à moins que vous ne sachiez bien ce que vous faites. Même dans ce cas, il est préférable d'utiliser la commande `CHANGE MASTER TO` .

NOTE : le contenu du fichier `master.info` est priorité par rapport a certaines versions spécifiées en ligne de commande, ou dans le fichier `my.cnf` . Voyez Options de réplication pour plus de détails.

Une fois que vous avez une sauvegarde, vous pouvez l'utiliser pour configurer d'autres esclaves, en suivant la procédure concernant l'esclave, ci−dessus. Vous n'aurez pas besoin d'une autre sauvegarde du maître.

6.5 Compatibilité de la réplication entre les versions de MySQL

Le format de log binaire original de la réplication a été développé en MySQL 3.23. Il a changé en MySQL 4.0, et encore en MySQL 5.0. Cela a des conséquences lorsque vous mettez à jour votre architecture de réplication, tel que décrit dans la section Mettre à jour une architecture de réplication .

Au niveau de la réplication, toutes les versions MySQL 4.1.x et 4.0.x sont identiques, car elles utilisent le même format de log binaire. Par conséquent, les serveurs dans cet intervalle de versions seront compatibles, et la réplication devrait fonctionner sans problèmes entre eux. Les exceptions à cette compatibilité sont que les versions de MySQL 4.0.0 à 4.0.2 étaient des versions de développement très récentes, et qu'elles ne doivent plus être utilisées. Elles représentent des versions alpha dans la série des 4.0. La compatibilité avec ces versions est toujours documentée dans le manuel, avec ces distributions.

La table suivante indique les compatibilités entre les esclaves et maîtres, pour différentes versions de MySQL.

		Maître	**Maître**	**Maître**
		3.23.33 et plus récent	**4.0.3 et plus récent or any 4.1.x**	**5.0.0**
Esclave	**3.23.33 et plus récent**	oui	non	non
Esclave	**4.0.3 et plus récent**	oui	oui	non
Esclave	**5.0.0**	oui	oui	oui

En général, nous recommandons d'utiliser des versions récentes de MySQL, car la réplication s'améliore continuellement. Nous recommendons aussi d'utiliser la même version pour le maître et les esclaves.

6.5.1 Changer de version de réplication

Lorsque vous mettez à jour vos serveurs dans une architecture de réplication, la procédure pour changer les versions dépend des versions que vous abandonnez et de celle vers laquelle vous allez.

6.5.2 Passer à la réplication version 4.0

Cette section s'applique aux situations de mise à jour depuis une architecture MySQL 3.23 vers 4.0 ou 4.1. Un serveur 4.0 doit être en version 4.0.3 ou plus récent, tel que mentionné dans la section Compatibilité de la réplication entre les versions de MySQL .

Lorsque vous mettez à jour un maître depuis MySQL 3.23 vers MySQL 4.0 ou 4.1, assurez vous d'abord que tous les esclaves et tous les maîtres sont déjà en versions 4.0 ou 4.1 (si ce n'est pas le cas, commencez par mettre à jour les esclaves comme indiqué ci−dessous). Une fois le maître mis à jour, vous ne devez pas relancer la réplication avec les vieux logs binaires 3.23, car cela va perturber les esclaves 4.0 et 4.1. La mise à jour peut être faites comme ceci, en supposant que vous avez un maître 3.23 à modifier, et des esclaves 4.0 ou 4.1 :

- Bloquez toutes les modifications sur le maître avec `FLUSH TABLES WITH READ LOCK` .
- Attendez que les esclaves ait rattrappé toutes les modifications du maître (utilisez `SHOW MASTER STATUS` sur le maître, et `SELECT MASTER_POS_WAIT()` sur les esclaves. Puis lancez `STOP SLAVE` sur les esclaves.
- Eteindez le serveur maître et passez le en MySQL 4.0 or 4.1.
- Relancez le serveur MySQL maître. Enregistrez le nom du nouveau log binaire du maître. Vous pouvez obtenir ce nom avec la commande `SHOW MASTER STATUS` sur le maître. Puis, lancez cette commande sur les esclaves :

```
 mysql> CHANGE MASTER TO MASTER_LOG_FILE='<name>', MASTER_LOG_POS=4;
mysql> START SLAVE;
```

6.5.3 Passer à la réplication version 5.0

Cette section s'applique aux situations de mise à jour depuis une architecture MySQL 3.23, 4.0 ou 4.1 vers une version 5.0.0. Un serveur 4.0 doit être en version 4.0.3 ou plus récent, tel que mentionné dans la section Compatibilité de la réplication entre les versions de MySQL .

D'abord, notez bien que MySQL 5.0.0 est actuellement en phase alpha; même s'il est supposé utilisable et meilleur que les vieilles versions (meilleure mise à jour, réplication de certaines variables de sessions importantes comme `SQL_MODE` ; voyez Changements de la version5.0.0 (22 Dec 2003: Alpha)), il n'est pas encore totalement testé. Nous vous recommandons donc de ne pas l'utiliser pour des environnements de production.

Lorsque vous passez de MySQL 3.23 or 4.0 en 4.1 ou 5.0.0, vous devriez vous assurer que tous les esclaves de ce maître sont déjà en version 5.0.0 (si ce n'est pas le cas, vous devriez commencer par mettre à jour vos esclaves, comme expliqué ci−dessous.

Alors, éteignez le maître, passez le en version 5.0.0 et relancez le. Le maître version 5.0.0 sera capable de relire les anciens logs binaires (d'avant la mise à jour), et de les envoyer aux esclaves 5.0.0 qui reconnaîtront le vieux format, et le comprendront. Les nouveaux logs binaires créés par le maître seront au format 5.0.0, et seront reconnus par les esclaves.

Pour mettre à jour les esclaves, commencez par les éteindre, puis passez les en version 5.0.0, et relancez les, ou relancez la réplication. Les esclaves de version 5.0.0 seront capables de relire les vieux fichiers de logs binaires (ceux d'avant la mise à jour), et exécuter les commandes qu'ils contiennent. Les logs de relais créé après la mise à jour seront au format 5.0.0.

En d'autres termes, il n'y a pas de mesure à prendre lorsque vous passez en version 5.0.0, sauf que les esclaves doivent être mis à jour avant le maître. Notez que si vous descendez de version, cela ne fonctionnera pas automatiquement : vous devez commencer par effacer les logs binaires et de relais au format 5.0.0 avant de procéder.

6.6 Fonctionnalités de la réplication et problèmes connus

La liste suivante explique ce qui est supporté ou pas. Des informations spécifiques `InnoDB` sur la réplication sont disponibles dans la section `InnoDB` et la réplication MySQL .

- La réplication s'effectue correctement sur les valeurs `AUTO_INCREMENT` , `LAST_INSERT_ID()` et `TIMESTAMP` .
- Les fonctions `USER()` et `LOAD_FILE()` sont répliquées dans modifications, et ne seront pas fiable une fois rendues sur le serveur esclave. C'est aussi vrai pour `CONNECTION_ID()` pour les esclaves de versions antérieures à la 4.1.1. La **nouvelle** fonction `PASSWORD()` de MySQL 4.1, est bien répliquée depuis les maîtres version 4.1.1; vos esclaves doivent être en version 4.1.0 ou plus récent pour la répliquer. Si vous avez d'anciens esclaves, et que vous devez répliquer la fonction `PASSWORD()` depuis un maître 4.1, vous devez lancer le maître avec l'option `--old-password` .
- Les variables `SQL_MODE` , `UNIQUE_CHECKS` , `SQL_AUTO_IS_NULL` sont répliquées depuis la version 5.0.0. Les variables `SQL_SELECT_LIMIT` et `TABLE_TYPE` ne sont pas répliquées pour le moment. `FOREIGN_KEY_CHECKS` est répliquée depuis la version 4.0.14.
- Vous devez utiliser le même jeu de caractères (`--default-character-set`) sur le maître et sur l'esclave. Sinon, vous risquez de rencontrer des erreurs de clés dupliquées, sur l'esclave, ces une valeur pourrait être considérée comme unique sur le serveur et non sur l'esclave. Les jeux de caractères seront répliqués en version 5.0.
- Si vous utilisez des tables transactionnelles sur le maître et non−transactionnelle sur l'esclave, pour les mêmes tables, vous rencontrerez des problèmes si l'esclave est interrompu au milieu d'un bloc `BEGIN/COMMIT` , car l'esclave reprendra ultérieurement au début du bloc `BEGIN` . Ce problème est sur notre liste de tâche, et sera corrigé prochainement.
- Les requêtes d' `UPDATE` qui utilisent des variables utilisateurs ne sont pas correctement répliquées sur les serveurs 3.23 et 4.0. C'est corrigé en 4.1. Notez que les noms de variables utilisateurs sont insensibles à la classe depuis la version 5.0, alors il est recommandé de prendre cela en compte lors de la configuration de la réplication entre un serveur version 5.0 et une version précédente.
- L'esclave peut se connecter au maître avec la sécurisation SSL, si le maître et l'esclave sont tous les deux en versions 4.1.1 ou plus récentes.
- Si la clause `DATA DIRECTORY` ou `INDEX DIRECTORY` est utilisée dans la commande `CREATE TABLE` sur le maître, la clause est aussi utilisée sur l'esclave. Cela peut causer des problèmes s'il n'existe pas de dossier correspondant sur le système de fichiers de l'esclave. Depuis MySQL 4.0.15, il y a une option de mode SQL `sql_mode` appelée `NO_DIR_IN_CREATE` . Si le serveur esclave fonctionne avec ce mode SQL, il va simplement ignorer ces clauses avant de répliquer les commandes `CREATE TABLE` . Le résultat est que les données `MyISAM` et les fichiers d'index seront créées dans le dossier de la base.
- Même si nous n'avons jamais vu d'occurrence de ce problème, il est théoriquement possible pour les données du maître et de l'esclave de différer si une requête non−déterministe est utilisée pour modifier les données, c'est à dire si elle est laissé au bon vouloir de l'optimiseur, ce qui n'est pas une bonne pratique même sans la réplication. Pour plus d'informations, voyez Bugs ouverts / limitations de MySQL .
- Avant MySQL 4.1.1, les commandes `FLUSH` , `ANALYZE` , `OPTIMIZE` , `REPAIR` n'étaient pas stockées dans le log binaire, et donc, elles n'étaient pas répliquées avec les esclaves. Ce n'est pas normalement un problème, car `FLUSH` ne modifie pas les tables. Cela peut signifier que vous avez modifié des droits dans les tables MySQL directement sans la commande `GRANT` et que vous avez répliqué les droits de `mysql` sans pouvoir faire de commande `FLUSH PRIVILEGES` sur vos

esclaves pour les prendre en compte. Depuis MySQL version 4.1.1, ces commandes sont écrites dans le log binaire, (hormis `FLUSH LOGS` , `FLUSH MASTER` , `FLUSH SLAVE` , `FLUSH TABLES WITH READ LOCK`) à moins que vous ne spécifiez `NO_WRITE_TO_BINLOG` ou son alias `LOCAL`). Pour un exemple d'utilisation de la syntaxe, voyez Syntaxe de `FLUSH` .

- MySQL supporte uniquement un maître et plusieurs esclaves. Ultérieurement, nous allons ajouter un algorithme de choix automatique du maître. Nous allons aussi introduire une notion d'agent, qui aideront à équilibrer la charge en envoyer les commandes `SELECT` aux différents esclaves.
- Lorsqu'un serveur s'arrête et repart, les tables `MEMORY` (`HEAP`) sont vidées. Depuis MySQL 4.0.18, le maître réplique cet effet comme ceci : la première fois que le maître utilise une table `MEMORY` après le démarrage, il indique aux esclaves que la table doit être vidée en ajoutant une commande `DELETE FROM` pour la table en question, dans son log binaire. Voyez Tables `HEAP` pour plus de détails.
- Les tables temporaires sont répliquées depuis la version 3.23.29, à l'exception des cas où vous éteignez le serveur esclave (et pas juste le thread esclave), que vous avez des tables temporaires ouvertes et qu'elles sont utilisées dans des modifications ultérieures. (Si vous éteignez l'esclave, les tables temporaires utilisées par ces commandes ne sont plus disponibles au redémarrage de l'esclave). Pour éviter ce problème, n'éteignez jamais un esclave qui a des tables temporaires actives. Utilisez cette procédure :
    - Utilisez la commande `SLAVE STOP` .
    - Vérifiez la variable de statut `Slave_open_temp_tables` pour vérifier si elle vaut bien 0.
    - Si elle vaut bien 0, exécutez `mysqladmin shutdown` .
    - Si le nombre n'est pas 0, redémarrez l'esclave avec la commande `SLAVE START` .
    - Répetez la procédure et voyez si vous avez plus de chance la prochaine fois.

  Nous envisageons de corriger ce problème prochainement.
- Il est possible de connecter les serveurs MySQL en chaîne bouclée (chaque serveur est le maître du précédent et l'esclave du suivant, en boucle), avec l'activation de l'option `log-slave-updates` . Notez que de nombreuses requêtes ne vont pas fonctionner dans ce type de configuration à moins que votre code client ne soit écrit avec beaucoup de soin, pour qu'il se charge des problèmes qui pourraient arriver dans différentes séquences de modifications sur différents serveurs.Cela signifie que vous pouvez réaliser une configuration comme ceci :

  ```
  A -> B -> C -> A
  ```

  Les identifiants de serveurs sont inscrits dans les événements. A saura qu'un événement qu'il a déjà exécuté lui est revenu, et il ne l'exécutera pas deux fois : il n'y a pas de risque de boucle infinie. Mais dans une configuration circulaire, vous devez vous assurer que le code client n'effectue pas de modifications conflictuelles. En d'autres termes, si vous insérez des données dans A et C, vous devez vous assurez qu'il n'y a pas de conflit de clé unique. Ne modifiez pas non plus deux lignes simultanément sur deux serveurs, si l'ordre des modifications a une importance pour vous.
- Si la requête sur l'esclave génère une erreur, le thread esclave s'arrête, et un message sera ajouté dans le fichier d'erreur. Vous devriez vous connecter pour corriger manuellement les données de l'esclave, puis relancer l'esclave avec la commande `SLAVE START` (disponible depuis la version 3.23.16. En version 3.23.15, vous devrez redémarrer le serveur.
- Si la connexion au maître est perdue, l'esclave tente de se reconnecter immédiatement, et en cas d'échec, il va retenter toutes les `master-connect-retry` (par défaut, 60) secondes. A cause de cela, il est sage d'éteindre le serveur maître et de le redémarrer régulièrement. L'esclave sera capable de gérer les problèmes réseau. Variables sytème du serveur .
- Eteindre l'esclave proprement est sûr, car il garde la trace du point où il en est rendu. Les extinctions sauvages vont produire des problèmes, surtout si le cache disque n'a pas été écrit sur le disque avant que le système ne s'arrête. Votre niveau de tolérance aux pannes sera grandement amélioré si vous avez de bons onduleurs.
- Etant donné la nature non transactionnelle des tables MySQL, il est possible qui va ne faire qu'une partie de la modification, et retourner une erreur. Cela peut arriver, par exemple, dans une insertion multiple dont une des lignes viole une contrainte d'unicité, ou si un très

long `UPDATE` est interrompu au milieu du stock de ligne. Si cela arrive sur le maître, l'esclave va s'arrêter et attendre que l'administrateur décide quoi faire, à moins que l'erreur soit légitime, et que la requête arrive à la même conclusion. Si le code d'erreur n'est pas désirable, certaines erreurs (voire même toutes), peuvent être masquées avec l'option `slave-skip-errors` , depuis la version 3.23.47.
- Si vous modifiez une table transactionnelle depuis une table transactionnelle, dans un bloc de transaction `BEGIN/COMMIT` , les modifications du log binaire peut être déphasées si un thread a fait une modification dans la table non−transactionnelle, avant la validation de la transaction. Les transactions sont écrites dans le log binaire au moment de leur validation.
- Avant la version 4.0.15, les modifications sur des tables non−transactionnelles sont écrites dans le log binaire immédiatement, alors que les modifications d'une transaction sont écrites au moment du `COMMIT` ou ignorées si vous utilisez un `ROLLBACK` ; vous devez prendre cela en compte lors de la modification de tables transactionnelles et non−transactionnelles dans la même transaction, si vous utilisez le log binaire pour les sauvegardes ou la réplication. En version 4.0.15, nous avons modifié le comportement du log pour les transactions, qui mèlent les modifications de tables transactionnelles et non−transactionnelles dans la même transaction, pour résoudre ce problème. L'ordre des requêtes est maintenant maintenu, et toutes les requêtes sont écrites, même en cas d'annulation `ROLLBACK` . Le problème qui reste est que lorsqu'une seconde connexion modifie une table non−transactionnelle durant la transaction de la première connexion, une erreur d'ordre dans les requêtes peut survenir, car la seconde transaction sera écrite immédiatement après sa réalisation.
- Lorsque l'esclave 4.x réplique une commande `LOAD DATA INFILE` depuis un maître 3.23, les valeurs des colonnes `Exec_Master_Log_Pos` et `Relay_Log_Space` pour `SHOW SLAVE STATUS` sont incorrectes. L'erreur de `Exec_Master_Log_Pos` va causer un problème lorsque vous stopperez et relancerez la réplication. Il est donc bon de corriger cela avec la commande `FLUSH LOGS` sur le maître. Ces bogues sont corrigés pour les esclaves en MySQL 5.0.0.

La table suivante liste les problèmes de MySQL 3.23 qui sont corrigés en MySQL 4.0 :

- `LOAD DATA INFILE` est correctement géré, tant que les données résident toujours sur le serveur maître au moment de la propagation.
- `LOAD LOCAL DATA INFILE` sera ignoré.
- En version 3.23 `RAND()` dans les modifications de lignes ne se propage pas correctement. Utilisez `RAND(some_non_rand_expr)` si vous répliquez des modifications qui incluent `RAND()` . Vous pouvez, par exemple, utiliser `UNIX_TIMESTAMP()` comme argument de `RAND()` . Ceci est corrigé en version 4.0.

6.7 Options de démarrage de la réplication

Sur le maître comme sur l'esclave, vous devez utiliser l'option `server-id` pour donner un identifiant unique ID à chaque serveur. Vous pouvez choisir un entier dans l'intervalle de 1 à 2^32 −{} 1 pour chaque maître et esclave. Exemple : `server-id=3`

Les options que vous pouvez utiliser sur le maître pour contrôler les logs sont décrites dans la section Le log binaire des mises à jour .

La table suivante décrit les options que vous pouvez utiliser sur les serveurs esclaves. Vous pouvez les spécifier en ligne de commande, ou dans le fichier d'options.

Les gestionnaires de réplication gèrent les options de manière spéciale, sans le sens où elles sont ignorées si un fichier `master.info` existe lorsque l'esclave est lancé, et qu'il contient des valeurs pour les options. Les options suivantes sont gérées de cette manière :

- `--master-host`
- `--master-user`
- `--master-password`
- `--master-port`
- `--master-connect-retry`

Depuis MySQL 4.1.1, les options suivantes sont gérées de manière particulière :

- `--master-ssl`
- `--master-ssl-ca`
- `--master-ssl-capath`
- `--master-ssl-cert`
- `--master-ssl-cipher`
- `--master-ssl-key`

Le format du fichier `master.info` de version 4.1.1 a changé pour inclure les options SSL. De plus, en version 4.1.1, le fichier inclut le nombre de lignes comme première ligne. Si vous passez d'une ancienne version vers un serveur 4.1.1, le nouveau serveur va mettre à jour le fichier `master.info` avec le nouveau format au démarrage. Toutefois, si vous rétrogradez en version 4.1.1, vous devrez supprimer la première ligne avant de relancer votre vieux serveur. Notez que dans ce cas, le serveur ancien ne pourra pas utiliser les connexions sécurisées pour communiquer avec le maître.

Si aucun fichier `master.info` n'existe lors du lancement de l'esclave, il utiliser les valeurs de ces options. Cela arrivera lorsque vous lancez un serveur de réplication en tant qu'esclave, pour la première fois, ou si vous avez utilisé la commande `RESET SLAVE` et arrêté puis relancé le serveur.

Cependant, si `master.info` existe lorsque l'esclave démarre, il utilisera les valeurs dans le fichier et ignorera les valeurs spécifiées en ligne de commande, ou dans le fichier d'options `master.info` .

Si vous redémarrez le serveur avec différentes options de démarrage que les valeurs qui sont dans le `master.info` , ces nouvelles valeurs n'auront pas d'effet, car le serveur continuera d'utiliser `master.info` . Pour utiliser différentes valeurs, vous devez relancer le serveur après avoir supprimé `master.info` , ou, de préférence, utilise la commande `CHANGE MASTER TO` pour remettre à zéro les valeurs durant l'exécution.

Supposez que vous spécifiez cette option dans votre fichier `my.cnf` :

```
 [mysqld]
master-host=un_hote
```

La première fois que vous démarrez le serveur en tant qu'esclave de réplication, il va lire et utiliser cette option dans le fichier `my.cnf` . Le serveur va ensuite enregistrer les valeurs courantes dans le fichier `master.info` . Au prochain démarrage du serveur, il va lire les valeurs dans le fichier `master.info` . Si vous modifiez `my.cnf` pour spécifier un nouvel hôte, cela n'aura pas d'effet. Vous devez utiliser la commande `CHANGE MASTER TO` .Comme le serveur donne la priorité au fichier `master.info` sur les options de démarrage décrites, vous pourriez ne pas souhaiter utiliser les options de démarrage pour ces valeurs, et plutôt, les spécifier avec la commande `CHANGE MASTER TO` . Voir `CHANGE MASTER TO` .

Cet exemple illustre une utilisation plus complète des options de démarrage pour configurer un serveur esclave :

```
 [mysqld]
server-id=2
master-host=db-master.mycompany.com
master-port=3306
master-user=pertinax
master-password=freitag
master-connect-retry=60
report-host=db-slave.mycompany.com
```

La liste suivante décrit les options de démarrage qui contrôlent la réplication : De nombreuses options peuvent être remises à zéro pendant que le serveur fonctionne, en utilisant la commande `CHANGE MASTER TO` . Sinon, des options comme `--replicate-*` peuvent être utilisées lorsque le serveur esclave démarre. Nous envisageons de corriger cela.

`--log-slave-updates`

Dit à l'esclave d'enregistrer les modifications effectuées par son thread SQL dans son propre log binaire. Par défaut, cette option est à Off. Pour que cette option ait un effet, l'esclave doit être lancé avec le log binaire activé : c'est l'option `--log-bin` option. `--log-slave-updates` sert lorsque vous voulez faire une chaîne de serveur de réplication. Par exemple :

```
A -> B -> C
```

C'est−à−dire, A sert de maître à l'esclave B, et B sert de maître à l'esclave C. Pour que cela fonctionne, avec B qui sert d'esclave et de maître simultanément, vous devez lancer B avec l'option `--log-slave-updates` . A et B doivent être lancés avec le log binaire activé.

`--log-warnings`
: Fait que l'esclave affiche plus de message sur ses activités. Par exemple, il vous alertera s'il réussi à se reconnecter après un problème de connexion, ou le démarrage de thread esclaves.Cette option n'est pas limitée à la réplication. Elle produit des alertes sur toutes la gamme des activités du serveur.

`--master-connect-retry=seconds`
: Le nombre de secondes qu'un esclave attend avant de tenter de se reconnecter au maître, dans le cas où le maître et l'esclave perdent la connexion. La valeur du fichier `master.info` a priorité, si elle est disponible. Par défaut, elle vaut 60.

`--master-host=host`
: Spécifie l'hôte ou l'IP du maître de réplication. Si cette option n'est pas fournie, le thread esclave ne sera pas lancé. La valeur inscrite dans le fichier `master.info` a priorité, si elle peut être lue. Un meilleur nom pour cette option aurait été `--bootstrap-master-host` , mais il est trop tard.

`--master-info-file=file_name`
: Le nom à utiliser pour le fichier dans lequel l'esclave stocke les informations sur le maître. Par défaut, c'est `mysql.info` , dans le dossier de données.

`--master-password=password`
: Le mot de passe que l'esclave utilise lors de l'identification auprès du maître. Si le mot de passe n'est pas configuré, la chaîne vide est utilisée. La valeur inscrite dans le fichier `master.info` a priorité, si elle peut être lue.

`--master-port=port_number`
: Le port du maître que l'esclave utilise lors de l'identification auprès du maître. Si le port n'est pas configuré, la valeur de la variable `MYSQL_PORT` est utilisée. Si vous n'y avez pas touché lors de la compilation avec `configure` , ce doit être 3306. La valeur inscrite dans le fichier `master.info` a priorité, si elle peut être lue.

`--master-ssl`
`--master-ssl-ca=file_name`
`--master-ssl-capath=directory_name`
`--master-ssl-cert=file_name`
`--master-ssl-cipher=cipher_list`
`--master-ssl-key=file_name`
: Ces options servent à configurer la réplication chiffrée, lorsque la connexion avec le maître utilise SSL. Leurs significations respectives est la même que les options `--ssl` , `--ssl-ca` , `--ssl-capath` , `--ssl-cert` , `--ssl-cipher` , `--ssl-key` décrites dans Options de ligne de commande pour SSL .

  Ces options sont disponibles depuis MySQL 4.1.1.

`--master-user=username`
: Le nom d'utilisateur que l'esclave utilise lors de l'identification auprès du maître. Le compte doit avoir les droits de `REPLICATION SLAVE` (avant MySQL 4.0.2, il devait avoir les droits de `FILE`). Si l'utilisateur maître n'est pas configuré, l'utilisateur `test` est utilisé. La valeur inscrite dans le fichier `master.info` a priorité, si elle peut être lue. Si l'utilisateur maître n'est pas configuré, la valeur `test` est utilisée.

`--max-relay-log-size=#`
: Pour faire la rotation automatique des logs. Syntaxe de `SHOW VARIABLES` .Cette option est disponible depuis MySQL 4.0.14.

`--read-only`
: Cette option fait que le serveur n'autorise aucune modification, hormis celles du thread esclave, ou celle des utilisateurs ayant les droits de `SUPER` . Cela peut être utile si vous voulez vous assurer que l'esclave ne reçoit aucune modification des clients.Cette option est disponible depuis MySQL 4.0.14.

`--relay-log=filename`

Pour spécifier la localisation et le nom qui doivent être utilisés pour les logs de relais. Les noms par défaut sont de la forme `host_name-relay-bin.nnn` , où `host_name` est le nom du serveur esclave et `nnn` indique le numéro de séquence du log de relais. Vous pouvez utiliser ces options pour avoir des noms de fichier de log de relais indépendants du nom d'hôte, ou si vos logs ont tendances à devenir très grands (et que vous ne voulez pas réduire la valeur de `max_relay_log_size`) et que vous devez les mettre dans un autre dossier, ou simplement pour accélérer la vitesse d'équilibrage entre deux disques.

`--relay-log-index=filename`

Pour spécifier la localisation et le nom qui doivent être utilisés pour le fichier d'index du log de relais. Le nom par défaut est `host_name-relay-bin.index` , où `host_name` est le nom du serveur esclave.

`--relay-log-info-file=filename`

Pour donner au fichier `relay-log.info` un autre nom ou pour le placer dans un autre dossier. Le nom par défaut est `relay-log.info` dans le dossier de données.

`--relay-log-purge={0|1}`

Active ou désactive la vidange automatique des logs de relais, dès qu'ils ne sont plus utiles. C'est une variable globale, qui peut être dynamiquement modifiée avec `SET GLOBAL RELAY_LOG_PURGE=0|1` . Sa valeur par défaut est 1.

Cette option est disponible depuis MySQL 4.1.1.

`--relay-log-space-limit=#`

Limite la taille maximale de tous les fichiers de logs de relais sur l'esclave (une valeur de 0 signifie ``sans limite''). C'est utile lorsque vous avez un petit disque sur votre machine esclave. Lorsque la limite est atteinte, le thread d'I/O fait une pause : il ne lit plus rien dans le log binaire du maître, jusqu'à ce que le thread SQL ait avancé, et effacé des fichiers de logs. Notez que cette limite n,est pas absolue : il se peut que le thread SQL requiert plusieurs événements pour être capable d'effacer les fichiers de log de relais. Dans ce cas, le thread d'I/O va dépasser la limite, jusqu'à ce que l'effacement devienne possible. Sans cela, des blocages pourraient survenir, ce qui arrivait sur les versions antérieures à la 4.0.13). Avec `--relay-log-space-limit` , il ne faut pas utiliser de valeur inférieure à deux fois la taille de `--max-relay-log-size` (ou `--max-binlog-size` si `--max-relay-log-size` vaut 0) car dans ce cas, il y a des chances que le thread d'I/O attende de l'espace libre par ce que `--relay-log-space-limit` est dépassée, mais que le thread SQL n'ait pas de logs à effacer, et ne peut donc libérer le thread d'I/O, forçant le thread d'I/O à ignorer temporairement `--relay-log-space-limit` .

`--replicate-do-db=db_name`

Indique à l'esclave qu'il doit restreindre la réplication aux commandes qui utilisent la base de données `db_name` par défaut (c'est à dire celle qui est sélectionnée avec la commande `USE`). Pour spécifier plusieurs base de données, utilisez cette option aussi souvent que nécessaire. Note que cela ne va pas autoriser les commandes multi−bases, comme `UPDATE some_db.some_table SET foo='bar'` si une base de données différente ou qu'aucune base de données n'est sélectionnée. Si vous avez besoin que les commandes multi−bases fonctionnent, assurez vous que vous avez MySQL 3.23.28 ou plus récent, et utilisez `--replicate-wild-do-table=db_name.%` . Lisez les notes qui suivent cette liste d'options.Un exemple qui pourrait ne pas fonctionner comme vous l'attendez : si l'esclave est lancé avec `--replicate-do-db=sales` et que vous émettez une commande sur le maître, la commande `UPDATE` suivante ne sera pas répliquée :

```
 USE prices;
UPDATE sales.january SET amount=amount+1000;
```

Si vous avez besoin de répliquer des commandes multi−bases, utilisez l'option `--replicate-wild-do-table=db_name.%` à la place.La raison principale de ce comportement ``vérifie juste la base par défaut'' est qu'il est difficile de savoir si une requête doit être répliquée, uniquement à partir de la requête. Par exemple, si vous utilisez une requête multi−tables `DELETE` oui multi−tables `UPDATE` , qui a des conséquences dans d'autres bases. La vérification de la base courante est aussi très rapide.

`--replicate-do-table=db_name.table_name`

Dit à l'esclave qu'il doit restreindre la réplication à une table spécifiée. Pour spécifier plusieurs tables, il faut utiliser cette directive plusieurs fois, une fois par table. Cela fonctionnera pour les mises à jours multi−bases, au contraire de `--replicate-do-db` . Lisez les notes qui suivent cette liste d'options.

`--replicate-ignore-db=db_name`

Indique à l'esclave qu'il doit ne doit pas assurer la réplication avec les commandes qui utilisent la base de données `db_name` par défaut (c'est à dire celle qui est sélectionnée avec la commande `USE`). Pour spécifier plusieurs base de données, utilisez cette option aussi souvent que nécessaire. Note que cela ne va pas autoriser les commandes multi−bases, comme `UPDATE some_db.some_table SET foo='bar'` si une base de données différente ou qu'aucune base de données n'est sélectionnée. Si vous avez besoin que les commandes multi−bases fonctionnent, assurez vous que vous avez MySQL 3.23.28 ou plus récent, et utilisez `--replicate-wild-do-table=db_name.%` . Lisez les notes qui suivent cette liste d'options.

Un exemple qui pourrait ne pas fonctionner comme vous l'attendez : si l'esclave est lancé avec `--replicate-ignore-db=sales` et que vous émettez une commande sur le maître, la commande `UPDATE` suivante ne sera pas répliquée :

```
 USE prices;
UPDATE sales.january SET amount=amount+1000;
```

Si vous avez besoin de répliquer des commandes multi−bases, utilisez l'option `--replicate-wild-ignore-table=db_name.%` à la place.

`--replicate-ignore-table=db_name.table_name`

Dit à l'esclave qu'il ne doit pas répliquer les commandes qui touche à la table spécifiée, même si d'autres tables sont modifiées dans la même commande. Pour spécifier plusieurs tables, il faut utiliser cette directive plusieurs fois, une fois par table. Cela fonctionnera pour les mises à jours multi−bases, au contraire de `--replicate-ignore-db` . Lisez les notes qui suivent cette liste d'options.

`--replicate-wild-do-table=db_name.table_name`

Dit à l'esclave qu'il doit restreindre la réplication aux tables dont le nom vérifie le masque spécifié. Le masque peut contenir les caractères '`%`' et '`_`' , qui ont la même signification que dans les expressions régulières de la clause `LIKE` . Pour spécifier plusieurs tables, il faut utiliser cette directive plusieurs fois, une fois par table. Cela fonctionnera pour les mises à jours multi−bases, au contraire de `--replicate-do-db` . Lisez les notes qui suivent cette liste d'options.

Exemple : `--replicate-wild-do-table=foo%.bar%` va répliquer les mises à jour qui surviennent sur toutes les tables de toutes les bases qui commencent par `foo` , et dont le nom de table commence par `bar` .

Notez que si vous utilisez `--replicate-wild-do-table=foo%.%` , alors la règle sera propagée à `CREATE DATABASE` et `DROP DATABASE` , c'est à dire que ces deux commandes seront répliquées si le nom de la base correspond au masque (`foo%` ici) (la magie est ici déclenchée par `%` comme masque de table.).

Si le masque de noms de tables est `%` , il accepte tous les noms de tables et les options s'appliquent aux commandes de niveau base de données (comme `CREATE DATABASE` , `DROP DATABASE` et `ALTER DATABASE`). Par exemple, si vous utilisez `--replicate-wild-do-table=foo%.%` , les commandes de niveau de base de données seront répliquées si le nom de la base de données est accepté par le masque `foo%` .

Si vous voulez faire la réplication des tables du type `ma_petite%base` (ceci est le nom exact de la base), mais que vous ne voulez pas répliquer la base `ma1petiteAABCbase` , vous devez protéger les caractères '`_`' et '`%`' : il faut utiliser une syntaxe équivalent à :

`replicate-wild-do-table=my\_own\%db` . Et si vous spécifiez cette option en ligne de commande, suivant votre système, vous devrez protéger aussi le caractère `\` (par exemple, en Shell `bash` , vous devez émettre une option sous la forme `--replicate-wild-do-table=my\\_own\\%db`).

`--replicate-wild-ignore-table=db_name.table_name`
: Dit à l'esclave qu'il ne doit pas répliquer les tables dont le nom vérifie le masque spécifié. Pour spécifier plusieurs tables, il faut utiliser cette directive plusieurs fois, une fois par table. Cela fonctionnera pour les mises à jours multi−bases, au contraire de `--replicate-do-db` . Lisez les notes qui suivent cette liste d'options.Exemple : `--replicate-wild-ignore-table=foo%.bar%` n'autorisera pas de modifications dans les tables des bases dont le nom commence par `foo` et dont le nom de table commence par `bar` .

  Pour des informations sur le fonctionnement du filtre, voyez l'option `--replicate-wild-ignore-table` . La règle pour inclure des caractères littéraux est la même que pour `--replicate-wild-ignore-table` .

`--replicate-rewrite-db=from_name->to_name`
: Dit à l'esclave de remplacer la base courante (celle qui est sélectionnée avec `USE`) par `to_name` si elle était `from_name` sur le maître. Seules les commandes impliquant des tables peuvent être affectées. (`CREATE DATABASE` , `DROP DATABASE` ne le seront pas), et uniquement si `from_name` était la base de données courante sur le maître. Cela ne fonctionnera pas pour les commandes multi−bases de données. Notez que la traduction est faite avant que les règles `--replicate-*` ne soient testées.

  Si vous utilisez cette option en ligne de commande, et que vous utilisez le caractère `'>'` , qui peut être spécial pour votre interpréteur Shell, protégez−le comme ceci :

```
shell> mysqld --replicate-rewrite-db="olddb->newdb"
```

`--replicate-same-server-id`
: A utiliser sur les serveurs esclaves. Généralement, vous pouvez spécifier la valeur 0 pour éviter les réplications infinies. Si cette option vaut 1, l'esclave n'ignorera pas les événements de réplication, même s'ils portent son propre numéro d'identification. Normalement, cela n'est utile que pour de très rares configurations. Vous ne pouvez pas mettre cette option à 1 si `--log-slave-updates` est utilisé. Faîtes attention en démarrant MySQL 4.1, par défaut le thread d'E/S n'écrit pas les événements dans le log de relais s'ils portent l'identification du serveur esclave (c'est une optimisation pour économiser l'espace disque, par rapport à la version 4.0). Si vous voulez utiliser `--replicate-same-server-id` avec les versions 4.1, assurez vous de démarrer l'esclave avec cette option avant que l'esclave ne lise ses propres événements et qu'il les fasse exécuter au thread SQL.

`--report-host=host`
: Le nom d'hôte ou l'adresse IP de l'esclave, qui doit être indiquée lors de l'enregistrement de l'esclave chez le maître. Cela apparaîtra dans l'affichage de la commande `SHOW SLAVE HOSTS` . Laissez cette option vide pour que l'esclave ne s'enregistre pas sur le maître. Notez qu'il n'est pas suffisant pour que le maître lise l'adresse IP de l'esclave sur la socket, une fois que l'esclave se connecte. à cause du `NAT` et des problèmes de routages, cette IP peut être invalide pour se connecter au maître depuis l'hôte ou les autres esclaves.

  Cette option est disponible depuis MySQL 4.0.0.

`--report-port=port_number`
: Le port de connexion indiqué par l'esclave lors de son enregistrement chez le maître. Configurez cette option si l'esclave utilise un port autre que le port par défaut, ou si vous avez installé un tunnel spécial pour le maître ou les autres esclaves. Dans le doute, laissez cette option vide.

  Cette option est disponible depuis MySQL 4.0.0.

`--skip-slave-start`

Dit à l'esclave de ne pas lancer les threads esclaves au démarrage du serveur L'utilisateur pourra les lancer manuellement, avec `START SLAVE` .

`--slave_compressed_protocol=#`
: Si cette option vaut 1, alors le protocole client/serveur compressé sera utilisé, si l'esclave et le maître le supportent.

`--slave-load-tmpdir=filename`
: Cette option vaut par défaut la variable `tmpdir` . Lorsque le thread SQL répliquer des commandes `LOAD DATA INFILE` , il extrait les fichiers à charger du log de relais dans un fichier temporaire, puis charge ce fichier dans la table. Si le fichier chargé sur le maître est immense, le fichier temporaire sera aussi grand. Il faudra donc dire à l'esclave que placer ces fichiers temporaires sur un grand disque, qui sera différent de `tmpdir` : utilisez cette option. Dans ce cas, vous pouvez aussi utiliser l'option `--relay-log` , car les fichiers de log de relais seront aussi grands. `--slave-load-tmpdir` doit pointer sur un système de fichier basés sur un disque, et non pas sur une portion de mémoire : l'esclave doit pouvoir accéder à ce fichier pour répliquer la commande `LOAD DATA INFILE` , même après un redémarrage.

`--slave-net-timeout=#`
: Le nombre de secondes à attendre des données du maître, avant d'annuler la lecture en considérant que la connexion est rompue, et de tenter de se reconnecter. La première reconnexion intervient immédiatement après l'expiration du délai. L'intervalle entre deux tentatives de connexion est contrôlé par l'option `--master-connect-retry` .

`--slave-skip-errors= [err_code1,err_code2,... | all]`
: Normalement, la réplication s'arrête lorsqu'une erreur survient, ce qui vous donne l'opportunité de résoudre les incohérences manuellement. Cette option Indique au thread SQL les erreurs qu'il doit ignorer durant la réplication.

  N'utilisez pas cette option si vous ne connaissez pas la raison des erreurs que vous rencontrez. S'il n'y a pas de bugs dans votre réplication, et qu'il n'y a pas de bug dans MySQL, vous ne devriez pas rencontrer d'erreurs, ni utiliser cette option. L'utilisation abusive de cette option conduit irrémédiablement l'esclave à être désynchronisé avec le maître sans que vous ne sachiez d'où vient l'erreur.

  Pour les codes d'erreur, il faut utiliser les numéros d'erreurs fournis par l'esclave dans le log d'erreur, et dans le résultat de `SHOW SLAVE STATUS` . La liste complète des messages d'erreurs est disponible dans la distribution source, dans le fichier `Docs/mysqld_error.txt` . Les codes d'erreur du serveur sont aussi disponibles sur Erreurs retournées .

  Vous pouvez (mais ne devez pas) utiliser la valeur très déconseillée de `all` , qui va ignorer tous les messages d'erreur, et continuer à touiller les données sans se préoccuper de cohérence. Inutile d'insister sur le fait que l'intégrité de vos données n'est plus du tout garantie. Ne vous plaignez pas si les données de votre esclave ne ressemblent même pas du tout à celle de votre maître : vous aurez été prévenu.

  Exemples :

```
 --slave-skip-errors=1062,1053
--slave-skip-errors=all
```

Voici l'ordre d'étude des règles `r--eplicate-*` , pour décider si une requête doit être exécutée par l'esclave ou ignorée :

- Existe−t−il des règles `--replicate-do-db` ou `--replicate-ignore-db` ?
  - ♦ Oui : les tester pour `--binlog-do-db` et `--binlog-ignore-db` (Le log binaire des mises à jour). Quel est le résultat?
    - ◊ ignorer la requête : ignore la requête et quitte.

      ◊ exécute la requête : n'exécute pas la requête immédiatement, reporte la décision, et passe à l'étape d'après.
   ♦ Non : passe à l'étape d'après.
* Y−t−il des règles `--replicate-*-table` ?
   ♦ Non : exécute la requête et quitte.
   ♦ Oui : passe à l'étape d'après. Seules les tables qui doivent être modifiées seront utilisées dans les règles : (`INSERT INTO sales SELECT * from prices` : seule `sales` sera utilisée pour évaluer les règles. Si plusieurs tables doivent être modifiées (modifications multi−tables), la première table (qui correspond à un ``do'' ou ``ignore'') gagne. C'est à dire que la première table est utilisée dans les règles de comparaison, et si aucune décision ne peut être prise, la seconde table est utilisée...
* Y a−t−il des règles `--replicate-do-table` ?
   ♦ Oui : Est−ce qu'une table entre dans cette liste?
      ◊ Oui : exécute la requête et quitte.
      ◊ Non : passe à l'étape d'après.
   ♦ Non : passe à l'étape d'après.
* Y a−t−il des règles `--replicate-ignore-table` ?
   ♦ Oui : Est−ce qu'une table entre dans cette liste?
      ◊ Oui : ignore la requête et quitte.
      ◊ Non : passe à l'étape d'après.
   ♦ Non : passe à l'étape d'après.
* Y a−t−il des règles `--replicate-wild-do-table` ?
   ♦ Oui : Est−ce qu'une table entre dans cette liste?
      ◊ Oui : exécute la requête et quitte.
      ◊ Non : passe à l'étape d'après.
   ♦ Non : passe à l'étape d'après.
* Y a−t−il des règles `--replicate-wild-ignore-table` ?
   ♦ Oui : Est−ce qu'une table entre dans cette liste?
      ◊ Oui : ignore la requête et quitte.
      ◊ Non : passe à l'étape d'après.
   ♦ Non : passe à l'étape d'après.
* Aucune règle n'a fonctionné avec `--replicate-*-table` . Y a−t−il d'autres tables à tester?
   ♦ Oui : boucle.
   ♦ Non : Nous avons testé toutes les tables à mettre à jour, et nous n'avons pas trouvé de règle les concernant. Y a−t−il des règles `--replicate-do-table` ou `--replicate-wild-do-table` ?
      ◊ Oui : ignore la requête et quitte.
      ◊ Non : exécute la requête et quitte.

6.8 FAQ de la réplication

Q : Comment puis−je configurer un esclave si le maître fonctionne déjà, et que je ne veux pas le stopper?

R : Il y a plusieurs solutions. Si vous avez effectué une sauvegarde du maître à un moment et enregistré le nom et l'offset du binlog (issu du résultat de la commande `SHOW MASTER STATUS`) correspondant à la sauvegarde, faites ceci :

* Assurez−vous qu'un identifiant unique est assigné à l'esclave.
* Exécutez la commande pour chaque valeur appropriée :

```
mysql> CHANGE MASTER TO
    ->     MASTER_HOST='master_host-name',
    ->     MASTER_USER='master_user_name',
    ->     MASTER_PASSWORD='master_pass',
    ->     MASTER_LOG_FILE='recorded_log_name',
    ->     MASTER_LOG_POS=recorded_log_pos;
```

* Exécutez la commande `SLAVE START`

Si vous n'avez pas de copie de sauvegarde, voici un moyen rapide d'en faire une :

* Exécutez cette commande MySQL :
  ```
  mysql> FLUSH TABLES WITH READ LOCK;
  ```
* Exécutez cette commande Shell, ou toute variation de cette commande :
  ```
  shell> tar zcf /tmp/backup.tar.gz /var/lib/mysql
  ```
* Utilisez cette commande pour vous assurer de bien noter les informations de réplication. Vous en aurez besoin ultérieurement.
  ```
  mysql> SHOW MASTER STATUS;
  ```
* Libérez les tables :
  ```
  mysql> UNLOCK TABLES;
  ```

Un autre alternative est de faire un export SQL du maître, au lieu de faire une copie comme indiqué ci−dessus : pour cela, vous pouvez utiliser l'utilitaire `mysqldump --master-data` sur votre maître, et exécuter ce script ultérieurement sur votre esclave. Cependant, c'est une méthode plus lente que de faire une copie binaire.

Quelque soit la méthode que vous adoptez, après cela, suivez les instructions comme pour le cas où vous avez déjà votre sauvegarde, et que vous avez enregistré le nom et l'offset du point de contrôle du log binaire. Tant que les logs binaires du serveur sont toujours là, vous allez pouvoir rattrapper tout ce qui se fait sur le serveur principal. Vous pourriez même attendre plusieurs jours ou mois avant de mettre en place votre esclave. En théorie, le temps d'attente peut être infini. En pratique, les limitations sont l'espace disque du maître, et le temps que cela prendra à l'esclave pour rattrapper le temps.

Vous pouvez aussi utiliser `LOAD DATA FROM MASTER` . C'est une commande pratique pour faire une copie de la base, l'envoyer à l'esclave, et ajutser le point de contrôle du log binaire, tout en une seule commande. Dans le future, `LOAD DATA FROM MASTER` sera la méthode recommandée pour configurer un esclave. Soyez prévenus, que le verrou de lecture posé par la commande sur le serveur peut rester en place un très long moment, si vous utilisez cette commande : elle n'est pas encore implémentée de manière efficace. Si vous avez de grandes tables, préférez donc la méthode qui utilise la sauvegarde via l'utilitaire `tar` après avoir exécuté la commande `FLUSH TABLES WITH READ LOCK` .

Q : Est ce que l'esclave doit être connecté en permanance au serveur?

R : Non, il n'est pas obligé. Vous pouvez éteindre l'esclave et le laisser déconnecter plusieurs heures ou jours, puis le reconnecter pour le voir récupérer les modifications et rattrapper le temps. Puis, se déconnecter à nouveau. De cette façon, vous pouvez, par exemple, configurer un esclave via une connexion modem, qui n'utilise que de brève période de connexions. L'implication de cela est qu'il n'est jamais garantit que l'esclave soit synchronisé avec le maître, à moins que vous ne preniez des mesures pour cela. Dans le futur, nous allons avoir l'option de bloquer le maître jusqu'à ce que au moins un des esclaves soit synchronisé.

Q : Comment puis−je mesurer le retard d'un esclave sur son maître? En d'autres termes, comme savoir quelle est la date de la dernière requête répliquée par l'esclave?

R : Si l'esclave est en version 4.1.1 pour plus récent, lisez la colonne `Seconds_Behind_Master` dans la commande `SHOW SLAVE STATUS` . Pour les versions plus anciennes, suivez cette procédure : Cela n'est possible que si un thread SQL existe, c'est à dire s'il existe dans la commande `SHOW PROCESSLIST` , Détails d'implémentation de la réplication .

En MySQL version 3.23, si le thread SQL esclave existe, c'est à dire, s'il apparait dans la commande `SHOW PROCESSLIST` , et s'il a exécuté au moins un événement lu auprès du maître, the thread modifie sa propre horloge pour prendre l'horaire du dernier événement répliqué (c'est pour cela que les colonnes `TIMESTAMP` sont bien répliquées. Dans la colonne `Time` du résultat de `SHOW PROCESSLIST` , le nombre de secondes affichées est le nombre de secondes entre la dernière commande exécutée sur le serveur maître et celle exécutée sur l'esclave. Notez que si votre esclave a été déconnecté du maître durant une heure, lorsqu'il se reconnecte, vous pouvez voir immédiatement la valeur 3600 dans la colonne `Time` , pour le thread esclave dans `SHOW PROCESSLIST` ... Ceci est du au fait que la dernière requête exécuté date d'une heure.

Q : Comment puis−je forcer le maître à bloquer les modifications jusqu'à ce que l'esclave ait tout rattrappé?

R : Exécutez les commandes suivantes :

- Sur le maître, exécutez ces commandes :
  ```
  mysql> FLUSH TABLES WITH READ LOCK;
  mysql> SHOW MASTER STATUS;
  ```
  Enregistrez le nom du fichier de log et l'offset, dans la commande `SHOW` .
- Sur l'esclave, utilisez la commande ci−dessous, où vous aurez reporté les arguments de coordonnées de réplication données par `MASTER_POS_WAIT()` :
  ```
  mysql> SELECT MASTER_POS_WAIT('log_name', log_offset);
  ```
  La commande `SELECT` va s'arrêter jusqu'à ce que l'esclave atteigne le fichier de log et l'offset. A ce point, l'esclave sera synchronisé avec le maître, et la commande se terminera.
- Sur le maître, utilisez la commande suivante pour permettre au maître de recommencer à traiter les modifications :
  ```
  mysql> UNLOCK TABLES;
  ```

Q : Quels sont vos conseils concernant la réplication bi−bidirectionnelle?

R : La réplication MySQL ne supporte aucun protocole de verrouillage entre le maître et l'esclave pour garantir l'atomicité d'une modification entre les serveurs. En d'autres termes, il est possible pour un client A de faire une modification sur le serveur 1 et que dans le même temps, avant que cela ne se soit propagé au serveur 2, un client B se connecte au serveur 2, et fasse une modification sur le serveur 2 qui ne débouchera pas sur le même état que celui dans lequel le serveur 1 est. C'est ainsi qu'il ne faut pas lier de cette façon deux serveurs, à moins que les modifications ne puisse se faire dans n'importe quel ordre, ou que vous sachiez prendre en charge des modifications anarchiques.

Vous devez aussi réaliser que la réplication bi−directionnelle n'améliore pas beaucoup les performances, tout au moins au niveau des modifications. Les deux serveurs doivent faire la même quantité de modifications, ainsi qu'un serveur seul le ferait. La seule différence est qu'il va y avoir moins de verrous, car les modifications qui proviennent d'un autre serveur seront optimisé par l'esclave. Cet avantage peut aussi être annulé par les délais réseau.

Q : Comment puis−je utiliser la réplication pour améliorer les performances de mon système ?

R : Vous devez configurer un serveur en maître et y diriger toutes les écritures, puis configurer les autres en esclaves dans la limite de vos moyens, et y distribuer les lectures. Vous pouvez aussi démarrer les esclaves en mode `--skip-bdb` , `--low-priority-updates` et `--delay-key-write=ALL` pour accélérer les esclaves. Dans ce cas, l'esclave va utiliser les tables non transactionnelles `MyISAM` au lieu des tables `BDB` pour obtenir plus de vitesse.

Q : Que dois−je faire pour préparer mon code client à la réplication?

R : Si la partie de votre code qui réalise les accès aux bases de données a été proprement modularisée, la convertir en une configuration qui supporte la réplication ne sera pas un problème :

modifiez simplement votre base pour qu'elle aille lire sur les esclaves et le maître, mais ne fasse que des modifications avec le maître. Si votre code n'a pas ce niveau d'abstraction, l'installation du système de réplication vous donnera alors la motivation ou la raison pour le faire. Vous devriez commencer par créer une couche d'abstraction ou un module avec les fonctions suivantes :

- `safe_writer_connect()`
- `safe_reader_connect()`
- `safe_reader_query()`
- `safe_writer_query()`

`safe_` signifie que la fonction devra prendre en charge toutes les conditions d'erreurs. Naturellement, vous pouvez utiliser des noms différents pour les fonctions. L'important est de savoir clairement laquelle se connecte en écriture et laquelle se connecte en lecture, et laquelle lit et laquelle écrit.

Vous devriez alors convertir votre code client pour qu'il utilise cette bibliothèque. Cela peut être un processus laborieux et déroutant, mais il va s'avérer payant dans le long terme. Toutes les applications qui suivent la technique ci−dessus pourront alors prendre avantage des solutions de réplication. Le code sera aussi bien plus facilement entretenu, et ajouter des options sera trivial. Vous devrez modifier une ou deux fonctions, comme par exemple pour enregistrer le temps de calcul de certaines requêtes, ou les requêtes qui vous retournent des erreurs.

Si vous avez écrit beaucoup de code jusqu'ici, vous pourriez vouloir automatiser la conversion en utilisant l'utilitaire de Monty, `replace` , qui est distribué avec la distribution standard de MySQL, ou bien simplement en écrivant un script Perl. Avec un peu de chance, votre code suit des conventions connues. Si ce n'est pas le cas, alors vous serez peut être conduit à réécrire votre application de toutes manières, ou bien, à lui appliquer des méthodes à la main.

Q : Quand et combien de réplications de MySQL permettent d'améliorer les performances de mon système?

R : La réplication MySQL est particulièrement avantageuse pour les systèmes qui gèrent des lectures fréquentes, et des écritures plus rares. En théorie, en utilisant uniquement un maître et beaucoup d'esclaves, vous pouvez augmenter les performances de votre système jusqu'à saturation de la bande passante ou du maître, pour les modifications.

Afin de déterminer le nombre d'esclaves que vous pouvez obtenir voir les performances de votre système s'améliorer, vous devez bien connaître les types de requêtes que vous utilisez, et empiriquement déterminer la relation entre le nombre de lectures et d'écritures (par secondes, ou maximum absolu), pour un maître et un esclave. L'exemple ci−dessous va vous montrer comment faire des calculs simples.

Imaginons que votre charge système soit constituée de 10% d'écriture et de 90% de lectures. Nous avons aussi déterminé que le maximum de lectures `max_reads` = 1200 −{} 2 * `max_writes` , ou, en d'autres mots, notre système peut voir des pics de 1200 lectures par secondes sans aucune écritures, notre temps d'écriture moyen est deux fois plus temps qu'une lecture, et la relation est linéaire. Supposons que notre maître et notre esclave sont de la même capacité, et que nous avons N esclaves et un maître. Nous avons alors pour chaque serveur (maître ou esclave) :

`lectures = 1200 - 2 * écriture` (issue des tests)

`lectures = 9* écriture / (N + 1)` (lectures réparties, mais toutes les écritures vont à tous les serveurs)

`9*écriture/(N+1) + 2 * écriture = 1200`

`écriture = 1200/(2 + 9/(N+1)`

- Si N = 0, ce qui signifie que nous n'avons pas de réplication, notre système peut gérer

1200/11, environs 109 écritures par secondes, ce qui signifie (que nous aurons 9 fois plus de lectures que d'écritures, étant donné la nature de notre application).
- Si N = 1, nous pouvons monter à 184 écriture par seconde.
- Si N = 8, nous pouvons monter à 400 écriture par seconde.
- Si N = 17, nous pouvons monter à 480 écriture par seconde.
- Eventuellement, si N se rapproche de l'infini (et notre budget de l'infini négatif), nous pourrons nous rapprocher de 600 écritures par secondes, en améliorant le système 5,5 fois. Toutefois, avec 8 serveurs, nous avons pu améliorer le système de 4 fois.

Notez que nos calculs ont supposés une bande passante infinie, et que nous avons négligé des facteurs qui pourraient être significatifs pour notre système. Dans de nombreux cas, nous ne pourrions pas faire de calculs précis pour prédire l'état de notre système avec N esclaves de réplication. Toutefois, répondre aux questions ci−dessus vous permettra de décider si la réplication est une solution à votre problème ou pas.

- Quel est le ratio d'écriture/lecture de votre système?
- Quelle est la charge maximale d'un serveur en écriture, si vous pouvez limiter les lectures?
- Combien d'esclaves votre réseau peut supporter?

Q : Comment puis−je utiliser la réplication pour fournir un système à haute tolérance de panne?

R : Avec les fonctionnalités actuellement disponible, vous devez configurer un serveur et un esclave (ou plusieurs esclaves), et écrire un script qui va surveiller le maître pour voir si il fonctionne , et instruire votre applcation et les esclaves d'un changement de maître en cas d'échec. Voici des suggestions :

- Utilisez la commande `CHANGE MASTER TO` pour changer un esclave en maître.
- Un bon moyen de garder votre application informé du maître courant est d'utiliser les DNS dynamiques, vous pouvez attribuer au maître. Avec `bind` , vous pouvez utiliser `nsupdate` pour modifier dynamiquement votre DNS.
- Vous devez faire fonctionner vos esclaves avec l'option `log-bin` et sans l'option `log-slave-updates` . De cette façon, l'esclave sera prêt à prendre le relais dès que vous lui enverrez la commande `STOP SLAVE` ; envoyez `RESET MASTER` et `CHANGE MASTER TO` aux autres esclaves. Par exemple, considérez l'architecture suivante (``M'' représente le maître, ``S'' les esclaves, ``WC'' les clients qui émettent des commandes de lecture et écriture. Les clients qui ne font que des lectures ne sont pas représentés, car ils n'ont pas à changer quoi que ce soit.

```
       WC
        \
         v
 WC----> M
       / | \
      /  |  \
     v   v   v
    S1   S2  S3
```

S1 (comme S2 et S3) est un esclave qui fonctionne avec les options `--log-bin` et sans `--log-slave-updates` . Comme les seules écritures faites sur S1 sont celles qui sont répliquées depuis M, le log binaire de S1 est vide : n'oubliez pas que S1 fonctionne sans `--log-slave-updates` . Puis, pour une raison quelconque, M devient inaccessible, et vous voulez que S1 devienne le nouveau maître (c'est à dire, les WC sont dirigées vers S1, et S2 et S3 répliquent S1.

Assurez vous que tous les esclaves ont fini de traîter les requêtes de leur log de relais. Sur chaque esclave, faites `STOP SLAVE IO_THREAD` , puis vérifiez le résultat de la commande `SHOW PROCESSLIST` jusqu'à ce que vous lisiez `Has read all relay log` . Lorsque cela est vrai pour tous les esclaves, ils peuvent être reconfigurés vers un nouveau maître. Faites `STOP SLAVE` sur chaque esclave, et `RESET MASTER` sur celui qui devient le maître, puis `CHANGE MASTER` sur les autres.

Aucun WC n'accède à M. Reconfigurez les WC pour qu'ils dirigent leurs requêtes sur S1. A partir de maintenant, les requêtes envoyées par WC à S1 sont écrites dans le log binaire. Le log binaire de S1 contient maintenant exactement chaque requête envoyée à S1 depuis que M est mort. Sur S2 (et S3), faites `STOP SLAVE` , `CHANGE MASTER TO MASTER_HOST='S1'` (où `'S1'` est remplacé par le vrai nom d'hôte de S1). Pour changer le maître, utilisez la commande `CHANGE MASTER` , et ajoutez les informations de connexion à S1 depuis S2 et S3 (utilisateur, mot de passe, port). Dans `CHANGE MASTER` , il n'y a pas besoin de spécifier le nom du fichier de log binaire de S1, ou la position dans le log : nous savons que c'est le premier fichier et le premier offset (position 4), car ce sont les coordonnées par défaut utilisées par `CHANGE MASTER` . Finalement, lances `START SLAVE` sur S2 et S3, et maintenant, vous avez ceci :

```
        WC
      /
      |
 WC   |  M(indisponible)
  \   |
    \ |
     v v
     S1<--S2  S3
      ^       |
      +-------+
```

Lorsque M est de nouveau disponible, vous devez utiliser la commande `CHANGE MASTER` comme vous l'avez fait avec S2 et S3, pour que M devienne l'esclave de S1 et rattrappe toutes les modifications que les WC ont émise, et qu'il a manqué. Puis, pour refaire de M le maître, suivez la même procédure que précédemment, comme si S1 était indispoinble et que M prenait le relais. Durant la procédure, n'oubliez pas d'utiliser la commande `RESET MASTER` sur M avant de faire de S1, S2 et S3 des esclaves de M, car ils risquent de reprendre les anciennes requêtes des WC, qui datent d'avant l'indisponibilité de M.

Nous travaillons actuellement à l'intégration automatique de l'élection d'un nouveau maître, mais jusqu'à ce que ce soit près, vous devez créer votre propre outil de surveillance.

6.9 Correction de problèmes courants

Si vous avez suivi les instructions, et que votre configuration de réplication ne fonctionne pas, commencez par supprimer les problèmes liés à l'utilisateur comme ceci :

- **Vérifiez les messages d'erreurs dans les logs** . De nombreux utilisateurs ont perdu du temps en ne faisant pas cela en premier.
- Est−ce que le maître enregistre dans le log binaire ? Vérifiez avec la commande `SHOW MASTER STATUS` . Si il le fait, la variable `Position` doit être non nulle. Si ce n'est pas le cas, vérifiez que vous avez donné au serveur l'option `log-bin` et que vous lui avez donné un `server-id` .
- Est−ce que l'esclave fonctionne? Vérifiez le avec `SHOW SLAVE STATUS` . La réponse se trouve dans la colonne `Slave_running` . Si ce n'est pas le cas, vérifiez les options de l'esclave, et vérifiez le fichier de log d'erreurs.
- Si l'esclave fonctionne, a−t−il établit une connexion avec le maître? Exécutez la commande `SHOW PROCESSLIST` , et recherchez un utilisateur avec la valeur `system user` dans la colonne `User` et `none` dans la colonne `Host` , et vérifiez la colonne `State` . Si elle indique `connecting to master` , vérifiez les droits de connexion pour l'utilisateur de réplication sur le serveur, ainsi que le nom de l'hôte, votre configuration DNS, le fonctionnement du maître, et si tout est OK, vérifiez le fichier de log d'erreurs.
- Si l'esclave fonctionnait, mais s'est arrêté, vérifiez le résultat de la commande SHOW SLAVE STATUS, et vérifiez le fichier de log d'erreurs. Il arrive que certaines requêtes réussissent sur le maître mais échouent sur l'esclave. Cela ne devrait pas arriver si vous avez pris la bonne sauvegarde du maître, et que vous n'avez jamais modifié les données sur le serveur esclave, autrement que par le truchement de l'esclave de réplication. Si c'est le cas, c'est un bogue, et vous devez le rapporter. Voyez plus loin pour savoir comment rapporter un bogue.

- Si une requête qui a réussit sur le maître, refuse de s'exécuter sur l'esclave, et qu'une synchronisation complète de la base ne semble pas possible, essayez ceci :
    - Commencez par voir s'il n'y a pas de lignes différentes de celles du maître. Essayez de comprendre comment elle a plus se trouver là, effacez−la, et essayez de redémarrer l'esclave avec `SLAVE START` . (cela peut être un bug : lisez les logs sur le manuel MySQL, http://www.mysql.com/documentation , pour savoir si c'est un bug et s'il est corrigé).
    - Si la solution ci−dessus ne fonctionne pas ou ne s'applique pas, essayez de comprendre si c'est risqué de faire une correction à la main (au besoin) puis, ignorez la prochaine requête du maître.
    - Si vous avez décidé que vous pouvez vous passer de la prochaine requête, utilisez la commande suivante :

      ```
       mysql> SET GLOBAL SQL_SLAVE_SKIP_COUNTER = n;
      mysql> START SLAVE;
      ```

      La valeur de `n` doit être de 1 si la requête n'utilise pas de valeur `AUTO_INCREMENT` ou `LAST_INSERT_ID()` . Sinon, la valeur doit être de 2. La raison pour utiliser la valeur 2 pour les requêtes qui utilisent `AUTO_INCREMENT` ou `LAST_INSERT_ID()` est qu'elles requièrent deux lignes dans le log binaire.
    - Si vous êtes sûrs que l'esclave est parfaitement synchronisé avec le maître, et que personne n'a mis à jour les tables impliquées, rapportez nous un bug.

6.10 Rapporter des bugs de réplication

Lorsque vous avez bien vérifié qu'il n'y a pas de problèmes avec les utilisateurs impliqués, et que la réplication ne fonctionne pas ou qu'elle est instable, il est temps d'envoyer un rapport de bug. Nous avons besoin d'autant d'information que possible pour rechercher le bug. N'hésitez pas à investir quelques efforts lors de la préparation d'u rapport de bug.

Si vous avez un moyen de reproduire le bug, alors indiquez le dans notre base de bugs à l'adresse http://bugs.mysql.com/ . Si vous avez un problème fantôme (un qui ne peut être réproduire ``à souhait''), utilisez la procéduire suivante :

- Vérifiez qu'il n'y a pas d'erreur utilisateur impliquée. Par exemple, si vous modifiez l'esclave sans passer par le thread esclave, les données seront désynchroniséez et vous pourrez alors rencontrer des problèmes de contraintes de clés uniques durant les modifications. Dans ce cas, l'esclave doit être arrêté et nettoyé manuellement pour être synchronisé avec le maître. Ce n'est pas un problème de réplication : c'est un problème d'interference extérieure, qui conduit à l'échec de la réplication.
- Exécutez l'esclave avec les options `--log-slave-updates` et `--log-bin` . Elles font que l'esclave va enregistrer les modifications qu'il reçoit dans ses propres logs binaires.
- Sauvez toutes les preuves avant de remettre à zéro l'état de la réplication. Si vous n'avez aucune information, ou seulement des informations partielles, cela nous prendra plus de temps pour rechercher le problème. Les preuves que vous devez rassembler sont :
    - Tous les logs binaires du maître
    - Tous les logs binaires de l'esclave
    - Le résultat de la commande `SHOW MASTER STATUS` sur le maître au moment du problème.
    - Le résultat de la commande `SHOW SLAVE STATUS` sur l'esclave au moment du problème.
    - Les logs d'erreur du maître et de l'esclave.
- Utilisez `mysqlbinlog` pour éxaminer les logs binaires. La commande suivante doit permettre d'identifier la requête coupable :

  ```
  mysqlbinlog −j pos_from_slave_status /path/to/log_from_slave_status | head
  ```

Une fois que vous avez rassemblez toutes ces preuves du problème fantôme, essayer de l'isoler dans des cas de tests indépendants. Puis, soumettez le problème dans notre base de bugs à l'adresse http://bugs.mysql.com/ avec toute autre information importante.

7 Optimisation de MySQL

L'optimisation est une tâche complexe car elle nécessite une parfaite compréhension du système en entier. Alors qu'il serait possible de faire quelques optimisations localement avec une faible connaissance de votre système ou de votre application, plus vous voulez un système optimal, plus il est nécessaire de le connaître.

Ce chapitre va tenter d'expliquer et de donner des exemples de différentes manières d'optimiser MySQL. Souvenez−vous, malgré tout, qu'il existe toujours d'autres moyens (de plus en plus difficiles) de rendre le système plus véloce.

7.1 Présentation de l'optimisation

Le plus important pour obtenir un système rapide est bien sûr le schéma utilisé. Il faut également savoir le genre de choses que fera le système et où seront les goulots d'étranglement.

Les ralentissements les plus courants sont :

- Les recherches sur les disques. Il faut du temps pour que le disque trouve un morceau de donnée. Avec les disques modernes en 1999, le temps d'accès est généralement inférieur à 10ms, donc on peut en théorie faire près de 100 recherches par seconde. Le temps s'améliore un peu avec les nouveaux disques et il est difficile de l'optimiser pour une seule table. La méthode pour l'optimiser est de disséminer les données sur plus d'un disque.
- Les accès disques en lecture/écriture. Une fois le disque à la bonne position, il faut lire les données. Avec des disques modernes en 1999, un disque fournit environ 10 à 20 Mo/s. Ceci est plus facile à optimiser que les recherches car on peut lire en parallèle sur plusieurs disques.
- Les cycles processeur. Quand les données sont dans la mémoire centrale (ou si elles y étaient déjà) il faut les traiter pour obtenir le résultat. Avoir des tables de petite taille par rapport à la mémoire est le facteur limitant le plus courant. Mais avec des petites tables, la célérité n'est souvent pas un souci.
- La bande passante de la mémoire. Quand le processeur a besoin de plus d'informations qu'il ne peut en stocker dans son cache, la bande passante de la mémoire centrale devient le goulot d'étranglement. C'est un étranglement peu courant pour la majorité des systèmes, mais il faut être averti.

7.1.1 Limitations et inconvénients des choix conceptuels de MySQL

Avec les tables de type MyISAM, MySQL utilise un verrouillage extrêmement rapide (plusieurs lectures / une seule écriture). Le plus gros problème avec ce type de table survient quand vous avez un mélange de flux de modifications et des sélections lentes sur la même table. Si c'est une problème sur plusieurs tables, vous pouvez utiliser un autre type de table pour celles ci. Types de tables MySQL .

MySQL peut utiliser à la fois des tables transactionnelles et des tables non−transactionnelle. Pour pouvoir travailler tranquillement avec des tables non−transactionnelles (qui n'ont pas la possibilité de revenir en arrière si quelque chose se passe mal) MySQL suit les règles suivantes:

- Toutes les colonnes ont une valeur par défaut.
- Si vous insérez une mauvaise valeur dans une colonne (par exemple `NULL` dans une colonne `NOT NULL`, ou encore une valeur numérique trop grande dans une colonne numérique), MySQL

prendra en compte "la meilleure valeur possible" plutôt que de sortir une erreur. Pour les valeurs numériques, il s'agit de 0, de la valeur la plus petite possible, ou de la valeur la plus grande possible. Pour les chaînes, il s'agit soit d'une chaîne vide, soit de la chaîne la plus longue que peut contenir la colonne.
- Toutes les expressions calculées retournent une valeur qui peut être utilisées à la place d'afficher un message d'erreur. Par exemple, 1/0 retourne `NULL`

Pour plus d'informations, voyez Comment MySQL gère les contraintes .

Ce qui précède signifie qu'il ne faut pas que le contrôle du contenu des champs soit fait au niveau de MySQL, mais au niveau de l'application.

7.1.2 Portabilité

Comme tous les serveurs SQL implémentent différemment le langage SQL, cela prend de solides connaissances pour écrire des applications SQL portables. Pour les insertions et sélections simples, c'est très simple, mais plus vos besoins se complexifient, plus c'est abscons. Si vous voulez une application qui fonctionne rapidement sur de nombreuses bases de données, c'est même encore plus difficile.

Pour rendre une application complexe portable, vous pouvez commencer par choisir une panoplie de serveurs SQL avec lesquels travailler.

Vous pouvez utiliser le programme/page web de MySQL appelé `crash-me` http://www.mysql.com/information/crash−me.php pour trouver les fonctions, types et limites que vous pouvez utiliser avec un panel de serveurs de bases de données. Les tests de `crash-me` ne vérifient pas tout, mais il est déjà très exhaustif avec plus de 450 points de tests.

Par exemple, vous ne devriez pas avoir de nom de colonne supérieur à 18 caractères, si vous voulez pouvoir utiliser Informix ou DB2.

Les programmes de tests `crash-me` et de performances de MySQL sont très indépendants du serveur. En regardant comment nous avons géré ces situations, vous pouvez comprendre comment rendre votre propre code indépendant du serveur. Les tests de performances sont situés dans le dossier `sql-bench` de la distribution source de MySQL. Ils sont écrits en Perl avec l'interface `DBI` , ce qui résout les problèmes de connexion.

Voyez http://www.mysql.com/information/benchmarks.html pour connaître les résultats de ces benchmarks.

Comme vous pouvez le voir avec ces résultats, toutes les bases de données ont leur point faible. En réalité, elles ont toutes une approche différente du même problème, et cela conduit à des comportements spécifiques.

Si vous avez besoin de l'indépendance au serveurs de bases de données, vous devez bien connaître les faiblesses de chaque serveur. MySQL est très rapide pour lire et modifier les données, mais peine lorsque les lectures et écritures sont lentes sur la même table. Oracle, d'un autre coté, a de gros problèmes lorsque vous essayez d'accéder aux données que vous avez modifié récemment (jusqu'à ce qu'elles soient écrites sur le disque). Les bases de données transactionnelles en général ne sont pas très douées pour générer des tables résumés à partir des tables de log, car dans ce cas, le verrouillage de ligne est inutile.

Pour rendre votre application **reellement** indépendante de la base de données, vous devez définir un classe très souple à travers laquelle vous allez vous interfacer pour manipuler vos données. Comme le langage C++ est disponible sur la plupart des systèmes, cela rend les classes C++ très pratiques pour cette tâche.

Si vous utilisez une fonctionnalité spécifique d'une base de données (comme la commande `REPLACE` de MySQL), il vous faut aussi coder la même commande pour les autres serveurs (qui sera alors plus lente). Avec MySQL, vous pouvez aussi utiliser la syntaxe `/*! */` pour utiliser des mots clés spécifiques de MySQL dans une requête. Le code entre `/* */` sera alors traité comme un commentaire et ignoré par la plupart des autres serveurs SQL.

Si les hautes performances sont plus importantes que l'exactitude, comme pour les applications web, il est possible de créer une couche application qui met en cache les résultats et vous donne de meilleures performances. En laissant les anciens résultats se périmer, vous pouvez garder un cache à jour. Cela vous donne une méthode pour gérer les grandes charges, durant lesquelles vous pouvez augmenter la taille du cache, et augmenter la durée de vie.

Dans ce cas, les informations de création de tables doivent contenir les informations de taille initiale du cache, et la fréquence de rafraîchissement des tables. Le cache de requêtes MySQL .

7.1.3 Pour quoi avons nous utilisé MySQL ?

Pendant le développement initial de MySQL, les fonctions de MySQL ont été créées pour convenir à un maximum de clients. Celles ci supporte des entrepôts de données pour deux des plus gros revendeurs suédois.

Nous recevons chaque semaine le résumé de toutes les transactions par carte de toutes les boutiques, et nous sommes chargés de fournir des informations utiles aux gérants des boutiques pour les aider à comprendre comment leurs propres campagnes publicitaires touchent leurs clients.

Les données sont assez énormes (près de 7 millions de résumés de transactions par mois), et nous avec les données de 4−10 ans que nous présentons aux utilisateurs. Nous avons chaque semaine des requêtes des clients qui veulent un accès 'instantané' aux nouveaux rapports sur ces données.

Nous avons réussi en stockant toutes les informations dans des tables de 'transactions' compressées. Nous avons une série de macros (scripts) qui génère des tables de résumés groupés par différents critères (groupe de produits, identifiant de client, boutique ...). ces rapports sont des pages web générées dynamiquement par un petit script Perl qui parcours une page web, exécute les requêtes SQL, et insère les résultats. Nous aurions bien utilisé PHP ou mod_perl à la place, mais ils n'étaient pas disponibles à cette époque.

Nous avons écrit un outil en `C` pour la représentation graphique des données qui génère des GIFs à partir du résultat de requêtes SQL (avec quelques traitements sur le résultat). Ceci est également effectué dynamiquement par le script Perl qui parcourt les fichiers `HTML` .

Pour la plupart des cas, un nouveau rapport peut simplement être fait en copiant un script existant, et en modifiant la requête SQL qu'il exécute. Dans certains cas, nous aurons besoin d'ajouter des champs à une table de résumé existante ou d'en générer une nouvelle, mais c'est tout de même toujours assez simple, car nous gardons toutes les tables de transactions sur disque. (Actuellement, nous avons au moins 50 Go de tables de transactions et 200 Go d'autres données sur les clients.)

Nous donnons également accès aux tables de résumés à nos clients directement avec ODBC, de sorte que les utilisateurs avancés puissent traiter les données eux−mêmes .

Nous n'avons eu aucun problème à supporter tout cela avec une relativement modeste Sun Ultra SPARCStation (2x200 MHz). Nous avons récemment amélioré l'un de nos serveurs en un bi−CPU 400 MHz UltraSPARC, et nous projetons actuellement de supporter les transactions au niveau du produit, ce qui signifie un décuplement des données. Nous pensons pouvoir y arriver uniquement en ajoutant des disques supplémentaires à nos systèmes.

Nous expérimentons aussi Intel−Linux, pour pouvoir avoir plus de puissance CPU pour moins cher. Comme nous utilisons désormais le format binaire portable pour les bases de données (nouveauté de la version 3.23), nous utiliserons cela pour quelques parties de l'application.

Nous avons au départ le sentiment que Linux s'acquitera mieux des faibles et moyennes charges tandis que Solaris fonctionnera mieux sur les grosses charges à cause des I/O disques extrêmes, mais nous n'avons actuellement aucune conclusion à ce propos. Après quelques discussion avec un développeur du noyau Linux, un effet de bord de Linux pourrait tant de ressources aux travaux de traitement que les performances de l'interface interactive peut devenir vraiment lente. Cela fait apparaître la machine très lente et sans réponse lorsque de gros traitements sont en cours. Heureusement, cela sera mieux géré dans les futurs noyaux de Linux.

7.1.4 La suite de tests MySQL

Ceci devrait comprendre une description technique de la suite de tests de performances de MySQL (et `crash-me`), mais cette description n'est pas encore écrite. Actuellement, vous pouvez vous faire une idée des tests en regardant le code et les résultats dans le répertoire `sql-bench` dans toutes les distributions de sources de MySQL.

Cette suite de test est censée permettre à utilisateur de comparer ce qu'une implémentation SQL donnée réussi bien ou mal.

Sachez que ces tests de performances lancent en un seul thread, donc il mesure le temps minimum pour chaque opération. Nous projetons pour le futur d'ajouter de nombreux tests multi−thread à cette suite de tests.

Par exemple, (tous ont été lancés sur une même machine NT 4.0)

Lecture de 2000000 lignes indexées	**Secondes**	**Secondes**
mysql	367	249
mysql_odbc	464	
db2_odbc	1206	
informix_odbc	121126	
ms−sql_odbc	1634	
oracle_odbc	20800	
solid_odbc	877	
sybase_odbc	17614	

Insertion de lignes (350768	**Secondes**	**Secondes**
mysql	381	206
mysql_odbc	619	
db2_odbc	3460	
informix_odbc	2692	
ms−sql_odbc	4012	
oracle_odbc	11291	
solid_odbc	1801	
sybase_odbc	4802	

Le test ci−dessus a été exécuté avec un index de cache de 8 Mo.

Nous avons rassemblé d'autres résultats de tests à http://www.mysql.com/information/benchmarks.html .

Notez que Oracle n'est pas inclus dans ces tests car ils ont demandé à être retirés. Tous les tests d'Oracle doivent être faits par Oracle! Nous croyons que cette politique va biaiser **fortement** les tests en faveur de Oracle, car les tests ci−dessus sont supposés montrer ce qu'une installation simple peut faire pour un client simple.

Pour utiliser la suite de tests, les prerequis suivants doivent être vérifiés :

- La suite de tests est disponible dans la distribution source de MySQL, et vous devez avoir téléchargé cette distribution. Vous pouvez télécharger la version publiée sur le site de http://www.mysql.com/downloads/ , ou utiliser celle du serveur de développement (Installer à partir des sources de l'arbre de développement).
- Les scripts de tests ont été écrits en Perl, et utilisent le module Perl `DBI` pour accéder aux serveurs, donc `DBI` doit être installée. Vous aurez aussi besoin des pilotes spécifiques DBD de chaque serveur que vous voulez tester. Par exemple, pour tester MySQL, PostgreSQL et DB2, les modules DBD::mysql, DBD::Pg et DBD::DB2 doivent être installés. Commentaires sur l'installation de Perl .

La suite de tests est située dans le dossier `sql-bench` de la distribution source de MySQL. Pour exécuter la suite de tests, compilez MySQL, puis allez dans le dossier `sql-bench` et exécutez le script `run-all-tests` :

```
 shell> cd sql-bench
shell> perl run-all-tests --server=server_name
```

`server_name` est un des serveurs supportés. Pour avoir la liste de toutes les options et serveurs supportés. utilisez cette commande :

```
 shell> perl run-all-tests --help
```

Le script `crash-me` est aussi situé dans le dossier `sql-bench` . `crash-me` essaie de déterminer quelles fonctionnalités un serveur supporte, et quelles sont ses limitations. Par exemple, le test détermine :

- Les types de colonnes supportés
- Le nombre d'index supportés
- Les fonctions supportées
- La taille maximale d'une requête
- La taille maximale d'une colonne `VARCHAR`

Vous pouvez retrouver les résultats de `crash-me` sur de nombreuses bases de données à http://www.mysql.com/information/crash−me.php .

7.1.5 Utiliser vos propres tests de performance

Vous devriez vraiment penser à préparer des tests de performances pour votre application et base, afin d'identifier les opérations les plus lentes. En les corrigeant (ou en remplaçant ces opérations des "modules simples") vous pouvez facilement identifier les autres opérations lentes (et ainsi de suite...). Même si la performance générale de votre application est suffisante, vous devriez prévoir où seront les prochains freins, et décider d'anticiper leur résolution, avant que vous n'ayez vraiment besoin de ces performances.

Pour avoir un exemple de programme de tests portables, voyez la suite de tests MySQL. La suite de tests comparatifs de MySQL . Vous pouvez prendre n'importe quel programme de cette suite, le modifier pour l'adapter à vos besoins, et essayer différentes solutions à votre problème : il suffit de tester et d'identifier la solution la plus rapide pour vous.

Une autre suite de tests est la " `Open Source Database Benchmark` ", disponible sur le site de http://osdb.sourceforge.net/ .

Il est très fréquent que des problèmes surviennent lorsque le système subit une forte charge. Nous avons de nombreux clients qui nous contactent lorsqu'ils ont mis leur système en production, et rencontré des problèmes de charge. Pour chacun d'entre eux, les problèmes étaient des problèmes simples de conceptions (les scans de tables ne sont **pas bons** sous forte charge) ou des problèmes liés au système d'exploitation ou les bibliothèques. La plupart auraient été vraiment **plus simples** à tester si le système n'était pas déjà en production.

Pour éviter des problèmes comme ceux−là, vous devriez mettre quelques efforts dans les tests de votre application dans son ensemble, avant de la mettre dans les pires conditions. Vous pouvez utiliser le programme `Super Smack` pour cela, qui est disponible à http://www.mysql.com/Downloads/super−smack/super−smack−1.0.tar.gz . Comme son nom le suggère, il va mettre votre système à genoux si vous lui demandez, alors assurez vous de ne l'utiliser qu'avec votre système de développement.

7.2 Optimisation des commandes SELECT et autres requêtes

Premièrement, ce qui affecte toutes les requêtes : plus votre système de droits est compliqué, plus vous aurez des baisses de performances.

Si vous n'avez aucun `GRANT` effectué, MySQL optimisera les vérifications de droits. Donc, si vous avez un système volumineux, il serait bénéfique d'éviter les grants. Sinon les performances seront réduites. Par exemple, si vous n'avez pas de droits de niveau table ou colonne, le serveur n'a pas à vérifier le contenu des tables `tables_priv` et `columns_priv` . Similairement, si vous n'avez pas de limites de ressources, le serveur n'a pas de comptes de ressources à faire. Si vous avez un très haut niveau de requêtes, il peut se révéler bénéfique d'utiliser une structure de droits simplifiée, pour réduire le temps de vérification.

Si votre problème est spécifique à une expression MySQL ou une fonction, vous pouvez utiliser la fonction `BENCHMARK()` du client `mysql` pour effectuer un test de performances. La syntaxe est `BENCHMARK(loop_count,expression)` . Par exemple :

```
 mysql> SELECT BENCHMARK(1000000,1+1);
+------------------------+
| BENCHMARK(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.32 sec)
```

Ce qui précède montre que MySQL peut exécuter 1 000 000 d'additions en 0.32 secondes sur un `PentiumII 400MHz` .

Toutes les fonctions MySQL sont sensé être optimisées, mais il peut y avoir quelques exceptions et la fonction `BENCHMARK(nombre_de_fois,expression)` est un très bon moyen de trouver ce qui cloche dans vos requêtes.

7.2.1 Syntaxe de EXPLAIN (Obtenir des informations sur les SELECT)

```
 EXPLAIN tbl_name
```

Ou :

```
EXPLAIN SELECT select_options
```

`EXPLAIN nom_de_table` est un synonyme de `DESCRIBE nom_de_table` ou `SHOW COLUMNS FROM nom_de_table` .

- La syntaxe `EXPLAIN tbl_name` est synonyme de `DESCRIBE tbl_name` ou `SHOW COLUMNS FROM tbl_name` .
- Lorsque vous faites précéder une commande `SELECT` avec le mot clé `EXPLAIN` , MySQL vous explique comment il va traiter la commande `SELECT` , choisir les tables et index pour les jointures.

Cette section fournit des informations sur comment utiliser `EXPLAIN` .

Avec l'aide de `EXPLAIN` , vous pouvez identifier les index à ajouter pour accélérer les commandes `SELECT` .

Vous devriez souvent utiliser la commande `ANALYZE TABLE` pour mettre à jour les statistiques de cardinalité de vos tables, qui affectent les choix de l'optimiseur. Syntaxe de `ANALYZE TABLE` .

Vous pouvez aussi voir si l'optimiseur fait les jointures dans un ordre vraiment optimal. Pour forcer l'optimiseur à utiliser un ordre spécifique de jointure dans une commande `SELECT` , ajoutez l'attribut `STRAIGHT_JOIN` à la clause.

Pour les jointures complexes, `EXPLAIN` retourne une ligne d'information pour chaque table utilisée dans la commande `SELECT` . Les tables sont listées dans l'ordre dans lequel elles seront lues. MySQL résout toutes les jointures avec une seule passe multi−jointure. Cela signifie que MySQL lit une ligne dans la première table, puis recherche les lignes qui correspondent dans la seconde, puis dans la troisième, etc. Lorsque toutes les tables ont été traitées, MySQL affiche les colonnes demandées, et il remonte dans les tables jusqu'à la dernière qui avait encore des lignes à traiter. La prochaine ligne est alors traitée de la même façon.

Avec MySQL version 4.1 l'affichage de `EXPLAIN` a été modifié pour mieux fonctionner avec les structures comme `UNION` , sous−requêtes, et tables dérivées. La plus importante évolution est l'addition de deux nouvelles colonnes : `id` et `select_type` .

Le résultat de la commande `EXPLAIN` est constitué des colonnes suivantes :

`id`
: identifiant de `SELECT` , le numéro séquentiel de cette commande `SELECT` dans la requête.

`select_type`
: Type de clause `SELECT` , qui peut être :

  `SIMPLE`
  : `SELECT` simple (sans utiliser de clause `UNION` ou de sous−requêtes).

  `PRIMARY`
  : `SELECT` extérieur.

  `UNION`
  : Second et autres `UNION SELECT` s.

  `DEPENDENT UNION`
  : Second et autres `UNION SELECTS` s, dépend de la commande extérieure.

  `SUBQUERY`
  : Premier `SELECT` de la sous−requête.

  `DEPENDENT SUBSELECT`
  : Premier `SELECT` , dépendant de la requête extérieure.

  `DERIVED`
  : Table dérivée `SELECT` .

`table`
: La table à laquelle la ligne fait référence.

`type`
: Le type de jointure. Les différents types de jointures sont les suivants, dans l'ordre du plus efficace au plus lent :

  `system`

La table a une seule ligne (c'est une table système). C'est un cas spécial du type de jointure `const` .

`const`

La table a au plus une ligne correspondante, qui sera lue dès le début de la requête. Comme il n'y a qu'une seule ligne, les valeurs des colonnes de cette ligne peuvent être considérées comme des constantes pour le reste de l'optimiseur. Les tables `const` sont très rapides, car elles ne sont lues qu'une fois.

`const` est utilisé lorsque vous comparez toutes les parties d'une clé `PRIMARY` / `UNIQUE` avec des constantes :

```
 SELECT * FROM const_table WHERE primary_key=1;

 SELECT * FROM const_table
WHERE primary_key_part1=1 AND primary_key_part2=2;
```

`eq_ref`

Une ligne de cette table sera lue pour chaque combinaison de ligne des tables précédentes. C'est le meilleur type de jointure possible, à l'exception des précédents. Il est utilisé lorsque toutes les parties d'un index sont utilisées par la jointure, et que l'index est `UNIQUE` ou `PRIMARY KEY` . `eq_ref` peut être utilisé pour les colonnes indexées, qui sont comparées avec l'opérateur `=` . L'élément comparé doit être une constante ou une expression qui utiliser les colonnes de la table qui est avant cette table.

Dans l'exemple suivant, `ref_table` sera capable d'utiliser `eq_ref` :

```
 SELECT * FROM ref_table,other_table
WHERE ref_table.key_column=other_table.column;

 SELECT * FROM ref_table,other_table
WHERE ref_table.key_column_part1=other_table.column
AND ref_table.key_column_part2=1;
```

`ref`

Toutes les lignes avec des valeurs d'index correspondantes seront lues dans cette table, pour chaque combinaison des lignes précédentes. `ref` est utilisé si la jointure n'utilise que le préfixe de gauche de la clé, ou si la clé n'est pas `UNIQUE` ou `PRIMARY KEY` (en d'autres termes, si la jointure ne peut pas sélectionner qu'une seule ligne en fonction de la clé). Si la clé qui est utilisée n'identifie que quelques lignes à chaque fois, la jointure est bonne. `ref` peut être utilisé pour les colonnes indexées, qui sont comparées avec l'opérateur `=` .

Dans les exemples suivants, `ref_table` sera capable d'utiliser `ref` .

```
 SELECT * FROM ref_table WHERE key_column=expr;

 SELECT * FROM ref_table,other_table
WHERE ref_table.key_column=other_table.column;

 SELECT * FROM ref_table,other_table
WHERE ref_table.key_column_part1=other_table.column
AND ref_table.key_column_part2=1;
```

`ref_or_null`

Comme `ref` , mais avec le coût supplémentaire pour les recherches couvrant les valeurs `NULL` . Ce type de jointure est nouveau en MySQL 4.1.1 est sert essentiellement à la résolution des sous−requêtes.Dans les exemples suivants, MySQL peut utiliser une jointure `ref_or_null` pour traiter `ref_table` :

```
 SELECT * FROM ref_table
WHERE key_column=expr OR key_column IS NULL;
```

Comment MySQL optimise les clauses `IS NULL` .

`index_merge`

Ce type de jointure indique que l'optimisation de type `Index Merge` est utilisée. Dans ce cas, la colonne `key` contient une liste d'index utilisés, et `key_len` contient la liste des plus longues parties de clés utilisées. Pour plus d'informations, voyez Comment MySQL optimise les clauses `OR` .

`unique_subquery`

Ce type remplace le type `ref` dans certaines sous−requêtes `IN` de la forme suivante :

```
value IN (SELECT primary_key FROM single_table WHERE some_expr)
```

`unique_subquery` est simplement une analyse d'index, qui remplace complètement la sous−requête pour une meilleure efficacité.

`index_subquery`

Ce type de jointure est similaire à `unique_subquery` . Elle remplace des sous−requêtes `IN` , mais elle fonctionne pour les index non−uniques dans les sous−requêtes de la forme suivante :

```
value IN (SELECT key_column FROM single_table WHERE some_expr)
```

`range`

Seules les lignes qui sont dans un intervalle donné seront lues, en utilisant l'index pour sélectionner les lignes. La colonne `key` indique quel est l'index utilisé. `key_len` contient la taille de la partie de la clé qui est utilisée. La colonne `ref` contiendra la valeur `NULL` pour ce type. `range` peut être utilisé lorsqu'une colonne indexée est comparée avec une constante comme `=` , `<>` , `>` , `>=` , `<` , `<=` , `IS NULL` , `<=>` , `BETWEEN` ou `IN` .

```
 SELECT * FROM tbl_name
WHERE key_column = 10;

 SELECT * FROM tbl_name
WHERE key_column BETWEEN 10 and 20;

 SELECT * FROM tbl_name
WHERE key_column IN (10,20,30);

 SELECT * FROM tbl_name
WHERE key_part1= 10 AND key_part2 IN (10,20,30);
```

`index`

C'est la même chose que `ALL` , hormis le fait que seul l'arbre d'index est étudié. C'est généralement plus rapide que `ALL` , car le fichier d'index est plus petit que le fichier de données.

Cette méthode peut être utilisée lorsque la requête utilise une colonne qui fait partie d'un index.

`ALL`

Une analyse complète de la table sera faîte pour chaque combinaison de lignes issue des premières tables. Ce n'est pas bon si la première table n'est pas une jointure de type `const` et c'est **très** mauvais dans les autres cas. Normalement vous pouvez éviter ces situations de `ALL` en ajoutant des index basée sur des parties de colonnes.

`possible_keys`

La colonne `possible_keys` indique quels index MySQL va pouvoir utiliser pour trouver les lignes dans cette table. Notez que cette colonne est totalement dépendante de l'ordre des tables. Cela signifie que certaines clés de la colonne `possible_keys` pourraient ne pas être utilisées dans d'autres cas d'ordre de tables.

Si cette colonne est vide, il n'y a pas d'index pertinent. Dans ce cas, vous pourrez améliorer les performances en examinant votre clause `WHERE` pour voir si des colonnes sont susceptibles d'être indexée. Si c'est le cas, créez un index approprié, et examinez le résultat avec la commande `EXPLAIN` . Syntaxe de `ALTER TABLE` .

Pour connaître tous les index d'une table, utilisez le code `SHOW INDEX FROM nom_de_table` .

`key`

La colonne `key` indique l'index que MySQL va décider d'utiliser. Si la clé vaut `NULL` , aucun index n'a été choisi. Pour forcer MySQL à utiliser un index listé dans la colonne `possible_keys` , utilisez `USE KEY/IGNORE KEY` dans votre requête. Syntaxe de `SELECT` .Pour les tables `MyISAM` et `BDB` , la commande `ANALYZE TABLE` va aider l'optimiseur à choisir les meilleurs index. Pour les tables `MyISAM` , `myisamchk --analyze` fera la même chose. Voyez Syntaxe de `ANALYZE TABLE` et Utilisation de `myisamchk` pour maintenir les tables et recouvrir les données .

`key_len`

La colonne `key_len` indique la taille de la clé que MySQL a décidé d'utiliser. La taille est `NULL` si la colonne `key` vaut `NULL` . Notez que cela vous indique combien de partie d'une clé multiple MySQL va réellement utiliser.

`ref`

La colonne `ref` indique quelle colonne ou quelles constantes sont utilisées avec la clé `key` , pour sélectionner les lignes de la table.

`rows`

La colonne `rows` indique le nombre de ligne que MySQL estime devoir examiner pour exécuter la requête.

`Extra`

Cette colonne contient des informations additionnelle sur comment MySQL va résoudre la requête. Voici une explication des différentes chaînes que vous pourriez trouver dans cette colonne :

`Distinct`

MySQL ne va pas continuer à chercher d'autres lignes que la ligne courante, après en avoir trouvé une.

`Not exists`

MySQL a été capable d'appliquer une optimisation de type `LEFT JOIN` sur la requête, et ne va pas examiner d'autres lignes de cette table pour la combinaison de lignes précédentes, une fois qu'il a trouvé une ligne qui satisfait le critère de `LEFT JOIN` .

Voici un exemple de cela :

```
 SELECT * FROM t1 LEFT JOIN t2 ON t1.id=t2.id
WHERE t2.id IS NULL;
```

Supposons que `t2.id` est défini comme `NOT NULL` . Dans ce cas, MySQL va scanner `t1` et rechercher des lignes dans `t2` via `t1.id` . Si MySQL trouve une ligne dans `t2` , il sait que `t2.id` ne peut pas être `NULL` , et il ne va pas scanner le reste des lignes de `t2` qui ont le même `id` . En d'autres termes, pour chaque ligne de `t1` , MySQL n'a besoin que de faire une recherche dans `t2` , indépendamment du nombre de lignes qui sont trouvées dans `t2` .

`range checked for each record (index map: #)`

MySQL n'a pas trouvé d'index satisfaisant à utiliser. Il va, à la place, pour chaque combinaison de lignes des tables précédentes, faire une vérification de quel index utiliser (si il en existe), et utiliser cet index pour continuer la recherche. Ce n'est pas très rapide, mais c'est plus rapide que de faire une recherche sans aucun index.

`Using filesort`

MySQL va avoir besoin d'un autre passage pour lire les lignes dans l'ordre. Le tri est fait en passant en revue toutes les lignes, suivant le `type de jointure` est stocker la clé de tri et le pointeur de la ligne pour chaque ligne qui satisfont la clause `WHERE` . Alors, les clés sont triées. Finalement, les lignes sont triées dans l'ordre.

`Using index`

Les informations de la colonne sont lues de la table, en utilisant uniquement les informations contenues dans l'index, sans avoir à faire d'autres lectures. Cela peut arriver lorsque toutes les colonnes utilisées dans une table font partie de l'index.

`Using temporary`

Pour résoudre la requête, MySQL va avoir besoin de créer une table temporaire pour contenir le résultat. C'est typiquement ce qui arrive si vous utilisez une clause `ORDER BY` sur une colonne différente de celles qui font partie de `GROUP BY` .

`Using where`
: Une clause `WHERE` sera utilisée pour restreindre les lignes qui seront trouvées dans la table suivante, ou envoyée au client. Si vous n'avez pas cette information, et que la table est de type `ALL` ou `index` , vous avez un problème dans votre requête (si vous ne vous attendiez pas à tester toutes les lignes de la table).

Si vous voulez rendre vos requêtes aussi rapide que possible, vous devriez examiner les lignes qui utilisent `Using filesort` et `Using temporary` .

Vous pouvez obtenir une bonne indication de la qualité de votre jointure en multipliant toutes les valeurs de la colonne `rows` dans la table de la commande `EXPLAIN` . Cela est une estimation du nombre de lignes que MySQL va examiner pour exécuter cette requête. C'est aussi ce nombre qui sera utilisé pour interrompre votre requête, grâce à la variable `max_join_size` . Choix des paramètres du serveur .L'exemple ci−dessous illustre comme une requête `JOIN` peut être optimisée avec les résultats de la commande `EXPLAIN` .

Supposons que vous avez la requête `SELECT` suivante, et que vous l'examinez avec `EXPLAIN` :

```
EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
           tt.ProjectReference, tt.EstimatedShipDate,
           tt.ActualShipDate, tt.ClientID,
           tt.ServiceCodes, tt.RepetitiveID,
           tt.CurrentProcess, tt.CurrentDPPerson,
           tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
           et_1.COUNTRY, do.CUSTNAME
       FROM tt, et, et AS et_1, do
       WHERE tt.SubmitTime IS NULL
           AND tt.ActualPC = et.EMPLOYID
           AND tt.AssignedPC = et_1.EMPLOYID
           AND tt.ClientID = do.CUSTNMBR;
```

Pour cette exemple, nous supposons que :

- Les colonnes utilisées sont déclarées comme ceci :

Table	**Colonne**	**Type de colonne**
tt	ActualPC	CHAR(10)
tt	AssignedPC	CHAR(10)
tt	ClientID	CHAR(10)
et	EMPLOYID	CHAR(15)
do	CUSTNMBR	CHAR(15)

- Les tables ont les index suivants :

Table	**Index**
tt	ActualPC
tt	AssignedPC
tt	ClientID
et	EMPLOYID (clé primaire)
do	CUSTNMBR (clé primaire)

- Les valeurs de `tt.ActualPC` ne sont pas réparties également.

Initialement, avant toute optimisation, la commande `EXPLAIN` produit les informations suivantes :

```
 table type possible_keys key  key_len ref  rows  Extra
et    ALL  PRIMARY       NULL NULL    NULL 74
do    ALL  PRIMARY       NULL NULL    NULL 2135
et_1  ALL  PRIMARY       NULL NULL    NULL 74
tt    ALL  AssignedPC,   NULL NULL    NULL 3872
           ClientID,
           ActualPC
      range checked for each record (key map: 35)
```

Comme le type `type` vaut `ALL` pour chaque table, le résultat indique que MySQL fait une analyse complète de toutes les tables. Cela va prendre un très long temps de calcul, car le nombre de lignes à examiner de cette façon est le produit du nombre de lignes de toutes les tables : dans notre cas, cela vaut `74 * 2135 * 74 * 3872 = 45,268,558,720` lignes. Si les tables étaient plus grandes, cela serait encore pire.Le premier problème que vous avons ici, est que MySQL ne peut pas (encore) utiliser d'index sur les colonnes, si elles sont déclarées différemment. Dans ce contexte, les colonnes `VARCHAR` et `CHAR` sont les mêmes, mais elles ont été déclarée avec des tailles différentes. Comme `tt.ActualPC` est déclarée comme `CHAR(10)` et que `et.EMPLOYID` est déclaré comme `CHAR(15)` , il y a un problème de taille.

Pour corriger cette disparité, utilisez la commande `ALTER TABLE` pour agrandir la colonne `ActualPC` de 10 caractères à 15 :

```
mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

Maintenant, `tt.ActualPC` et `et.EMPLOYID` sont tous les deux des colonnes de type `VARCHAR(15)` . Exécuter la commande `EXPLAIN` produit maintenant le résultat suivant :

```
 table type   possible_keys key     key_len ref         rows    Extra
tt    ALL    AssignedPC,   NULL    NULL    NULL        3872    Using
             ClientID,                                         where
             ActualPC
do    ALL    PRIMARY       NULL    NULL    NULL        2135
      range checked for each record (key map: 1)
et_1  ALL    PRIMARY       NULL    NULL    NULL        74
      range checked for each record (key map: 1)
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC 1
```

Ce n'est pas parfait, mais c'est bien mieux. Le produit de toutes les lignes a été divisé par 74). Cette version s'exécute en quelques secondes.Une autre modification peut être faîte pour éliminer les problèmes de taille de colonne pour `tt.AssignedPC = et_1.EMPLOYID` et `tt.ClientID = do.CUSTNMBR` :

```
mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
    ->                MODIFY ClientID   VARCHAR(15);
```

Maintenant, `EXPLAIN` produit le résultat suivant :

```
 table type   possible_keys key      key_len ref           rows Extra
et    ALL    PRIMARY       NULL     NULL    NULL          74
tt    ref    AssignedPC,   ActualPC 15      et.EMPLOYID   52   Using
             ClientID,                                         where
             ActualPC
et_1  eq_ref PRIMARY       PRIMARY  15      tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY  15      tt.ClientID   1
```

C'est presque aussi bon que cela pourrait l'être.

Le problème final est que, par défaut, MySQL supporte que les valeurs de la colonne `tt.ActualPC` sont uniformément répartie, et que ce n'est pas le cas pour la table `tt` . Mais il est facile de le dire à MySQL :

```
mysql> <userinput>ANALYZE TABLE tt;</userinput>
```

Maintenant, la jointure est parfaite, et la commande `EXPLAIN` produit ce résultat :

```
 table type   possible_keys key     key_len ref           rows Extra
tt    ALL    AssignedPC    NULL    NULL    NULL          3872 Using
             ClientID,                                        where
             ActualPC
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC   1
et_1  eq_ref PRIMARY       PRIMARY 15      tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY 15      tt.ClientID   1
```

Notez que la colonne `rows` dans le résultat de `EXPLAIN` est une prédiction éclairée de l'optimiseur de jointure MySQL. Pour optimiser une requête, vous devriez vérifier si ces nombres sont proches de la réalité. Si ce n'est pas le cas, vous pourriez obtenir de meilleures performances avec l'attribut `STRAIGHT_JOIN` dans votre commande `SELECT` , et en choisissant vous même l'ordre de jointure des tables dans la clause `FROM` .

7.2.2 Mesurer les performances d'une requête

Dans la plupart des cas, vous pouvez mesurer la performance d'une requête en comptant le nombre d'accès disques. Pour les tables de petite taille, vous pouvez généralement obtenir une seule lecture (car l'index est probablement en cache). Pour les tables plus grandes, vous pouvez estimer que vous aurez besoin de (en utilisant les index `B-tree`) : `log(row_count) / log(index_block_length / 3 * 2 / (index_length + data_pointer_length)) + 1` lectures pour trouver une ligne.

Pour MySQL, un bloc d'index vaut généralement 1024 octets, et le pointeur de données vaut 4 octets. Une table de 500,000 avec un index de taille 3 (entier moyen) vous donnera : `log(500,000)/log(1024/3*2/(3+4)) + 1` = 4 lectures.

Comme l'index ci−dessus vous serait de taille 500 000 * 7 * 3/2 = 5.2 Mo, (en supposant que les index des tampons sont remplit aux 2/3, ce qui est typique), vous aurez probablement l'essentiel de l'index en mémoire, et vous n'aurez alors besoin que de 1 ou 2 lectures pour lire le reste des lignes.

Pour les écritures, toutefois, vous aurez besoin de 4 lectures (comme ci−dessus), pour trouver la place du nouvel index, et normalement, deux autres lectures pour modifier l'index et la ligne.

Notez que le raisonnement ci−dessus n'indique pas que votre application va dégénérer en fonction du logarithme népérien! Tant que tout est mis en cache par l'OS ou le serveur SQL, les performances ne vont se réduire que marginalement, même si la table grossit beaucoup. Une fois que les données seront trop importantes pour être en cache, votre application va ralentir car le serveur devra faire des lectures sur le disque (ce qui va accroître le log). Pour éviter cela, augmentez le cache d'index au fur et à mesure que votre index grossit. Choix des paramètres du serveur .

7.2.3 Vitesse des requêtes SELECT

En général, lorsque vous voulez rendre un `SELECT ... WHERE` plus rapide, la première chose à faire est de voir si vous pouvez ajouter des index. Toutes les références entre les tables doivent normalement être faites avec des index. Vous pouvez utiliser la commande `EXPLAIN` pour déterminer les index utilisés pour le `SELECT` . Voyez aussi Comment MySQL utilise les index et Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT`) .

Quelques conseils généraux :

- Pour aider MySQL à mieux optimiser les requêtes, exécutez `myisamchk --analyze` sur une table après l'avoir remplie avec quelques données consistantes. Cela met à jour une valeur pour chaque partie de l'index qui indique le nombre moyen de lignes qui ont la même valeur. (Pour les index uniques, c'est toujours 1, bien sûr.) MySQL utilisera cela pour décider quel index choisir pour connecter deux tables avec une "expression non−constante". Vous pouvez vérifier le retour de l'exécution d' `analyze` en faisant `SHOW INDEX FROM nom_de_table` et examiner la colonne `Cardinality` .
- Pour trier un index et des données par rapport à un index, utilisez `myisamchk --sort-index --sort-records=1` (si vous voulez trier selon le premier index). Si vous avec un index unique à partir duquel vous voulez lire toutes les lignes en prenant comme ordre cet index, c'est un bon moyen de rendre les traitements plus rapides. Notez, toutefois, que ce tri n'est pas le plus optimal et prendra beaucoup de temps pour une grosse table !

7.2.4 Comment MySQL optimise les clauses WHERE

Les optimisation de la clause `WHERE` sont présentées avec la commande `SELECT` car elles sont généralement utilisées avec la commande `SELECT` , mais les mêmes optimisations peuvent s'appliquer aux clauses `WHERE` des commandes `DELETE` et `UPDATE` .

Notez aussi que cette section est incomplète. MySQL fait de très nombreuses optimisations, et nous n'avons pas eu le temps de toutes les documenter.

Certaines des optimisations effectuées par MYSQL sont présentées ici :

- Suppression des parenthèses inutiles :

```
    ((a AND b) AND c OR (((a AND b) AND (c AND d))))
-> (a AND b AND c) OR (a AND b AND c AND d)
```

- Remplacement des constantes :

```
    (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```

- Suppression des conditions constantes (nécessaires pour le remplacement des constantes) :

```
    (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6
```

- Les expressions constantes utilisées par les index sont évaluées une fois.
- `COUNT(*)` sur une table simple, sans clause `WHERE` est lu directement dans les informations de la table pour les tables `MyISAM` et `HEAP` . Cela peut aussi être fait avec les expressions `NOT NULL` lorsqu'elles sont utilisées sur une seule table. table.
- Détection précoce est expressions constantes invalides. MySQL détecte rapidement les commandes `SELECT` qui sont impossibles, et ne retourne aucune ligne.
- `HAVING` est combiné avec la clause `WHERE` si vous n'utilisez pas la clause `GROUP BY` ou les fonctions de groupe (`COUNT()` , `MIN()` ...).
- Pour chaque sous−jointure, une clause `WHERE` simplifiée est construite pour accélérer l'évaluation de `WHERE` pour chaque sous−jointure, et aussi essayer d'ignorer les lignes le plus tôt possible.
- Toutes les tables constantes sont lues en premier, avant toute autre table de la requête. Une table constante est une table :
    - Une table vide ou une table d'une ligne.
    - Une table qui est utilisée avec la clause `WHERE` sur un index de type `UNIQUE` , ou avec une clé primaire `PRIMARY KEY` , dont toutes les parties sont des expressions constantes, et les parties de l'index sont identifiées comme `NOT NULL` .

  Toutes les tables suivantes sont considérées comme constantes :

```
 mysql> SELECT * FROM t WHERE primary_key=1;
mysql> SELECT * FROM t1,t2
    ->          WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

- La meilleure combinaison de jointure est obtenue en testant toutes les possibilités. Si toutes les colonnes des clauses `ORDER BY` et `GROUP BY` proviennent de la même table, cette table sera utilisée de préférence comme première table dans la jointure.
- Si il y a une clause `ORDER BY` et une clause `GROUP BY` différente, ou si la clause `ORDER BY` ou `GROUP BY` contient des colonnes issues des tables autres que la première, une table temporaire est créée.
- Si vous utilisez `SQL_SMALL_RESULT` , MySQL va utiliser une table temporaire en mémoire.
- Chaque index de table est interrogé, et le meilleur index qui représente moins de 30% des lignes est utilisé. Si un tel index ne peut être identifié, un scan rapide de la table est fait.
- Dans certains cas, MySQL peut lire des lignes depuis l'index sans même consulter le fichier de données. Si toutes les colonnes de l'index sont des nombres, alors seul l'arbre d'index sera utilisé pour résoudre la requête.
- Avant chaque affichage de ligne, celles qui ne satisfont pas les critères de la clause `HAVING` sont ignorées.

Quelques exemples de requêtes très rapides :

```
 mysql> SELECT COUNT(*) FROM tbl_name;
mysql> SELECT MIN(key_part1),MAX(key_part1) FROM tbl_name;
mysql> SELECT MAX(key_part2) FROM tbl_name
    ->        WHERE key_part_1=constant;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1,key_part2,... LIMIT 10;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1 DESC,key_part2 DESC,... LIMIT 10;
```

Les requêtes suivantes ne sont résolues qu'avec l'arbre d'index (en supposant que les colonnes sont numériques) :

```
mysql> SELECT key_part1,key_part2 FROM tbl_name WHERE key_part1=val;
mysql> SELECT COUNT(*) FROM tbl_name
    ->        WHERE key_part1=val1 AND key_part2=val2;
mysql> SELECT key_part2 FROM tbl_name GROUP BY key_part1;
```

Les requêtes suivantes utilisent l'indexation pour lire les lignes dans un ordre donnés, dans faire de tri supplémentaire :

```
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1,key_part2,... ;
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1 DESC,key_part2 DESC,... ;
```

7.2.5 Comment MySQL optimise la condition OR

La méthode `Index Merge` est utilisée pour lire les lignes ayant plusieurs scans de type `ref` , `ref_or_null` et `range` , et qui combinent les résultats en un seul. Cette méthode est employée lorsque la condition de table est une disjonction de conditions, pour lesquelles `ref` , `ref_or_null` et `range` peuvent être utilisée avec différentes clés.

Ce type d'optimisation de ``jointure'' est nouveau en MySQL 5.0.0, et représente une modification significative par rapport aux index, car l' **ancienne** règle était que le serveur ne pouvait utiliser qu'un seul index pour chaque table référencée.

Dans le résultat de la commande `EXPLAIN` , cette méthode apparaît comme `index_merge` , dans la colonne `type` . Dans ce cas, la colonne `key` contient une liste des index utilisés, et `key_len` contient une liste des parties de clés les plus longues pour ces index.

Exemples :

```
SELECT * FROM tbl_name WHERE key_part1 = 10 OR key_part2 = 20;

SELECT * FROM tbl_name
   WHERE (key_part1 = 10 OR key_part2 = 20) AND non_key_part=30;

SELECT * FROM t1,t2
   WHERE (t1.key1 IN (1,2) OR t1.key2 LIKE 'value%')
   AND t2.key1=t1.some_col

SELECT * FROM t1,t2
   WHERE t1.key1=1
   AND (t2.key1=t1.some_col OR t2.key2=t1.some_col2)
```

7.2.6 Comment MySQL optimise IS NULL

MySQL peut exploiter certaines optimisation sur les conditions `column IS NULL` , comme il peut le faire avec les conditions `column = constant_value` . Par exemple, MySQL peut utiliser des index et des intervalles pour rechercher des valeurs `NULL` avec `IS NULL` .

```
SELECT * FROM table_name WHERE key_col IS NULL;

SELECT * FROM table_name WHERE key_col <=> NULL;

SELECT * FROM table_name WHERE key_col=# OR key_col=# OR key_col IS NULL
```

Si vous utilisez `column_name IS NULL` sur une colonne `NOT NULL` dans une clause `WHERE` , sur une table qui ne fait pas partie d'une jointure `OUTER JOIN` , l'expression sera optimisée immédiatement.

MySQL 4.1.1 peut aussi optimiser des combinaisons `column = expr AND column IS NULL` , une forme qui est fréquente avec les sous−requêtes. `EXPLAIN` vous indiquera `ref_or_null` lorsque cette optimisation est utilisée.

Cette optimisation peut gérer une condition `IS NULL` avec toute partie de clé.

Quelques exemples de requêtes qui sont optimisées (en supposant qu'il existe une clé sur `t2` (a,b) :

```
SELECT * FROM t1 WHERE t1.a=expr OR t1.a IS NULL;

SELECT * FROM t1,t2 WHERE t1.a=t2.a OR t2.a IS NULL;

SELECT * FROM t1,t2 WHERE (t1.a=t2.a OR t2.a IS NULL) AND t2.b=t1.b;

SELECT * FROM t1,t2 WHERE t1.a=t2.a AND (t2.b=t1.b OR t2.b IS NULL);

SELECT * FROM t1,t2 WHERE (t1.a=t2.a AND t2.a IS NULL AND ...) OR (t1.a=t2.a AND t2.a IS NULL AND ...);
```

`ref_or_null` fonctionnee en lisant la clé de référence, et après ça, fait une recherche différente pour les valeurs `NULL` .

Notez que l'optimisation ne peut gérer qu'un seul niveau de conditions `IS NULL` .

```
SELECT * FROM t1,t2 where (t1.a=t2.a AND t2.a IS NULL) OR (t1.b=t2.b AND t2.b IS NULL);
```

Dans le cas ci−dessus, MySQL va uniquement utiliser une recherche de clé pour la partie `(t1.a=t2.a AND t2.a IS NULL)` et ne sera pas capable d'utiliser la clé pour `b` .

7.2.7 Comment MySQL optimise DISTINCT

`DISTINCT` combiné avec un `ORDER BY` aura dans la plupart des cas recours à une table temporaire.

Notez que comme `DISTINCT` peut utiliser `GROUP BY` , apprenez comment MySQL fonctionne avec les champs de `ORDER BY` et `HAVING` qui ne sont pas dans la liste des colonnes sélectionnées. `GROUP BY` avec les champs cachés .

Quand vous combinerez `LIMIT #` avec `DISTINCT` , MySQL stoppera dès qu'il trouvera `#` lignes uniques.

Si vous n'utilisez pas de colonnes de toutes les tables utilisées, MySQL arrête de scanner la table non−utilisée dès qu'il trouve la première correspondance.

```
SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;
```

Dans ce cas, en supposant que `t1` est utilisée avant `t2` (vérifiez avec `EXPLAIN`), MySQL arrêtera de lire à partir de `t2` (pour cette ligne particulière de `t1`) lorsque la première ligne de `t2` est trouvée.

7.2.8 Comment MySQL optimise les clauses LEFT JOIN et RIGHT JOIN

`A LEFT JOIN B` est implémenté dans MySQL comme suit :

- La table `B` est censée être dépendante de la table `A` et de toutes les tables dont dépend `A` .
- La table `A` est censée être dépendante de toutes les tables (à part `B`) qui sont utilisées dans la condition du `LEFT JOIN` .
- Toutes les conditions du `LEFT JOIN` sont transmises à la clause `WHERE` .
- Toutes les optimisations standards de jointures sont effectuées, à l'excepté qu'une table est toujours lue après celles dont elle dépend. S'il y a une dépendance circulaire, MySQL retournera une erreur.
- Toutes les optimisations standards de `WHERE` sont effectuées.
- S'il y a une ligne dans `A` qui répond à la clause `WHERE` , mais qu'il n'y avait aucune ligne dans `B` qui répondait à la condition du `LEFT JOIN` , alors une ligne supplémentaire de `B` est générée avec toutes les colonnes mises à `NULL` .

- Si vous utilisez `LEFT JOIN` pour trouver les enregistrements qui n'existent pas dans d'autres tables et que vous effectuez le test suivant : `nom_colonne IS NULL` dans la partie `WHERE` , où `nom_colonne` est une colonne qui est déclarée en tant que `NOT NULL` , alors MySQL arrêtera de chercher d'autres lignes (pour une combinaison de clefs particulière) après avoir trouvé une ligne qui répond à la condition du `LEFT JOIN` .

`RIGHT JOIN` est implémenté de manière analogue à `LEFT JOIN` .

L'ordre de lecture de tables forcé par `LEFT JOIN` et `STRAIGHT JOIN` aidera l'optimiseur de jointures (qui calcule l'ordre dans lequel les tables doivent être jointes) à faire son travail plus rapidement, puisqu'il y aura moins de permutations de tables à vérifier.

Notez que ce qui précède signifie que si vous faites une requête de la sorte :

```
SELECT *
   FROM a,b LEFT JOIN c ON (c.key=a.key) LEFT JOIN d ON (d.key=a.key)
   WHERE b.key=d.key;
```

Un palliatif est de changer la requête en :

```
SELECT *
   FROM b,a LEFT JOIN c ON (c.key=a.key) LEFT JOIN d ON (d.key=a.key)
   WHERE b.key=d.key;
```

Depuis la version 4.0.14, MySQL effectue l'optimisation `LEFT JOIN` suivante : si la condition `WHERE` est toujours fausse pour la ligne `NULL` générée, la jointure `LEFT JOIN` est transformée en jointure normale.Par exemple, dans la requête suivante, la clause `WHERE` sera fausse si `t2.column` est `NULL` : il est donc valide de convertir la jointure en une jointure normale.

```
SELECT * FROM t1 LEFT JOIN t2 ON (column1) WHERE t2.column2=5;
```

Par conséquent, il est possible de convertir la requête en jointure normale :

```
SELECT * FROM t1,t2 WHERE t2.column2=5 AND t1.column1=t2.column1;
```

Cela peut se faire plus rapidement, car MySQL peut maintenant utiliser la table `t2` avant la table `t1` si les relations sont plus favorables. Pour forcer l'utilisation spécifique d'un ordre de table, utilisez `STRAIGHT JOIN` .

7.2.9 Comment MySQL optimise ORDER BY

Dans certain cas, MySQL peut utiliser un index pour répondre à une requête `ORDER BY` ou `GROUP BY` sans faire aucun tri.

L'index peut être utilisé même si le `ORDER BY` ne correspond pas exactement à l'index, tant que toutes les parties inutilisée de l'index et les colonnes du `ORDER BY` sont constantes dans la clause `WHERE` . Les requêtes suivantes utilisent l'index pour répondre aux parties `ORDER BY` / `GROUP BY` :

```
 SELECT * FROM t1 ORDER BY partie_clef1,partie_clef2,...
SELECT * FROM t1 WHERE partie_clef1=constante ORDER BY partie_clef2
SELECT * FROM t1 WHERE partie_clef1=constante GROUP BY partie_clef2
SELECT * FROM t1 ORDER BY partie_clef1 DESC,partie_clef2 DESC
SELECT * FROM t1 WHERE partie_clef1=1 ORDER BY partie_clef1 DESC,partie_clef2 DESC
```

Quelques cas où MySQL ne peut **pas** utiliser les index pour répondre à `ORDER BY` : (Notez que MySQL utilisera quand même les indexes pour trouver les lignes qui correspondent à la clause `WHERE`) :

- Vous effectuez un `ORDER BY` sur des clefs différentes :
  ```
  SELECT * FROM t1 ORDER BY key1, key2;
  ```
- Vous effectuez un `ORDER BY` en utilisant des parties de clef non consécutives.
  ```
  SELECT * FROM t1 WHERE key2=constant ORDER BY key_part2;
  ```
- Vous mélangez `ASC` et `DESC` .
  ```
  SELECT * FROM t1 ORDER BY key_part1 DESC, key_part2 ASC;
  ```

* La clef utilisée pour extraire les résultats n'est pas la même que celle utilisée lors du groupement `ORDER BY` :

```
SELECT * FROM t1 WHERE key2=constant ORDER BY key1;
```

* Vous faites une jointure entre plusieurs tables et les colonnes sur lesquelles vous faites un `ORDER BY` ne font pas toutes parties de la première table non− `const` qui est utilisée pour récupérer les lignes (C'est la première table dans l'affichage d' `EXPLAIN` qui n'utilise pas une méthode de récupération sur une ligne constante).
* Vous avez plusieurs expressions `ORDER BY` et `GROUP BY` .
* L'index de table utilisé est un type d'index qui n'enregistre pas les lignes dans l'ordre. (comme le type d'index `HASH` dans les tables `HEAP`).

Dans les cas où MySQL doit trier les résultats, il utilisera l'algorithme suivant :

* Lit toutes les lignes en fonction d'un index ou par scan de la table. Les lignes qui ne vérifient pas la condition `WHERE` sont ignorées.
* Stocke les valeurs des clés de tri dans un buffer. La taille du buffer est la valeur de la variable système `sort_buffer_size` .
* Lorsque le buffer se remplit, fait un tri rapide et stocke le résultat dans une fichier temporaire. Sauve le pointeur dans un bloc trié. Si toutes les lignes tiennent dans le buffer de tri, aucun fichier temporaire n'est créé.
* Répète les étapes précédentes jusqu'à ce que toutes les lignes aient été lues.
* Fait une combinaison multiple jusqu'à `MERGEBUFF` (7) régions en un bloc, dans un autre fichier temporaire. Répète l'opération jusqu'à ce que le premier fichier soit dans le second.
* Répète la suite jusqu'à ce qu'il y ait moins de `MERGEBUFF2` (15) bloc libres.
* Dans la dernière combinaison multiple, seul le pointeur de ligne (la dernière partie de la clé de tri), est écrite dans le fichier de résultat.
* Lit les lignes triées en utilisant les pointeurs de lignes du fichier de résultat. Pour optimiser cela, on lit un gros bloc de pointeur, on les trie, et on les utilise pour lire les lignes en ordre dans un buffer. La taille du buffer est la valeur de la variable système `read_rnd_buffer_size` . Le code de cette étape est dans le fichier source `sql/records.cc` .

Vous pouvez vérifier avec `EXPLAIN SELECT ... ORDER BY` si MySQL peut utiliser des index pour répondre à cette requête. Si vous obtenez un `Using filesort` dans la colonne `extra` , c'est que MySQL ne peut utiliser d'index pour résoudre cet `ORDER BY` . Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT`) .

Si vous voulez plus de rapidité avec les `ORDER BY` , vous devez d'abord voir si vous pouvez faire en sorte que MySQL utilises des index au lieu de passer par des phases de tri en plus. Si cela se révèle impossible, vous pouvez :

* Augmenter la taille de la variable `sort_buffer` .
* Augmenter la taille de la variable `record_rnd_buffer` .
* Changer `tmpdir` pour qu'il pointe vers un disque dédié avec beaucoup d'espace libre. Si vous utilisez MySQL version 4.1 ou plus récent, vous pouvez répartir la charge entre plusieurs disques physiques en donnant à l'option `tmpdir` une liste de chemin, séparés par des deux−points (`':'`) ou des points−virgules `';'` sous Windows). Ils seront utilisés circulairement. **Note :** ces chemins doivent aboutir à différents disques **physiques** , et non pas différentes partitions du même disque.

Par défaut, MySQL trie les requêtes `GROUP BY x,y[,...]` comme si vous aviez spécifié l'ordre `ORDER BY x,y[,...]` . Si vous ajoutez une clause `ORDER BY` explicite, MySQL l'optimise aussi sans perte de vitesse, même si un tri a lieu. Si la requête inclut une clause `GROUP BY` mais que vous voulez éviter le surcoût du tri, vous pouvez supprimer le tri en spécifiant `ORDER BY NULL` :

```
INSERT INTO foo SELECT a,COUNT(*) FROM bar GROUP BY a ORDER BY NULL;
```

7.2.10 Comment MySQL optimise LIMIT

Dans certains cas, MySQL va gérer la requête différemment avec la clause `LIMIT #` , si la clause `HAVING` n'est pas utilisée :

- Si vous ne sélectionnez que quelques lignes avec `LIMIT` , MySQL va utiliser les index dans certains cas, où il aurait préféré utiliser un scan de table complet.
- Si vous utilisez `LIMIT #` avec la clause `ORDER BY` , MySQL va arrêter de trier dès qu'il a trouvé la première `#` au lieu de trier toute la table.
- Lorsque vous combinez `LIMIT #` avec `DISTINCT` , MySQL va s'arrêter dès qu'il a trouvé `#` lignes distinctes.
- Dans certains cas, la clause `GROUP BY` peut être appliquée en lisant les clés dans l'ordre (ou en faisant un tri sur la clé), puis en calculant un sommaire, jusqu'à ce que la clé soient modifiée. Dans ce cas, `LIMIT #` ne va pas appliquer les éléments non nécessaires de la clause `GROUP BY` s.
- Aussitôt que MySQL a envoyé les premières `#` lignes au client, il annule le reste de la requête (si vous n'utilisez pas la fonction `SQL_CALC_FOUND_ROWS`).
- `LIMIT 0` va toujours retourner rapidement un résultat vide. C'est pratique pour vérifier une requête et lire les types de colonnes du résultat, sans exécuter réellement la requête.
- Lorsque le serveur utilise des tables temporaire pour résoudre les requêtes, la clause `LIMIT #` est utilisée pour calculer l'espace nécessaire.

7.2.11 Comment éviter les scans de tables,,,

`EXPLAIN` affiche la vlaeur `ALL` dans la colonne `type` lorsque MySQL utilise n scan de table pour résoudre une requête. Cela arrive lorsque :

- La table est si petite qu'il est plus rapide de scanner la table que d'utiliser les index. C'est un cas courant pour les tables de moins de 10 lignes, et de taille de ligne faible.
- Il n'y a pas de restriction exploitable sur les conditions `ON` et `WHERE` , avec les colonnes indexées.
- Vous comparez des colonnes indexées avec des constantes, et MySQL a calculé, en se basant sur l'arbre d'index, que les constantes couvrent une trop grande partie de la table : un scan devrait être plus rapide. Comment MySQL optimise les clauses `WHERE` .
- Vous utilisez une clé avec une cardinalité faible (c'est à dire, beaucoup de lignes sont trouvées). MySQL va alors supposer que l'utilisation de l'index va lui imposer beaucoup de recherches, et qu'un scan de table sera plus rapide.

Ce que vous pouvez faire pour éviter les scans de grosses tables : s

- Utilisez `ANALYZE TABLE` sur les tables scannées pour optimiser la distribution des clés. Syntaxe de `ANALYZE TABLE` .
- Utilisez `FORCE INDEX` sur les tables scannnées, pour dire à MYSQL que les scans de tables sont trop coûteux, comparé à l'utilisation de l'index. Syntaxe de `SELECT` .
  ```
  SELECT * FROM t1,t2 force index(index_for_column) WHERE t1.column=t2.column;
  ```
- Lancez `mysqld` avec `--max-seeks-for-key=1000` ou faites `SET MAX_SEEKS_FOR_KEY=1000` pour dire à l'optimiseur que les scans sans index ne généreront pas plus de 1000 recherches dans les index. Variables sytème du serveur .

7.2.12 Vitesse des requêtes INSERT

Le temps d'insertion d'une ligne est constitué comme ceci :

* Connexion : (3)
* Envoi au serveur : (2)
* Analyse de la requête : (2)
* Insertion de la ligne : (1 x taille de la ligne)
* Insertion des index : (1 x nombre d'index)
* Fermeture : (1)

où les nombres représentent une partie proportionnelle du temps total. Le calcul ne prend pas en compte les coûts d'administration initiaux de l'ouverture des tables (qui est fait une fois pour chaque requête simultanée).

La taille de la table ralentit les opérations d'insertion des index par un facteur de log N (`B-tree` s).

Quelques méthodes pour accélérer les insertions :

* Si vous insérez plusieurs lignes depuis le même client, en même temps, utilisez les valeurs multiples de la commande `INSERT` . C'est bien plus rapide (et parfois beaucoup plus rapide) que d'utiliser des commandes `INSERT` distinctes. Si vous ajoutez des données dans une table non vide, vous pouvez ajuster la variable `bulk_insert_buffer_size` pour l'accélérer encore plus. Syntaxe de `SHOW VARIABLES` .
* Si vous insérez de nombreuses lignes depuis différents clients, vous pouvez accélérer les insertions en utilisant la commande `INSERT DELAYED` . Syntaxe de `INSERT` .
* Avec les tables `MyISAM` , vous pouvez insérer des lignes en même temps que vous utilisez des commandes `SELECT` , du moment qu'il n'y a pas d'effacement de ligne dans la table.
* Lorsque vous chargez une table depuis un fichier texte, utilisez la commande `LOAD DATA INFILE` . Elle est généralement 20 fois plus rapide que l'équivalent en commandes `INSERT` . Syntaxe de `LOAD DATA INFILE` .
* Il est possible, avec un peu de travail supplémentaire, d'accélérer encore la vitesse des commandes `LOAD DATA INFILE` . Utilisez la procédure standard :
    * Créez optionnellement une table avec `CREATE TABLE` . Par exemple, en utilisant `mysql` ou Perl `DBI` .
    * Exécutez une commande `FLUSH TABLES` ou la commande en ligne shell `mysqladmin flush-tables` .
    * Utilisez `myisamchk --keys-used=0 -rq /path/to/db/tbl_name` . Cela va supprimer l'utilisation des index dans la table.
    * Insérez vos données dans la table, avec `LOAD DATA INFILE` . Les index ne seront pas modifiés, et donc, très rapides.
    * Si vous allez uniquement lire la table dans le futur, utilisez `myisampack` pour la réduire de taille. Caractéristiques des tables compressées .
    * Re−créez les index avec `myisamchk -r -q /path/to/db/tbl_name` . Cette commande va créer l'arbre d'index en mémoire, avant de l'écrire sur le disque, ce qui est bien plus rapide, car il n'y a que peu d'accès disques. L'arbre final sera aussi parfaitement équilibrés.
    * Exécutez une commande `FLUSH TABLES` ou utilisez la commande en ligne shell `mysqladmin flush-tables` .

  Notez que la ocmmande `LOAD DATA INFILE` fait aussi les optimisations ci−dessus, si vous faites les insertions dans une table vide. La différence principale avec la procédure ci−dessus est que vous pouvez laisser `myisamchk` allouer plus de mémoire temporaire pour la création d'index, que vous ne pourriez le faire pour chaque recréation.

  Depuis MySQL 4.0 vous pouvez aussi utiliser `ALTER TABLE tbl_name DISABLE KEYS` au lieu de `myisamchk --keys-used=0 -rq /path/to/db/tbl_name` et `ALTER TABLE tbl_name ENABLE KEYS` au lieu de `myisamchk -r -q /path/to/db/tbl_name` . De cette façon, vous pouvez aussi éviter l'étape `FLUSH TABLES` .
* Vous pouvez accélérer les insertions qui sont faites avec plusieurs requêtes en verrouillant vos tables :

```
 mysql> LOCK TABLES a WRITE;
mysql> INSERT INTO a VALUES (1,23),(2,34),(4,33);
mysql> INSERT INTO a VALUES (8,26),(6,29);
```

```
mysql> UNLOCK TABLES;
```

La principale différence de vitesse est que l'index de buffer est écrit sur le disque une fois, après toutes les insertions `INSERT` terminées. Normalement, il aurait du y avoir de nombreuses écritures, une pour chaque commande `INSERT` . Le verrouillage n'est pas nécessaire si vous pouvez insérer toutes les lignes d'une seule commande.

Pour les tables transactionnelles, vous devriez utiliser `BEGIN/COMMIT` au lieu de `LOCK TABLES` pour accélérer les opérations.

Le verrouillage va aussi réduire le nombre total de tests de connexions, mais le temps d'attente maximum de certains threads va augmenter (car il va y avoir la queue pour les verrous). Par exemple :

```
 thread 1 fait 1000 insertions
thread 2, 3, et 4 font 1 insertion
thread 5 fait 1000 insertions
```

Si vous ne voulez pas utiliser le verrouillage, les threads 2, 3 et 4 auront fini avant les 1 et 5. Si vous utilisez le verrouillage, 2, 3 et 4 me finiront probablement pas avant 1 ou 5, mais la durée globale de l'opération sera 40% plus courte.Comme les commandes `INSERT` , `UPDATE` et `DELETE` sont très rapides avec MySQL, vous obtiendrez de meilleures performances générales en ajoutant des verrous autour de toutes vos opérations de 5 insertions o modifications simultanées. Si vous faîtes de très nombreux insertions dans une ligne, vous pouvez utiliser `LOCK TABLES` suivi de `UNLOCK TABLES` une fois de temps en temps (par exemple, toutes les 1000) pour permettre aux autres threads d'accéder à la table. Cela vous donnera quand même une bonne accélération.

Bien sur, `LOAD DATA INFILE` reste bien plus rapide pour charger les données.

- Pour accélérer `LOAD DATA INFILE` et `INSERT` , agrandissez le buffer de clé. Choix des paramètres du serveur .

7.2.13 Vitesses des commandes UPDATE

Les requêtes de modification sont optimisées comme les requêtes de `SELECT` avec le coût supplémentaire de l'écriture. La vitesse d'écriture dépend de la taille des données qui sont modifiées, et du nombre d'index que cela va impacter. Les index ne sont pas modifiés tant que la ligne n'est pas écrite. Les index qui ne sont pas modifiés ne seront pas réécrit.

De plus, une autre méthode pour obtenir des accélérations avec les modifications est de retarder les modifications, et d'en faire plusieurs d'un coup. Faire plusieurs modifications d'un coup est bien plus rapide que d'en faire une à chaque fois.

Notez que, avec le format de ligne dynamique, la modification d'une ligne peut déboucher sur la fragmentation de la ligne. Si vous le faite souvent, il est très important d'appliquer `OPTIMIZE TABLE` sur ces tables, pour les optimiser. Syntaxe de `OPTIMIZE TABLE` .

7.2.14 Rapidité des requêtes DELETE

Si vous voulez effacer toutes les lignes d'une table, vous devez utiliser `TRUNCATE TABLE nom_de_table` . Syntaxe de `TRUNCATE` .

Le temps de suppression d'une ligne est exactement proportionnel au nombre d'index. Pour effacer les enregistrements plus rapidement, vous pouvez augmenter la taille du cache d'index. Choix des paramètres du serveur .

7.2.15 Autres conseils d'optimisation

Quelques conseils en vrac pour accélérer le serveur :

- Utilisez les connexions persistantes à la base, pour éviter les coûts récurrents de connexion. Si vous ne pouvez pas utiliser de connexions persistantes, et que vous faites de nombreuses connexions à la base, essayez de modifier la valeur de la variable `thread_cache_size` . Choix des paramètres du serveur .
- Vérifiez toujours que vos requêtes utilisent vraiment les index que vous avez créé dans les tables. Avec MySQL, vous pouvez utiliser la commande `EXPLAIN` . Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT`) .
- Essayez d'éviter les requêtes `SELECT` complexes sur les tables `MyISAM` qui sont souvent modifiées. Cela évitera des problèmes de verrouillage.
- Les nouvelles tables `MyISAM` peuvent insérer des lignes sans en effacer d'autre, tout en lisant dans cette table. Si c'est important pour vous, vous pouvez considérer d'autres méthodes où vous n'avez pas à effacer de lignes, ou bien utilisez `OPTIMIZE TABLE` après avoir effacé beaucoup de lignes.
- Utilisez `ALTER TABLE ... ORDER BY expr1,expr2...` si vous lisez les colonnes dans l'ordre `expr1,expr2...` . Avec cette option, après de grosses modifications dans la table, vous pourriez obtenir de meilleures performances.
- Dans certains cas, cela vaut la peine d'ajouter une colonne qui est une combinaison (" `hashed` ") des informations des autres colonnes. Si cette colonne est courte, et plutôt exemptes de doublons, elle peut se révéler plus rapide qu'un gros index sur plusieurs colonnes. Avec MySQL, il est très facile d'utiliser une telle colonne :

```
SELECT * FROM tbl_name
   WHERE hash_col=MD5(CONCAT(col1,col2))
   AND col1='constant' AND col2='constant';
```

- Pour les tables qui sont souvent modifiées, vous devriez essayer d'éviter les colonnes `VARCHAR` et `BLOB` . Vous obtiendrez des lignes à format dynamique si vous utilisez ne serait−ce qu'une seule colonne `VARCHAR` ou `BLOB` . Types de tables MySQL .
- Normalement, cela ne sert à rien de séparer une table en différentes tables plus petites, juste parce que vos lignes deviennent grosses. Pour accéder à une ligne, le plus long est le temps d'accès au premier octets de la ligne. Après cela, les disques modernes vont lire très rapidement la ligne, et suffisamment pour la plus par des applications. Le seul cas où cela peut être important est si vous êtes capables de dégager une table à format de ligne fixe (voir ci−dessus), ou si vous avez besoin de scanner régulièrement la table, mais que vous n'avez pas besoin de toutes les colonnes. Types de tables MySQL .
- Si vous avez besoin de calculer souvent des expressions en fonction des informations placées dans de nombreuses lignes (comme compter des lignes), il est probablement plus efficace d'introduire une nouvelle table qui va mettre à jour ce compteur en temps réel. Une modification du type présenté ci−dessous est très rapide!

```
UPDATE tbl_name SET count_col=count_col+1 WHERE key_col=constant;
```

  C'est très important lorsque vous utilisez les types de tables `MyISAM` et `ISAM` , qui ne dispose que d'un verrouillage de table (plusieurs lecteurs, un seul qui écrit). Cela va aussi améliorer les performances avec la plus par des bases, car le gestionnaire de verrouillage de ligne aura moins de tâches à faire.
- Si vous devez rassembler des statistiques issues de grosses tables de log, utiliser les tables de sommaires plutôt que la table complète. Entretenir un sommaire est bien plus rapide que de regénérer des tables à partir des logs à chaque modification (suivant l'importance de vos informations), plutôt que de modifier l'application qui fonctionne.
- Si possible, essayez de marquer les rapports comme "direct" ou "statistique", où les données nécessaires pour les rapports statistiques ne sont générées qu'à partir de tables de sommaires, calculées depuis les données réelles.
- Utilisez les valeurs par défaut des colonnes. N'insérez des valeurs explicitement que lorsque la valeur diffère de la valeur par défaut. Cela réduit le temps d'analyse de MySQL, et

améliore les insertions.

- Dans certains cas, il est pratique de compacter et stocker les données dans un `BLOB` . Dans ce cas, vous devez ajouter du code supplémentaire pour compacter et décompacter les données dans le `BLOB` , mais cela pourra vous faire économiser de nombreux accès. C'est pratique lorsque vous avez des données qui ne peuvent s'adapter facilement à une structure de base de données.
- Normalement, vous devriez essayer de garder vos données non redondantes (ce qui s'appelle la troisième forme normale dans les théories de bases de données), mais ne vous empêchez pas de duplique des données ou de créer des tables de sommaire, pour gagner de la vitesse.
- Les procédures stockées ou `UDF` (fonctions utilisateur) peuvent être une bonne façon de gagner en performance. Dans ce cas, vous devriez avoir une méthode pour appliquer les mêmes fonctions d'une autre manière, si votre base ne supporte les procédures stockées.
- Vous pouvez aussi gagner de la vitesse en utilisant des caches de requêtes dans vos applications, et en essayant de rassembler les nombreuses insertions ou modifications. Si votre base de données supporte le verrouillage de table (comme MySQL et Oracle), cela vous aidera à vous assurer que le cache d'index est vidé après chaque modifications.
- Utilisez `INSERT /*! DELAYED */` lorsque vous n'avez pas besoin d'être assuré que vos données sont écrites. Cela accélère les insertions, car de nombreuses lignes seront écrites en une seule fois.
- Utilisez `INSERT /*! LOW_PRIORITY */` lorsque vous voulez que vos sélections soient prioritaires.
- Utilisez `SELECT /*! HIGH_PRIORITY */` pour rendre les sélections prioritaires. C'est à dire, les sélections seront désormais faites même si un autre programme attend pour écrire.
- Utilise la commande `INSERT` multiple pour insérer plusieurs lignes en une seule commande SQL (plusieurs serveurs SQL le supporte).
- Utilisez `LOAD DATA INFILE` pour charger de grande quantité de données dans une table. C'est généralement plus rapide que des insertions, et sera même encore plus rapide une fois que `myisamchk` sera intégré dans `mysqld` .
- Utilisez les colonnes `AUTO_INCREMENT` pour avoir des valeurs uniques.
- Utilisez `OPTIMIZE TABLE` une fois de temps en temps, pour éviter la fragmentation lors de l'utilisation de tables avec un format de ligne dynamique. Syntaxe de `OPTIMIZE TABLE` .
- Utilisez la tables de type `HEAP` pour accélérer les traitements au maximum. Types de tables MySQL .
- Avec un serveur web normal, les images doivent être stockées dans des fichiers. C'est à dire, ne stockez qu'une référence au fichier d'image dans la base. La raison principale à cela est qu'un serveur web est bien meilleur pour mettre en cache des fichiers que le contenu d'une base de données. Il est donc plus rapide si vous utilisez des fichiers.
- Utilisez des tables en mémoire pour les données non critiques, qui ont besoin d'être lues souvent (comme des informations sur la dernière bannière affichée pour les utilisateurs sans cookies).
- Les colonnes contenant des informations identiques dans différentes tables doivent être déclarées identiquement lors de la création des tables, et porter des noms identiques. Avant la version 3.23, vous pouviez ralentir les jointures.

  Essayez de garder des noms simples (utilisez `nom` au lieu de `nom_du_client` dans la table de clients). Pour rendre vos noms de colonnes portables vers les autres serveurs SQL, vous devriez essayer de les garder plus petits que 18 caractères.
- Si vous avez vraiment besoin de très haute vitesse, vous devriez considérer les interfaces de bas niveau pour le stockage des données que les différents serveurs SQL supportent. Par exemple, en accédant directement aux tables MySQL `MyISAM` , vous pourriez obtenir un gain de vitesse de l'ordre de 2 à 5 fois, en comparaison avec l'interface SQL. Pour cela, les données doivent être sur le même serveur que l'application, et généralement, elles ne doivent être manipulées que par un seul programme à la fois (car le verrouillage externe de fichiers est très lent). Vous pouvez éliminer ces problèmes en créant des commandes `MyISAM`

de bas niveau dans le serveur MySQL (cela peut se faire facilement pour améliorer les performances). Soyez très prudent dans la conception de votre interface, mais il est très facile de supporter ce type d'optimisation.
- Dans de nombreux cas, il est plus rapide d'accéder aux données depuis une base (en utilisant une connexion ouverte) que d'accéder à un fichier texte, car la base de données est plus compacte que le fichier texte (si vous utilisez des données numériques), et cela entraîne moins d'accès disques. Vous allez aussi économiser du code, car vous n'aurez pas à analyser le fichier texte pour repérer les limites de lignes.
- Vous pouvez aussi utiliser la réplication pour accélérer le serveur. Réplication avec MySQL .
- Déclarer une table avec `DELAY_KEY_WRITE=1` va accélérer la mise à jour des index, car ils ne seront pas écrit sur le disque jusqu'à ce que le fichier de données soit refermé. L'inconvénient est que vous devez exécuter l'utilitaire `myisamchk` sur ces tables avant de lancer `mysqld` pour vous assurer que les index sont bien à jour, au cas où le processus aurait été interrompu avant d'enregistrer les données. Comme les informations d'index peuvent toujours être regénérées, vous ne perdrez pas de données avec `DELAY_KEY_WRITE` .

7.3 Verrouillage de tables

7.3.1 Méthodes de verrouillage

Actuellement, MySQL ne supporte que le verrouillage de table pour les tables `ISAM` / `MyISAM` et `MEMORY` (`HEAP`), le verrouillage de page pour les tables `BDB` et le verrouillage de ligne pour `InnoDB` .

Dans de nombreux cas, vous pouvez faire prévoir le type de verrouillage qui sera le plus efficace pour une application, mais il est très difficile de savoir si un type de verrou est meilleur que l'autre. Tout dépend de l'application, et des différentes composants qui utilisent les verrous.

Pour décider si vous voulez utiliser un type de table avec verrouillage de ligne, vous devez commencer par étudier ce que votre application fait, et quel est le schéma d'utilisation des sélections et modifications. Par exemple, la plupart des applications Web font de nombreuses sélections, peu d'effacements, des modifications basées sur des clés, et des insertions dans des tables spécifiques. Le moteur de base MySQL `MyISAM` est très bien optimisé pour cette application.

Toutes les méthodes de verrouillage de MySQL sont exemptes de blocage, sauf pour les tables `InnoDB` et `BDB` . Ceci fonctionne en demandant tous les verrous d'un seul coup, au début de la requête, et en verrouillant les tables toujours dans le même ordre.

Les tables `InnoDB` obtiennent automatiquement leur verrou de ligne et les tables `BDB` leur verrou de page, durant le traitement de la requête SQL, et non pas au démarrage de la transaction.

La méthode de verrouillage des tables de MySQL en écriture (`WRITE`) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrous d'écriture.

La méthode de verrouillage des tables de MySQL en lecture (`READ`) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrou de lecture.

Lorsqu'un verrou est libéré, le verrou est donné aux threads de la queue de verrou en écriture, puis à ceux de la queue de verrou en lecture.

Cela signifie que si vous avez de nombreuses modifications dans une table, la commande SELECT va attendre qu'il n'y ait plus d'écriture avant de lire.

Depuis MySQL 3.23.33, vous pouvez analyser le comportement des verrous sur une table avec les variables de statut Table_locks_waited et Table_locks_immediate :

```
mysql> SHOW STATUS LIKE 'Table%';
+-----------------------+---------+
| Variable_name         | Value   |
+-----------------------+---------+
| Table_locks_immediate | 1151552 |
| Table_locks_waited    | 15324   |
+-----------------------+---------+
```

Depuis MySQL 3.23.7 (3.23.25 pour Windows), vous pouvez librement mélanger des commandes INSERT et SELECT sur une table MyISAM sans verrous, si les commandes INSERT sont sans conflit. C'est à dire, vous pouvez insérer des lignes dans une table MyISAM en même temps que d'autres clients lisent la même table. Aucun conflit ne survient si la table ne contient aucun bloc libre dans les données, et que les lignes sont insérées à la fin de la table. Les trous sont des lignes qui ont été effacées. S'il y a des trouvs, les insertions concurrentes sont réactivées automatiquement, lorsque les trous sont bouchés par de nouvelles données.

Pour contourner ce problème dans les cas où vous voulez faire de nombreuses INSERT et SELECT sur la même table, vous pouvez insérer les lignes dans une table temporaire, et ne modifier la table réelle que de temps en temps, à partir de la table temporaire.Ceci peut être fait comme ceci :

```
mysql> LOCK TABLES real_table WRITE, insert_table WRITE;
mysql> INSERT INTO real_table SELECT * FROM insert_table;
mysql> TRUNCATE TABLE insert_table;
mysql> UNLOCK TABLES;
```

InnoDB utilise un verrouillage de ligne, et BDB utilise un verrouillage de page. Pour les moteurs InnoDB et BDB storage, un blocage de verrou est possible. Cela est dû au fait que InnoDB obtient automatiquement un verrou de ligne, et BDB pose le verrou de page durant le traitement SQL, et non pas au démarrage de la transaction.Avantages du verrouillage de ligne :

- Moins de conflits de lignes, lorsque les mêmes lignes sont utilisées par différents threads.
- Moins de modifications pour les annulations (ROLLBACK)
- Rend possible le verrouillage d'une ligne pour une longue durée.

Inconvénients du verrouillage de ligne :

- Prend plus de mémoire que les verrous de page ou de table.
- Est plus lent que les verrous de page ou de table, lorsqu'il est utilisé sur une grand partie de la table, car il faut alors poser plusieurs verrous.
- Est vraiment bien pire que les autres verrous si vous utilisez souvent la requête GROUP BY sur la majeure partie des données, ou si vous avez à scanner toute la table.
- Avec des verrous de plus haut niveau, vous pouvez aussi supporter des verrous d'autres types, pour optimiser l'application, car le coût de l'administration est moindre que pour le verrouillage de ligne.

Les verrous de tables sont supérieurs aux verrous de page ou de ligne dans les cas suivants :

- Les lectures.
- Les lectures et les modifications sur des clés strictes : c'est le cas si une modification ou un effacement de ligne peut être lu en une seule opération dans l'index.
  ```
  UPDATE table_name SET column=value WHERE unique_key#
  DELETE FROM table_name WHERE unique_key=#
  ```
- SELECT combiné avec INSERT (et quelques UPDATE et DELETE rares).
- De nombreux scans / GROUP BY sur toute la table, sans aucune écriture.

Autres possibiltés alternatives au verrouillage de ligne ou de page :

Le versionnage (comme celui que nous utilisons pour les insertions simultanées avec MySQL), où vous pouvez avoir un thread qui écrit et de nombreux autres qui lisent. Cela signifie que les bases ou tables supportent différentes vues des données, suivants le moment d'accès aux données. D'autres noms pour cette techniques sont `time travel` , `copy on write` ou `copy on demand` .

La `copy on demand` (copie sur demande) est dans de nombreuses situations bien meilleure que le verrouillage de page ou de ligne. Le pire reste l'utilisation de mémoire, qui est bien plus forte qu'avec les verrous normaux.

Au lieu d'utiliser le verrouillage de ligne, vous pouvez utiliser des verrous au niveau de l'application (comme les get_lock/release_lock de MySQL). Cela ne fonctionne qu'avec les applications bien élevées.

7.3.2 Problème de verrouillage de tables

MySQL utilise le verrouillage de table (au lieu du verrouillage de ligne ou de colonne) sur tous les types de tables, sauf `InnoDB` et `BDB` , pour obtenir un système de verrou à très haute vitesse.

Pour les tables `InnoDB` et `BDB` , MySQL n'utilise le verrouillage de table que vous le demandez explicitement avec `LOCK TABLES` . Pour ces tables, nous vous recommandons de ne jamais utiliser la commande `LOCK TABLES` , car `InnoDB` utilise un verrouillage de ligne automatique, et `BDB` utilise un verrouillage de pages, pour assurer l'isolation des transactions.

Pour les grandes tables, le verrouillage de table est meilleur que le verrouillage de lignes, pour la plupart des applications, mais il recèle quelque pièges.

Le verrouillage de tables permet à de nombreux threads de lire dans la même table, mais si un thread désire écrire dans la table, il doit obtenir un verrou en écriture pour avoir un accès exclusif. Durant la modification, les autres threads qui voudront lire dans cette table, devront attendre.

Comme les modifications de tables sont considérées comme plus importantes que les lectures avec `SELECT` , toutes les commandes qui modifient la table ont priorités sur les lectures. Cela devrait vous assurer que les modifications ne sont pas retenues trop longtemps, à cause de nombreuses lectures sur une même table. Vous pouvez toutefois modifier cela avec l'option `LOW_PRIORITY` des commandes de modification, et l'option `HIGH_PRIORITY` de `SELECT`).

Depuis MySQL version 3.23.7, vous pouvez utiliser la variable `max_write_lock_count` pour forcer MySQL à laisser temporairement la place à toutes les commandes `SELECT` , après un certain nombre de modifications dans la table.

Le verrouillage de table est une mauvaise technique dans les situations suivantes :

- Un client exécute une commande `SELECT` qui prend très longtemps.
- Un autre client exécute une commande `UPDATE` sur la table. Ce client va devoir attendre que la commande `SELECT` soit finie.
- Un autre client exécute une autre commande `SELECT` sur la même table. Comme `UPDATE` a la priorité sur `SELECT` , cette commande `SELECT` va attendre que `UPDATE` soit finit. Il va donc attendre que le premier `SELECT` soit fini.

Des solutions aux problèmes sont :

- Essayez d'accélérer au maximum les commandes `SELECT` . Vous pourriez passer par une table de sommaire pour cela.
- Démarrez `mysqld` avec l'option `--low-priority-updates` . Cela va donner aux commandes de modification une priorité plus faible que `SELECT` . Dans ce cas, c'est la commande `SELECT` du précédent scénario qui s'exécutera avant la commande `INSERT` .
- Vous pouvez donner à une commande spécifique `INSERT` , `UPDATE` ou `DELETE` , une priorité plus basse avec l'attribut `LOW_PRIORITY` .
- Démarrez `mysqld` avec une valeur faible pour `max_write_lock_count` afin de donner plus souvent la chance aux verrous `READ` la possibilité de lire des données, entre deux verrous `WRITE` .
- Vous pouvez spécifier que toutes les modifications d'un thread spécifique doivent être faites avec un priorité basse, en utilisant la commande SQL : `SET LOW_PRIORITY_UPDATES=1` . Syntaxe de `SET` .
- Vous pouvez spécifier qu'une requête particulière `SELECT` est très importante, en utilisant l'attribut `HIGH_PRIORITY` . Syntaxe de `SELECT` .
- Si vous avez des problèmes avec des `INSERT` combinés avec des `SELECT` , utilisez les tables `MyISAM` car elle supportent les commandes `SELECT` s et `INSERT` simultanées.
- Si vous voulez mélanger les commandes `INSERT` et `SELECT` , utilisez l'attribut `DELAYED` de la commande `INSERT` pour résoudre ce problème. Syntaxe de `INSERT` .
- Si vous avez des problèmes avec des combinaisons de `SELECT` et `DELETE` , l'option `LIMIT` de `DELETE` peut aider. Syntaxe de `DELETE` .
- Utiliser `SQL_BUFFER_RESULT` avec les commandes `SELECT` peut aider à réduire la durée des verrous. Syntaxe de `SELECT` .
- Vous pouvez changer le code de verrouillage dans le fichier `mysys/thr_lock.c` pour n'utiliser qu'une queue unique. Dans ce cas, les lectures et écritures auront la même priorité, ce qui peut aider certaines applications.

Voici quelques conseils avec le système de verrouillage de MySQL :

- Les accès concurents ne sont pas un problème si vous ne mélangez pas les sélections et les modifications de nombreuses lignes dans la même table.
- Vous pouvez utiliser `LOCK TABLES` pour accélérer les opérations : de nombreuses modifications dans un même verrou seront plus rapides. Répartir le contenu de la table en plusieurs tables peut aussi aider.
- Si vous rencontrez des problèmes de vitesse avec les verrous de tables, vous devez être capables d'améliorer les performances en convertissant certaines tables en `InnoDB` ou `BDB` . Le moteur de tables `InnoDB` . Tables `BDB` ou `BerkeleyDB` .

7.4 Optimiser la structure de la base de données

7.4.1 Conception

MySQL conserve les données et les index dans deux fichiers séparés. De nombreux (et en fait presque toutes) les autres bases mélangent les données et les index dans le même fichier. Nous pensons que le choix de MySQL est bien meilleur pour un grand nombre de systèmes modernes.

Une autre méthode de stockage des données est de conserver les informations de chaque colonne dans une zone séparée (par exemple SDBM et Focus). Cela va réduire les performances qui accèdent à plus d'une colonne. Comme cela dégénère vite lorsque plus d'une colonne est utilisée, nous pensons que ce modèle n'est pas bon pour une base de données généraliste.

Les cas les plus courants sont que les index et les données sont stockées ensemble (comme Oracle/Sybase). Dans ce cas, vous aurez aussi les informations de lignes dans la page finale de l'index. L'intérêt d'une telle organisation est que, dans de nombreuses situations, dépendament du cache d'index, vous économisez des lectures disques. Les problèmes de cette organisation sont :

- Le scan des tables est bien plus lent, car vous devez lire les index pour obtenir les données.
- Vous ne pouvez pas utiliser uniquement l'index pour lire des données pour une requête.
- Vous utilisez beaucoup d'espace, et vous devez dupliquer des index de noeuds (car vous ne pouvez pas simplement stocker des lignes dans les noeuds).
- Les suppressions vont perturber la table (comme les index ne sont pas modifiés lors de l'effacement).
- Il est plus difficile de ne mettre en cache que les données.

7.4.2 Rendre vos tables aussi compactes que possible

Une des optimisations simple est de réduire au maximum la taille de vos données et de vos index sur le disque et en mémoire. Cela peut donner des accélérations impressionnantes, car les lectures sur le disque sont plus rapides, et moins de mémoire centrale sera utilisée. L'indexation de colonnes de petites taille prend aussi moins de ressources.

MySQL supporte un grand nombre de type de tables et de format de ligne. Choisir ces types peut vous conduire à des améliorations de performances. Types de tables MySQL .

Vous pouvez obtenir des gains de performances sur les tables et minimiser l'espace disque en utilisant les techniques ci−dessous :

- Utilisez les types les plus efficaces et les plus petits possibles. MySQL a différents types spécialisés qui épargnent de l'espace disque et de la mémoire.
- Utilisez les types d'entiers les plus petits possible pour réduire les tables. Par exemple, `MEDIUMINT` est souvent préférable à `INT` .
- Déclarez les colonnes pour qu'elle soient `NOT NULL` si possible. Cela accélère les traitements, et vous fait gagner un bit par colonne. Notez que si vous avez vraiment besoin d'une valeur `NULL` dans votre application, il est recommandé de l'utiliser. Evitez simplement de l'utiliser par défaut sur toutes les colonnes.
- Si vous n'avez pas de colonne de taille variable (`VARCHAR` , `TEXT` ou `BLOB`), un format de ligne à taille fixe est utilisé. C'est plus rapide, mais cela prend plus d'espace sur le disque. Format de table `MyISAM` .
- La clé primaire doit être aussi courte que possible. Cela rend l'identification des lignes plus efficace.
- Ne créez que des index dont vous avez besoin. Les index sont bons pour accélérer les lectures, mais sont plus lents lorsque vous écrivez des données. Si vous accéder essentiellement à votre table en lecture avec des combinaisons de colonnes, faites un index avec ces colonnes. Le premier index doit être la colonne la plus utilisée. Si vous utilisez **constamment** de nombreuses colonnes, vous devriez utiliser la colonne avec le plus de doublons en premier, pour obtenir une meilleure compression.
- Si il est probable qu'une colonne a un préfixe unique avec les premiers caractères, il est mieux de n'indexer que ce préfixe. MySQL supporte les index sur une partie de colonne. Les index les plus courts sont les plus efficaces car ils prennent moins d'espace disque, et aussi, car ils absorbent plus de requêtes grâce au cache en mémoire. Choix des paramètres du serveur .
- Dans certaines circonstances, il peut être intéressant de séparer en deux une table qui est scannée très souvent. C'est particulièrement vrai pour les formats de tables dynamiques, et si possible, utilisez un format de table statique pour les colonnes les plus pertinentes.

7.4.3 Index de colonnes

Tous les types de colonnes ed MySQL peuvent être indexés. L'utilisation des index sur les colonnes pertinentes est la meilleur façon d'améliorer les performances de opérations de `SELECT` .

Le nombre maximum de clefs et la longueur maximale des index sont définis pour chaque type de table. Types de tables MySQL . Vous pouvez avec tous les gestionnaires de tables avoir au moins 16 clefs et une taille totale d'index d'au moins 256 octets.

Pour les colonnes `CHAR` et `VARCHAR` , il est possible d'indexer un préfixe de la colonne. C'est plus rapide et plus économe en espace disque que l'indexation de la colonne entière. La syntaxe pour indexer le début d'une colonne au moment de la création de la table ressemble à cela: Index de colonne .

Les moteurs de tables `MyISAM` et (depuis MySQL 4.0.14) `InnoDB` supportent aussi l'indexation des colonnes `BLOB` et `TEXT` . Lors de l'indexation d'une colonne `BLOB` ou `TEXT` , vous **devez** spécifier une taille pour l'index. Par exemple :

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```

Les préfixes peuvent atteindre 255 octets de longeur (ou 1000 octets pour les tables `MyISAM` et `InnoDB` depuis MySQL 4.1.2). Notez que les limites de préfixes sont mesurées en octets, alors que la limite de préfixe dans la commande `CREATE TABLE` est interprétée comme un nombre de caractères. Prenez le en compte lorsque vous spécifiez une taille de préfixe pour une colonne qui utilise un jeu de caractères multi−octets.Depuis MySQL 3.23.23, vous pouvez aussi créer des index `FULLTEXT` . Ils sont utilisés pour les recherches en texte plein. Seules les tables `MyISAM` supportent les index `FULLTEXT` et uniquement pour les colonnes `CHAR` , `VARCHAR` , et `TEXT` . L'indexation se fait sur toute la largeur de la colonne. L'indexation par préfixe n'est pas possible. Voyez la section Fonctions de recherche en texte plein pour plus de détails.

Depuis MySQL 4.1.0, vous pouvez créer des index spatiaux. Actuellement, le type de données spatial n'est supporté que par les tables `MyISAM` . Les données spatiales utilisent un `R−tree` .

Le moteur de tables `MEMORY` (`HEAP`) supporte les index hash. Depuis MySQL 4.1.0, ce moteur supporte aussi les index `B−tree` .

7.4.4 Index sur plusieurs colonnes

MySQL peut créer des index sur plusieurs colonnes. Un index peut comprendre jusqu'à 15 colonnes. (sur les colonnes de type `CHAR` ou `VARCHAR` , vous pouvez utiliser uniquement le début de la colonne pour l'indexation.) (Index de colonne).

Un index sur plusieurs colonnes peut être compris comme un tableau trié contenant des valeurs créées par concaténation des valeurs des colonnes indexées.

MySQL utilise les index sur plusieurs colonnes de telle sorte que les requêtes sont accélérées quand on spécifie une quantité connue de la première colonne de l'index dans un clause `WHERE` , même si on ne spécifie pas la valeur des autres colonnes.

On suppose qu'une table est créée avec les paramètres suivant:

```
mysql> CREATE TABLE test (
    ->       id INT NOT NULL,
    ->       nom CHAR(30) NOT NULL,
    ->       prenom CHAR(30) NOT NULL,
    ->       PRIMARY KEY (id),
    ->       INDEX nom_index (nom,prenom));
```

Alors l'index `nom_index` est un index de `nom` et de `prenom` . Cela sera utile pour les requêtes qui spécifient des valeurs dans une gamme donnée de `nom` , ou pour à la fois `nom` et `prenom` . Ainsi l'index `nom_index` sera utilisé pour les requêtes suivantes:

```
mysql> SELECT * FROM test WHERE nom="Widenius";

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                  AND prenom="Michael";

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                  AND (prenom="Michael" OR prenom="Monty");

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                  AND prenom >="M" AND prenom < "N";
```

Cependant, l'index `nom_index` **ne sera pas** utilisé pour les requêtes suivantes :

```
mysql> SELECT * FROM test WHERE prenom="Michael";

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                  OR prenom="Michael";
```

Pour plus d'informations sur la méthode de MySQL pour utiliser les index dans le but d'améliorer les performance des requêtes, voyez la section suivante.

7.4.5 Comment MySQL utilise les index

Les index sont utilisés pour trouver des lignes de résultat avec une valeur spécifique, très rapidement. Sans index, MySQL doit lire successivement toutes les lignes, et à chaque fois, faire les comparaisons nécessaires pour extraire un résultat pertinent. Plus la table est grosse, plus c'est coûteux. Si la table dispose d'un index pour les colonnes utilisées, MySQL peut alors trouver rapidement les positions des lignes dans le fichier de données, sans avoir à fouiller toute la table. Si une table à 1000 lignes, l'opération sera alors 100 fois plus rapide qu'une lecture séquentielle. Notez que si vous devez lire la presque totalité des 1000 lignes, la lecture séquentielle se révélera alors plus rapide, malgré tout.

Tous les index de MySQL (`PRIMARY` , `UNIQUE` et `INDEX`) sont stockés sous la forme de `B-tree` . Les chaînes sont automatiquement préfixée et leurs espaces terminaux sont supprimés. Syntaxe de `CREATE INDEX` .Les index sont utilisés pour :

- Trouver rapidement des lignes qui satisfont une clause `WHERE` .
- Ecarter rapidement des lignes. S'il y a un choix à faire entre plusieurs index, MySQL utilise généralement celui qui retourne le plus petit nombre de lignes.
- Lire des lignes dans d'autres tables lors des jointures.
- Trouver les valeurs `MAX()` et `MIN()` pour une colonne indexée. C'est une opération qui est optimisée par le preprocesseur, qui vérifie si vous utilisez la constante `WHERE key_part_# =` sur toute les parties de clés inférieures à < N. Dans ce cas, MySQL va faire une simple recherche de clé et remplacer l'expression par une constante. Si toutes les expressions sont remplacées par des constantes, la requête va alors être rapidement calculée :
  ```
  SELECT MIN(key_part2),MAX(key_part2) FROM table_name where key_part1=10
  ```
- Trier ou grouper des lignes dans une table, si le tri ou le regroupement est fait avec un préfixe à gauche utilisable (par exemple, `ORDER BY key_part_1,key_part_2`). La clé est lue en ordre inverse, si toutes les parties de clés sont suivies du mot clé `DESC` . Comment MySQL optimise `ORDER BY` .
- Dans certains cas, la requête peut être optimisée pour lire des valeurs sans consulter le fichier de données. Si cette possibilité est utilisée avec des colonnes qui sont toutes numériques, et forme le préfixe de gauche d'une clé, les valeurs peuvent être lues depuis l'index, à grande vitesse :
  ```
  SELECT key_part3 FROM table_name WHERE key_part1=1
  ```

Supposez que vous utilisiez la commande `SELECT` suivante :

```
mysql> SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;
```

Si un index multi−colonne existe sur les colonnes `col1` et `col2` , les lignes appropriées seront directement lues. Si des index séparés sur les colonnes `col1` et `col2` existent, l'optimiseur va essayer de trouver l'index le plus restrictif des deux, en décidant quel index débouche sur le moins de lignes possibles. Si une table a un index multi−colonne, tout préfixe d'index peut être utilisé par l'optimiseur pour trouver des lignes. Par exemple, si vous avez un index à trois colonnes `(col1,col2,col3)` , vous pouvez faire des recherches accélérées sur les combinaisons de colonnes `(col1)` , `(col1,col2)` et `(col1,col2,col3)` .

MySQL ne peut utiliser d'index partiel sir les colonnes ne forment pas un préfixe d'index. Supposez que vous avez la commande `SELECT` suivante :

```
mysql> SELECT * FROM tbl_name WHERE col1=val1;
mysql> SELECT * FROM tbl_name WHERE col2=val2;
mysql> SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;
```

Si un index existe sur les colonnes `(col1,col2,col3)` , seule la première requête pourra utiliser l'index ci−dessus. Les deux autres requêtes utilisent des colonnes indexées, mais les colonnes `(col2)` et `(col2,col3)` ne font pas partie du préfixe des colonnes `(col1,col2,col3)` .

MySQL utilise aussi les index lors des comparaisons avec l'opérateur `LIKE` si l'argument de `LIKE` est une chaîne constante qui ne commence pas par un caractère joker. Par exemple, les requêtes `SELECT` suivantes utilisent des index :

```
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "Patrick%";
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "Pat%_ck%";
```

Dans le premier exemple, seules les lignes avec `"Patrick" <= key_col < "Patricl"` sont considérées. Dans le second exemple, `"Pat" <= key_col < "Pau"` sont considérées.Les commandes `SELECT` suivantes n'utilisent pas d'index :

```
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "%Patrick%";
mysql> SELECT * FROM tbl_name WHERE key_col LIKE other_col;
```

Dans la première requête, la valeur associée à `LIKE` commence avec un caractère joker. Dans le second exemple, la valeur associée à `LIKE` n'est pas une valeur constante.MySQL 4.0 fait une autre optimisation avec l'opérateur `LIKE` . Si vous utilisez `... LIKE "%string%"` et que `string` est plus grand que 3 caractères, MySQL va utiliser l'algorithme `Turbo Boyer−Moore` qui prend une valeur initiale pour résoudre le masque, et l'exploite pour accélérer la recherche.

Les recherches qui utilisent la fonction `column_name IS NULL` vont utiliser les index si `column_name` sont des index.

MySQL normalement utilise l'index qui génère le moins de lignes possible. Un index est utilisé avec les colonnes que vous spécifiez, et les opérateurs suivants : `=` , `>` , `>=` , `<` , `<=` , `BETWEEN` et l'opérateur `LIKE` sans préfixe joker, c'est à dire de la forme `'quelquechose%'` .

Un index qui ne s'applique pas à tous les niveaux de `AND` dans une requête `WHERE` , ne sera pas utilisé pour optimiser la requête. En d'autres termes, pour être capable d'utiliser un index pour optimiser une requête, un préfixe de l'index doit être utilisé dans toutes les parties de la formule logique contenant `AND` .

Les clauses `WHERE` suivantes utilisent des index :

```
... WHERE index_part1=1 AND index_part2=2 AND other_column=3
... WHERE index=1 OR A=10 AND index=2      /* index = 1 OR index = 2 */
... WHERE index_part1='hello' AND index_part_3=5
        /* optimisé par "index_part1='hello'" */
... WHERE index1=1 and index2=2 or index1=3 and index3=3;
        /* peut utiliser un index sur index1 mais pas sur index2 ou index 3 */
```

Ces clauses WHERE n'utilisent **pas** d'index :

```
 ... WHERE index_part2=1 AND index_part3=2  /* index_part_1 n'est pas utilisé */
... WHERE index=1 OR A=10                  /* Index n'est pas utilisé sur les deux parties du AND */
... WHERE index_part1=1 OR index_part2=10  /* Aucun index ne s'applique à toutes les colonnes */
```

Notez que dans certains cas, MySQL ne va pas utiliser un index, même s'il y en a un disponible. Si l'utilisation de l'index requiert que MySQL accède à plus de 30% des lignes de la table (dans ce cas, un scan de table est probablement plus rapide, et demandera moins d'accès disques). Notez que si une telle requête utilise la clause LIMIT pour ne lire qu'une partie des lignes, MySQL utilisera tout de même l'index, car il va trouver plus rapidement les quelques lignes de résultat.

Les index hash ont des caractéristiques différentes de celles présentées :

- Elles sont utilisées uniquement pour les comparaisons avec les opérateurs = ou <=> (mais elles sont **très** rapides).
- L'optimiseur ne peut pas utiliser un index hash pour accélérer une clause ORDER BY . Ce type d'index ne peut être utilisé que pour rechercher la prochaine ligne dans l'ordre.
- MySQL ne peut déterminer approximativement le nombre de lignes qui sont présentes entre deux valeurs : cette valeur est utilisée par l'optimiseur d'intervalle pour décider quel index utiliser. Cela affecte certaines requêtes, si vous changez la table MyISAM en table MEMORY .
- Seules les clés entières peuvent être recherchées, pour une ligne. Avec un index B-tree , un préfixe peut être utilisé pour trouver les lignes.

7.4.6 Le cache de clé des tables MyISAM

Pour réduire les accès aux disques, le moteur MyISAM emploie une stratégie utilisé par de nombreux systèmes de bases de données. Il utilise un cache qui garde en mémoire les blocs de tables les plus souvent utilisés.

- Pour les blocs d'index, une structure spéciale appelée cache de clés (buffer de clés) est entretenue. La structure contient un certain nombre de bloc de mémoire, où les blocs d'index les plus souvent sollicités résident.
- Pour les blocs de données, MySQL n'utilise pas de cache. Au lieu de cela, il exploite le cache natif du système de fichiers.

Cette section décrit les opérations basiques du cache de clés MyISAM . Puis, elle présente les modifications apportées en MySQL version 4.1 pour améliorer les performances du cache de clés, et vous donner un meilleur contrôle sur les opérations de cache.

- Les accès aux caches de clés ne sont pas séquentiels entre les threads. Des accès simultanés sont désormais possibles.
- Vous pouvez configurer plusieurs caches de clés, et assigner différents index de tables, spécifiquement.

Le mécanisme de cache de clés est aussi utilisé par les tables ISAM . Toutefois, ce n'est pas significatif. Les tables ISAM sont de moins en moins utilisée depuis l'introduction en MySQL 3.23 des tables MyISAM . MySQL 4.1 va plus loin : les tables ISAM sont désactivées par défaut.

Vous pouvez contrôler la taille du cache de clé avec la variable système key_buffer_size . Si cette variable vaut zéro, le cache ne sera pas utilisé. Le cache de clés est aussi désactivé si la valeur de key_buffer_size est trop petite pour allouer le nombre minimal de blocs de buffers (8).

Lorsque le cache de clés n'est pas opérationnel, les fichiers d'index sont lus avec le cache du système de fichiers, fourni par le système d'exploitation. En d'autres termes, les index sont lus avec la même technique que les blocs de données.

Un bloc d'index est une adresse unitaire pour le fichier d'index `MyISAM` . Généralement, la taille d'un bloc d'index est égal à la taille des noeuds de l'index `B−tree` . Les index sont représentés sur le disque en utilisant un arbre `B−tree` . Les noeuds terminaux sont appelés des feuilles. Les noeuds qui ne sont pas des feuilles sont dits non−terminaux.

Tous les blocs de buffer dans la structure de cache de clés ont la même taille. Cette taille peut être égale, supérieure ou inférieure à la taille de bloc d'index de la table. Généralement, un de ces deux valeurs est un multiple de l'autre.

Lorsque des données d'un bloc d'index de table doivent être lues, le serveur commence par vérifier si elles sont disponibles dans le cache de clés, plutôt que sur le disque. C'est à dire, qu'il va préférer écrire ou lire dans le cache de clés que sur le disque. Sinon, le serveur choisit un bloc de cache contenant un index d'une autre table, et remplace les données par celles de la table qu'il manipule. Dès que le bloc est dans le cache, les données d'index sont accessibles.

Si un des blocs sélectionnés pour être écrasé, a été modifié, le bloc est considéré comme ``sale.'' Dans ce cas, avant d'être remplacé, il est d'abord écrit dans le fichier d'index, sur le disque.

Généralement, le serveur suit une heuristique `LRU` (`Least Recently Used` : le moins utilisé) : lorsqu'il choisit un bloc pour être remplacé, il sélectionne le bloc qui a été accédé le moins souvent. Pour faciliter ce choix, le module de cache de clés entretient une queue (la chaîne LRU) de tous les blocs utilisés. Lorsqu'un bloc doit être remplacé, les blocs du début de la queue sont les moins souvent sélectionnés, et sont les candidats au remplacement.

7.4.6.1 Accès au cache de clé partagé

Avant MySQL 4.1, les accès au cache de clé étaient sérialisé : deux threads ne pouvaient y accéder simultanément. Les processus serveur demandent un bloc après avoir finit de traiter la requête précédente. En conséquence, une requête pour un bloc d'index qui n'est pas présente dans le cache de clés bloque l'accès aux autres threads lorsque le cache est complété avec le nouveau bloc.

Depuis la version 4.1.0, le serveur supporte un accès partagé au cache de clés :

- Un buffer qui n'est pas modifié peut être lu par plusieurs threads.
- Un buffer qui est modifié fait attendre les threads qui doivent l'utiliser jusqu'à la fin de la modification.
- Plusieurs threads peuvent initier des requêtes qui engendreront des remplacement de cache, tant qu'ils n'interfèrent pas les uns avec les autres : c'est à dire tant qu'ils ont besoin de blocs différents, et qu'ils remplacent des blocs différents.

L'accès partagé au cache de clé permet au serveur d'améliorer considérablement la vitesse d'exécution.

7.4.6.2 Caches multiples de clés

Les accès partagés à un cache de clés améliorent grandement les performances, mais ne règlent pas les contentieux entre les threads. Ils sont toujours en compétition pour les structures de contrôle qui donnent l'accès aux caches de clés. Pour réduire ces frictions, MySQL 4.1.1 dispose de caches multiples de clés. Cela vous permet d'assigner différentes clés à différents caches.

Lorsqu'il peut y avoir un cache de clé multiple, le serveur doit savoir quel cache utiliser lors du traitement d'une requête, pour une table `MyISAM` . Par défaut, les index des tables `MyISAM` dans le cache par défaut. Pour assigner un index à un cache spécifique, utilisez la commande `CACHE INDEX` .

Par exemple, les deux commandes suivantes assignent les index des tables `t1` , `t2` et `t3` au cache de clé appelé `hot_cache` :

```
mysql> CACHE INDEX t1, t2, t3 IN hot_cache;
+---------+--------------------+----------+----------+
| Table   | Op                 | Msg_type | Msg_text |
+---------+--------------------+----------+----------+
| test.t1 | assign_to_keycache | status   | OK       |
| test.t2 | assign_to_keycache | status   | OK       |
| test.t3 | assign_to_keycache | status   | OK       |
+---------+--------------------+----------+----------+
```

Note : si le serveur a été compilé en incluant le moteur de table `ISAM` , les tables `ISAM` utilise le mécanisme de cache de clés. Mais les index `ISAM` utilisent uniquement le cache de clés par défaut, et ils ne peuvent pas être assignés à un autre cache.

Le cache de clés indiqué dans la commande `CACHE INDEX` peut être créé en spécifiant sa taille avec le paramètre `SET GLOBAL` ou en utilisant les options de démarrage. Par exemple :

```
mysql> SET GLOBAL keycache1.key_buffer_size=128*1024;
```

Pour détruire un cache de clé, donnez lui une taille de zéro :

```
mysql> SET GLOBAL keycache1.key_buffer_size=0;
```

Les variables de cache de clés sont des variables systèmes structurées, qui ont un nom et des composants. Pour `keycache1.key_buffer_size` , `keycache1` est le nom de variable du cache, et `key_buffer_size` est le composant de cache. Voyez Variables système structurées pour une description de la syntaxe utilisée pour faire référence aux variables système structurée.Par défaut, les index de tables sont assignés dans le cache de clé principal, dit aussi cache par défaut. Ce cache est créé au démarrage. Lorsqu'un cache de clés est détruit, les index qui lui étaient assignés sont assignés au cache par défaut.

Pour un serveur en charge, nous recommandons la stratégie suivante pour les caches de clés :

- Un cache de clés principal qui représente 20% de l'espace alloué pour tous les caches de clés. Il sera utilisé par les tables qui sont le plus sollicitées, mais qui ne sont pas modifiées.
- Un cache de clés minoritaire qui représente 20% de l'espace alloué pour tous les caches de clés. Il sera utilisé pour les tables de taille intermédiaires, qui sont intensivement modifiées, comme des tables temporaires, par exemple.
- Un cache de clés secondaire qui représente 60% de l'espace alloué pour tous les caches de clés. C'est le cache de clé par défaut, il sera utilisé pour toutes les autres tables.

Une raison pour utiliser les trois caches de clés est que l'accès à un des caches ne bloque pas l'accès aux autres. Les requêtes qui accèdent aux index d'un des caches ne sont pas en compétition avec les requêtes qui utilisent les index dans les autres cache. Les gains de performances sont aussi dûs à :

- Le cache principal est utilisé pour les requêtes en lecture et son contenu est jamais modifié. Par conséquent, lorsqu'un bloc d'index doit être lu sur le disque, le contenu du bloc remplacé n'a pas besoin d'être sauvé.
- Pour un index assigné au cache principal, s'il n'y a pas de requêtes qui font des scans d'index, il y a une haute probabilité que tous les blocs d'index qui ne sont pas terminaux resteront dans le cache.
- Une opération de modification sur une table temporaire est effectuée plus rapidement lorsque le noeud à modifier est déjà dans le cache, et n'a pas besoin d'être lu dans le

disque. Si la taille des index de la table temporaire est comparable à la taille du cache minoritaire, la probabilité est très haute que l'index soit déjà dans le cache.

7.4.6.3 Stratégie d'insertion au milieu

Par défaut, le système de gestion de cache de clé de MySQL 4.1 utilise la stratégie LRU pour choisir les blocs de cache qui doivent être remplacés, mais il accepte aussi une autre méthode plus sophistiquée, appelée "stratégie de l'insertion au milieu".

Lors de l'utilisation de la stratégie d'insertion au milieu, la chaîne LRU est divisée en deux parties : une sous−chaîne principale, et une sous−chaîne secondaire. Le point de division entre les deux parties n'est pas fixé, mais le système s'assure que la partie principale n'est pas ``trop petite'', et qu'elle contient au moins `key_cache_division_limit` % de bloc de cache de clés. `key_cache_division_limit` est un composant d'une variable structurée de cache de clé, et sa valeur peut être modifiée indépendamment pour chaque cache.

Lorsqu'un bloc d'index est lu dans une table, depuis le cache de clé, il est placé à la fin de la sous−chaîne secondaire. Après un certain nombre d'accès, il est promu dans la sous−chaîne principale. Actuellement, le nombre d'accès requis pour passer un bloc et le même pour tous les blocs d'index. Dans le futur, nous allons permettre au compteur d'accès d'utiliser le niveau de `B−tree` : moins d'accès seront nécessaires à un noeud s'il contient un noeud non−terminal d'un des niveaux supérieur de l'index `B−tree` .

Un bloc de la chaîne principale est placé à la fin de la chaîne. Le bloc circule alors dans la la sous−chaîne. Si le bloc reste à la fin de la sous−chaîne suffisamment longtemps, il est rétrogradé dans la chaîne secondaire. Ce temps est déterminé par la valeur du composant `key_cache_age_threshold` .

La valeur de seuil prescrit que, pour un cache de clé contenant `N` blocs, le bloc au début de la chaîne principale qui n'est pas accédé dans les derniers `N*key_cache_age_threshold/100` accès doit être placé au début de la chaîne secondaire. Il devient le premier candidat à l'éviction, car les blocs de remplacement sont toujours pris au début de la chaîne secondaire.

La stratégie de l'insertion au milieu vous permet de garder les blocs les plus utilisés dans le cache. Si vous préférez utiliser la stratégie LRU classique, laissez la valeur de `key_cache_division_limit` à 100.

La stratégie d'insertion au milieu aide à améliorer les performances lorsque l'exécution d'une requête qui requiert un scan d'index place dans le cache toutes les valeurs de l'index. Pour éviter cela, vous devez utiliser la stratégie d'insertion au milieu, avec une valeur très inférieure à 100 pour `key_cache_division_limit` . Les blocs les plus utilisés seront conservés dans le cache durant un tel scan.

7.4.6.4 Pré−chargement des index

S'il y a suffisamment de blocs dans le cache de clé pour contenir tout un index, ou au moins les blocs correspondant aux blocs non−terminaux, alors cela vaut la peine de pré−charger l'index avant de commencer à l'utiliser. Le pré−chargement vous premet de mettre les blocs d'index dans un buffer de cache le plus efficacement : il lit les blocs séquentiellement sur le disque.

Sans le pré−chargement, les blocs seront placées dans le cache de clé, au fur et à mesure des besoins des requêtes. Même si les blocs resteront dans le cache, puisqu'il y a de la place pour tous, ils seront pris sur le disque dans un ordre aléatoire, et non séquentiellement.

Pour pré−charger un index dans un cache, utilisez la commande `LOAD INDEX INTO CACHE` . Par exemple, la commande suivante précharge les index des tables t1 et t2 :

```
mysql> LOAD INDEX INTO CACHE t1, t2 IGNORE LEAVES;
+---------+--------------+----------+----------+
| Table   | Op           | Msg_type | Msg_text |
+---------+--------------+----------+----------+
| test.t1 | preload_keys | status   | OK       |
| test.t2 | preload_keys | status   | OK       |
+---------+--------------+----------+----------+
```

L'option `IGNORE LEAVES` fait que les blocs non−terminaux seuls seront lus dans l'index. Par conséquent, la commande ci−dessus va charger tous les blocs de l'index de `t1` , mais uniquement les blocs non−terminaux de `t2` .Si un index a été assigné à un cache de clé en utilisant la commande `CACHE INDEX` , le pré−chargement place les blocs dans ce cache. Sinon, l'index est chargé dans le cache par défaut.

7.4.6.5 Taille des blocs du cache de clé

MySQL 4.1 propose une nouvelle variable `key_cache_block_size` , pour chaque clé. Cette variable spécifie la taille du bloc de buffer pour le cache de clé. Il sert à optimiser les performances d'E/S pour les fichiers d'index.

Les meilleurs performances d'E/S sont obtenues lorsque la taille du buffer de lecture est égale à la taille des opérations natives d'E/S système. Mais en donnant une taille de bloc de clé égale à la taille du buffer d'E/S ne donne pas les meilleures performances. Lors de la lecture de grands blocs terminaux, le serveur charge beaucoup de données inutiles, ce qui empêche la lecture d'autres noeuds.

Actuellement, vous ne pouvez pas contrôler la taille des blocs d'index dans la table. Cette taille est fixée par le serveur lorsque le fichier d'index `.MYI` est créé, en fonction de la taille des index de la table. Dans la plupart des cas, il est choisi égal à la taille du buffer d'E/S. Dans le futur, cela sera changé, et la variable `key_cache_block_size` sera exploitée.

7.4.6.6 Restructurer le cache de clé

Un cache de clé peut être restructuré à tout moment, en modifiant les valeurs de ses paramètres. Par exemple :

```
mysql> SET GLOBAL cold_cache.key_buffer_size=4*1024*1024;
```

Si vous assignez une nouvelle valeurs aux variables `key_buffer_size` ou `key_cache_block_size` , le serveur va détruire l'ancienne structure du cache, et en recréer un, en se basant sur les nouvelles valeurs. Si le cache contient des blocs sales, le serveur va les sauver sur le disque avant de détruire et recréer le cache. La restructuration n'intervient pas si vous modifiez un autre paramètre du cache.Lors de la restructuration d'un cache de clé, le serveur commence par vider le contenu des buffers sales sur le disque. Après cela, le contenu du cache devient inaccessible. Cependant, la restructuration ne bloque pas les requêtes qui utilisent des index qui sont dans le cache. Au lieu de cela, le serveur accède directement à la table et utilise le cache du système de fichiers. Le cache du système de fichiers n'est pas aussi efficace que celui du cache de clés : si les requêtes s'exécutent toujours, elles seront un peu ralenties. Une fois que le cache a été restructuré, il redevient accessible, et les blocs peuvent être placés dedans. Le cache du système de fichiers n'est plus utilisé.

7.4.7 Comment MySQL compte les tables ouvertes

Quand vous utiliserez la commande `mysqladmin status` , vous verrez quelque chose de ce genre :

```
Uptime: 426 Running threads: 1 Questions: 11082 Reloads: 1 Open tables: 12
```

Cela vous laissera perplexe si vous n'avez que 6 tables.

MySQL est multi−threadé, il peut donc exécuter plusieurs requêtes sur la même table simultanément. Pour minimiser les interférences entre deux threads ayant différentes actions sur le même fichier, la table est ouverte indépendamment par chacun des threads. Cela nécessite un peu de mémoire, mais augmente les performances. Avec les tables au format `ISAM` et `MyISAM` , cela requière aussi un fichier additionnel de description du fichier des données. Avec ce type de tables, le fichier décrivant l'index est partagé entre tous les threads.

Vous pourrez lire plus sur le sujet à la section suivante : Comment MySQL ouvre et ferme les tables .

7.4.8 Quand MySQL ouvre et ferme les tables

`table_cache` , `max_connections` et `max_tmp_tables` affectent le nombre maximum de tables que le serveur garde ouvertes. Si vous augmentez l'une de ces valeurs, vous pourriez rencontrer une des limites de votre système d'exploitation. Cependant, vous pourrez augmenter ces limites sur de nombreux systèmes d'exploitation. Consultez votre documentation système pour voir comment faire cela, car la méthode pour modifier la limite est différente pour chaque système.

`table_cache` est lié au `max_connections` . Par exemple, pour 200 connexions simultanées, vous devriez avoir un cache de table d'environ `200 * n` , où `n` est le nombre maximum de table dans une jointure. Vous devez aussi réserver des pointeurs de fichiers supplémentaires pour les tables temporaires et les fichiers.

Assurez vous que votre système d'exploitation peut gérer le nombre de pointeurs de fichiers demandé par l'option `table_cache` . Si `table_cache` est trop grand, MySQL peut être à court de pointeurs, et refuser des connexions, échouer à l'exécution de requêtes, ou être très instable. Vous devez aussi prendre en compte que les tables `MyISAM` peuvent avoir besoin de deux pointeurs de fichiers pour chaque table différente. Vous pouvez augmenter le nombre de pointeurs de fichiers disponibles pour MySQL avec l'option de démarrage `--open-files-limit=#` . Fichier non trouvé .

Le cache de tables ouvertes reste au niveau de `table_cache` entrées (par défaut, 64; cela peut être modifié avec l'option `-O table_cache=#` de `mysqld`). Notez que MySQL peut ouvrir temporairement plus de tables, pour être capable d'exécuter des requêtes.

Une table qui n'est pas utilisée est refermée, et supprimée du cache de table, dans les circonstances suivantes :

- Lorsque le cache est plein, et qu'un thread essaie d'ouvrir une table qui n'est pas dans le cache.
- Lorsque le cache contient plus de `table_cache` lignes, et qu'aucun thread n'utilise cette table.
- Lorsque quelqu'un utilise la commande `mysqladmin refresh` ou `mysqladmin flush-tables` .
- Lorsque quelqu'un exécute la commande `FLUSH TABLES` .

Lorsque le cache de table se remplit, le serveur utilise la procédure suivante pour identifier une entrée du cache, pour la supprimer :

- Les tables qui n'est pas en cours d'utilisation est libérée, en utilisant la table qui a été accédé depuis plus longtemps en premier.
- Si le cache est plein, et qu'aucune table ne peut être libérée, mais qu'une nouvelle table doit être ouverte, le cache est temporairement étendu.
- Si le cache est dans un état d'extension, et qu'une table passe de l'état d'utilisation à non utilisation, la table est immédiatement fermée et libérée du cache.

Une table est ouverte pour chaque accès simultané. Cela signifie que si vous avez deux threads qui accèdent à la même table, ou accèdent à la même table deux fois dans la requête (avec `AS`), la table devra être ouverte deux fois. La première ouverture d'une table prendre deux pointeurs de fichiers. Chaque utilisation supplémentaire de la même table ne prendra qu'un pointeur supplémentaire. Le pointeur de fichier supplémentaire de la première table est celui du fichier d'index. Ce pointeur est partagé entre les threads.

Si vous ouvrez une table avec `HANDLER table_name OPEN` , un objet de table dédié sera alloué pour le thread. Cet objet de table n'est pas partagé avec les autres threads, et il ne sera pas fermé avant que le thread n'appelle `HANDLER table_name CLOSE` , ou que le thread ne meurt. Syntaxe de `HANDLER` . Lorsque cela arrive, la table est placée dans le cache de table (si il n'est pas plein).

Vous pouvez vérifier si votre cache de table n'est pas trop petit en vérifiant la variable de `mysqld` appelée `Opened_tables` . Si cette valeur est grande, même si vous n'avez pas trop abusé de la commande `FLUSH TABLES` , vous devrez augmenter la taille du cache. Syntaxe de `SHOW STATUS` .

7.4.9 Inconvénients de la création d'un grand nombre de tables dans la même base de données

Si vous avez beaucoup de fichiers dans un dossier, les opérations d'ouverture, fermeture, et création seront ralenties. Si vous exécutez une requête `SELECT` sur plusieurs tables, il y aura une légère perte lorsque le cache de tables sera plein, car pour chaque table ouverte, une autre doit être fermée. Vous pouvez réduire cette table en augmentant la taille du cache de tables.

7.5 Optimiser le serveur MySQL

7.5.1 Réglage du système, au moment de la compilation, et paramètres du démarrage

Nous démarrons par le niveau du système, car certaines décisions à ce niveau doivent être prises très tôt. Dans d'autres cas, un regard rapide à cette partie doit suffire, car ce n'est pas tellement important pour les gros gains. Toutefois, il est toujours sympathique de sentir combien on peut gagner en changeant des choses à ce niveau.

Le choix du système d'exploitation est vraiment important! Pour utiliser au maximum les capacités de machines multi−processeurs, il vaut mieux choisir Solaris (car les threads marchent vraiment très bien) ou Linux (car le noyau 2.2 supporte très bien le SMP). Mais les plates−formes Linux 32 bits limitent par défaut la taille des fichiers à 2 Go. Heureusement, cela sera bientôt réparé avec l'arrivée des nouveaux systèmes de fichier (XFS/Reiserfs). Si vous souhaitez désespérement utiliser des fichiers de plus de 2 Go sur Linux−intel 32 bits, vous devriez utiliser le patch de LFS pour le système de fichier ext2

Comme nous n'avons pas utilisé MySQL en production sur énormément de plates−formes, nous vous conseillons de tester votre plate−forme avant de la choisir définitivement.

Autres astuces:

- Si vous avez suffisamment de RAM, vous pouvez supprimer toutes les partitions d'échange (swap). Certains systèmes d'exploitation utilisent parfois la partition d'échange quand bien même il reste de la mémoire libre.
- L'utilisation de l'option `--skip-external-locking` de MySQL empêche les verrous externes. Cela n'influencera pas les fonctionnalités de MySQL tant que vous n'utilisez qu'un seul serveur. Il faut cependant penser à arrêter le serveur (ou bien de verrouiller les parties pertinentes) avant d'utiliser `myisamchk` . Sur certains systèmes, cette option est inutile car les verrous externes ne fonctionnent pas du tout.

  L'option `--skip-external-locking` est activée par défaut quand on compile avec `MIT-pthreads` , car `flock()` n'est pas totalement supporté sur toutes les plates−formes par `MIT-pthreads` . Elle l'est également sur Linux, cat le verrouillage des fichiers de Linux n'est pas encore sûr.

  Les seuls cas où on ne peut pas utiliser `--skip-external-locking` sont si on utilise plusieurs **serveurs** (pas de clients) MySQL sur les mêmes données, ou si on lance `myisamchk` sur une table sans vider son tampon et sans la verrouiller au préalable.Il est toujours possible d'utiliser `LOCK TABLES` / `UNLOCK TABLES` même si vous utilisez `--skip-external-locking` .

7.5.2 Réglage des paramètres du serveur

Vous pouvez obtenir les tailles par défaut des tampons du serveur `mysqld` avec la commande:

```
shell> mysqld --help
```

Cette commande génère une liste de toutes les options de `mysqld` et des variables configurables. Cette sortie comprend les valeurs par défaut et ressemble à cela :

```
 Possible variables for option --set-variable (-O) are:
back_log                 current value: 5
bdb_cache_size           current value: 1048540
binlog_cache_size        current value: 32768
connect_timeout          current value: 5
delayed_insert_timeout   current value: 300
delayed_insert_limit     current value: 100
delayed_queue_size       current value: 1000
flush_time               current value: 0
interactive_timeout      current value: 28800
join_buffer_size         current value: 131072
key_buffer_size          current value: 1048540
lower_case_table_names   current value: 0
long_query_time          current value: 10
max_allowed_packet       current value: 1048576
max_binlog_cache_size    current value: 4294967295
max_connections          current value: 100
max_connect_errors       current value: 10
max_delayed_threads      current value: 20
max_heap_table_size      current value: 16777216
max_join_size            current value: 4294967295
max_sort_length          current value: 1024
max_tmp_tables           current value: 32
max_write_lock_count     current value: 4294967295
myisam_sort_buffer_size  current value: 8388608
net_buffer_length        current value: 16384
net_retry_count          current value: 10
net_read_timeout         current value: 30
net_write_timeout        current value: 60
read_buffer_size         current value: 131072
record_rnd_buffer_size   current value: 131072
slow_launch_time         current value: 2
sort_buffer              current value: 2097116
table_cache              current value: 64
thread_concurrency       current value: 10
tmp_table_size           current value: 1048576
```

```
thread_stack            current value: 131072
wait_timeout            current value: 28800
```

Si un serveur `mysqld` est en cours d'exécution, vous pouvez voir les valeurs que les variables utilisent réellement en exécutant la commande :

```
mysql> SHOW VARIABLES;
```

Vous pouvez obtenir les statistiques et différents indicateurs de statut pour un serveur en fonctionnement avec cette commande :

```
mysql> SHOW STATUS;
```

Les variables de serveur et de statut sont aussi accessibles avec `mysqladmin` :

```
shell> mysqladmin variables
shell> mysqladmin extended-status
```

Vous pouvez trouver une description complète de toutes les variables système dans les sections Variables sytème du serveur et Variables de statut du serveur .

MySQL utilise des algorithmes très extensibles, donc vous pouvez utiliser très peu de mémoire. Si malgré tout vous fournissez plus de mémoire à MySQL, vous obtiendrez également de meilleures performances.

Les deux variables les plus importantes au moment du réglage d'un serveur MySQL sont `key_buffer_size` et `table_cache` . Vous devriez vous assurer que celles sont sont bien paramétrées avant de modifier les autres variables.

Les exemples suivants indiquent quelques valeurs typiques pour différentes valeurs de configuration. Les exemples utilisent le script `mysqld_safe` et utilisent la syntaxe `--name=value` pour donner à la variable appelée `name` la valeur `value` . Cette syntaxe est disponible depuis MySQL 4.0. Pour les anciennes versions de MySQL, prenez en compte ces différences :

- Utilisez `safe_mysqld` plutôt que `mysqld_safe` .
- Utilisez la syntaxe de modification des variables `--set-variable=nom=valeur` ou `-O name=value` .
- Pour les noms de variables qui finissent par `_size` , vous pouvez les spécifier sans le suffise `_size` . Par exemple, l'ancien nom de `sort_buffer_size` est `sort_buffer` . L'ancien nom de `read_buffer_size` est `record_buffer` . Pour voir quelles variables votre version du serveur reconnaît, utilisez `mysqld --help` .

Si vous avez beaucoup de mémoire (>=256 Mo) et beaucoup de tables, et que vous désirez des performances maximales avec un faible de nombre de clients, vous devriez essayer quelque chose cela:

```
shell> safe_mysqld -O key_buffer=64M -O table_cache=256 \
           -O sort_buffer=4M -O read_buffer_size=1M &
```

Si vous n'avez que 128 Mo et seulement quelques tables, mais que vous demandez beaucoup de classements, vous pouvez essayer cela :

```
shell> safe_mysqld -O key_buffer=16M -O sort_buffer=1M
```

Si vous avez peu de mémoire et beaucoup de connections, essayez cela:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=100k \
           -O read_buffer_size=100k &
```

Ou encore:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=16k \
           -O table_cache=32 -O read_buffer_size=8k -O net_buffer_length=1K &
```

Si vous utilisez `GROUP BY` ou `ORDER BY` sur des fichiers de taille supérieure à la mémoire disponible, vous devriez augmenter la valeur de `record_rnd_buffer` pour accélérer la lecture des lignes après que le classement ait été fait.

A l'installation de MySQL, un répertoire `support-files` est créé, et contient plusieurs exemples de fichiers `my.cnf` : `my-huge.cnf` , `my-large.cnf` , `my-medium.cnf` et `my-small.cnf` . Vous pouvez les utiliser comme base pour optimiser votre système.

Si vous avez vraiment beaucoup de connections, des problèmes peuvent apparaître avec le fichier d'échange si `mysqld` n'a pas été configuré pour utiliser peu de mémoire pour chaque connexion. `mysqld` fonctionne mieux si vous avec suffisamment de mémoire pour toutes les connections, bien sûr !

Notez que si vous changez une option de `mysqld` , elle ne prendra effet qu'au prochain démarrage du serveur.

Pour voir les effets d'un changement de paramètre, essayez quelque chose comme ca:

```
shell> mysqld -O key_buffer=32m --help
```

Les valeurs des variables sont listées vers la fin du résultat. Assurez vous bien de la présence de l'option `--help` en fin de ligne; si ce n'est pas le cas, les options listées après dans la ligne de commande ne seront pas prises en compte à la sortie.Pour plus d'information sur le paramétrage du moteur `InnoDB` , voyez la section Paramétrage optimale d' `InnoDB` .

7.5.3 Influences de la compilation et des liaisons sur la vitesse de MySQL

La plupart des tests suivants ont été réalisés sous Linux avec les outils comparatifs de MySQL, mais ils peuvent donner quelques indications pour d'autres systèmes d'exploitation et sur une charge de travail différente.

Les exécutables les plus rapides sont obtenus en liant avec `-static` .

Sur Linux, le code le plus rapide sera obtenu en compilant avec `pgcc` et `-O3` . Pour compiler `sql_yacc.cc` avec ces options, il faut environ 200 Mo de mémoire car `gcc/pgcc` demande beaucoup de mémoire pour créer toutes les fonctions d'une traite. Il est aussi possible d'utiliser `CXX=gcc` à la configuration de MySQL pour éviter l'inclusion de la bibliothèque `libstdc++` (qui n'est pas nécessaire). Sachez que pour certaines versions de `pgcc` , le code résultant ne fonctionnera que sur de vrais processeurs Pentium, même si vous utilisez l'option du compilateur qui doit générer du code fonctionnant sur tour les types de processeurs x586 (comme AMD).

L'utilisation du meilleur compilateur et/ou de la meilleur option de compilation permet de gagner 10 à 30% de vitesse dans vos applications. C'est très important quand vous compilez le serveur SQL vous−même !

Nous avons compilé avec les compilateurs de Cygnus CodeFusion et de Fujitsu, mais aucun des deux n'était suffisamment exempt d'erreurs pour permettre la compilation de MySQL avec l'optimisation.

A la compilation de MySQL, vous devriez uniquement utiliser le support des caractères que vous allez utiliser. (Option `--with-charset=xxx` .) Les distributions binaires standards de MySQL sont compilées avec le support de toutes les gammes de caractères.Voici une liste des mesures que nous avons effectués:

- L'utilisation de `pgcc` et la compilation complète avec l'option `-O6` donne un serveur `mysqld` 1% plus rapide qu'avec `gcc` 2.95.2.
- Si vous utilisez la liaison dynamique (sans `-static`), le résultat est 13% plus lent sur Linux. Sachez que vous pouvez néanmoins utiliser la liaison dynamique pour les bibliothèques de MySQL. Seul le serveur a des performances critiques.
- Si vous allégez votre binaire `mysqld` avec l'option `strip libexec/mysqld` , vous obtenez un binaire jusqu'à 4% plus rapide.

- Si vous utilisez TCP/IP plutôt que les sockets Unix, le résultat est 7.5% plus lent sur le même ordinateur. (Si vous vous connectez sur `localhost` , MySQL utilisera les sockets par défaut.)
- Si vous vous connectez en TCP/IP depuis un autre ordinateur avec un lien Ethernet 100 Mo/s, le résultat sera 8 à 11% plus lent.
- L'utilisation de connections sécurisées (toutes les données chiffrées par le support interne de SSL) pour nos tests comparatifs a provoqué une perte de vitesse de 55%.
- Si vous compilez avec `--with-debug=full` , vous perdrez 20% de performances sur la plupart des requêtes, mais la perte peut être plus importante sur certaines requêtes (La suite de tests de MySQL tourne 35% plus lentement). Si vous utilisez `--with-debug` , vous ne perdrez que 15%. En démarrant une version de `mysqld` , compilée avec `--with-debug=full` , avec `--skip-safemalloc` , le résultat final devrait être proche d'une compilation avec `--with-debug` .
- Sur un Sun UltraSPARC−IIe, Forte 5.0 est 4% plus rapide que `gcc` 3.2.
- Sur un Sun UltraSPARC−IIe, Forte 5.0 est 4% plus rapide en mode 32 bit qu'en mode 64 bit.
- La compilation avec `gcc` 2.95.2 sur UltraSPARC avec l'option `-mcpu=v8 -Wa,-xarch=v8plusa` améliore les performances de 4%.
- Sur Solaris 2.5.1, `MIT-pthreads` est 8−12% plus lent que la gestion native des threads de Solaris sur mono−processeur. Avec plus de charge ou de CPU, la différente devrait être encore plus grande.
- La compilation sur Linux−x86 avec gcc sans les pointeurs `-fomit-frame-pointer` ou `-fomit-frame-pointer -ffixed-ebp` rend `mysqld` 1 à 4% plus rapide.

Autrefois les distributions fournies par MySQL AB de MySQL−Linux étaient compilées avec `pgcc` , mais nous avons dû revenir au simple gcc à cause d'un bogue dans `pgcc` qui générait du code qui ne fonctionnait pas sur AMD. Nous continuerons à utiliser gcc tant que ce bogue ne sera pas corrigé. Néanmoins, si vous avez une machine non−AMD, vous pouvez obtenir des binaires plus rapides en compilant avec `pgcc` . Le binaire standard de MySQL pour Linux est lié statiquement pour être plus rapide et plus portable.

7.5.4 Comment MySQL gère la mémoire

La liste suivante indique certaines techniques utilisées par le serveur `mysqld` pour gérer la mémoire. Lorsque c'est possible, la variable serveur liée à la mémoire est indiquée :

- Le buffer de clés (variable `key_buffer_size`) est partagé par tous les threads. Les autres buffers sont sont alloués par le serveur suivant les besoins. Choix des paramètres du serveur .
- Chaque connexion utilise un espace spécifique au thread :
    - une pile (par défaut, 64 ko, variable `thread_stack`),
    - un buffer de connexion (variable `net_buffer_length`),
    - un buffer de résultat (variable `net_buffer_length`).

  Le buffer de connexion et celui de résultat sont dynamiquement élargit jusqu'à `max_allowed_packet` suivant les besoins. Lorsque la requête s'exécute, une copie de la chaîne de requête est aussi allouée.
- Tous les threads partagent la même mémoire de base.
- Seules les tables compressées `ISAM` / `MyISAM` sont copiées en mémoire. Ceci est dû au fait que pour un espace de 32 bits, il n'y a pas de place pour les grosses tables en mémoire. Lorsque les systèmes de 64 bits seront plus répandus, nous pourrions généraliser le support pour la copie en mémoire.
- Chaque requête qui effectue une analyse séquentielle d'une table, alloue un buffer de lecture (variable `record_buffer`).
- Lors de la lecture de lignes en ordre 'aléatoire' (par exemple, après un tri), un buffer de lecture aléatoire est allouée pour éviter les accès disques (variable `record_rnd_buffer`).
- Toutes les jointures sont faîtes en une seule passe, et la plupart des jointure sont faîtes

sans utiliser de table temporaire. La plupart des table temporaires sont faîtes en mémoire (table HEAP). Les tables temporaires avec beaucoup de données (calculées comme la somme des tailles de toutes les colonnes) ou qui contiennent des colonnes de type BLOB sont sauvées sur le disque.Un problème avec les versions de MySQL antérieures à la version 3.23.2 est que si une table HEAP dépassait la taille maximale de tmp_table_size , vous obteniez une erreur The table tbl_name is full . Dans les nouvelles versions, ce problème est géré en passant automatiquement la table HEAP en une table MyISAM sur le disque. Pour contourner ce problème, vous pouvez augmenter la taille maximale des tables en mémoire en modifiant l'option tmp_table_size de mysqld , ou en modifiant l'option SQL BIG_TABLES dans le programme client. Syntaxe de SET . En MySQL version 3.20, la taille maximale de la table temporaire est record_buffer*16 , ce qui fait que si vous utilisez cette version, vous aurez à augmenter la valeur de record_buffer . Vous pouvez aussi démarrer mysqld avec l'option --big-tables pour toujours stocker les tables temporaires sur le disque. Cependant, cela va affecter la vitesse de votre serveur pour les requêtes complexes.

- La plupart des requêtes qui sont triées allouent un buffer de tri, et entre 0 et 2 fichiers temporaires, suivant la taille du résultat. Où MySQL conserve les fichiers temporaires .
- Toute l'analyse et les calculs sont faits en mémoire locale. Aucune mémoire supplémentaire n'est nécessaire pour les petits calculs, et les allocations et libérations de mémoire sont évités. La mémoire n'est allouée que pour les chaînes très grandes (ceci se fait via malloc() et free()).
- Chaque fichier d'index est ouvert une fois, et le fichier de données est ouvert pour chaque thread concurrent. Pour chaque thread concurrent, une structure de table, une structure de colonne pour chaque colonne et un buffer de taille 3 * n est alloué (où n est la taille maximale de ligne, en dehors des colonnes de type BLOB). Une colonne de type BLOB utilise 5 à 8 octets de plus que la taille des données du BLOB . Les gestionnaires de table ISAM / MyISAM utilisent un buffer d'une ligne de plus pour leur utilisation interne.
- Pour chaque table qui a une colonne BLOB , un buffer est dynamiquement agrandi pour lire les valeurs BLOB . Si vous analysez toute une table, un buffer aussi grand que la plus grande valeur de la colonne BLOB sera alloué.
- Les gestionnaires de tables pour les tables en cours d'utilisation sont sauvées dans un cache, et géré comme une pile FIFO. Normalement, ce cache contient 64 lignes. Si une table doit être utilisée par deux threads concurrents simultanément, le cache contiendra deux entrées pour la table. Comment MySQL ouvre et ferme les tables .
- La commande mysqladmin flush-tables ferme toute les tables qui ne sont pas utilisées, et marque toutes les tables en cours d'utilisation pour qu'elles soient fermées dès la fin du thread. Cela va libérer l'essentiel de la mémoire utilisée.

ps et d'autres commandes de statut système peuvent indiquer que mysqld utilise beaucoup de mémoire. Ceci est peut être dû a des erreurs de comptabilité. Par exemple, sous Solaris, ps compte la mémoire inutilisée entre les threads comme de la mémoire utilisée. Vous pouvez le vérifier en regardant l'état de la swap avec swap -s . Nous avons testé mysqld avec les détecteurs de fuite mémoire commerciaux, et il n'y a aucune fuite.

7.5.5 Comment MySQL utilise le DNS

Quand un nouveau thread se connecte à mysqld , mysqld crée nouveau thread pour traiter la requête. Ce thread contrôle d'abord si le nom de l'hôte est dans le cache des noms d'hôte. Si ce n'est pas le cas, le thread va appeler gethostbyaddr_r() et gethostbyname_r() pour résoudre le nom de l'hôte.

- Si le système d'opération supporte les fonctions gethostbyaddr_r() et gethostbyname_r() , compatibles avec les threads, elles sont utilisées pour la résolution.
- Si le système d'exploitation ne supporte pas les appels précédents, le thread va verrouiller un "mutex" et appeler gethostbyaddr() et gethostbyname() à la place. Sachez que dans ce cas, aucun

autre thread ne peut résoudre de nom d'hôte qui n'est pas dans le cache tant que le premier thread n'a pas fini.

Il est possible de désactiver la recherche du nom par DNS en démarrant `mysqld` avec l'option `--skip-name-resolve` . Dans ce cas, il est toujours possible d'utiliser les adresses IP dans les tables de privilèges de MySQL.

Si votre service DNS est très lent et que vous avez beaucoup d'hôtes, vous pouvez améliorer les performances soit en désactivant le DNS avec `--skip-name-resolve` , soit en augmentant la taille de `HOST_CACHE_SIZE` (par défaut: 128) et en recompilant `mysqld` .

Il est possible de désactiver le cache de noms d'hôte avec `--skip-host-cache` . Il est possible de vider le cache des noms d'hôtes avec `FLUSH HOSTS` ou avec `mysqladmin flush-hosts` .

Si vous ne voulez pas autoriser les connections par `TCP/IP` , vous pouvez utiliser l'option `--skip-networking` au démarrage de `mysqld` .

7.6 Problèmes avec les disques

- Comme mentionné plus tôt, les accès disques représentent une limitation. Ce problème devient de plus en plus apparent, au fur et à mesure que les données sont de plus en plus nombreuses, et que les techniques de cache deviennent impossibles. Pour les grandes bases de données, lorsque vous accédez aux données plus ou moins aléatoirement, vous pouvez être sûr que vous aurez besoin d'un accès disque pour lire, et de plusieurs autres pour écrire. Pour minimiser le problème, utilisez des disques avec des temps d'accès très faibles.
- Augmentez le nombre de disques disponibles (et donc, réduisez le coût d' un accès), en plaçant des données sur d'autres fichiers via des liens symboliques.
- Pour plus de robustesse, vous pouvez utiliser des disques en RAID 0+1 (parallélisme et réplication), mais dans ce cas, vous aurez besoin de 2*N disques pour contenir vos données sur N disques. C'est probablement l'option la plus sûre, si vous avez le budget pour cela. Vous risquez aussi d'avoir à investir dans un système de gestion de gros volume de données pour gérer cela efficacement.
- Une bonne option est de garder les données semi−importantes (qui peuvent être regénérées) sur un disque RAID 0 tandis que les données vraiment importantes (comme les informations d'hôtes et les log) sur un disque de type RAID 0+1 ou RAID N. RAID N peut être un problème si vous avez de nombreux accès en écrire, à cause du temps de modification des bits de parité.
- Sous Linux, vous pouvez améliorer les performances (jusqu'à 100% en charge n'est pas difficile) en utilisant `hdparm` pour configurer votre interface disque. La commande suivante doit être une série de bonnes options de `hdparm` pour MySQL (et probablement d'autres applications) :

  ```
  hdparm −m 16 −d 1
  ```

  Notez que la performances et la robustesse des solutions ci−dessus dépendent de votre matériel, et nous vous conseillons vivement de tester votre système soigneusement après avoir utilisé `hdparm` ! Consultez le manuel de `hdparm` pour plus de détails. Si `hdparm` n'est pas utilisé correctement, le système de fichiers peut être corrompu. Sauvegardez tout avant d'expérimenter.
- Vous pouvez aussi modifier les paramètres suivants sur le système de fichiers que la base de données utilise : Si vous n'avez pas besoin de savoir quand un fichier a été accédé la dernière fois (ce qui n'est pas utile avec un serveur de base de données), vous pouvez monter votre système de fichier avec l'option `−o noatime` .

Sur de nombreux systèmes d'exploitation, vous pouvez monter des disques avec l'option `-o async` pour que le système de fichiers soit modifié de manière asynchrone. Si votre serveur est raisonnablement stable, vous devriez obtenir de bonne performances sans sacrifier la stabilité (cette option est activée par défaut sur Linux).

7.6.1 Utiliser des liens symboliques

Vous pouvez déplacer les dossiers de bases de données et les placer dans un autre endroit, puis remplacer les dossiers eux−mêmes par des liens symboliques vers ces autres endroits. Vous pourriez vouloir faire cela pour mettre la base de données sur un système de fichier plus rapide, ou pour gagner de l'espace disque sur le système central, ou encore répartir vos tables sur différents disques.

Le mieux, pour cela, est de faire des liens symboliques des bases vers les différents disques, et de ne faire des liens symboliques sur les tables qu'en dernier ressort.

7.6.1.1 Utiliser les liens symboliques pour les bases

Pour créer des liens symboliques sur les bases de données, vous devez commencer par créer un dossier sur un disque de destination, puis faire un lien symbolique depuis le dossier de données vers votre dossier de destination.

```
 shell> mkdir /dr1/databases/test
shell> ln −s /dr1/databases/test mysqld−datadir
```

MySQL n'accepte pas que vous fassiez le lien depuis plusieurs bases sur le même dossier. Remplacer une base par un lien symbolique sera correct tant que vous n'essayez pas de faire des liens symboliques dans la même base. Supposez que vous la base `db1` dans le dossier de données MySQL, puis que vous fassiez un lien symbolique `db2` qui pointe sur `db1` :

```
 shell> cd /path/to/datadir
shell> ln −s db1 db2
```

Maintenant, pour toute table `tbl_a` de `db1`, il en apparaît aussi `tbl_a` dans `db2`. Si un thread modifie `db1.tbl_a` et un autre `db2.tbl_a`, il va y avoir un conflit.Si vous avez vraiment besoin de cette fonctionnalité, vous devez changer le code suivant dans le fichier C `mysys/mf_format.c` :

```
 if (!(MyFlags & MY_RESOLVE_LINK) ||
     (!lstat(filename,&stat_buff) && S_ISLNK(stat_buff.st_mode)))
```

Avant MySQL 4.0, recherchez cette instruction dans le fichier `mysys/mf_format.c` :

```
 if (flag & 32 || (!lstat(to,&stat_buff) && S_ISLNK(stat_buff.st_mode)))
```

Remplacez l'instruction par :

```
 if (1)
```

Sous Windows, vous pouvez utiliser des liens internes symboliques pour relier des bases en compilant MySQL avec l'option `-DUSE_SYMDIR`. Cela vous permettra de placer vos bases de données sur différentes partitions. Utiliser les liens symboliques pour les bases sous Windows .

7.6.1.2 Utiliser les liens symboliques avec les tables sous Unix

Avant MySQL 4.0, vous ne devez pas utiliser les liens symboliques avec les tables, si vous n'êtes pas **très** prudents avec. Le problème est que si vous exécutez `ALTER TABLE`, `REPAIR TABLE` ou `OPTIMIZE TABLE` sur une table symbolique, le lien sera supprimé et remplacé par le fichier original. Cela arrive car les commandes ci−dessus fonctionnent en créant un fichier temporaire dans le dossier de base, et lorsque l'opération est faite, l'original est remplacé par la copie.

Vous ne devez pas utiliser des liens symboliques sur les tables, sur les systèmes qui ne supportent pas complètement la fonction `realpath()` . (Au moins Linux et Solaris supportent `realpath()`)

En MySQL 4.0, les liens symboliques sont complètement supportés par les tables `MyISAM` . Les autres types de tables vous donneront des résultats étranges lorsque vous les utilisez comme indiqué ci−dessus.

La gestion des liens symboliques de MySQL 4.0 fonctionne comme ceci (uniquement pour les tables `MyISAM`) :

Dans le dossier de données, vous allez toujours trouver le fichier de définition de table, le fichier de structure et le fichier d'index.

- Dans le dossier de données, vous devez toujours avoir le fichier de définition de table, le fichier de données et le fichier d'index. Les fichiers de données et d'index peuvent être déplacés ailleurs, et remplacés dans le dossier de données par des liens symboliques. Mais le fichier de définition ne le peut pas.
- Vous pouvez utiliser un lien symbolique avec le fichier d'index et celui de données, pour placer ces fichiers dans d'autres dossiers.
- Le lien symbolique peut être fait via le système d'exploitation (si `mysqld` ne fonctionne pas) ou avec la commande `INDEX/DATA DIRECTORY="path-to-dir"` dans `CREATE TABLE` . Syntaxe de `CREATE TABLE` .
- `myisamchk` ne va pas remplacer un lien symbolique avec les données ou le fichier d'index, mais il va travailler directement sur le fichier vers lequel le lien pointe. Tous les fichiers temporaires seront créé dans le même dossier que le dossier qui contient les données ou le fichier d'index.
- Lorsque vous détruisez une table qui utilise un lien symbolique, le fichier et le lien symbolique sont détruits. C'est une bonne raison pour **ne pas** exécuter `mysqld` en tant que `root` ou donner des droits d'écriture à d'autres personnes dans les dossiers de données de MySQL.
- Si vous renommez une table avec `ALTER TABLE RENAME` vous n'avez pas à déplacer la table dans une autre base, le lien symbolique du dossier de base sera renommé avec le nouveau nom.
- Si vous utilisez la commande `ALTER TABLE RENAME` pour déplacer la table dans une autre base, la table sera déplacée dans l'autre base, et l'ancien lien symbolique et le fichier vers lequel il pointait seront détruits (en d'autres termes, la nouvelle table ne sera pas un lien symbolique).
- Si vous n'utilisez pas de lien symbolique, vous devriez utiliser l'option `--skip-symlink` de `mysqld` pour vous assurer que personne n'efface ou ne renomme un fichier en dehors du dossier de données de MySQL.

`SHOW CREATE TABLE` n'indique pas si une table a des liens symboliques, avant la version 4.0.15. C'est aussi vrai pour `mysqldump` , qui utilise `SHOW CREATE TABLE` pour générer les commandes `CREATE TABLE` .

Ce qui n'est pas encore supporté :

- `ALTER TABLE` ignore toutes les options `INDEX/DATA DIRECTORY="path"` .
- `BACKUP TABLE` et `RESTORE TABLE` ne respectent pas les liens symboliques.
- Le fichier `.frm` **ne doit jamais** être un lien symbolique (Comme indiqué précédemment, seul les fichiers d'index et de données peuvent être des liens symboliques. Si jamais vous le faites malgré tout, vous générerez des erreurs de cohérence. Supposez que vous une base `db1` dans le dossier de données `MySQL` , et une table `tbl1` dans cette base, et dans le dossier `db1` , vous faites un lien symbolique `tbl2` qui pointe sur `tbl1` :
  ```
  shell> cd /path/to/datadir/db1
  shell> ln -s tbl1.frm tbl2.frm
  shell> ln -s tbl1.MYD tbl2.MYD
  shell> ln -s tbl1.MYI tbl2.MYI
  ```
  Il va y avoir des problèmes si un thread lit `db1.tbl1` et qu'un autre modifie `db1.tbl2` :

- Le cache de requête sera induit en erreur (il va croire que `tbl1` a été mis à jour, et retournera des résultats incohérents).
- La commande `ALTER` de la table `tbl2` va aussi échouer.

7.6.1.3 Utiliser des liens symboliques pour les bases de données sous Windows

Depuis MySQL 3.23.16, les serveurs `mysqld-max` et `mysql-max-nt` de la stribution MySQL sont compilé avec l'option `-DUSE_SYMDIR` . Cela vous permet de disposer d'un dossier de base de données sur un autre disque, en utilisant un lien symbolique vers ce dossier, même si la procédure à suivre pour configurer ce lien est différent.

Depuis MySQL 4.0, les liens symboliques sont activés par défaut. Si vous n'en avez pas besoin, vous pouvez les désactiver avec l'option `skip-symbolic-links` :

```
 [mysqld]
skip-symbolic-links
```

Avant MySQL 4.0, les liens symboliques sont désactivés par défaut. Pour les activer, vous pouvez ajouter la ligne suivante dans votre fichier `my.cnf` ou `my.ini` :

```
 [mysqld]
symbolic-links
```

Sous Windows, vous créez un lien symbolique vers une base de données MySQL en créant un fichier qui contient le nom du dossier de destination. Sauvez le fichier dans le dossier de données, en utilisant le nom `db_name.sym` , où `db_name` est le nom de la base.

Supposons que le dossier de données MySQL est `C:\mysql\data` et que votre base `foo` soit placée dans le dossier `D:\data\foo` , vous pouvez configurer les liens symboliques comme ceci :

- Assurez vous que le dossier `D:\data\foo` existe bien, en le créant si nécessaire. Si vous avez déjà un dossier appelé `foo` dans le dossier de données, vous devez le déplacer dans `D:\data` . Sinon, le lien symbolique sera inopérant. Pour éviter les problèmes, le serveur ne doit pas fonctionner lorsque vous déplacez le dossier.
- Créez le fichier `C:\mysql\data\foo.sym` qui contient le chemin `D:\data\foo\` .

Après cela, toutes les tables créées dans la base `foo` seront créées dans le dossier `D:\data\foo` . Notes que les liens symboliques ne seront pas utilisés si un dossier du même nom existe dans le dossier de données MySQL.

8 MySQL Scripts clients et utilitaires

Il y a de nombreux programmes clients de MySQL, qui se connectent au serveur pour accéder aux bases ou effectuer des opérations administratives. D'autres utilitaires sont aussi disponibles. Ils ne communiquent pas avec le serveur, mais effectuent des opérations liées à MySQL.

Ce chapitre fournit un bref aperçu de ces programmes, et vous en dit plus long sur le fonctionnement de chacun. Les descriptions indiquent comment invoquer les programmes, et quelles options ils comprennent. Voyez Using MySQL Programs pour des informations plus générales sur les programmes et leur options.

8.1 Présentation des scripts serveurs et utilitaires

Voici une brève liste des scripts et utilitaires pour le serveur :

`myisampack`
: Un utilitaire qui compresse les tables `MyISAM` pour produire des tables en lecture seule très compactes. `myisampack` , le générateur de tables MySQL compressées .

`mysql`
: Le client en ligne de commande, pour envoyer des requêtes à MySQL, interactivement, ou en batch. L'outil de ligne de commande .

`mysqlaccess`
: Un script qui vérifie les droits d'accès du trio hôte, utilisateur et base de données.

`mysqladmin`
: Un utilitaire pour réaliser des opérations d'administration de la base, telles que les créations de bases, le rafraîchissement des tables de droits, l'écriture des tables sur le disque et la réouverture des fichiers de log. à `mysqladmin` permet aussi de lire la version, les processus et les informations de statut du serveur. `mysqladmin` , administrer un serveur MySQL .

`mysqlbinlog`
: Utilitaire de lecture des requêtes au format binaire. Peut être utilisé après un crash, sur une vieille sauvegarde. L'utilitaire binaire `mysqlbinlog` .

`mysqlcc`
: Interface graphique avec le serveur MySQL. `mysqlcc` , le centre de contrôle MySQL .

`mysqlcheck`
: Un client d'entretien de tables, qui vérifie, répare, analyse et optimise les tables. Utiliser `mysqlcheck` pour l'entretien des tables et les réparations .

`mysqldump`
: Exporte une base de données MySQL dans un fichier sous la forme de requêtes SQL, ou de fichiers texte, avec la tabulation comme séparateur. Un freeware amélioré, d'après une idée originale de Igor Romanenko. `mysqldump` , exporter les structures de tables et les données .

`mysqlhotcopy`
: Un utilitaire qui fait des sauvegardes rapides des tables `MyISAM` et `ISAM` alors que le serveur fonctionne. `mysqlhotcopy` , copier les bases et tables MySQL .

`mysqlimport`
: Importe les fichiers textes dans les tables, en utilisant la commande `LOAD DATA INFILE` . `mysqlimport` , importer des données depuis des fichiers texte .

`mysqlshow`
: Lister les bases, tables et colonnes .

`perror`
: Un utilitaire qui affiche la signification des codes d'erreur MySQL. `perror` , expliquer les codes d'erreurs .

`replace`
: `replace` modifie des chaînes dans des fichiers, ou sur l'entrée standard. L'utilitaire de remplacement `replace` .

Chaque programme MySQL accepte différentes options. Mais chaque programme dispose de l'option `--help` qui vous donne une description complète des options du programme. Par exemple, `mysql --help` .

Les clients MySQL qui communiquent avec le serveur, utilisent la bibliothèque `mysqlclient` est les variables d'environnements suivantes :

MYSQL_UNIX_PORT	Le fichier de socket Unix par défaut. Utilisé pour les connexions à localhost
MYSQL_TCP_PORT	Le port par défaut. Utilisé par les connexions TCP/IP.
MYSQL_PWD	Le mot de passe par défaut.
MYSQL_DEBUG	Les options de débogage.
TMPDIR	Le dossier où les tables et fichiers temporaires sont placés.

Utiliser `MYSQL_PWD` n'est pas sécuritaire, et est fortement déconseillé. Sécuriser votre mot de passe .

Vous pouvez remplacer les valeurs par défaut des options en spécifiant les valeurs dans un fichier d'options, ou en ligne de commande. Options spécifiques des programmes .

8.2 myisampack , le générateur de tables MySQL compressées en lecture seule

`myisampack` sert à compresser des tables MyISAM et `pack_isam` sert à compresser les tables ISAM. Comme les tables ISAM sont obsolètes, nous ne traiterons que de `myisampack` , mais tout ce qui est dit au sujet de `myisampack` est aussi vrai pour `pack_isam` .

`myisampack` fonctionne en compressant séparément chaque colonne de la table. Les informations nécessaires à la décompression sont lues en mémoire lorsque la table est ouverte. Cela donne de bien meilleures performances lors de l'accès à des lignes individuelles, car nous n'avez qu'à décompresser exactement une des lignes, et non pas un bloc de disque entier. Généralement, `myisampack` compresse le fichier avec un gain de 40 à 70 %.

MySQL utilise la carte mémoire (`mmap()`) sur les tables compressées et utilise les outils classiques de lecture et écriture si `mmap()` ne fonctionne pas.Notez bien ceci :

- Si `mysqld` a été appelé avec l'option `--skip-external-locking` , ce n'est pas une bonne idée que d'appeler `myisampack` si la table risque d'être mise à jour par le processus principal.
- Après avoir compressé la table, celle−ci n'est plus accessible qu'en lecture. C'est souvent un état voulu (par exemple, pour être gravée sur un CD). De plus, autoriser les écritures dans une table compressée fait partie de notre liste de tâche, mais avec une très faible priorité.
- `myisampack` peut aussi compresser des colonnes `BLOB` ou `TEXT` . L'ancien `pack_isam` (pour les tables `ISAM`) ne peut le faire.

`myisampack` est invoqué comme ceci :

```
shell> myisampack [options] filename ...
```

Chaque nom de fichier doit être le nom d'un fichier d'index (`.MYI`). Si vous n'êtes pas dans le dossier de données, vous devez spécifier le chemin complet jusqu'au fichier. Il est toléré que vous omettiez l'extension du fichier `.MYI` .

`myisampack` supporte les options suivantes :

`--help` , `-?`

Affiche le message d'aide et quitte.

`--backup` , `-b`

Fait une sauvegarde de la table sous le nom de `tbl_name.OLD` .

`--debug[=debug_options]` , `-# [debug_options]`

Affiche le log de débogage. La chaîne `debug_options` vaut souvent `'d:t:o,filename'` .

`--force` , `-f`

Force la compression de la table, même si elle grossit ou si le fichier temporaire existe déjà. `myisampack` crée un fichier temporaire appelé `tbl_name.TMD` lors de la compression. Si vous tuez `myisampack` , le fichier `.TMD` peut ne pas être effacé. Normalement, `myisampack` se termine avec une erreur s'il découvre que le fichier `tbl_name.TMD` existe. Avec `--force` , `myisampack` reprendra le travail.

`--join=big_tbl_name` , `-j big_tbl_name`

Rassemble toutes les tables indiquées dans la ligne de commande dans une seule table appelée `big_tbl_name` . Toutes les tables qui seront combinées **doivent** être identiques (mêmes noms de colonnes, mêmes types, mêmes index, etc.)

`--packlength=#` , `-p #`

Spécifie la taille de stockage de la longueur de ligne, en octets. Cette valeur doit être 1, 2, ou 3. (`myisampack` stocke toutes les lignes avec des pointeurs de lignes de 1, 2 ou 3 octets. Dans les cas normaux, `myisampack` peut déterminer la taille correcte avant de compresser le fichier, mais il peut aussi se rendre compte durant le processus qu'une autre taille aurait été plus appropriée, ou plus courte. Dans ce cas, `myisampack` va imprimer une note pour que vous le sachiez lors de la prochaine compression du même fichier.

`--silent` , `-s`

Mode silencieux. Seules les erreurs seront affichées.

`--test` , `-t`

Ne compresse pas la table, mais teste juste la compression.

`--tmp_dir=path` , `-T path`

Utilise le dossier indiqué comme dossier pour les fichiers temporaires.

`--verbose` , `-v`

Mode détaillé. Toutes les informations sur la progression de la compression seront affichées.

`--version` , `-V`

Affiche la version et quitte.

`--wait` , `-w`

Attend et reessaie, si la table était déjà en cours d'utilisation. Si le serveur `mysqld` a été démarré avec l'option `--skip-external-locking` , ce n'est pas une bonne idée d'appeler `myisampack` , car la table risque d'être modifiée durant la compression.

La séquence de commande illustre la session de compression :

```
 shell> ls -l station.*
-rw-rw-r--   1 monty    my         994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

 shell> myisamchk -dvv station

 MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-02-02  3:06:43
Data records:             1192  Deleted blocks:             0
Datafile: Parts:          1192  Deleted data:               0
Datafile pointer (bytes):     2  Keyfile pointer (bytes):     2
Max datafile length:   54657023  Max keyfile length:   33554431
Recordlength:              834
Record format: Fixed length

 table description:
Key Start Len Index   Type                     Root  Blocksize    Rec/key
1   2     4   unique  unsigned long            1024       1024          1
2   32    30  multip. text                    10240       1024          1
```

```
 Field Start Length Type
1     1     1
2     2     4
3     6     4
4     10    1
5     11    20
6     31    1
7     32    30
8     62    35
9     97    35
10    132   35
11    167   4
12    171   16
13    187   35
14    222   4
15    226   16
16    242   20
17    262   20
18    282   20
19    302   30
20    332   4
21    336   4
22    340   1
23    341   8
24    349   8
25    357   8
26    365   2
27    367   2
28    369   4
29    373   4
30    377   1
31    378   2
32    380   8
33    388   4
34    392   4
35    396   4
36    400   4
37    404   1
38    405   4
39    409   4
40    413   4
41    417   4
42    421   4
43    425   4
44    429   20
45    449   30
46    479   1
47    480   1
48    481   79
49    560   79
50    639   79
51    718   79
52    797   8
53    805   1
54    806   1
55    807   20
56    827   4
57    831   4

 shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

 normal:     20  empty-space:      16  empty-zero:        12  empty-fill:  11
pre-space:   0  end-space:        12  table-lookups:      5  zero:         7
Original trees:  57  After join: 17
- Compressing file
87.14%

 shell> ls -l station.*
-rw-rw-r--   1 monty    my         127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

 shell> myisamchk -dvv station

 MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-04-17 19:04:26
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     1
Max datafile length:   16777215  Max keyfile length:     131071
Recordlength:               834
Record format: Compressed

 table description:
Key Start Len Index   Type                        Root  Blocksize    Rec/key
1   2     4   unique  unsigned long              10240       1024          1
2   32    30  multip. text                       54272       1024          1
```

```
 Field Start Length Type                         Huff tree  Bits
1     1     1      constant                             1     0
2     2     4      zerofill(1)                          2     9
3     6     4      no zeros, zerofill(1)                2     9
4     10    1                                           3     9
5     11    20     table-lookup                         4     0
6     31    1                                           3     9
7     32    30     no endspace, not_always              5     9
8     62    35     no endspace, not_always, no empty    6     9
9     97    35     no empty                             7     9
10    132   35     no endspace, not_always, no empty    6     9
11    167   4      zerofill(1)                          2     9
12    171   16     no endspace, not_always, no empty    5     9
13    187   35     no endspace, not_always, no empty    6     9
14    222   4      zerofill(1)                          2     9
15    226   16     no endspace, not_always, no empty    5     9
16    242   20     no endspace, not_always              8     9
17    262   20     no endspace, no empty                8     9
18    282   20     no endspace, no empty                5     9
19    302   30     no endspace, no empty                6     9
20    332   4      always zero                          2     9
21    336   4      always zero                          2     9
22    340   1                                           3     9
23    341   8      table-lookup                         9     0
24    349   8      table-lookup                        10     0
25    357   8      always zero                          2     9
26    365   2                                           2     9
27    367   2      no zeros, zerofill(1)                2     9
28    369   4      no zeros, zerofill(1)                2     9
29    373   4      table-lookup                        11     0
30    377   1                                           3     9
31    378   2      no zeros, zerofill(1)                2     9
32    380   8      no zeros                             2     9
33    388   4      always zero                          2     9
34    392   4      table-lookup                        12     0
35    396   4      no zeros, zerofill(1)               13     9
36    400   4      no zeros, zerofill(1)                2     9
37    404   1                                           2     9
38    405   4      no zeros                             2     9
39    409   4      always zero                          2     9
40    413   4      no zeros                             2     9
41    417   4      always zero                          2     9
42    421   4      no zeros                             2     9
43    425   4      always zero                          2     9
44    429   20     no empty                             3     9
45    449   30     no empty                             3     9
46    479   1                                          14     4
47    480   1                                          14     4
48    481   79     no endspace, no empty               15     9
49    560   79     no empty                             2     9
50    639   79     no empty                             2     9
51    718   79     no endspace                         16     9
52    797   8      no empty                             2     9
53    805   1                                          17     1
54    806   1                                           3     9
55    807   20     no empty                             3     9
56    827   4      no zeros, zerofill(2)                2     9
57    831   4      no zeros, zerofill(1)                2     9
```

Les informations affichées par `myisampack` sont décrites ici :

`normal`
: Le nombre de colonnes pour lesquelles aucune compression n'est utilisée.

`empty-space`
: Le nombre de colonnes dont les valeurs ne contiennent que des octets : elles n'occuperont plus qu'un octet.

`empty-zero`
: Le nombre de colonnes dont les valeurs ne contiennent que des zéros : elles n'occuperont plus qu'un octet.

`empty-fill`
: Le nombre de colonnes de type entier qui n'occupent pas la totalité de l'espace de leur type. Elles seront réduites en taille (par exemple, une colonne de type `INTEGER` sera transformée en `MEDIUMINT`).

`pre-space`
: Le nombre de colonnes de nombres à virgule flottante qui ont des valeurs stockées avec des espaces initiaux. Dans ce cas, chaque valeur va contenir le nombre d'espace initiaux.

`end-space`
: Le nombre de colonnes qui ont de nombreux espaces terminaux. Dans ce cas, chaque valeur va contenir un compte du nombre d'espaces terminaux.

`table-lookup`
: La colonne n'a que quelques valeurs différentes, qui seront converties en une

colonne de type `ENUM` avant une compression de type Huffman.

`zero`
: Le nombre de colonnes pour lesquelles toutes les valeurs sont zéro.

`Original trees`
: Le nombre initial d'arbres Huffman.

`After join`
: Le nombre d'arbres Huffman distincts obtenus après avoir joint les arbres pour économiser de l'espace d'entête.

Après la compression d'une table, `myisamchk -dvv` affiche des informations supplémentaires pour chaque champ :

`Type`
: Le type de fichier peut contenir les informations suivantes :

  `constant`
  : Toutes les lignes ont la même valeur.

  `no endspace`
  : Ne stocke pas les espaces finaux.

  `no endspace` , `not_always`
  : Ne stocke pas les espaces finaux et ne compresse pas les espaces finaux pour toutes les valeurs.

  `no endspace` , `no empty`
  : Ne stocke pas les espaces finaux. Ne stocke pas les valeurs vides.

  `table-lookup`
  : La colonne a été convertie en `ENUM` .

  `zerofill(n)`
  : Les `n` chiffres significatifs sont toujours 0, et n'ont pas été stockés.

  `no zeros`
  : Ne stocke pas les zéros.

  `always zero`
  : Les valeurs 0 sont stockées sur un octet.

`Huff tree`
: L'arbre Huffman associé au champ.

`Bits`
: Le nombre de bits utilisés par l'arbre Huffman.

Après la compression de `pack_isam` / `myisampack` vous devez exécuter la commande `isamchk` / `myisamchk` pour recréer l'index. A ce moment, vous pouvez aussi trier les blocs d'index et créer des statistiques nécessaires pour l'optimiseur MySQL :

```
shell> myisamchk -rq --sort-index --analyze tbl_name.MYI
```

Une procédure similaire s'applique aux tables `ISAM` . Après avoir utilisé `pack_isam` , utilisez `isamchk` pour recréer les index :

```
shell> isamchk -rq --sort-index --analyze tbl_name.ISM
```

Après avoir installé la table compressée dans un dossier de données MysQL, vous devez exécutez la commande `mysqladmin flush-tables` pour forcer `mysqld` à utiliser cette nouvelle table.Si vous voulez décompresser une table compressée, vous pouvez le faire avec l'option `--unpack` de la commande `isamchk` ou `myisamchk` .

8.3 mysql , l'outil en ligne de commande

`mysql` est un simple script SQL (qui exploite `GNU readline`). Il supporte une utilisation interactive et non−interactive. Lorsqu'il est utilisé interactivement, les résultats des requêtes sont présentés sous la forme d'une table au format ASCII. Lorsqu'il est utilisé non−interactivement, par exemple, comme filtre, le résultat est fourni au format de liste avec séparation par tabulation (le format d'affichage peut être modifié en utilisant les options de ligne de commande).

Si vous avez des problèmes liés à des insuffisances de mémoire avec le client, utilisez l'option `--quick` ! Cela force `mysql` à utiliser `mysql_use_result()` plutôt que `mysql_store_result()` pour lire les résultats.

Utiliser `mysql` est très simple. Il suffit de le démarrer comme ceci :

```
shell> mysql db_name
```

ou :

```
shell> mysql --user=user_name --password=your_password db_name
```

Tapez une commande SQL, puis terminez−la avec `';'` , `'\g'` ou `'\G'` , et finissez avec entrée.

Vous pouvez exécuter un script avec :

```
shell> mysql db_name < script.sql > output.tab
```

`mysql` supporte les options suivantes :

`--help` , `-?`
: Affiche cette aide et quitte.

`--batch` , `_B`
: Affiche les résultats avec une tabulation comme résultat, et chaque ligne avec une nouvelle ligne. N'utilise pas l'historique.

`--character-sets-dir=path`
: Le dossier où les jeux de caractères sont créés. Le jeu de caractères utilisé pour les données et le stockage .

`--compress` , `_C`
: Utilise la compression avec le protocole client / serveur.

`--database=db_name` , `-D db_name`
: La base de données à utiliser. C'est particulièrement pratique dans le fichier d'options `my.cnf` .

`--debug[=debug_options]` , `-# [debug_options]`
: Génère un log de débogage. La chaîne `debug_options` vaut souvent `'d:t:o,file_name'` . Par défaut, la valeur est `'d:t:o,/tmp/mysql.trace'` .

`--debug-info` , `-T`
: Affiche des informations de débogage lorsque le programme se termine.

`--default-character-set=charset`
: Configure le jeu de caractères par défaut. Le jeu de caractères utilisé pour les données et le stockage .

`--execute=statement` , `-e statement`
: Exécute une commande et quitte. Le résultat est au format de l'option `--batch` .

`--force` , `f`
: Continue même si vous recevez une erreur SQL.

`--host=host_name` , `-h host_name`
: Connexion avec l'hôte indiqué.

`--html` , `H`
: Produit un résultat au format HTML.

`--ignore-space` , `i`
: Ignore les espaces après les noms de fonctions. L'effet de cette commande est décrit dans la discussion sur `IGNORE_SPACE` de la section Le mode SQL du serveur .

`--local-infile[={0|1}]`
: Active ou désactive la possibilité d'utiliser la commande `LOCAL` pour `LOAD DATA INFILE` . Sans valeur, cette option active `LOCAL` . Elle peut être configuré avec `--local-infile=0` ou `--local-infile=1` pour explicitement activer ou désactiver `LOCAL` . Activer `LOCAL` n'a pas d'effet si le serveur ne le supporte pas de son coté.

`--named-commands` , `-G`
: Les commandes nommées sont **activées** . Utilisez la forme `\*` uniquement, ou utilisez les commandes nommées au début d'une ligne se terminant par un point−virgule (`';'`). Depuis la version 10.9, le client démarre avec cette option **activée** par défaut. Avec l'option `-g` , le format long des commandes va continuer à fonctionner.

--no-auto-rehash , -A

Pas de rehashage automatique. Cette option fait que mysql se lance plus rapidement, mais vous devez utiliser la commande rehash si vous voulez utiliser la completion de nom de tables.

--no-beep , -b

Ne fait pas de bip, lorsqu'une erreur survient.

--no-named-commands , -g

Les commandes nommées sont désactivées. Utilisez uniquement la forme * ou bien utilisez les commandes nommées en début de ligne, et terminez la avec un point−virgule (';'). Depuis MySQL 3.23.22, mysql démarre avec cette option **activée** par défaut! Cependant, même si cette avec cette option, les formats de commandes longues fonctionneront sur la première ligne.

--no-pager

Désactive le système de page, et affiche directement dans la sortie standard. Plus de détails dans la section Commandes mysql .

--no-tee

Désactive le fichier de sortie. Voyez l'aide interactive (\h). Plus de détails dans la section Commandes mysql .

--one-database , O

Ne modifie que la base par défaut. C'est pratique pour éviter les modifications dans les autres bases dans le fichier de log.

--pager[=command]

Type d'affichage. Par défaut, la variable d'environnement ENV vaut PAGER . Les pagineurs valides sont less , more , cat [> filename] , etc. Voyez l'aide interactive (\h). Cette option n'est pas fonctionnelle en mode batch. Les pagineurs ne fonctionnent qu'avec Unix. Plus de détails dans la section Commandes mysql .

--password[=password] , -p[password]

Le mot de passe utilisé lors de la connexion sur le serveur. S'il n'est pas donné en ligne de commande, il sera demandé interactivement. Notez que si vous utilisez la forme courte -p , vous **ne devez pas** laisser d'espace entre l'option et le mot de passe.

--port=port_num , -P port_num

Le numéro de port TCP/IP pour la connexion.

--prompt=format_str

Modifie le format de l'invite de commandes (prompt). Par défaut, c'est mysql> . Les séquences spéciales sont présentées dans la section Commandes mysql .

--protocol={TCP | SOCKET | PIPE | MEMORY}

Spécifie le protocole de connexion à utiliser. Nouveau en MySQL version 4.1.

--quick , -q

Ne met pas en cache le résultat, et l'affiche ligne par ligne. C'est plus lent pour le serveur, si le résultat est interrompu. N'utilise pas le fichier d'historique.

--raw , -r

Ecrit les valeurs des colonnes sans les conversions de protections. Utilisé en mode --batch

--reconnect

Si la connexion est perdue, essaie de se reconnecter automatiquement au serveur, juste une fois. Pour supprimer la reconnexion automatique, utilisez --skip-reconnect . Nouveau en MySQL 4.1.0.

--safe-updates , --i-am-a-dummy , -U

N'autorise que les commandes UPDATE et DELETE qui utilisent des clés. Voir plus bas pour des informations sur cette option. Vous pouvez annuler cette option si vous l'avez dans le fichier d'option my.cnf en utilisant la syntaxe --safe-updates . Voyez la section Conseils mysql pour plus d'informations sur cette option.

--silent , -s

Mode très silencieux.

----skip-column-names , -N

N'écrit pas les noms de colonnes dans les résultats.

`--skip-line-numbers` , `-L`
: N'écrit pas les numéros de lignes dans les erreurs. Très pratique lorsque vous voulez comparer des résultats qui incluent des messages d'erreurs.

`--socket=path` , `-S path`
: Le fichier de socket à utiliser pour la connexion.

`--table` , `-t`
: Affichage au format de table. C'est le mode par défaut pour le mode non−batch.

`--tee=file_name`
: Ajoute tout dans le fichier de sortie. Voyez l'aide interactive (\h). Ne fonctionne pas en mode batch. Cette option est détaillée dans Commandes `mysql` .

`--unbuffered` , `-n`
: Vide le buffer de requête après chaque requête.

`--user=user_name` , `-u user_name`
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur Unix courant.

`--verbose` , `-v`
: Affichage plus détaillé (`-v -v -v` indique le format d'affichage de table).

`--version` , `-V`
: Affiche la version et quitte.

`--vertical` , `-E`
: Affiche le résultat d'une requête verticalement. Sans cette option, vous pouvez aussi obtenir ce format en terminant votre requête avec `\G` .

`--wait` , `-w`
: Attend et retente si la connexion s'interrompt, au lieu de quitter.

`--xml` , `-X`
: Affiche le résultat au format XML.

Vous pouvez aussi spécifier les variables suivantes avec la syntaxe `--var=option` :

`connect_timeout`
: Nombre de secondes avant que la connexion n'expire. Valeur par défaut : 0.

`max_allowed_packet`
: Taille maximale du paquet de communication avec le serveur. Valeur par défaut : 16777216.

`max_join_size`
: Limite automatique pour les commandes de jointure avec l'option `--i-am-a-dummy` . Valeur par défaut : 1 000 000 (un million).

`net_buffer_length`
: Buffer pour les communications TCP/IP et socket. Valeur par défaut : 16 ko.

`select_limit`
: Limite automatique pour les commandes `SELECT` avec l'option `--i-am-a-dummy` Valeur par défaut : 1000.

Il est aussi possible de configurer les variables en utilisant `--set-variable=var_name=value` ou la syntaxe `-O var_name=value` . Cependant, cette syntaxe est obsolète depuis MySQL 4.0.

Sous Unix, le client `mysql` écrit l'historique des requêtes dans un fichier. Par défaut, le fichier de requêtes s'appelle `.mysql_history` , et il est créé dans votre dossier racine. Pour spécifier un fichier différent, modifiez la variable d'environnement `MYSQL_HISTFILE` .

Si vous ne voulez pas entretenir un fichier d'historiques, supprimez `.mysql_history` s'il existe, puis utiliser l'une des techniques suivantes :

- Donnez à `MYSQL_HISTFILE` la valeur de `/dev/null` . Pour que cela soit pris en compte à chaque redémarrage, ajoutez cette ligne à votre script de démarrage.
- Faites un lien symbolique depuis `.mysql_histfile` vers `/dev/null` :

  ```
  shell> ln -s /dev/null $HOME/.mysql_history
  ```

  Il suffira de faire cela une seule fois.

8.3.1 Commandes mysql

mysql envoie des requêtes SQL que vous avez saisie au serveur, pour exécution. Il y a aussi des commandes que le client mysql interprête. Si vous tapez ' help ' en ligne de commande, mysql va afficher les commandes qu'il supporte :

```
mysql> help

MySQL commands:
?         (\h)    Synonym for `help'.
clear     (\c)    Clear command.
connect   (\r)    Reconnect to the server.
                  Optional arguments are db and host.
delimiter (\d)    Set query delimiter.
edit      (\e)    Edit command with $EDITOR.
ego       (\G)    Send command to mysql server,
                  display result vertically.
exit      (\q)    Exit mysql. Same as quit.
go        (\g)    Send command to mysql server.
help      (\h)    Display this help.
nopager   (\n)    Disable pager, print to stdout.
notee     (\t)    Don't write into outfile.
pager     (\P)    Set PAGER [to_pager].
                  Print the query results via PAGER.
print     (\p)    Print current command.
prompt    (\R)    Change your mysql prompt.
quit      (\q)    Quit mysql.
rehash    (\#)    Rebuild completion hash.
source    (\.)    Execute an SQL script file.
                  Takes a file name as an argument.
status    (\s)    Get status information from the server.
system    (\!)    Execute a system shell command.
tee       (\T)    Set outfile [to_outfile].
                  Append everything into given outfile.
use       (\u)    Use another database.
                  Takes database name as argument.
```

Les commandes edit , nopager , pager et system ne fonctionnent que sous Unix.

La commande status donne des détails sur la connexion et le serveur utilisés. Si vous fonctionnez en mode --safe-updates , status va aussi afficher les valeurs des variables de mysql qui affectent vos requêtes.

Pour enregistrer les requêtes et leur résultat, utilisez la commande tee . Toutes les données affichées à l'écran seront ajoutées à un fichier donné. Cela peut être très pratique pour déboguer. Vous pouvez activer cette fonctionnalité en ligne de commande, avec l'option --tee , ou interactivement avec la commande tee . Le fichier tee peut être désactivé interactivement avec la commande notee . Exécuter tee à nouveau ré−active le log. Sans paramètre, le fichier précédent sera utilisé. Notez que tee envoie les requêtes dans le fichier après chaque commande, juste avant que mysql ne l'affiche.

La lecture et la recherche dans les résultats de requêtes en mode interactif est possible en utilisant les programmes Unix less , more , ou tout autre programme similaire, avec l'option --pager . Si vous ne spécifier par de valeur d'option, mysql regarde la valeur de la variable d'environnement PAGER , et utilise ce pager. Vous pouvez aussi l'activer interactivement avec la commande pager et la désactiver avec nopager . La commande prend un argument optionnel : s'il est fournit, le programme de pager indiqué sera utilisé. Sinon, le pager sera celui indiqué en ligne de commande, ou stdout si aucun pager n'était indiqué.

La pagination de sortie ne fonctionne que sous Unix, car elle utilise la fonction popen() , qui n'existe pas sous Windows. Pour Windows, la commande tee peut être utilisée pour sauver le résultat, même si ce n,est pas aussi pratique que pager pour naviguer dans le résultat.

Quelques conseils avec la commande pager :

- Vous pouvez l'utiliser pour écrire les résultats dans un fichier :

  ```
  mysql> pager cat > /tmp/log.txt
  ```

Vous pouvez lui passer les options que le page comprendra :

```
mysql> pager less -n -i -S
```

- Dans le précédent exemple, notez l'option `-S` . Vous la trouverez pratique pour naviguer dans des résultats très larges. Parfois, un résultat très large est difficile à lire à l'écran. L'option `-S` de `less` rend le résultat plus lisible, car vous pouvez aussi scroller horizontalement, avec les flèches de droite et de gauche. Vous pouvez aussi utiliser interactivement `-S` dans `less` pour activer ou désactiver la navigation horizontale. Pour plus d'informations, voyez le manuel de `less` :

```
shell> man less
```

- Vous pouvez spécifier des commandes de pages très complexe :

```
mysql> pager cat | tee /dr1/tmp/res.txt \
        | tee /dr2/tmp/res2.txt | less -n -i -S
```

Dans cet exemple, la commande va envoyer les résultats de la commande dans deux fichiers différents, dans deux dossiers différents, placés sur deux devis `/dr1` et `/dr2` , mais affichera toujours le résultat à l'écran via `less` .

Vous pouvez aussi combiner les deux fonctions ci−dessus : activer le `tee` , spécifier le `pager` '`less`' et vous serez capable de naviguer dans les résultats avec le `less` Unix, tout en enregistrant tous les résultats dans un fichier. La différence entre le `tee` d'Unix utilisé avec le `pager` et le `tee` intégré du client `mysql` , est que le `tee` intégré fonctionne même si vous n'avez pas de `tee` Unix disponible. Le `tee` enregistre tout ce qui est affiché à l'écran, alors que le `tee` Unix utilisé avec `pager` n'en note pas autant. Enfin, le `tee` interactif est plus facile à activer et désactiver, lorsque vous souhaitez enregistrer un résultat dans un fichier, mais que vous voulez désactiver cette fonctionnalité à d'autres moments.

Depuis MySQL version 4.0.2, il est possible de modifier l'invite de commande de `mysql` . La chaîne de définition de l'invite de commande accepte les séquences suivantes :

Option	Description
\v	version de mysqld
\d	database en cours
\h	hôte MySQL
\p	port de connexion
\u	nom d'utilisateur
\U	Identifiant complet username@host
\\	'\'
\n	nouvelle ligne
\t	tabulation
\	espace
_	espace
\R	heure 24h (0−23)
\r	heure 12h (1−12)
\m	minutes
\y	année sur deux chiffres
\Y	année sur quatre chiffres
\D	format de date complet
\s	secondes
\w	jour de la semaine en trois lettres (Mon, Tue, ...)
\P	am/pm
\o	mois au format numérique
\O	mois en trois lettres (Jan, Feb, ...)
\c	compteur du nombre de commande

'`\`' suivi de n'importe quelle lettre représente la lettre littéralement.

Si vous spécifiez une commande `prompt` sans argument, `mysql` utilisera l'invite de commande par défaut de `mysql>` .

Vous pouvez modifier l'invite de commande comme ceci :

- Variable d'environnement Vous pouvez utiliser la variable d'environnement `MYSQL_PS1` , en lui donnant la chaîne d'invite. Par exemple :

```
shell> export MYSQL_PS1="(\u&#64;\h) [\d]> "
```

- Utiliser le fichier d'optionsVous pouvez configurer l'invite de commandes `prompt` dans le fichier d'options MySQL, comme `/etc/my.cnf` ou `.my.cnf` , dans le groupe `mysql` . Par exemple :

  ```
  [mysql]
  prompt=(\\u&#64;\\h) [\\d]>\\_
  ```

  Dans cet exemple, notez que les anti−slash sont doublés. Si vous configurez l'invite de commande `prompt` dans un fichier d'options, il est recommandé de doubler les anti−slash, lorsque vous utilisez des options. Il y a des recoupements entre les séquences protégées et les options. Ces séquences sont listées dans Utiliser les fichiers d'options . Ce recoupement peut vous causer des problèmes avec des anti−slashs uniques. Par exemple, `\s` sera interprété comme un espace, plutôt que comme le nombre de secondes. La valeur suivante montre comment définir une invite avec l'heure au format `HH:MM:SS>` :

  ```
  [mysql]
  prompt="\\r:\\m:\\s> "
  ```
- Utilisez une option de ligne de commande

  Vous pouvez utiliser l'option `--prompt` dans la commande `mysql` . Par exemple :

  ```
  shell> mysql --prompt="(\u&#64;\h) [\d]> "
  (user&#64;host) [database]>
  ```
- Interactivement

  Vous pouvez aussi utiliser la commande `prompt` (ou `\R`) depuis le client pour modifier interactivement l'invite de commande. Par exemple :

  ```
  mysql> prompt (\u&#64;\h) [\d]>\_
  PROMPT set to '(\u&#64;\h) [\d]>\_'
  (user&#64;host) [database]>
  (user&#64;host) [database]> prompt
  Returning to default PROMPT of mysql>
  mysql>
  ```

8.3.2 Comment exécuter des commandes SQL depuis un fichier texte

Le client `mysql` peut être utilisé en mode interactif comme ceci :

```
shell> mysql database
```

Toutefois, il est aussi possible de rassembler les commandes SQL dans un fichier, et d'indiquer à `mysql` de lire les commandes dans ce fichier. Pour faire cela, créez un fichier texte `fichier_texte` qui contient les commandes SQL que vous souhaitez exécuter. Puis, exécutez ce fichier avec `mysql` comme ceci :

```
shell> mysql database < fichier_texte
```

Vous pouvez aussi démarrer votre fichier texte avec la commande `USE nom_base` . Dans ce cas, il n'est pas nécessaire de spécifier le nom de la base de données dans la ligne de commande :

```
shell> mysql < fichier_texte
```

Si vous avez déjà démarré le client `mysql` , vous pouvez exécuter un script SQL en utilisant la commande `source` :

```
mysql> source nom_fichier;
```

Pour plus d'informations sur le mode batch, consultez Utiliser `mysql` en mode batch .

8.3.3 Conseils avec mysql

Cette section décrit certaines techniques qui vous aideront à utiliser `mysql` plus efficacement.

8.3.3.1 Affichage des résultats verticalement

Certaines requêtes sont bien plus lisibles une fois affichées verticalement, au lieu de horizontalement. Par exemple, si la taille du texte est bien plus grande que la largeur de l'écran, ou qu'il y a des retours à la ligne, le format vertical est préférable :

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 LIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty&#64;no.spam.com
  mail_to: "Thimble Smith" <tim&#64;no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:

Thimble> Hi.  I think this is a good idea.  Is anyone familiar
Thimble> with UTF-8 or Unicode? Otherwise, I'll put this on my
Thimble> TODO list and see what happens.

 Yes, please do that.

 Regards,
Monty
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

8.3.3.2 Utilisation de l'option --safe-updates

Pour les débutants, une option de démarrage pratique est `--safe-updates` (ou `--i-am-a-dummy` , qui a le même effet). Cette option a été introduite en MySQL 3.23.11. Elle est pratique si vous avec émis des commandes `DELETE FROM tbl_name` mais que vous avez oublié la clause `WHERE` . Normalement, une telle commande va effacer toutes les lignes de la table. Avec `--safe-updates` , vous pouvez effacer uniquement les lignes dont vous spécifiez les valeurs de clé pour les identifier. Cela évite les accidents.

Lorsque vous utilisez l'option `--safe-updates` , `mysql` émet l'alerte suivante lorsqu'il se connecte à MySQL :

```
SET SQL_SAFE_UPDATES=1,SQL_SELECT_LIMIT=1000,
   SQL_MAX_JOIN_SIZE=1000000;
```

Syntaxe de `SET` .La commande `SET` a l'effet suivant :

- Vous n'êtes pas autorisés à exécuter de commandes `UPDATE` et `DELETE` à moins que vous ne spécifiez une contrainte de clé dans la clause `WHERE` où que vous fournissiez une clause `LIMIT` , ou les deux. Par exemple :
  ```
  UPDATE tbl_name SET not_key_column=# WHERE key_column=#;

  UPDATE tbl_name SET not_key_column=# LIMIT 1;
  ```
- Tous les résultats importants de `SELECT` sont automatiquement limités à 1000, à moins que la commande n'inclut la clause `LIMIT` .
- Les commandes `SELECT` multi−tables qui devront étudier plus d'un million de lignes sont annulées.

Pour spécifier des limites autres que 1000 et 1000000, vous pouvez remplacer les maxima avec `--select_limit` et `--max_join_size` :

```
shell> mysql --safe-updates --select_limit=500 --max_join_size=10000
```

8.3.3.3 Désactiver la reconnexion automatique de mysql

Si le client `mysql` perd la connexion au serveur durant l'envoi d'une requête, il va immédiatement et automatiquement essayer de se reconnecter une fois au serveur, puis essayer d'envoyer à nouveau la requête. Toutefois, même si `mysql` réussit à se reconnecter, l'ancienne connexion a été fermée, et tous les objets temporaires ont été perdus : tables temporaires, configuration en auto_commit, variables utilisateur et de session. Ce comportement peut se révéléer dangeureux pour vous, comme dans l'exemple suivant, où le serveur est stoppé, puis relancé sans que vous le sachiez :

```
 mysql> SET &#64;a=1;
Query OK, 0 rows affected (0.05 sec)

 mysql> INSERT INTO t VALUES(&#64;a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    1
Current database: test

 Query OK, 1 row affected (1.30 sec)

 mysql> SELECT * FROM t;
+------+
| a    |
+------+
| NULL |
+------+
1 row in set (0.05 sec)
```

La variable utilisateur `@a` a été perduée, et lors de la reconnexion, elle est indéfinie. S'il est important que `mysql` génère une erreur lors de la perte de connexion, vous pouvez lancez le client `mysql` avec l'option `--skip-reconnect` .

8.4 mysqladmin , administration d'un serveur MySQL

`mysqladmin` est un utilitaire pour exécuter des commandes d'administration. Vous pouvez l'utiliser pour vérifier la configuration du serveur, créer et effacer des bases, etc.

La syntaxe de `mysqladmin` est :

```
shell> mysqladmin [OPTIONS] command [command-option] command ...
```

Le `mysqladmin` actuel supporte les commandes suivantes :

create databasename
: Crée une nouvelle base.

drop databasename
: Efface une base et toutes ces tables.

extended−status
: Affiche un message de statut du serveur très complet.

flush−hosts
: Vide tous les hôtes mis en cache.

flush−logs
: Vide de la mémoire tous les logs.

flush−privileges
: Recharger les tables de droits (identique à la commande reload).

flush−status
: Remet à zéro les variables de statut.

flush−tables
: Vide de la mémoire toutes les tables.

flush−threads
: Vide les threads de cache. Nouveau en MySQL 3.23.16.

kill id,id,...
: Termine un thread MySQL.

password new−password
: Spécifie un nouveau mot de passe. Modifie l'ancien mot de passe en `new-password` pour le compte que vous utilisez lors de la connexion avec `mysqladmin` .

ping
: Vérifie si mysqld fonctionne ou pas.

processlist
: Affiche la liste des processus du serveur. Cela revient à la commande `SHOW PROCESSLIST` . Si `--verbose` est utilisé, le résultat est le même que `SHOW FULL PROCESSLIST` .

reload
: Recharge les tables de droits.

refresh
: Vide de la mémoire toutes les tables, puis ferme et réouvre les fichiers de logs.

shutdown
: Eteind le serveur.

slave−start
: Démarre l'esclave de réplication.

status
: Affiche le message de statut court du serveur.

slave−stop
: Eteind l'esclave de réplication.

variables
: Affiche les variable disponibles.

version
: Affiche la version du serveur.

Toutes les commandes peuvent être réduites à leur préfixe simple. Par exemple :

```
 shell> mysqladmin proc stat
+----+-------+-----------+----+-------------+------+-------+------+
| Id | User  | Host      | db | Command     | Time | State | Info |
+----+-------+-----------+----+-------------+------+-------+------+
| 6  | monty | localhost |    | Processlist | 0    |       |      |
+----+-------+-----------+----+-------------+------+-------+------+
Uptime: 10077  Threads: 1  Questions: 9  Slow queries: 0
Opens: 6 Flush tables: 1  Open tables: 2
Memory in use: 1092K  Max memory used: 1116K
```

La commande `mysqladmin status` liste les colonnes suivantes :

Uptime
: Nombre de secondes de vie du serveur MySQL.

Threads
: Nombre de threads actifs (clients).

Questions
: Nombre de questions reçu des clients depuis le démarrage de `mysqld` .

Slow queries
: Nombre de requêtes qui ont pris plus de `long_query_time` secondse. Le log des requêtes lentes .

Opens
: Combien de tables sont ouvertes par `mysqld` .

Flush tables
: Nombre de commandes `flush ...` , `refresh` et `reload` .

Open tables
: Nombre de tables qui sont ouvertes actuellement.

Memory in use
: Mémoire allouée directement par `mysqld` (uniquement disponible si MySQL a été compilé avec l'option −−with−debug=full).

Maximum memory used
: Maximum de mémoire allouée directement par `mysqld` (uniquement disponible si MySQL a été compilé avec l'option −−with−debug=full).

Si vous exécutez `mysqladmin shutdown` en vous connectant à un serveur local, avec un fichier de socket Unix, `mysqladmin` va attendre que le fichier de processus du serveur soit supprimé, pour s'assurer que le serveur est bien arrété.

`mysqladmin` supporte les options suivantes :

−−help, −?
: Affiche le message d'aide et quitte.

−−character−sets−dir=path
: Le dossier où les jeux de caractères sont stockés. Le jeu de caractères utilisé pour les données et le stockage .

−−compress, −C
: Compresse toutes les informations entre le client et le serveur, si les deux le supporte.

−−count=#, −c #
: Le nombre d'itération à faier. Cela fonctionne uniquement avec `--sleep` (`-i`).

−−debug[=debug_options], −# [debug_options]
: Écrit un log de débogage. La chaîne `debug_options` est souvent `'d:t:o,file_name'` . La valeur par défaut est `'d:t:o,/tmp/mysqladmin.trace'` .

−−force, −f
: Ne demande pas de confirmation pour la commande `drop database` . Avec des commandes multiples, continue même si une erreur survient.

−−host=host_name, −h host_name
: Connexion au serveur MYSQL avec le nom d'hôte donné.

−−password[=password], −p[password]
: Le mot de passe utilisé lors de la connexion sur le serveur. S'il n'est pas donné en ligne de commande, il sera demandé interactivement. Notez que si vous utilisez la forme courte `-p` , vous **ne devez pas** laisser d'espace entre l'option et le mot de passe.

−−port=port_num, −P port_num
: Le numéro de port TCP/IP pour la connexion.

−−protocol={TCP | SOCKET | PIPE | MEMORY}
: Spécifie le protocole de connexion à utiliser. Nouveau en MySQL version 4.1.

−−relative, −r
: Affiche la différence entre la valeur courante et la valeur précédente, lorsqu'utilisé avec l'option `-i` . Actuellement, cette option fonctionne avec la commande `extended-status` .

−−silent, −s
: Mode très silencieux.

−−sleep=delay, −i delay
: Exécute les commandes encore et encore, avec `delay` secondes entre deux.

−−socket=path, −S path
: Le fichier de socket à utiliser pour la connexion.

−−user=user_name, −u user_name
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur Unix courant.

−−verbose, −v
: Affichage plus détaillé (−v −v −v indique le format d'affichage de table).

−−version, −V
: Affiche la version et quitte.

−−vertical, −E

Print output vertically. This is similar to `--relative` , but prints output vertically.

−−wait[=#], −w[#]
Si la connexion n'a pu être établie, attent et ressaie au lieu d'abandonner. Si une valeur est spécifiée, elle indique le nombre de tentatives. La valeur par défaut est 1 fois.

Vous pouvez aussi configurer ces options avec la syntaxe `--var_name=value` :

connect_timeout
Le nombre de secondes avant une éxpiration de connexion. (Par défaut, 0.)

shutdown_timeout
Le nombre de seconde d'attente de l'extinction. (Par défaut, 0.)

Il est aussi possible de configurer les variables en utilisant `--set-variable=var_name=value` ou la syntaxe `-O var_name=value` . Cependant, cette syntaxe est obsolète depuis MySQL 4.0.

8.5 mysqlbinlog , Exécuter des requêtes dans le log binaire

Les fichiers de log sont écrits dans un format binaire. Vous pouvez éxaminer le log binaire avec l'utilitaire `mysqlbinlog` . Il est disponible depuis MySQL 3.23.14.

Appelez `mysqlbinlog` comme ceci :

```
shell> mysqlbinlog [options] log-file ...
```

Par exemple, pour afficher le contenu du fichier de log binaire `binlog.000003` , utilisez cette commande :

```
shell> mysqlbinlog binlog.0000003
```

Le résultat est toutes les requêtes contenues dans le fichier de log binaire `binlog.000003` , avec différentes informations (durée de la requête, identifiant du thread qui l'a émise, timestamp d'émission, etc.).Normalement, vous utilisez `mysqlbinlog` pour lire les fichiers de log directement, et les envoyer au serveur MySQL local. Il est aussi possible de lire le fichier binaire sur un serveur distant en utilisant l'option `--read-from-remote-server` . Cependant, c'est une technique abandonnée, car nous préférons rendre plus simple l'utilisation des logs binaires sur un serveur MySQL local.

Lorsque vous lisez des logs binaires distants, les options de connexion peuvent être données pour indiquer comment se connecter au serveur, mais ils sont ignorés à moins que vous ne spécifiez aussi l'option `--read-from-remote-server` . Ces options sont `--host` , `--password` , `--port` , `--protocol` , `--socket` et `--user` .

Vous pouvez aussi utiliser `mysqlbinlog` pour relayer des fichiers de log écrits par un serveur esclave, dans une architecture de réplication. Les logs de relais sont au même format que le log binaire.

Le log binaire est présenté en détails dans la section Le log binaire des mises à jour .

`mysqlbinlog` supporte les optiosn suivantes :

−−help, −?
Affiche cette aide et quitte.

−−database=db_name, −d db_name
Limite les lignes à cette base de données (log local uniquement).

−−force−read, −f
Continue même si vous obtenez une erreur SQL.

−−host=host_name, −h host_name

Lit le log binaire depuis le serveur MySQL distant.

−−local−load=path, −l path
: Prépare les fichiers temporaires destinés aux commandes `LOAD DATA INFILE` dans le dossier spécifié.

−−offset=N, −o N
: Ignore les `N` première lignes.

−−password[=password], −p[password]
: Le mot de passe utilisé lors de la connexion sur le serveur. S'il n'est pas donné en ligne de commande, il sera demandé interactivement. Notez que si vous utilisez la forme courte `-p` , vous **ne devez pas** laisser d'espace entre l'option et le mot de passe.

−−port=port_num, −P port_num
: Le numéro de port TCP/IP pour la connexion.

−−position=N, −j N
: Comment la lecture dans le log binaire à la position `N` .

−−protocol={TCP | SOCKET | PIPE | MEMORY}
: Spécifie le protocole de connexion à utiliser. Nouveau en MySQL version 4.1.

−−read−from−remote−server, −R
: Read the binary log from a MySQL server. Les options de connexion distantes seront ignorées à moins que cette option ne soit donné. Ces options sont `--host` , `--password` , `--port` , `--protocol` , `--socket` et `--user` .

−−result−file=name, −r name
: Export direct vers le fichier spécifié.

−−short−form, −s
: Affiche uniquement les commandes du log, sans les informations supplémentaires.

−−socket=path, −S path
: Le fichier de socket à utiliser pour la connexion.

−−user=user_name, −u user_name
: Le nom d'utilisateur MySQL lors de la connexion à distance.

−−version, −V
: Affiche la version et quitte.

Vous pouvez aussi configurer les variables suivantes avec l'option `--var_name=value` :

open_files_limit
: Spécifie le nombre de pointeurs de fichiers à réserver.

Vous pouvez envoyer le résultat de `mysqlbinlog` vers un client `mysql` avec un pipe : c'est une technique pour restaurer le serveur après un crash (Sauvegardes de base de données) :

```
shell> mysqlbinlog hostname-bin.000001 | mysql
```

ou :

```
shell> mysqlbinlog hostname-bin.[0-9]* | mysql
```

Vous pouvez aussi rediriger le résultat de `mysqlbinlog` dans un fichier texte, modifier ce fichier (supprimer les requêtes qui vous gênent), puis exécuter ces requêtes, depuis le fichier, avec `mysql` . Après edition du fichier, exécutez les commandes qu'il contient comme d'habitude, avec le programme `mysql` .

`mysqlbinlog` dispose de l'option `--position` qui affiche les requêtes du log binaire à partir de la position spécifiée.

Si vous avez plus d'un fichier de log binaire à exécuter sur le serveur MySQL, la méthode sûre est de tout faire avec la même connexion MySQL. Voici la méthode **dangeureuse** :

```
 shell> mysqlbinlog hostname-bin.000001 | mysql # DANGER!!
shell> mysqlbinlog hostname-bin.000002 | mysql # DANGER!!
```

Cela va causer des problèmes si le premier log contient des commandes de type CREATE TEMPORARY TABLE et que le second contient des requêtes d'utilisation de cette table : lorsque le premier mysql termine son exécution, il va détruire la table, et le second va rencontrer des erreurs ``unknown table''. Pour éviter cela, utilisez une seule connexion, surtout si vous utilisez des tables temporaires. Voici deux méthodes possibles :

```
shell> mysqlbinlog hostname-bin.000001 hostname-bin.000002 | mysql
```

La seconde méthode :

```
shell> mysqlbinlog hostname-bin.000001 >  /tmp/statements.sql
shell> mysqlbinlog hostname-bin.000002 >> /tmp/statements.sql
shell> mysql -e "source /tmp/statements.sql"
```

Depuis MySQL 4.0.14, mysqlbinlog peut préparer des requêtes valides pour mysql , afin d'il utilise la commande LOAD DATA INFILE depuis le log binaire. COmme le log contient les données à charger (c'est vrai depuis MySQL 4.0; MySQL 3.23 n'inscrivait pas les données à charger dans le log binaire, et le fichier original était nécessaire pour exécuter le contenu du log binaire), mysqlbinlog va copier ces données dans un fichie temporaire et émettre une commande LOAD DATA INFILE pour que mysql le charge. Le dossier du fichier temporaire est le dossier temporaire par défaut : il peut être modifié avec l'option local-load de mysqlbinlog .

Comme mysqlbinlog convertit les commandes LOAD DATA INFILE en commandes LOAD DATA LOCAL INFILE (c'est à dire qu'il ajoute LOCAL), le client et le serveur que vous utilisez pour traiter les commandes doivent être configuré pour autoriser l'option LOCAL . Problèmes de sécurité avec LOAD DATA LOCAL .

ATTENTION : lorsque vous exécutez mysqlbinlog sur un fichier binaire, il va créer un fichier temporaire pour chaque commande LOAD DATA INFILE . Ces fichiers **ne seront pas** automatiquement effacés, car vous en aurez besoin lorsque vous exécuterez les commandes SQL générées. Il faudra les effacer manuellement lorsque vous n'en aurez plus besoin. Les fichiers portent le nom de temporary-dir/original_file_name-#-# .

Dans le futur, nous allons corriger ce problème, en permettant à mysqlbinlog de se connecter directement au serveur mysqld . Dans ce cas, nous pourrons supprimer tranquillement les fichiers de log, lorsqu'ils auront été utilisés.

Avant MySQL 4.1, mysqlbinlog ne pouvait pas préparer de log SQL pour mysql lorsque le log binaire contenait des requêtes de différents threads, utilisant des tables temporaires de même nom, si les requêtes étaient mélangées. Ceci est corrigé en MySQL 4.1.

8.6 mysqlcc , MySQL Control Center

mysqlcc , MySQL Control Center, est un client portable qui fournit une interface graphique (GUI) au serveur MySQL. Il supporte l'utilisation interactive, y compris la coloration syntaxique et la complétion. Il permet de gérer les tables et bases, et d'administrer le serveur.

Actuellement, mysqlcc fonctionne sur Windows et Linux.

mysqlcc n'est pas inclus avec les distribution MySQL, mais il peut être téléchargé séparément à http://dev.mysql.com/downloads/ . Actuellement, mysqlcc fonctionne sur Windows et Linux.

Lancez mysqlcc en double−cliquant sur son icone en environnement graphique. En ligne de commande, utilisez ceci :

```
shell> mysqlcc [options]
```

mysqlcc supports the following options:

−−help, −?

Affiche cette aide.

−−blocking_queries, −b
: Utilise les requêtes bloquantes.

−−compress, −C
: Utilise la compression avec le protocole client/serveur.

−−connection_name=name, −c name
: Synonyme de `--server` .

−−database=db_name, −d db_name
: Base de données à utiliser. C'est généralement utile dans un fichier d'options.

−−history_size=#, −H #
: Taille de l'historique de la fenêtre de requête.

−−host=host_name, −h host_name
: Hôte de connexion.

−−local−infile[={0|1}]
: Active ou désactive les fonctionnalités `LOCAL` de `LOAD DATA INFILE` . Sans valeur, cette option active `LOCAL` . `LOCAL` peut être spécifié sous la forme `--local-infile=0` ou `--local-infile=1` pour être désactivée ou activée. Activer `LOCAL` n'a pas d'effet si le serveur ne le supporte pas.

−−password[=password], −p[password]
: Le mot de passe lors de la connexion au serveur. Si le mot de passe n'est pas donné en ligne de commande, il vous sera demandé. Notez que si vous utilisez la forme courte `-p` , vous **ne devez pas** mettre d'espace entre l'option et le mot de passe.

−−plugins_path=name, −g name
: Chemin du dossier de module de MySQL Control Center.

−−port=port_num, −P port_num
: Numéro de port TCP/IP pour la connexion.

−−query, −q
: Ouvre une fenêtre de requête au démarrage.

−−register, −r
: Ouvre la fenêtre de dialogue 'Register Server'.

−−server=name, −s name
: Nom de la connexion de MySQL Control Center.

−−socket=path, −S path
: Fichier de socket à utiliser pour la connexion.

−y, −−syntax
: Active la coloration syntaxique et la complétion.

−−syntax_file=name, −Y name
: Fichier de syntaxe pour la complétion.

−−translations_path=name, −T name
: Chemin jusqu'au dossier de traductions de MySQL Control Center.

−−user=user_name, −u user_name
: Nom d'utilisateur.

−−version, −V
: Affiche la version.

Vous pouvez spécifier les variables suivantes avec l'option `-O` ou `--set-variable` .

`connect_timeout`
: Nombre de seconde avant que la connexion expire. (Valeur par défaut 0)

`max_allowed_packet`
: Taille maximale des paquets à échanger avec le serveur (Valeur par défaut 16777216)

`max_join_size`
: Limite automatique du nombre de lignes dans une jointure avec l'option `--safe-updates` (Valeur par défaut 1000000)

`net_buffer_length`

Buffer de communication TCP/IP (Valeur par défaut 16384)

`select_limit`
: Limite automatique pour les commandes `SELECT` avec l'option `--safe-updates` (Valeur par défaut 1000)

Notez que les syntaxes `--set-variable=name=value` et `-O name=value` sont obsolètes depuis MySQL 4.0. Utilisez `--name=value` .

8.7 Utiliser mysqlcheck pour l'entretien et la réparation

Vous pouvez utiliser `mysqlcheck` comme outil d'entretien et de réparation pour les tables `MyISAM` . `mysqlcheck` est disponible depuis MySQL version 3.23.38.

`mysqlcheck` est similaire à `myisamchk` mais il fonctionne différemment. `mysqlcheck` doit être utilisé lorsque le serveur `mysqld` fonctionne, alors que `myisamchk` doit être utilisé lorsque le serveur ne fonctionne pas. L'intérêt st que vous n'avez plus besoin d'interrompre le serveur pour vérifier ou réparer vos tables.

`mysqlcheck` utilise les commandes du serveur MySQL `CHECK` , `REPAIR` , `ANALYZE` et `OPTIMIZE` , d'une manière pratique pour l'utilisateur.

Il y a trois façons différentes d'utiliser `mysqlcheck` :

```
 shell> mysqlcheck [options] db_name [tables]
shell> mysqlcheck [options] --databases DB1 [DB2 DB3...]
shell> mysqlcheck [options] --all-databases
```

Il peut aussi être utilisé comme `mysqldump` lorsqu'il faut choisir les bases et tables à traiter. `mysqlcheck` dispose d'une fonctionnalité spéciale, comparé aux autres clients : le comportement par défaut, c'est à dire la vérification des tables, peut être modifiée en renommant le fichier binaire. Si vous voulez avoir un fichier qui réparer les tables par défaut, il suffit de copier `mysqlcheck` sur votre disque, et de l'appeler `mysqlrepair` , ou bien, de faire un lien symbolique sur l'exécutable et de l'appeler `mysqlrepair` . Si vous appelez `mysqlrepair` , il va réparer les tables par défaut.

Les noms que vous pouvez utiliser pour que `mysqlcheck` réparer automatiquement les tables sont :

mysqlrepair	L'option par défaut est --repair
mysqlanalyze	L'option par défaut est --analyze
mysqloptimize	L'option par défaut est --optimize

Les options disponibles pour `mysqlcheck` sont listées ici. Vérifiez que votre version les supporte avec la commande `mysqlcheck --help` .

`-?` , *{−−help}*
: Affiche ce message d'aide, et termine.

`--all-databases` , `-A`
: Vérifie toutes les bases. C'est la même chose que `--databases` dans toutes les bases sélectionnées.

`--all-in-1` , `-1`
: Au lieu de faire une requête par table, exécute toutes les requêtes dans une requête, séparément pour chaque base. Les noms de tables seront séparés par une virgule.

`--analyze` , `-a`
: Analyse les tables indiquées.

`--auto-repair`
: Si une table vérifiées est corrompue, la corrige automatiquement. La réparation sera faite après la vérification de toutes les tables, si des tables corrompues ont été découvertes.

`--character-sets-dir=...`

Dossier contenant le jeu de caractères. Le jeu de caractères utilisé pour les données et le stockage .

`--check` , `-c`
: Vérifie les tables en erreur

`--check-only-changed` , `-C`
: Vérifie uniquement les tables qui ont été modifiées depuis la dernière modification, ou qui n'ont pas été correctement fermées.

`--compress`
: Utiliser la compression du protocole client/serveur.

`--databases` , `-B`
: Pour tester plusieurs bases de données. Notez que la différence d'utilisation : dans ce cas, aucune table n'est précisé. Tous les arguments de noms sont considérés comme des noms de base.

`--debug[=debug_options]` , `-# [debug_options]`
: Affiche le log de débogage. Souvent, la chaîne `debug_options` vaut `'d:t:o,nom_de_fichier'` .

`--default-character-set=...`
: Spécifie le jeu de caractères par défaut. Le jeu de caractères utilisé pour les données et le stockage .

`--extended` , `-e`
: Si vous utilisez cette option avec `CHECK TABLE` , elle va s'assurer que la table est totalement cohérente, mais prendre un très long temps.

Si vous utilisez cette option avec REPAIR TABLE, elle va réaliser une réparation exhaustive de la table, qui peut non seulement prendre un temps très long, mais produire de nombreuses lignes erronées.

`--fast` , `-F`
: Ne vérifie que les tables qui n'ont pas été correctement fermées.

`--force` , `-f`
: Continue même si on rencontre une erreur SQL.

`--host=host_name` , `-h host_name`
: Connexion à l'hôte.

`--medium-check` , `-m`
: Plus rapide que la vérification complète, mais ne trouvera que 99.99 % de toutes les erreurs. Cela devrait être la bonne option pour la plupart des situations.

`--optimize` , `-o`
: Optimise la table.

`--password[=password]` , `-p[password]`
: Le mot de passe à utiliser lors de la connexion au serveur. Si aucun mot de passe n'est fourni, il sera demandé en ligne de commande. Il ne **faut pas** laisser d'espace entre l'option −p et le mot de passe.

`--port=port_num` , `-P port_num`
: Le numéro de port de la connexion.

−−protocol={TCP | SOCKET | PIPE | MEMORY}
: Pour spécifier le protocole à utiliser pour la connexion. Nouveau en MySQL 4.1.

`--quick` , `-q`
: Si vous utilisez cette option avec `CHECK TABLE` , elle va éviter que l'analyse ne traite toutes les lignes pour vérifier les mauvais liens. C'est la méthode d'analyse la plus rapide.

Si vous utilisez cette option avec REPAIR TABLE, elle va essayer de ne réparer que le fichier d'index. C'est la méthode la plus rapide pour la réparation.

`--repair` , `-r`
: Peut corriger presque tout, sauf les problèmes de doublons pour les clés uniques.

`--silent` , `-s`
: Affiche moins de messages d'erreurs.

`--socket=path` , `-S path`
: Nom du fichier de socket à utiliser pour la connexion.

`--tables`
: Remplace l'option `--databases` ou `-B` . Tous les arguments suivants sont considérés comme des noms de tables.

`--user=user_name` , `-u user_name`
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur courant.

`--verbose` , `-v`
: Afficher des informations sur les différentes étapes.

`--version` , `-V`
: Affiche les informations de version, et termine.

8.8 mysqldump , sauvegarde des structures de tables et les données

Utilitaire qui permet d'exporter une base ou un groupe de bases vers un fichier texte, pour la sauvegarde ou le transfert entre deux serveurs (pas nécessairement entre serveurs MySQL). L'export contiendra les requêtes SQL nécessaires pour créer la table et la remplir.

Si vous faîtes une sauvegarde du serveur, vous devriez aussi utiliser la commande `mysqlhotcopy` . `mysqlhotcopy` , copier les bases et tables MySQL .

Il y a plusieurs méthodes pour invoquer `mysqldump` :

```
 shell> mysqldump [options] db_name [tables]
shell> mysqldump [options] --databases DB1 [DB2 DB3...]
shell> mysqldump [options] --all-databases
```

Si vous ne spécifiez pas de table, ou si vous utilisez l'option `--databases` ou `--all-databases` , la base de données complète sera exportée.

Vous pouvez obtenir une liste des options valides pour votre version de `mysqldump` avec la commande `mysqldump --help` .

Notez que si vous exécutez `mysqldump` sans l'option `--quick` ou `--opt` , `mysqldump` va charger la totalité du résultat en mémoire, avant de l'écrire. Cette option peut résoudre des problèmes de mémoire si vous exportez de grosses tables.

Notez que si vous utilisez une nouvelles copie du programme `mysqldump` , et que vous allez faire un export qui sera lu dans une vieille version de MySQL, vous ne devriez pas utiliser les options `--opt` et `-e` .

Les valeurs numériques hors des plages de validité comme `-inf` et `inf` , ainsi que `NaN` (`not-a-number` , pas un nombre) sont exportées par `mysqldump` comme `NULL` . Vous pouvez le voir dans la table suivante :

```
 mysql> CREATE TABLE t (f DOUBLE);
mysql> INSERT INTO t VALUES(1e+111111111111111111111);
mysql> INSERT INTO t VALUES(-1e111111111111111111111);
mysql> SELECT f FROM t;
+------+
| f    |
+------+
|  inf |
| -inf |
+------+
```

Pour cette table, `mysqldump` produit l'export suivant :

```
 --
-- Dumping data for table `t`
--
```

```
 INSERT INTO t VALUES (NULL);
INSERT INTO t VALUES (NULL);
```

La signification de ce comportement est que si vous voulez exporter puis restaurer une table, le nouveau contenu sera peut être différent de l'original. Notez que depuis MySQL 4.1.2 vous ne pouvez pas insérer la valeur `inf` dans la table, et ce comportement de `mysqldump` ne sera pertinent qu'avec les anciens serveurs.

`mysqldump` supporte les options suivantes :

`--help` , `-?`
: Affiche le message d'aide et quitte.

`--add-drop-table`
: Ajoute une commande `drop table` avant chaque requête de création de table.

`--add-locks`
: Ajoute une commande `LOCK TABLES` avant l'export de table et une commande `UNLOCK TABLE` après(Pour accélérer les insertions dans MySQL). Vitesse des requêtes `INSERT` .

`--all-databases` , `-A`
: Exporte toutes les tables. C'est l'équivalent de l'option `--databases` avec toutes les bases de données sélectionnées.

`--allow-keywords`
: Permet la création de colonnes ayant des noms de mots réservés. Cela fonctionne en préfixant chaque nom de colonne avec le nom de la table.

`--comments[={0|1}]`
: Si cette option prend `0` , elle supprime les informations additionnelles (comme les versions de programme, les versions d'hôte) dans les exports. L'option `--skip-comments` fait la même chose. Par défaut, la valeur de cette option est `1` , pour conserver ces informations. Nouveau en MySQL 4.0.17.

`--compatible=name`
: Produit un résultat qui est compatible avec les autres bases de données, ou avec d'anciennes versions de MySQL. Les valeurs possibles de `name` sont `mysql323` , `mysql40` , `postgresql` , `oracle` , `mssql` , `db2` , `sapdb` , `no_key_options` , `no_table_options` , ou `no_field_options` . Pour utiliser plusieurs valeurs, séparez les par des virgules. Ces valeurs ont la même signification que les options correspondantes de configuration du mode SQL. Le mode SQL du serveur .

  Cette option requiert la version 4.1.0 ou plus récente. Avec les anciens serveurs, cela ne fait rien.

`--complete-insert` , `-c`
: Utilise des commandes `INSERT` complètes, avec les noms de colonnes.

`-C` , `--compress`
: Compresse toutes les informations entre le client et le serveur, les deux supporte la compression.

`--create-options`
: Inclut toutes les options spécifiques MySQL de création de table dans les commandes `CREATE TABLE` . Avant MySQL 4.1.2, utilisez `--all` .

`--databases` , `-B`
: Pour exporter plusieurs bases de données. Notez la différence d'utilisation. Dans ce cas, aucune table n'est spécifié. Tous les arguments de noms sont considérés comme des noms de base. Une ligne `USE db_name;` sera ajoutée dans l'export avant chaque base de données.

`--debug[=debug_options]` , `-# [debug_options]`
: Active l'historique de débogage. La chaîne de format est généralement `'d:t:o,file_name'` .

`--default-character-set=charset`
: Configure le jeu de caractères par défaut pour l'export. S'il n'est pas spécifié, `mysqldump` 10.3 (MySQL−4.1.2) ou plus récent va utiliser `utf8` . Les versions plus anciennes utiliseront `latin1` .

`--delayed`
: Les insertions se font avec la commande `INSERT DELAYED` .

`--delete-master-logs`

Sur un maître de réplication, efface le log binaire une fois que l'opération d'export est faite. Cette option active automatiquement `--first-slave` . Elle a été ajoutée en MySQL 3.23.57 (pour MySQL 3.23) et MySQL 4.0.13 (pour MySQL 4.0).

`--disable-keys` , `-K`

Pour chaque table, entoure les commandes d' `INSERT` avec les commandes `/*!40000 ALTER TABLE tbl_name DISABLE KEYS */;` et `/*!40000 ALTER TABLE tbl_name ENABLE KEYS */;` . Cela accélère les chargements du fichier d'export pour MySQL 4.0 car les index ne sont créés qu'après l'insertion. Cette option n'est effective que pour les tables `MyISAM` .

`--extended-insert` , `-e`

Utilise la nouvelle syntaxe multi−ligne `INSERT` . (Cela donne des insertions plus courtes et plus efficaces).

`--fields-terminated-by=...`
`--fields-enclosed-by=...`
`--fields-optionally-enclosed-by=...`
`--fields-escaped-by=...`
`--lines-terminated-by=...`

Ces options sont utilisées avec l'option `-T` et ont la même signification que les clauses correspondantes de la commande `LOAD DATA INFILE` . Syntaxe de `LOAD DATA INFILE` .

`--first-slave` , `-x`

Verrouille toutes les tables de toutes les bases de données.

`--flush-logs` , `-F`

Ecrit tout le fichier de log du serveur avant de commencer l'export. Notez que si vous utilisez cette option avec `--all-databases` (ou l'option `-A`), les logs seront vidés **pour chaque base de données exportée** .

`-f` , `--force,`

Continue même si une erreur SQL survient durant l'export.

`--host=host_name` , `-h host_name`

Exporte les données depuis le serveur MySQL vers l'hôte indiqué. L'hôte par défaut est `localhost` .

`--lock-tables` , `-l`

Verrouille toutes les tables avant de commencer l'export. Les tables sont verrouillées avec `READ LOCAL` pour permettre des insertions concurrentes sur les tables `MyISAM` .Notez que lorsque vous exportes des tables de bases différentes, l'option `--lock-tables` va verrouiller chaque base séparément. Cette option ne vous garantira pas que vos tables seront logiquement cohérente entre les bases. Des tables de différentes bases pourraient être exportées dans des états très différents.

`--master-data`

Cette option est similaire à `--first-slave` , mais produit aussi une commande `CHANGE MASTER TO` qui fait que le serveur esclave va commencer à la bonne position dans le log du maître, si vous utilisez cette exportation pour configurer initialement l'esclave.

`--no-create-db` , `-n`

`CREATE DATABASE /*!32312 IF NOT EXISTS*/ db_name;` ne sera pas ajouté dans l'export. Sinon, la ligne ci−dessus sera ajoutée, si l'une des options `--databases` ou `--all-databases` ont été activée.

`--no-create-info` , `-t`

N'écrit pas les informations de création de table (la requête `CREATE TABLE`).

`--no-data` , `-d`

N'écrit aucune ligne d'informations sur la table. C'est très pratique si vous voulez simplement exporter la structure de la table.

`--opt`

Identique à `--quick --add-drop-table --add-locks --extended-insert --lock-tables` . Vous obtiendrez l'export le plus rapide à importer dans un serveur MySQL.

`--password[=password]` , `-p[password]`

Le mot de passe à utiliser lors de la connexion au serveur. Notez que si vous utilisez l'option courte `-p` , vous **ne devez pas** laisser d'espace entre l'option et le mot de passe. Si vous spécifiez en omettant la partie `'=your_pass'` , `mysqldump` vous demandera le mot de passe en ligne de commande.

`--port=port_num` , `-P port_num`

Le port TCP/IP à utiliser avec l'hôte.

`--protocol={TCP | SOCKET | PIPE | MEMORY}`
: Pour spécifier le protocole de connexion à utiliser. Nouveau en MySQL 4.1.

`--quick` , `-q`
: Ne garde pas en buffer les requêtes, mais écrit immédiatement dans la sortie. Utilise `mysql_use_result()` pour cela.

`--quote-names` , `-Q`
: Protège les noms des tables et colonnes avec le caractère '`' .

`--result-file=file` , `-r file`
: Ecrit directement dans le fichier indiqué. Cette option doit être utilisé sur MSDOS, car cela évite que la nouvelle ligne `'\n'` soient converties en `'\n\r'` (nouvelle ligne et retour chariot).

`--single-transaction`
: Cette option ajoute la commande SQL `BEGIN` avant d'exporter les données vers le serveur. C'est généralement pratique pour les tables `InnoDB` et le niveau d'isolation de transaction `READ_COMMITTED` , car ce mode va exporter l'état de la base au moment de la commande `BEGIN` sans bloquer les autres applications.

  Lorsque vous utilisez cette option, pensez bien que seules les tables transactionnelles seront exportées dans un état cohérent, c'est à dire que les tables `MyISAM` ou `HEAP` qui seront exportées avec cette option, pourront changer d'état.

  L'option `--single-transaction` a été ajoutée en version 4.0.2. Cette option est mutuellement exclusive avec l'option `--lock-tables` car `LOCK TABLES` va valider une transaction interne précédente.

`--socket=path` , `-S path`
: Le fichier de socket à utiliser pour les connexions locale (à `localhost`), qui est l'hôte par défaut.

`--skip-comments`
: Identique à que `--comments` = `0` .

`--tab=path` , `-T path`
: Crée un fichier `table_name.sql` , qui contient les commandes SQL `CREATE` , et un fichier `table_name.txt` , qui contient les données, pour chaque table. Le format du fichier `.txt` est celui qui est spécifié par les options `--fields-xxx` et `--lines--xxx` . **Note** : cette option ne fonctionne qui si `mysqldump` est exécuté sur la même machine que le démon `mysqld` , et que le nom d'utilisateur et le groupe de `mysqld` (normalement l'utilisateur `mysql` , et le groupe `mysql`) doivent avoir des permission pour créer et écrire un fichier dans le dossier que vous spécifiez.

`--tables`
: Remplace l'option `--databases` ou `-B` . Tous les arguments suivant les options sont considérés comme des noms de tables.

`--user=user_name` , `-u user_name`
: Le nom d'utilisateur MySQL lors de la connexion à un serveur distant.

`--verbose` , `-v`
: Mode détaillé. Affiche plus d'informations sur les faits et gestes du programme.

`--version` , `-V`
: Affiche la version du programme et quitte.

`--where='where-condition'` , `-w 'where-condition'`
: Exporte uniquement les lignes sélectionnées. Notez que les guillemets sont obligatoires.

  Exemples :

```
 "--where=user='jimf'"
"-wuserid>1"
"-wuserid<1"
```

`-X` , `--xml`
: Exporte la base au format XML.

Vous pouvez aussi configurer les variables systèmes suivantes avec la syntaxe `--var_name=value` :

`max_allowed_packet`
: La taille maximale du buffer pour les communications client / serveur. La valeur de cette variable peut être au maximum de 16 Mo avant MySQL 4.0, et jusqu'à 1 Go depuis MySQL 4.0. Lors de la création de commandes d'insertions multiples (avec l'option `--extended-insert` ou `--opt`), `mysqldump` va créer des lignes ayant une taille maximale de `max_allowed_packet` octets. Si vous augmentez la valeur de cette variable, assurez vous que `max_allowed_packet` est assez grande dans le serveur.

`net_buffer_length`
: La taille initiale du buffer de communication.

Il est aussi possible de configurer les variables en utilisant `--set-variable=var_name=value` ou `-O var_name=value` . Mais cette syntaxe est obsolète depuis MySQL 4.0.L'usage normal de `mysqldump` est probablement de faire des sauvegardes de bases.

```
mysqldump --opt database > backup-file.sql
```

Vous pouvez importer les données dans la base MySQL avec :

```
mysql database < backup-file.sql
```

ou

```
mysql -e "source /patch-to-backup/backup-file.sql" database
```

Cependant, il est très pratique pour remplir un autre serveur MySQL avec des informations depuis une base :

```
mysqldump --opt database | mysql ---host=remote-host -C database
```

Il est possible d'exporter plusieurs bases de données en une seule commande :

```
mysqldump --databases database1 [database2 ...] > my_databases.sql
```

Si vous souhaitez exporter toutes les bases, vous pouvez utiliser :

```
mysqldump --all-databases > all_databases.sql
```

Pour plus d'informations sur les sauvegardes, voyez Sauvegardes de base de données .

8.9 mysqlhotcopy , copier les bases et tables MySQL

`mysqlhotcopy` est un script Perl qui utilise `LOCK TABLES` , `FLUSH TABLES` et `cp` ou `scp` pour faire rapidement des sauvegardes de bases. C'est la méthode la plus rapide pour faire une sauvegarde. C'est aussi le moyen le plus sûr pour copier des tables et bases, mais il ne peut fonctionner que sur la machine qui contient les fichiers de données. `mysqlhotcopy` fonctionne uniquement sous Unix, et il ne fonctionne qu'avec les tables de type `MyISAM` et `ISAM` .

```
shell> mysqlhotcopy db_name [/path/to/new_directory]
shell> mysqlhotcopy db_name_1 ... db_name_n /path/to/new_directory
shell> mysqlhotcopy db_name./regex/
```

`mysqlhotcopy` supporte les options suivantes :

−?, −−help
: Affiche un écran d'aide et quitte.

−−allowold
: Ne pas annuler si la sauvegarde existe déjà (renomme la simplement en _old)

−−checkpoint=db_name.tbl_name
: Insère un point de contrôle dans la table spécifiée (base.table)

−−debug
: Active le débogage.

−−dryrun, −n
: Rapporte les actions réalisées sans les faire.

−−flushlog
: Vide les logs sur le disque une fois que toutes les tables sont verrouillées.

−−keepold

Ne pas effacer une sauvegarde précédente (celle qui a été renommée) lorsque c'est terminé.

−−method=#
: Méthode de copie (`cp` ou `scp`).

−−noindices
: Ne pas inclure les fichiers d'index complet dans la copie, pour faire des fichiers de sauvegarde plus petit et plus rapide. Les index peuvent toujours être reconstruits plus tard avec `myisamchk -rq.` .

−p, −−password=#
: Mot de passe utilisé pour la connexion au serveur.

−−port=port_num, −P port_num
: Port utilisé pour la connexion au serveur.

−−quiet, −q
: Mode silencieux. N'affiche que les erreurs.

−−regexp=expr
: Copie toutes les bases dont le nom vérifie un masque d'expression régulière.

−−socket=path, −S path
: Socket utilisée pour la connexion au serveur.

−−suffix=str
: Suffixe des noms des bases copiées.

−−tmpdir=path
: Dossier temporaire (au lieu de `/tmp`).

−−user=user_name, −u user_name
: Nom d'utilisateur pour la connexion au serveur.

Vous pouvez essayer `perldoc mysqlhotcopy` pour avoir plus de documentation sur `mysqlhotcopy` . `mysqlhotcopy` lit les options du groupe `[client]` et `[mysqlhotcopy]` dans le fichier d'options.

Pour être capable d'exécuter `mysqlhotcopy` , vous avez besoin des droits d'écriture dans le dossier de sauvegarde, et le droit de `SELECT` dans les tables que vous souhaitez copier, ainsi que les droits MySQL de `RELOAD` (pour utiliser la commande `FLUSH TABLES`).

```
shell> perldoc mysqlhotcopy
```

8.10 mysqlimport , importer des données depuis des fichiers texte

`mysqlimport` fournit une interface en ligne de commande à la commande SQL `LOAD DATA INFILE` . La plupart des options de `mysqlimport` correspondent directement aux mêmes options de `LOAD DATA INFILE` . Syntaxe de `LOAD DATA INFILE` .

`mysqlimport` est appelé comme ceci :

```
shell> mysqlimport [options] database textfile1 [textfile2 ...]
```

Pour chaque fichier texte indiqué dans la ligne de commande, `mysqlimport` supprime toute extension du nom du fichier, et utilise le résultat pour déterminer le nom de la table qui va recevoir le contenu du fichier. Par exemple, pour des fichiers appelés `patient.txt` , `patient.text` et `patient` seront tous importés dans la table `patient` . `mysqlimport` supporte les options suivantes :

`--help` , `-?`
: Affiche le message d'aide et quitte.

`--columns=column_list` , `-c column_list`

Cette option prend une liste de noms de colonnes, séparés par des virgules. Ce champs est utilisé pour créer une commande `LOAD DATA INFILE` correcte, qui sera alors passée à MySQL.

`--compress` , `-C`

Compresse toutes les informations entre le client et le serveur, si c'est possible.

`--debug[=debug_options]` , `-# [debug_options]`

Active le débogage. la valeur de `debug_options` est souvent : `'d:t:o,file_name'` .

`--delete` , `-D`

Vide la table avant d'importer le fichier texte.

`--fields-terminated-by=...`
`--fields-enclosed-by=...`
`--fields-optionally-enclosed-by=...`
`--fields-escaped-by=...`
`--lines-terminated-by=...`

Ces options ont la même signification que les clauses correspondantes de `LOAD DATA INFILE` . Syntaxe de `LOAD DATA INFILE` .

`--force` , `-f`

Ignore les erreurs. Par exemple, si une table n'existe pas pour un fichier texte, `mysqlimport` va continuer de traiter les autres fichiers. Sans `--force` , `mysqlimport` se termine dès qu'une erreur survient.

`--host=host_name` , `-h host_name`

Importe les données sur le serveur MySQL, avec l'hôte spécifié. La valeur par défaut est `localhost` .

`--ignore` , `-i`

Voir la description de `--replace` .

`--ignore-lines=n`

Ignore les `n` premières lignes du fichier de données.

`--local` , `-L`

Lit le fichier d'entrée dans le client. Par défaut, les fichiers textes sont supposés être lus par le serveur, si vous vous connectez à `localhost` (qui l'hôte par défaut).

`--lock-tables` , `-l`

Verrouille **toutes** les tables en écriture avant de ne traiter les fichiers textes. Cela assure que toutes les tables sont synchronisée sur le serveur.

`--password[=password]` , `-p[password]`

Le mot de passe à utiliser lors de la connexion au serveur. Notez que si vous utilisez l'option courte (`-p`), vous **ne pouvez pas** laisser d'espace entre l'option est le mot de passe. Si vous ne spécifiez pas le mot de passe avec l'option, `mysqlimport` va vous demander le mot de passe en ligne.

`--port=port_num` , `-P port_num`

Le port TCP/IP utilisé avec l'hôte. Cela sert pour les connexions à des hôtes qui ne sont pas `localhost` , pour lequel la socket Unix est utilisée.

`--protocol={TCP | SOCKET | PIPE | MEMORY}`

Spécifie le protocole à utiliser pour se connecter. Nouveau en MySQL version 4.1.

`--replace` , `-r`

Les options `--replace` et `--ignore` contrôles la gestion des lignes lues envers les lignes qui existent déjà sur le serveur. Si vous spécifiez l'option `--replace` , les nouvelles lignes remplaceront les lignes existantes. Si vous spécifiez `--ignore` , les lignes qui sont en double dans une table qui dispose d'une colonne de type unique. Si vous ne spécifiez pas ces options, une erreur surviendra lorsqu'une clé en double sera trouvée, et la lecture du reste du fichier sera annulé.

`--silent` , `-s`

Mode silencieux. N'affiche que les erreurs qui surviennent.

`--socket=path` , `-S path`

Le fichier de socket à utiliser lors de la connexion à `localhost` (qui est l'hôte par défaut).

`--user=user_name` , `-u user_name`

Le nom de l'utilisateur MySQL à utiliser lors de la connexion au serveur MySQL. La valeur par défaut est celui de votre utilisateur Unix.

`--verbose` , `-v`

`--verbose` , `-v`
: Mode détaillé. Affiche bien plus d'informations sur les actions du programme.

Affiche la version et quitte.

Voici un exemple d'utilisation de `mysqlimport` :

```
 $ mysql --version
mysql  Ver 9.33 Distrib 3.22.25, for pc-linux-gnu (i686)
$ uname -a
Linux xxx.com 2.2.5-15 #1 Mon Apr 19 22:21:09 EDT 1999 i586 unknown
$ mysql -e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
$ ed
a
100     Max Sydow
101     Count Dracula
.
w imptest.txt
32
q
$ od -c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   C   o   u   n   t       D   r   a   c   u   l   a  \n
0000040
$ mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$ mysql -e 'SELECT * FROM imptest' test
+------+---------------+
| id   | n             |
+------+---------------+
|  100 | Max Sydow     |
|  101 | Count Dracula |
+------+---------------+
```

8.11 Afficher les bases, tables et colonnes

`mysqlshow` peut être utilisé pour lister les bases qui existent, leurs tables et leurs colonnes.

Avec le programme `mysql` vous pouvez obtenir des informations avec la commande `SHOW` . Les mêmes informations sont disponibles en utilisant ces commandes directement. Par exemple, vous pouvez utiliser le client `mysql` . Syntaxe de `SET` et `SHOW` .

`mysqlshow` est utilisé comme ceci :

```
 shell> mysqlshow [OPTIONS] [database [table [column]]]
```

- Si aucune base n'est indiquée, toutes les bases de données sont listées.
- Si aucune table n'est nommée, toutes les tables de la base sont affichées.
- Si aucune colonne n'est nommée, toutes les colonnees et leur type sont affichés.

Notez que dans les nouvelles versions de MySQL, vous ne verrez que les bases de données, tables et colonnes pour lesquelles vous avez des droits.

Si le dernier argument contient un caractère joker shell ou SQL (`'*'` , `'?'` , `'%'` ou `'_'`) alors seules les entités qui valident ce masque sont affichées. Si une base contient des caractères soulignés, ils doivent être protégés avec un anti−slash (certains shell Unix en demande même deux), afin de lister correctement les tables et les colonnes. Les `'*'` et `'?'` sont convertis en joker SQL `'%'` and `'_'` . Cela peut causer des confusions lorsque vous essayez d'afficher des colonnes qui contiennent un souligné `'_'` , comme c'est le cas avec `mysqlshow` qui ne vous affiche que les colonnes qui vérifient le masque. Ceci est facilement corrigé en ajoutant un caractère `'%'` en plus dans la ligne de commande (comme argument séparé).

`mysqlshow` supporte les options suivantes :

−−help, −?
: Affiche cette aide et quitte.

−−character−sets−dir=path
: Le dossier où les jeux de caractères sont créés. Le jeu de caractères utilisé pour les données et le stockage .

−−compress, −C
: Utilise la compression avec le protocole client / serveur.

−−debug[=debug_options], −# [debug_options]
: Génère un log de débogage. La chaîne `debug_options` vaut souvent `'d:t:o,file_name'` . Par défaut, la valeur est `'d:t:o,/tmp/mysql.trace'` .

−−default−character−set=charset
: Configure le jeu de caractères par défaut. Le jeu de caractères utilisé pour les données et le stockage .

−−host=host_name, −h host_name
: Connexion avec l'hôte indiqué.

−−keys, −k
: Affiche les index de la table.

−−password[=password], −p[password]
: Le mot de passe utilisé lors de la connexion sur le serveur. S'il n'est pas donné en ligne de commande, il sera demandé interactivement. Notez que si vous utilisez la forme courte `-p` , vous **ne devez pas** laisser d'espace entre l'option et le mot de passe.

−−port=port_num, −P port_num
: Le numéro de port TCP/IP pour la connexion.

−−protocol={TCP | SOCKET | PIPE | MEMORY}
: Spécifie le protocole de connexion à utiliser. Nouveau en MySQL version 4.1.

−−socket=path, −S path
: Le fichier de socket à utiliser pour la connexion.

−−status, −i
: Affiche des informations supplémentaires sur chaque table.

−−user=user_name, −u user_name
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur Unix courant.

−−verbose, −v
: Affichage plus détaillé (`-v -v -v` indique le format d'affichage de table).

−−version, −V
: Affiche la version et quitte.

8.12 perror , expliquer les codes d'erreurs

Pour la plupart des erreurs système, MySQL va, en plus d'un message interne, aussi afficher un code d'erreur, dans l'un des styles suivants :

```
 message ... (errno: #)
message ... (Errcode: #)
```

Vous pouvez découvrir ce que ce code d'erreur signifie soit en examinant la documentation de votre système, soit en utilisant l'utilitaire `perror` . `perror` affiche une description pour le code d'erreur, ou, pour une erreur du gestionnaire de tables MyISAM/ISAM.

`perror` est appelé comme ceci :

```
shell> perror [options] errorcode ...
```

Exemple :

```
 shell> perror 13 64
Error code  13:  Permission denied
Error code  64:  Machine is not on the network
```

Notez que les messages d'erreurs sont dépendants du système. Un message d'erreur peut avoir différentes explications sur différents systèmes.

8.13 L'utilitaire de remplacement de chaînes replace

L'utilitaire `replace` modifie les chaînes dans des fichiers. Elle utilise une machine à états finis pour rechercher d'abord les plus grandes chaînes. Elle peut être utilisée pour faire des échanges de chaînes, entre les chaînes `a` et `b` de deux fichiers `file1` et `file2` :

```
shell> replace a b b a -- file1 file2 ...
```

Utilisez l'option `--` pour indiquer où la liste de chaînes de remplacement s'arrête, et où commence la liste de fichiers.

Tout fichier nommé en ligne de commande est modifié directement : il est recommandé de faire des sauvegardes de fichiers originaux.

Si aucun fichier n'est nommé en ligne de commande, `replace` lit l'entrée standard, et écrit dans la sortie standard. Dansa ce cas, aucune option `--` n'est nécessaire.

Le programme `replace` sert à `msql2mysql` . `msql2mysql` , convertit des programmes mSQL vers MySQL .

`replace` supporte les options suivantes :

−?, −I
: Affiche le message d'aide et quitte.

−# debug_options
: Écrit un log de débogage. La chaîne `debug_options` vaut souvent `'d:t:o,file_name'` .

−s
: Mode silencieux. Affiche moins d'informations sur les activités du programme.

−v
: Mode prolifique. Affiche plus d'informations sur les activités du programme.

−V
: Affiche la version, et quitte.

9 Référence du langage MySQL

MySQL possède une interface SQL assez complexe, mais facile à apprendre. Ca chapitre décrit les différentes commandes, types, et fonctions que vous aurez besoin de connaître pour utiliser MySQL de façon optimale. Ce chapitre sert aussi de référence pour toutes les fonctionnalités incluses au sein de MySQL. Afin d'utiliser ce chapitre efficacement, vous pourrez trouver utile de vous reporter aux différents index.

- Voyez la section Structure dy langage pour des informations sur l'écriture des valeurs, des identifiants et des commentaires.
- Voyez la section Types de colonnes pour des informations sur les différents types de colonnes que MySQL supporte dans les tables.
- Voyez la section Fonctions et opérateurs pour une liste de fonctions et opérateurs à utiliser pour écrire des expressions.
- Voyez la section Syntaxe SQL pour la description des commandes SQL que MySQL supporte.

Afin d'utiliser efficacement ces chapitres, vous pouvez vous reporter aux différents index.

10 Structure du langage

Ce chapitre présente les règles d'écriture des commandes SQL avec MySQL :

- Les valeurs littérales telles que les nombres et chaînes
- Les identifiants de tables et colonnes
- Les variables utilisateur et système
- Les commentaires
- Les mots réservés

10.1 Littéraux : comment écrire les chaînes et les nombres

Cette section décrit les différents façons d'écrire les chaînes et les nombres en MySQL. Elle couvre aussi les différentes nuances et quiproquos que vous pouvez rencontrer lorsque vous manipulez ces types de données.

10.1.1 Chaînes

Une chaîne est une séquence de caractères, entourée de guillemets simples (`'''`) ou doubles (`'"'`). Exemples:

Si le serveur SQL est en mode `ANSI_QUOTES` , les chaînes littérales ne peuvent être mises qu'entre guillemets simples. Une chaîne avec des guillemets double sera interprétée comme un identifiant.

```
 'une chaîne'
"une autre chaîne"
```

Depuis MySQL 4.1.1, les littéraux disposent d'une option de jeu de caractères et de collation avec la clause `COLLATE` :

```
 [_charset_name]'string' [COLLATE collation_name]
```

Exemples :

```
 SELECT _latin1'string';
SELECT _latin1'string' COLLATE latin1_danish_ci;
```

Pour plus d'informations sur ces formes de chaîne de caractères, voyez Character String Literal Character Set and Collation .

A l'intérieur d'une chaîne, certains séquences de caractères ont une signification spéciale. Chacune d'elle commence par un anti–slash (`'\'`), connu comme le **caractère d'échappement** . `MySQL` reconnaît les séquences suivantes :

`\0`	Un 0 ASCII (NUL).
`\'`	Un guillemet simple (''').
`\"`	Un guillemet double ('"').
`\b`	Un effacement.
`\n`	Une nouvelle ligne.
`\r`	Un retour chariot.
`\t`	Une tabulation.
`\z`	ASCII(26) (Contrôle-Z). Ce caractère peut être encodé pour vous éviter des problèmes avec Windows, vu qu'il équivaut à une fin de fichier sur cet OS. (ASCII(26) vous posera des problèmes si vous utilisez mysql base < fichier .)
`\\`	Un anti-slash ('\').
`\%`	Un signe pourcentage littéral : '%' . Voir les notes ci-dessous.
`\_`	Un signe souligné littéral : '_' . Voir les notes ci-dessous.

Ces séquences sont sensibles à la casse. Par exemple, '\b' est interprétée comme un anti−slash, mais '\B' est interprété comme la lettre 'B' .

Les caractères '\%' et '_' sont utilisés pour rechercher des chaînes littérales '%' et '_' dans un contexte d'expressions régulières. Sinon, ces caractères sont interprétés comme des caractères joker. Fonctions de comparaison de chaînes de caractères . Notez que si vous utilisez '\%' ou '_' dans d'autres contextes, ces séquences retourneront '\%' et '_' et non '%' et '_' .

Il y a plusieurs façons d'intégrer un guillemet dans une chaîne :

- Un `'` à l'intérieur d'une chaîne entourée de `'` peut être noté `''` .
- Un `"` à l'intérieur d'une chaîne entourée de `"` peut être noté `""` .
- Vous pouvez faire précéder le guillemet par caractère d'échappement (`\`).
- Un guillemet simple `'` à l'intérieur d'une chaîne à guillemets doubles `"` n'a besoin d'aucun traitement spécial (ni doublage, ni échappement). De même, aucun traitement spécial n'est requis pour un guillemet double `"` à l'intérieur d'une chaîne à guillemets simples `'` .

Le `SELECT` montré ici explique comment les guillemets et les échappements fonctionnent :

```
 mysql> SELECT 'bonjour', '"bonjour"', '""bonjour""', 'bon''jour', '\'bonjour';
+---------+-----------+-------------+----------+----------+
| bonjour | "bonjour" | ""bonjour"" | bon'jour | 'bonjour |
+---------+-----------+-------------+----------+----------+

 mysql> SELECT "bonjour", "'bonjour'", "''bonjour''", "bon""jour", "\"bonjour";
+---------+-----------+-------------+----------+----------+
| bonjour | 'bonjour' | ''bonjour'' | bon"jour | "bonjour |
+---------+-----------+-------------+----------+----------+

 mysql> SELECT "Voilà\n3\nlignes";
+--------------------+
| Voilà
3
lignes |
+--------------------+
```

Si vous voulez insérer des données binaires dans un champ chaîne (comme un `BLOB`), les caractères suivants doivent être échappés :

NUL	ASCII 0. Représentez le avec '\0' (un anti−slash suivi du caractère ASCII '0').
\	ASCII 92, anti−slash. A représenter avec '\\' .
'	ASCII 39, guillemet simple. A représenter avec '\'' .
"	ASCII 34, guillemet double. A représenter avec '\"' .

Lorsque vous écrivez des applications, toutes les chaînes qui risquent de contenir ces caractères spéciaux doivent être protégés avant d'être intégrée dans la commande SQL. Vous pouvez faire cela de deux manières différentes :

- passez la chaîne à une fonction qui protège les caractères spéciaux. Par exemple, en langage C, vous pouvez utiliser la fonction `mysql_real_escape_string()` . `mysql_real_escape_string()` . L'interface `Perl DBI` fournit une méthode basée sur les `guillemets` pour convertir les caractères spéciaux en leur séquence correspondante. Interface Perl .
- Au lieu de protéger explicitement tous les caractères, de nombreuses interfaces MySQL fournissent un système de variables qui vous permettent de mettre des marqueurs dans la requête, et de lier les variables à leur valeur au moment de leur exécution. Dans ce cas, l'interface se charge de protéger les caractères spéciaux pour vous.

10.1.2 Nombres

Les entiers sont représentés comme une séquence de chiffres. Les décimaux utilisent `.` comme séparateur. Tous les types de nombres peuvent être précédés d'un `-` pour indiquer une valeur négative.

Exemples d'entiers valides :

```
 1221
0
-32
```

Exemples de nombres à virgule flottante :

```
 294.42
-32032.6809e+10
148.00
```

Un entier peut être utilisé dans un contexte décimal, il sera interprété comme le nombre décimal équivalent.

10.1.3 Valeurs hexadécimales

MySQL supporte les valeurs hexadécimales. Dans un contexte numérique, elles agissent comme des entiers (précision 64−bit). Dans un contexte de chaîne, elles agissent comme une chaîne binaire où chaque paire de caractères hexadécimaux est convertie en caractère :

```
 mysql> SELECT x'4D7953514C';
        -> 'MySQL'
mysql> SELECT 0xa+0;
        -> 10
mysql> SELECT 0x5061756c;
        -> 'Paul'
```

En MySQL 4.1 (et en MySQL 4.0 si vous utilisez l'option `--new`), le type par défaut d'une valeur hexadécimale est chaîne. Si vous voulez vous assurer qu'une telle valeur est traitée comme un nombre, vous pouvez utiliser `CAST(... AS UNSIGNED)` :

```
 mysql> SELECT 0x41, CAST(0x41 AS UNSIGNED);
        -> 'A', 65
```

La syntaxe `0x` est basée sur ODBC. Les chaînes hexadécimales sont souvent utilisées par ODBC pour fournir des valeurs aux colonnes `BLOB` . La syntaxe `x'hexstring'` est nouvelle en 4.0 et est basée sur le standard SQL.Depuis MySQL 4.0.1, vous pouvez convertir une chaîne ou nombre en chaîne au format hexadécimal avec la fonction `HEX()` :

```
 mysql> SELECT HEX('cat');
        -> '636174'
mysql> SELECT 0x636174;
        -> 'cat'
```

10.1.4 Valeurs booléennes

Depuis MySQL version 4.1, la constante `TRUE` vaut `1` et la constante `FALSE` vaut `0` . Les noms des constantes peuvent être écrit en minuscules ou majuscules.

```
 mysql> SELECT TRUE, true, FALSE, false;
        -> 1, 1, 0, 0
```

10.1.5 Valeurs NULL

La valeur `NULL` signifie ``pas de données'' et est différente des valeurs comme `0` pour les nombres ou la chaîne vide pour les types chaîne. Problèmes avec les valeurs `NULL` .

`NULL` peut être représenté par `\N` lors de la récupération ou écriture avec des fichiers (`LOAD DATA INFILE` , `SELECT ... INTO OUTFILE`). Syntaxe de `LOAD DATA INFILE` .

10.2 Noms de bases, tables, index, colonnes et alias

Les noms des bases de données, tables, index, colonnes et alias suivent tous les mêmes règles en MySQL.

La table suivante décrit la taille maximale et les caractères autorisées pour chaque type d'identifiant.

Identifiant	**Longueur maximale**	**Caractères autorisés**
Base de données	64	Tous les caractères autorisés dans un nom de dossier à part '/' , '\' et '.' .
Table	64	Tous les caractères autorisés dans le nom d'un fichier à part '/' et '.' .
Colonne	64	Tous.
Index	64	Tous.
Alias	255	tous.

Notez qu'en plus de ce qui précède, vous n'avez pas droit au caractères ASCII(0) ou ASCII(255) dans un identifiant. Avant MySQL 4.1, les identifiants que pouvaient pas contenir de guillemets.

Depuis MySQL 4.1, les identifiants sont stockés en Unicode (UTF8). Cela s'applique aux identifiants stockés dans les tables de définitions du fichier `.frm` , et aux identifiants stockés dans les tables de droits de la base `mysql` . Même si les identifiants Unicode peuvent inclure des caractères multi−octets, notez que les tailles maximales affichées dans la table sont donnés en octets. Si un identifiant contient un caractère multi−octet, le nombre de **caractères** autorisé est alors inférieur aux chiffres affichés.

Un identifiant peut être entre guillemet ou pas. Si un identifiant est un mot réservé, ou qu'il contient des caractères spéciaux, vous **devez** le mettre entre guillemets lorsque vous l'utilisez. Pour une liste des mots réservés, voyez Est−ce que MySQL est sensible aux mots réservés ? . Les caractères spéciaux sont hors de la plage des caractères alpha−numérique et `'_'` et `'$'` .

Notez que si un identifiant est un mot réservé, ou contient des caractères spéciaux, vous devez absolument le protéger avec `'`'` :

Le caractère de protection des identifiants est le guillemet oblique `'`'` :

```
mysql> SELECT * FROM `select` WHERE `select`.id > 100;
```

Si vous utilisez MySQL avec les modes `MAXDB` ou `ANSI_QUOTES` , il est aussi possible d'utiliser les guillemets doubles pour les identifiants :

```
mysql> CREATE TABLE "test" (col INT);
ERROR 1064: You have an error in your SQL syntax. (...)
mysql> SET SQL_MODE="ANSI_QUOTES";
mysql> CREATE TABLE "test" (col INT);
Query OK, 0 rows affected (0.00 sec)
```

Choisir son mode SQL .

Depuis MySQL 4.1, les guillemets peuvent être inclus dans les noms d'identifiants. Si le caractère inclus dans l'identifiant est le même que celui qui est utilisé pour protéger l'identifiant, doublez−le. La commande suivante crée la table `a`b` , qui contient la colonne `c"d` :

```
mysql> CREATE TABLE `a``b` (`c"d` INT);
```

La protection d'identifiant par guillemets a été introduite en MySQL 3.23.6 pour permettre l'utilisation de mots réservés ou des caractères spéciaux dans les noms de colonnes. Avant la version 3.23.6, vous ne pouviez pas utiliser les identifiants qui utilisent des guillemets, et les règles sont plus restrictives :

- Un nom d'identifiant est constitué de caractères alpha−numériques issus du jeu de caractères courant, plus `'_'` et `'$'` . Le jeu de caractères par défaut est ISO−8859−1 (Latin1). Cela peut être modifié avec l'option de démarrage `--default-character-set` de `mysqld` . Le jeu de

caractères utilisé pour les données et le stockage .
- Un nom peut commencer avec n'importe quel caractère qui est valide dans un identifiant. En particulier, un nom peut commencer par un chiffre : cela est différent dans de nombreuses autres bases de données. Cependant, un nom sans guillemets ne peut pas contenir **uniquement** des chiffres.
- Vous ne pouvez pas utiliser le caractère '.' dans les noms, car il est utilisé pour les formats complets de références aux colonnes, en utilisant les noms de tables ou de bases. (Identifier Qualifiers).

Il est recommandé de ne pas utiliser de noms comme `1e` , car une expression comme `1e+1` est ambiguÎ. Elle pourrait être interprétée comme l'expression `1e + 1` ou comme le nombre `1e+1` , suivant le contexte.

10.2.1 Identifiants

MySQL autorise l'utilisation de nom qui sont constitués d'un seul identifiant, ou d'identifiants multiples. Les composants d'un tel nom doivent être séparés par le caractère point ('.'). Le premier composant détermine le contexte dans lequel l'identifiant final est interprété.

En MySQL, vous pouvez faire référence à une colonne en utilisant l'une des trois formes suivantes :

Référence de colonne	**Signification**
col_name	La colonne col_name de n'importe quelle table de la requête qui contient une colonne de ce nom.
tbl_name.col_name	La colonne col_name de la table tbl_name de la base de données par défaut.
db_name.tbl_name.col_name	La colonne col_name de la table tbl_name de la base db_name . Cette syntaxe n'est pas disponible avant MySQL version 3.22.

Si un composant d'un nom complexe requiert des guillemets de protection, vous devez protéger chaque composant individuellement plutôt que l'identifiant dans son ensemble. Par exemple, `` `my-table`.`my-column` `` est valide, mais `` `my-table.my-column` `` ne l'est pas.

Vous n'êtes pas obligé de spécifier le nom d'une table `tbl_name` ou le nom de la base `db_name.tbl_name` comme préfixe dans une requête, à moins que la référence soit ambigue. Supposez que les tables `t1` et `t2` contiennent toutes les deux une colonne `c` , et que vous lisez le contenu de la colonne `c` dans une commande `SELECT` qui utilise les deux tables `t1` et `t2` . Dans ce cas, `c` est ambigue, car ce n'est pas un nom unique dans les différentes tables de la commande. Vous devez le préciser avec un nom de table, comme dans `t1.c` ou `t2.c` . Similairement, si vous lisez des colonnes dans une table `t` de la base `db1` et dans la table `t` de la base `db2` dans la même commande, vous devez utiliser les noms complets de colonnes, comme `db1.t.col_name` et `db2.t.col_name` .

La syntaxe `.tbl_name` correspond à la table `tbl_name` de la base courante. Cette syntaxe est acceptée pour la compatibilité avec ODBC, car certains programmes ODBC préfixent le nom de la table avec un caractère point '.' .

10.2.2 Sensibilité à la casse pour les noms

En MySQL, les bases et les tables correspondent à des dossiers et des fichiers. Les tables dans une base correspondent au moins à un fichier dans le dossier de base et possiblement plusieurs, suivant le moteur de table utilisé. Par conséquent, la sensibilité à la casse du système déterminera la sensibilité à la casse des noms de bases de données et tables. Cela signifie que les noms sont insensibles à la casse sous Windows, et sensibles sous la plupart des variétés Unix. Mac OS X est une exception car il est basé sur Unix, mais le système de fichiers par défaut (HFS+) n'est pas sensible à la casse. Cependant, Mac OS X supporte aussi les volumes UFS, qui sont sensibles à la casse, comme les autres Unix. Extensions MySQL à l'ANSI SQL92 .

Note : même si les noms ne sont pas sensibles à la casse sous Windows, vous ne devez pas vous référer à une entité en utilisant différentes casse dans la même requête. La requête suivante ne fonctionnera pas car elle se référe à une table avec `ma_table` et `MA_TABLE` :

```
mysql> SELECT * FROM ma_table WHERE MA_TABLE.col=1;
```

Les noms de colonnes et d'alias sont insensibles à la casse dans tous les cas.

Les alias sur tables sont sensibles à la casse avant MySQL 4.1.1. La requête suivante ne marchera pas car elle se réfère à `a` et `A` :

```
mysql> SELECT nom_de_colonne FROM nom_de_table AS a
    ->                 WHERE a.nom_de_colonne = 1 OR A.nom_de_colonne = 2;
```

Si vous avez du mal à vous souvenir de la casse des noms de bases et de tables, adoptez une convention, comme toujours créer les bases et les tables en utilisant des minuscules.

La façon de stocker les noms sur le disque et leur utilisation dans les syntaxes MySQL est définie par la variable `lower_case_table_names` , qui peuvent être spécifié au lancement de `mysqld` . `lower_case_table_names` peut prendre l'une des valeurs suivantes :

Valeur	Signification
0	Les noms de tables et bases sont stockées sur le disque avec la casse utilisée dans la commande CREATE TABLE ou CREATE DATABASE . Les comparaisons de nom sont sensibles à la casse. C'est le comportement par défaut sous Unix. Notez que si vous forcer cette valeur à 0 avec l'option --lower-case-table-names=0 sur un système insensible à la casse, et que vous accéder à la table avec une autre casse, alors vous pouvez avoir des corruptions d'index.
1	Les noms de tables sont stockées en minuscules sur le disque, et les comparaisons de nom de tables sont insensibles à la casse. Ce comportement s'applique aussi aux noms de bases de données depuis MySQL 4.0.2, et aux alias de tables depuis 4.1.1. C'est la valeur par défaut sur les systèmes Windows et Mac OS X.
2	Les tables et bases sont stockées sur le disque avec la casse spécifiée dans CREATE TABLE et CREATE DATABASE , mais MySQL les convertis en minuscules lors des recherches. Les comparaisons de noms sont insensibles à la casse. **Note** : cela **ne fonctionne que** sur les systèmes de fichiers qui sont insensibles à la casse. Les noms de tables InnoDB sont stockées en minuscules, comme pour lower_case_table_names=1 . Donne à lower_case_table_names la valeur de 2 est possible depuis MySQL 4.0.18.

Si vous utilisez MySQL sur une seule plate−forme, vous n'aurez pas à changer la valeur de `lower_case_table_names` variable. Cependant, vous pouvez rencontrer des problèmes lors des transferts entre plates−formes, où les systèmes de fichiers différent de par leur sensibilité à la casse. Par exemple, sous Unix, vous pouvez avoir deux tables `ma_table` et `MA_TABLE` , alors que sous Windows, ces deux noms représentent la même table. Pour éviter les problèmes de transferts de noms, vous avez deux choix :

- Utilisez l'option `lower_case_table_names=1` sur tous les systèmes. L'inconvénient le plus sérieux est que lorsque vous utilisez `SHOW TABLES` ou `SHOW DATABASES` , vous ne verrez pas la casse originale des noms.
- Utilisez `lower_case_table_names=0` sous Unix et `lower_case_table_names=2` sous Windows. Cela préserver la casse des noms de tables ou bases. L'inconvénient est que vous devez vous assurer que les requêtes utilisent toujours la bonne casse sous Windows. Si vous transférez vos requêtes vers Unix, où la casse des noms aura son importance, les requêtes ne fonctionneraient plus.

Notez qu'avant de passer la valeur de `lower_case_table_names` à 1 sous Unix, vous devez commencer par convertir vos anciens noms de tables et bases en minuscules, avant de redémarrer `mysqld` .

10.3 Variables utilisateur

MySQL supporte les variables utilisateur spécifiques à la connexion avec la syntaxe `@variablename` . Un nom de variable consiste de caractères alpha−numériques, basés sur le jeu de caractères courant, de `'_'` , `'$'` , et `'.'` . Le jeu de caractères par défaut est ISO−8859−1 Latin1. Cette valeur peut être changée en utilisant l'option `--default-character-set` de `mysqld` . Le jeu de caractères utilisé pour les données et le stockage .

Les variables n'ont pas besoin d'être initialisés. Elles sont à `NULL` par défaut et peuvent contenir un entier, un réel ou une chaîne. Toutes les variables d'un thread sont automatiquement libérées lorsque le thread se termine.

Vous pouvez déclarer une variable avec la syntaxe de `SET` :

```
SET &#64;variable= &#123; expression entier | expression réel | expression chaîne &#125;
[,&#64;variable= ...].
```

Vous pouvez aussi assigner une valeur à une variable avec d'autres commande que `SET` . Par contre, dans ce cas là, l'opérateur d'assignation est `:=` au lieu de `=` , parce que `=` est réservé aux comparaisons dans les requêtes autres que `SET` :

```
mysql> SELECT &#64;t1:=(&#64;t2:=1)+&#64;t3:=4,&#64;t1,&#64;t2,&#64;t3;
+----------------------+------+------+------+
| &#64;t1:=(&#64;t2:=1)+&#64;t3:=4 | &#64;t1  | &#64;t2  | &#64;t3  |
+----------------------+------+------+------+
|                    5 |    5 |    1 |    4 |
+----------------------+------+------+------+
```

Les variables utilisateur peuvent être utilisés là où les expressions sont allouées. Notez que cela n'inclut pas pour l'instant les contextes où un nombre est explicitement requis, comme ce qui est le cas avec la clause `LIMIT` dans une requête `SELECT` , ou la clause `IGNORE nombre LINES` dans une requête `LOAD DATA` . **Note** : dans une requête `SELECT` , chaque expression est n'évaluée que lors de l'envoi au client. Cela signifie que pour les clauses `HAVING` , `GROUP BY` , ou `ORDER BY` , vous ne pouvez vous référer à une expression qui implique des variables qui sont définies dans la partie `SELECT` . Par exemple, la requête suivante **ne produira pas** le résultat escompté :

```
mysql> SELECT (&#64;aa:=id) AS a, (&#64;aa+3) AS b FROM nom_de_table HAVING b=5;
```

La raison est que `@aa` ne contiendra pas la valeur de la ligne courante, mais celle de `id` pour la dernière ligne acceptée.La règle générale est de ne jamais assigner **et** utiliser la même variable dans la même commande.

Un autre problème avec l'affectation de variable et son utilisation dans la même commande est que le type de valeur par défaut est basé sur le type de la variable dans la commande de départ. Une variable non−assignée est supposée être de type `NULL` ou de type `STRING` . L'exemple suivant illustre bien ce cas de figure :

```
mysql> SET &#64;a="test";
mysql> SELECT &#64;a,(&#64;a:=20) FROM table_name;
```

Dans ce cas, MySQL va indiquer au client que la colonne 1 est une chaîne, et il convertira tous les accès à la variable `@a` en chaîne, même si @a recevra finalement un nombre dans la seconde ligne. Après l'exécution de la commande, `@a` sera considéré comme un nombre.

Si vous avez des problèmes avec cela, essayer d'éviter d'affecter et d'utiliser la même variable dans la même commande, ou bien initialisez la variable avec les valeurs de 0, 0.0 ou "" avant de l'utiliser.

10.4 Variables système

A partir de la version 4.0.3, nous fournissons un meilleur accès à beaucoup de variables système et variables de connexion. On peut changer la plupart d'entre elle sans avoir à stopper le serveur.

Le serveur `mysqld` dispose de deux types de variables. Les variables globales, qui affectent l'ensemble du serveur. Les variables de session qui affectent des connexions individuelles.

Lorsque `mysqld` démarre, toutes les variables globales sont initialisées à partir des arguments passés en ligne de commande et des fichiers de configuration. Vous pouvez changer ces valeurs avec la commande `SET GLOBAL` . Lorsqu'un nouveau thread est créé, les variables spécifiques aux threads sont initialisées à partir des variables globales et ne changeront pas même si vous utilisez la commande `SET GLOBAL` .

Le serveur entretient aussi un jeu de variables de session pour chaque client qui se connecte. Les variables de session d'un client sont initialisées au moment de la connexion, en utilisant les valeurs correspondantes globales. Pour les variables de session qui sont dynamiques, le client peut les changer avec la commande `SET SESSION var_name` . Modifier les variables de session d'une connexion ne requiert aucun droit spécifique, mais le client ne peut changer que ses propres variables de session, et non pas celle d'un autre client.

Une modification à une variable globale est visible à tous les clients qui accèdent à cette variable. Mais, cela n'affecte la variable de session correspondante que lors de la prochaine connexion. Les connexions déjà établies ne sont pas affectées par un changement de variable globale. (pas même le client qui a émis la commande `SET GLOBAL`).

Pour définir la valeur d'une variable `GLOBAL` , vous devez utiliser l'une des syntaxes suivantes. Ici nous utilisons la variable `sort_buffer_size` à titre d'exemple.

Pour donner la valeur à une variable `GLOBAL` , utilisez l'une de ces syntaxes :

```
 mysql> SET GLOBAL sort_buffer_size=value;
mysql> SET &#64;&#64;global.sort_buffer_size=value;
```

Pour définir la valeur d'une variable `SESSION` , vous devez utiliser l'une des syntaxes suivantes :

```
 mysql> SET SESSION sort_buffer_size=value;
mysql> SET &#64;&#64;session.sort_buffer_size=value;
mysql> SET sort_buffer_size=value;
```

`LOCAL` est un synonyme de `SESSION` .Si vous ne spécifiez pas `GLOBAL` ou `SESSION` alors `SESSION` est utilisé. Syntaxe de `SET` .

Pour récupérer la valeur d'une variable de type `GLOBAL` vous pouvez utiliser l'une des commandes suivantes :

```
 mysql> SELECT &#64;&#64;global.sort_buffer_size;
mysql> SHOW GLOBAL VARIABLES like 'sort_buffer_size';
```

Pour récupérer la valeur d'une variable de type `SESSION` vous pouvez utiliser l'une des commandes suivantes :

```
 mysql> SELECT &#64;&#64;sort_buffer_size;
mysql> SELECT &#64;&#64;session.sort_buffer_size;
mysql> SHOW SESSION VARIABLES like 'sort_buffer_size';
```

Ici aussi, `LOCAL` est un synonyme de `SESSION` .Lorsque vous récupérez une valeur de variable avec la syntaxe `@@nom_variable` et que vous ne spécifiez pas `GLOBAL` ou `SESSION` , MySQL retournera la valeur spécifique au thread (`SESSION`) si elle existe. Sinon, MySQL retournera la valeur globale.

Pour la commande `SHOW VARIABLES` , si vous ne spécifiez pas `GLOBAL` , `SESSION` ou `LOCAL` , MySQL retourne les valeurs de `SESSION` .

La raison d'imposer la présence du mot `GLOBAL` pour configurer une variable de type `GLOBAL` mais non pour la lire est pour être sur que vous n'aurez pas de problèmes plus tard si vous voulez introduire ou effacer une variable spécifique au thread qui aurait le même nom. Dans ce cas, vous pourriez changer accidentellement l'état du serveur pour toutes les connexions (et non la votre uniquement).

Plus d'informations sur les options de démarrage du système et les variables système sont dans les sections Options de ligne de commande `mysqld` et Variables sytème du serveur . Une liste des variables qui peuvent être modifiées durant l'exécution est présentée dans Variables dynamiques système .

10.4.1 Variables système structurées

Les variables système structurées sont supportées depuis MySQL 4.1.1. Une variable structurée diffère d'une variable système classique sur deux points :

- Sa valeur est une structure avec des composants qui spécifient des paramètres serveurs qui sont étroitement liés.
- Il peu y avoir plusieurs instances d'une même variable système structurée. Chacun d'entre elles a un nom différent, et fait référence à une ressource différente, gérée sur le serveur.

Actuellement, MySQL supporte un type de variables système structurées. Il spécifie les paramètres qui gouvernement les caches de clé. Une variable système structurée de cache de clé contient les composants suivants :

- `key_buffer_size`
- `key_cache_block_size`
- `key_cache_division_limit`
- `key_cache_age_threshold`

L'objectif de cette section est de décrire la syntaxe d'opération avec les variables système structurées. Les variables du cache de clé sont utilisées comme exemple syntaxique, mais les détails concernant le cache de clé sont disponibles dans la section Le cache de clés `MyISAM` .

Pour faire référence à un composant d'une variable système structurée, vous pouvez utiliser un nom composé, au format `nom_d_instance.nom_du_composant` . Par exemple :

```
 hot_cache.key_buffer_size
hot_cache.key_cache_block_size
cold_cache.key_cache_block_size
```

Pour chaque variables système structurée, un instance avec le nom `default` est toujours prédéfinie. Si vous faites référence à un composant d'une variables système structurée sans aucun nom d'instance, l'instance `default` sera utilisée. Par conséquent, `default.key_buffer_size` et `key_buffer_size` font références à la même variable système structurée.

Les règles de nommage pour les variables système structurées et ses composants sont les suivantes :

- Pour un type donnée de variables système structurées, chaque instance doit avoir un nom unique **à l'intérieur** de ce type. Cependant, les noms d'instances doivent être unique **à l'intérieur** des types de variables système structurées. Par exemple, chaque variable système structurée aura une instance `default` , ce qui fait que `default` n'est pas unique à l'intérieur des types de variables.
- Les noms de composants de chaque variable système structurée doit être unique à travers tous les noms de variables systèmes. Si ce n'était pas vrai (c'est à dire, si deux types de variables structurées pouvaient partager des noms de composants), il ne serait pas facile de trouver la variable structurée par défaut, pour ce type.
- Si un nom de variable système structurée n'est pas valide en tant qu'identifiant non protégé, il faut utiliser les guillemets obliques pour le protéger. Par exemple, `hot-cache` n'est pas valide, mais `` `hot-cache` `` l'est.

- `global` , `session` , et `local` ne sont pas des noms de composants valides. Cela évite les conflits avec des notations comme `@@global.var_name` , qui fait référence à des variables système non−structurées.

Actuellement, les deux premières règles ne peuvent pas être violées, puisqu'il n'y a qu'un seul type de variables système structurées, celui des caches de clés. Ces règles prendront leur importance si d'autres types de variables structurées sont créées dans le futur.

A une exception près, il est possible de faire référence à une variable système structurée en utilisant des noms composés, dans un contexte où un nom de variable simple est utilisé. Par exemple, vous pouvez assigner une valeur à une variable structurée en utilisant la ligne de commande suivante :

```
shell> mysqld --hot_cache.key_buffer_size=64K
```

Dans un fichier d'options, faites ceci :

```
[mysqld]
hot_cache.key_buffer_size=64K
```

Si vous lancez le serveur avec cette option, il va créer un cache de clé appelé `hot_cache` , avec une taille de 64 ko, en plus du cache de clé par défaut, qui a une taille de 8 Mo.

Supposez que vous démarriez le serveur avec ces options :

```
shell> mysqld --key_buffer_size=256K \
        --extra_cache.key_buffer_size=128K \
        --extra_cache.key_cache_block_size=2096
```

Dans ce cas, le serveur utilise une taille par défaut de 256 ko pour le cache. Vous pourriez aussi écrire `--default.key_buffer_size=256K` . De plus, le serveur crée un second cache de clé appelé `extra_cache` , de taille 128 ko, avec une taille de bloc de buffer de 2096 octets.

L'exemple suivant démarre le serveur avec trois cache de clé différents, avec des tailles de ratio 3:1:1 :

```
shell> mysqld --key_buffer_size=6M \
        --hot_cache.key_buffer_size=2M \
        --cold_cache.key_buffer_size=2M
```

Les variables système structurées peuvent être affectées et lue durant l'exécution. Par exemple, pour créer un cache de clé appelé `hot_cache` , de taille 10 Mo, utilisez une des commandes suivantes :

```
mysql> SET GLOBAL hot_cache.key_buffer_size = 10*1024*1024;
mysql> SET &#64;&#64;global.hot_cache.key_buffer_size = 10*1024*1024;
```

Pour lire la taille du cache, faites ceci :

```
mysql> SELECT &#64;&#64;global.hot_cache.key_buffer_size;
```

Cependant, la commande suivante ne fonctionne pas. La variable n'est pas interprétée comme un nom composé, mais comme une simple chaîne pour l'opérateur `LIKE` :

```
mysql> SHOW GLOBAL VARIABLES LIKE 'hot_cache.key_buffer_size';
```

C'est la exception où vous ne pouvez pas utiliser une variable système structurée, là où une variable simple fonctionnerait.

10.5 Syntaxe des commentaires

Le serveur MySQL supporte trois types de commentaires :

- Depuis '`#`' jusqu'à la fin de la ligne.
- Depuis '`-- `' jusqu'à la fin de la ligne. Ce style est supporté depuis MySQL 3.23.3. Notez que le commentaire '`-- `' (double−tiret) requiert que le second tiret soit suivi d'un espace ou par

un autre caractère de contrôle comme une nouvelle ligne. Cette syntaxe diffère légèrement des standards SQL, comme discuté dans la section `'--'` comme début de commentaire .
- Depuis `'/*'` jusqu'à `'*/'` . La séquence de fermeture n'a pas besoin d'être sur la même ligne, ce qui permet de répartir le commentaire sur plusieurs lignes.

Voici un exemple avec les trois types de commentaires :

```
 mysql> SELECT 1+1;     # Ce commentaire se continue jusqu'à la fin de la ligne
mysql> SELECT 1+1;     -- Ce commentaire se continue jusqu'à la fin de la ligne
mysql> SELECT 1 /* Ceci est un commentaire dans la ligne */ + 1;
mysql> SELECT 1+
/*
Ceci est un commentaire
sur plusieurs lignes
*/
1;
```

La syntaxe des commentaires décrites ici s'applique à l'analyseur du serveur `mysqld` , lorsqu'il traite les commandes SQL. Le client `mysql` peut aussi effectuer des analyses de commandes avant de les envoyer : par exemple, il recherche les limites de requêtes dans les commandes multi−lignes. Cependant, il y a des limitations dans la façon de gérer les commentaires `/* ... */` :

- Les guillemets (simples et doubles) sont considérés comme des indications de début de chaîne, même dans un commentaire. Si le guillemet n'est pas refermé (par un second guillemet), l'analyseur ne réalisera pas que le commentaire est fini. Si vous utilisez mysql interactivement, vous pouvez vous en apercevoir, car il va modifier l'invite de commande de `mysql>` en `'>` ou `">` .
- Un point−virgule sert à indiquer la fin de la commande SQL, et tout ce qui suit un point−virgule est considéré comme étant une nouvelle requête.

Ces limitations s'appliquent aussi bien à `mysql` en ligne de commande, que lorsque vous demandez à `mysql` de lire des commandes depuis un fichier (`mysql < un-fichier`).

10.6 Cas des mots réservés MySQL

Un problème récurrent provient de la tentative de création de tables avec des noms de colonnes qui sont des types de champs ou des fonctions natives de MySQL, comme `TIMESTAMP` ou `GROUP` . Il vous st permis de le faire (par exemple `ABS` est permis comme nom de colonne), mais les espaces ne sont pas permis entre le nom d'une fonction et la première `'('` suivante lors de l'utilisation de fonctions qui sont aussi des noms de colonnes.

Un effet secondaire de ce comportement est que l'omission d'espace dans certains contexte fait que l'identifiant est interprété comme un nom de fonction. Par exemple, cette commande est valide :

```
 mysql> CREATE TABLE abs (val INT);
```

Mais omettre l'espace après `abs` génère une erreur de syntaxe, car la commande semble utiliser la fonction `ABS()` :

```
 mysql> CREATE TABLE abs(val INT);
```

Si vous lancez le serveur l'option de mode `IGNORE_SPACE` , le serveur autorisera l'appel de fonction avec un espace entre le nom de la fonction et le caractère de parenthèse ouvrante `'('` suivant. Les noms de fonctions sont alors considérés comme des mots réservés. Comme pour les résultats, les noms de colonnes qui sont identiques au nom de fonctions doivent être placés entre guillemets, tels que décrit dans Noms de base de données, tables, index, colonnes et alias . Le mode SQL du serveur est contrôlé par la procédure de la section Choisir son mode SQL .

Les mots suivants sont explicitement réservés en MySQL. La plupart sont interdits par ANSI SQL92 en tant que nom de colonnes ou de tables (par exemple, `GROUP`). Quelques uns sont réservés parce que MySQL en a besoin et utilise (actuellement) un analyseur `yacc` :

Les symboles suivants (issus de la table ci−dessus) sont interdits par ANSI SQL mais permis par MySQL en tant que noms de colonnes ou de tables. Cela est dû au fait que ces noms sont très courants, et de nombreux programmeur les ont déjà utilisés.

- `ACTION`
- `BIT`
- `DATE`
- `ENUM`
- `NO`
- `TEXT`
- `TIME`
- `TIMESTAMP`

11 Jeux de caractères et Unicode

L'amélioration du support des jeux de caractères est un des progrès de MySQL en version 4.1. Ce chapitre couvre :

- Ce que sont les jeux de caractères et les collations
- Le système par défaut à plusieurs niveaux
- Nouvelle syntaxe en MySQL 4.1
- Fonctions et opérations affectées
- Support d'Unicode
- La signification de chaque jeu de caractères et collation

Le support des jeux de caractères est inclus dans les tables `MySISAM` , `MEMORY` (`HEAP`), et depuis MySQL 4.1.2, `InnoDB` . Le moteur de table `ISAM` n'inclut pas le support des jeux de caractères, et il n'y a pas de plan pour le faire car `ISAM` est abandonnée.

11.1 Jeux de caractères et collation : généralités

Un **jeu de caractères** est un ensemble de symboles et de codes. Une **collation** est un ensemble de règles permettant la comparaisons de caractères dans un jeu. Pour rendre ces définitions plus concrètes, voici un exemple avec un alphabet imaginaire.

Supposons que nous avons un alphabet de 4 lettres : `'A'` , `'B'` , `'a'` , `'b'` . Nous assignons à chaque lettre un nombre comme ceci : `'A'` = 0, `'B'` = 1, `'a'` = 2, `'c'` = 3. La lettre `'A'` est un symbole, le chiffre 0 est le **code** de `'A'` , et la combinaison des quatre lettres et de leur code forme le **jeu de caractères** .

Maintenant, supposons que nous voulions comparer deux chaînes de caractères : `'A'` et `'B'` . Le plus simple pour cela est de regarder leurs codes : 0 pour `'A'` et 1 pour `'B'` , et comme 0 est inférieure à 1, nous pouvons dire que `'A'` est plus petit que `'B'` . Ce que nous venons de faire est une collation pour notre jeu de caractères. la collation est un ensemble de règle, qui se résume à ceci dans notre cas : ``compare les codes''. Cette règle est la plus simple collation **binaire** .

Si nous devons différentier les majuscules des minuscules, nous aurons au moins deux règles : (1) traiter les minuscules `'a'` et `'b'` comme des équivalents de `'A'` et `'B'` ; (2) puis comparer leurs codes respectifs. Nous appelons cette règle une collation **sensible à la casse** . C'est un peu plus complexe que la règle précédente.

En réalité, la plupart des jeux de caractères ont de nombreux caractères : ce n'est pas simplement `'A'` et `'B'` mais des alphabets entiers, ou des systèmes d'écriture orientaux avec des milliers de caractères, incluant des caractères spéciaux et la ponctuation. Dans la vraie vie, une collation a de très nombreuses règles, concernant la sensibilité à la casse ou encore l'insensibilité aux accents (un accent est une marque attachée aux lettres comme le `'ó'`) allemand) et les caractères multiples comme le e dans l'o `''` = `'oe'` de l'une des deux collations allemandes.

MySQL 4.1 peut faire cela pour vous :

- Stocker des chaînes dans différents jeux de caractères
- Comparer des chaînes à l'aide de différentes collations
- Mélanger différents jeux de caractères et collations sur le même serveur, la même base ou même la même table.
- Permettre la spécification du jeu de caractère et de la collation à n'importe quel niveau

Sous cet angle, MySQL 4.1 est bien plus souple que MySQL 4.0 et que les autres bases de données. Mais, pour pouvoir utiliser ces nouvelles fonctionnalités, vous devez savoir quels sont les jeux de caractères et les collations disponibles, comment les modifier ou comment les utiliser avec les opérateurs.

11.2 Jeux de caractères et collation dans MySQL

Un jeu de caractères a toujours au moins une collation. Pour lister les jeux de caractères disponibles, utilisez la commande `SHOW CHARACTER SET` :

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+
| Charset  | Description                 | Default collation   |
+----------+-----------------------------+---------------------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |
| dec8     | DEC West European           | dec8_swedish_ci     |
| cp850    | DOS West European           | cp850_general_ci    |
| hp8      | HP West European            | hp8_english_ci      |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |
| latin1   | ISO 8859-1 West European    | latin1_swedish_ci   |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |
...
```

Le résultat inclut en réalité une autre colonne, qui n'est pas présentée dans cette page, à des fins de publication.Tout jeu de caractères a toujours au moins une collation. Il peut en avoir plusieurs.

Pour lister les collations d'un jeu de caractères, utilisez la commande `SHOW COLLATION` . Par exemple, pour afficher les collations du jeu de caractères `latin1` (``ISO−8859−1 West European''), utilisez cette commande, et recherchez les noms de collation qui commencent par `latin1` :

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------+---------+
| Collation         | Charset | Id | Default | Compiled | Sortlen |
+-------------------+---------+----+---------+----------+---------+
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       1 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       1 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
| latin1_spanish_ci | latin1  | 94 |         |          |       0 |
+-------------------+---------+----+---------+----------+---------+
```

Les collations `latin1` ont les significations suivantes :

Collation	**Signification**
latin1_bin	Binaire, suivant l'encodage latin1
latin1_danish_ci	Danois/Norvégien
latin1_general_ci	Multilingue
latin1_general_cs	Multilingue, sensible à la casse
latin1_german1_ci	Allemand DIN-1
latin1_german2_ci	Allemand DIN-2
latin1_spanish_ci	Espagnol moderne
latin1_swedish_ci	Suédois/Finlandais

Les collations ont les caractéristiques suivantes :

- Deux jeux de caractères différents ne peuvent pas avoir la même collation.
- Chaque jeu de caractère a une collation qui est la **collation par défaut** . Par exemple, la collation par défaut de `latin1` est `latin1_swedish_ci` .
- Il y a une convention pour les noms de collation : elles commencent par le nom du jeu de caractères auquel elles sont associées; elles incluent généralement un nom de langue, et finissent par `_ci` (`case insensitive` , insensible à la casse), `_cs` (`case sensitive` , sensible à la casse), `_bin` (binaire), ou `_uca` (Algorithme Unicode, Unicode Collation Algorithm).

11.3 Déterminer le jeu de caractères et la collation par défaut

Il y a une configuration par défaut pour chaque jeu de caractères et collation à quatre niveaux : serveur, base de données, table, connexion. La description suivante peut paraître complexe, mais il a été montré qu'en pratique la mise par défaut par niveaux multiples mène à des résultats simples et évidents.

11.3.1 Jeu de caractères et collation serveur

Le serveur MySQL a un jeu de caractères et une collation serveur, qui ne peuvent pas être nuls.

MySQL détermine le jeu de caractères et la collation serveurs comme suit :

- En fonction de l'option de configuration active quand le serveur démarre.
- En fonction des valeurs de configuration à l'exécution.

A ce niveau, la décision est simple. Le jeu de caractères serveur et sa collation dépendent des options que vous utilisez au démarrage de `mysqld` . Vous pouvez utiliser `--default-character-set=character_set_name` comme jeu de caractères et vous pouvez en même temps ajouter `--default-collation=collation_name` pour la collation. Si vous n'indiquez pas de jeu de caractères, cela revient à dire `--default-character-set=latin1` . Si vous indiquez un jeu de caractères (par exemple, `latin1`) mais pas de collation, cela revient à dire : `--default-charset=latin1 --collation=latin1_swedish_ci` car `latin1_swedish_ci` est la collation par défaut de `latin1` . par conséquent, les trois commandes suivantes ont toutes le même effet :

```
shell> mysqld
shell> mysqld --default-character-set=latin1
shell> mysqld --default-character-set=latin1
          --default-collation=latin1_swedish_ci
```

Une façon de changer la configuration par défaut est de recompiler MySQL. Si vous voulez changer le jeu de caractères et la collation par défaut du serveur quand vous compilez depuis les sources, utilisez `--with-character-set` et `--with-collation` comme arguments pour `configure` . Par exemple :

```
shell> ./configure --with-character-set=latin1
```

ou :

```
shell> ./configure --with-character-set=latin1
          --with-collation=latin1_german1_ci
```

`mysqld` et `configure` vérifient aussi que la combinaison jeu de caractères/collation est valide. Ces programmes affichent un message erreur et se terminent si la combinaison n'est pas valide.

11.3.2 Jeu de caractères et collation de base de données

Toute base de données a un jeu de caractères de base de données et une collation de base de données, qui ne peuvent pas être nulles. Les commandes `CREATE DATABASE` et `ALTER DATABASE` permettent de utiliser optionnellement ces deux attributs :

```
CREATE DATABASE db_name
  [DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]]

ALTER DATABASE db_name
   [DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]]
```

Exemple :

```
CREATE DATABASE db_name
  DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
```

MySQL choisit le jeu de caractères et la collation de base de données comme ceci :

- Si les deux clauses `CHARACTER SET X` et `COLLATE Y` ont été spécifiées, alors leurs valeurs sont utilisées.
- Si `CHARACTER SET X` a été spécifiée sans `COLLATE` , alors le jeu de caractères est `X` et sa collation par défaut.
- Sinon, le jeu de caractères et la collation par défaut du serveur sont utilisés.

La syntaxe MySQL `CREATE DATABASE ... DEFAULT CHARACTER SET ...` est analogue à la syntaxe du standard SQL `CREATE SCHEMA ... CHARACTER SET ...` . Il est donc possible de créer des bases de données avec différents jeux de caractères et collations, sur le même serveur MySQL.

Le jeu de caractères et la collation sont utilisées comme valeur par défaut pour les tables, lorsque ces informations ne sont pas spécifiées dans les commandes `CREATE TABLE` . Elles n'ont pas d'autres utilité.

11.3.3 Jeu de caractères de tables et collation

Chaque table a un jeu de caractères et une collation de table qui ne peut pas être nulle. Les commandes `CREATE TABLE` et `ALTER TABLE` ont maintenant des options pour préciser le jeu de caractères et la collation :

```
CREATE TABLE table_name ( column_list )
  [DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]]

ALTER TABLE table_name
  [DEFAULT CHARACTER SET character_set_name] [COLLATE collation_name]
```

Exemple :

```
CREATE TABLE t1 ( ... ) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

MySQL choisit le jeu de caractère et la collation :

- Si `CHARACTER SET X` et `COLLATE Y` sont précisés tous les deux, alors on adopte le jeu de caractères `X` et la collation `Y` .
- Si `CHARACTER SET X` était précisé sans `COLLATE` , alors on adopte le jeu de caractères `X` et sa collation par défaut.
- Sinon, le jeu de caractères et la collation de la base de données sont utilisés.

Le jeu de caractères et la collation sont utilisés comme valeurs par défaut si ces deux attributs ne sont pas précisés par la définition d'une colonne. Le jeu de caractères et la collation sont des extensions MySQL : il n'y a pas de telles fonctionnalités en SQL standard.

11.3.4 Jeu de caractères de colonne et collation

Chaque colonne de `` `character` '' (c'est à dire les colonnes de type `CHAR` , `VARCHAR` , ou `TEXT`) a un jeu de caractères de colonne et une collation de colonne qui ne peut pas être nulle. La syntaxe de définition de colonne a maintenant des clauses optionnelles pour préciser le jeu de caractères de colonne et la collation :

```
column_name &#123;CHAR | VARCHAR | TEXT&#125; (column_length)
   [CHARACTER SET character_set_name [COLLATE collation_name]]
```

Exemple :

```
CREATE TABLE Table1
(
   column1 VARCHAR(5) CHARACTER SET latin1 COLLATE latin1_german1_ci
);
```

MySQL choisit le jeu de caractères et la collation de colonne comme ceci :

- Si `CHARACTER SET X` et `COLLATE Y` sont spécifiés, alors le jeu de caractères est `X` et la collation `Y` .
- Si `CHARACTER SET X` a été spécifié sans la clause `COLLATE` , alors le jeu de caractères est `X` et la collation est sa collation par défaut.
- Sinon, MySQL utilise le jeu de caractères et la collation par défaut.

Les clauses `CHARACTER SET` et `COLLATE` font partie du standard SQL.

11.3.5 Exemples d'attribution de jeu de caractères et collation

Les exemples suivants montrent comment MySQL détermine le jeu de caractère par défaut et les valeurs de collation. **Exemple 1 : définition de table et colonne**

```
CREATE TABLE t1
(
  c1 CHAR(10) CHARACTER SET latin1 COLLATE latin1_german1_ci
) DEFAULT CHARACTER SET latin2 COLLATE latin2_bin;
```

Vous avez ici une colonne avec un jeu de caractères `latin1` et une collation `latin1_german1_ci` . La définition est explicite, c'est donc direct. Veuillez noter qu'il n'y a pas de problème à stocker une colonne `latin1` dans une table `latin2` . **Exemple 2 : définition de table et colonne**

```
CREATE TABLE t1
(
   c1 CHAR(10) CHARACTER SET latin1
) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

Cette fois−ci nous avons une colonne avec un jeu de caractères `latin1` et une collation par défaut. Maintenant, bien que cela puisse sembler naturel, la collation par défaut n'est pas spécifiée au niveau de la table. A la place, comme la collation par défaut de `latin1` est toujours `latin1_swedish_ci` , la colonne `c1` aura la collation `latin1_swedish_ci` (et non `latin1_danish_ci`).

Exemple 3 : définition de table et colonne

```
CREATE TABLE t1
(
   c1 CHAR(10)
) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

Nous avons une colonne avec un jeu de caractères par défaut et une collation par défaut. Dans ces conditions, MySQL regarde au niveau de la table pour déterminer le jeu de caractères et la collation. Par conséquent le jeu de caractères de la colonne `c1` est `latin1` et sa collation est `latin1_danish_ci` . **Exemple 4: définition de base, table et colonne**

```
CREATE DATABASE d1 DEFAULT CHARACTER SET latin2 COLLATE latin2_czech_ci;
USE d1;
CREATE TABLE t1
(
   c1 CHAR(10)
);
```

Nous créons une colonne sans préciser son jeu de caractère ni sa collation. Nous ne spécifions pas non plus de jeu de caractères ni de collation au niveau de la table. Dans ces conditions, MySQL cherche au niveau de la base de données. (La configuration de la base de donnée devient celle de la table, et par conséquent celle de la colonne.) Donc le jeu de caractères de colonne `c1` est `latin2` et sa collation est `latin2_czech_ci` .

11.3.6 Jeux de caractères et collations de connexion

Plusieurs variables contrôlent les jeux de caractères et collation du système pour un client. Certaines ont déjà été mentionnées précédemment :

- Le jeu de caractères et la collation sont disponibles dans les variables `character_set_server` et `collation_server` .
- Le jeu de caractères et la collation par défaut de la base de données sont disponibles dans `character_set_database` et `collation_database` .

D'autres variables et collations sont impliquées dans la gestion des connexions entre un client et un serveur. Chaque client a un jeu de caractères et une collation attitrés.

Pensez à ce qu'est une "connexion" : c'est ce que vous faites lorsque vous vous connectez au serveur. Le client envoie des SQL commandes comme des requêtes, au travers de la connexion, vers le serveur. Le serveur renvoie des réponses, comme des jeux de résultats, au client, au travers de la connexion. Ceci mène à plusieurs questions telles que :

- Dans quel jeu de caractères est la requête lorsqu'elle quitte le client?

  Le serveur utilise la variable `character_set_client` pour connaître le jeu de caractères des requêtes émises par le client.
- dans quel jeu de caractère le serveur devrait il traduire la requête après l'avoir reçue?

  Pour cela, `character_set_connection` et `collation_connection` sont utilisées par le serveur. Il convertit les requêtes envoyées par le client de `character_set_client` en `character_set_connection` (hormis les chaînes littérales qui sont précédées de `_latin1` ou `_utf8`). `collation_connection` est importante pour les comparaisons de chaînes littérales. Pour les comparaisons de chaînes avec des colonnes, la collation de la colonne a la priorité.
- Dans quel jeu de caractères le serveur devrait−il traduire les résultats ou messages d'erreur avant de les renvoyer au client?

  La variable `character_set_results` indique le jeu de caractères que le serveur utilise pour retourner les résultats aux clients. Cela inclut les données telles que les noms de colonnes ou les meta−données.

Vous pouvez configurez ces options là, ou vous pouvez vous fier aux configurations par défaut (auquel cas vous pouvez sauter cette section).

Il y a deux commandes qui permettent de modifier le jeu de caractères de la connexion :

```
 SET NAMES 'charset_name'
SET CHARACTER SET charset_name
```

`SET NAMES` indique ce qui est dans la commande SQL que le client envoie. Par conséquent, `SET NAMES cp1251` indique au serveur : ``les futurs messages fournis par ce client seront dans le jeu de caractères `cp1251` '' et le serveur est libre de les traduire dans son propre jeu de caractères, éventuellement.La commande `SET NAMES 'x'` est équivalente à ces trois commandes :

```
 mysql> SET character_set_client = x;
mysql> SET character_set_results = x;
mysql> SET character_set_connection = x;
```

`SET CHARACTER SET` est similaire, mais spécifie le jeu de caractères et la collation par défaut des bases pour la connexion. Une commande `SET CHARACTER SET x` est équivalente à :

```
 mysql> SET character_set_client = x;
mysql> SET character_set_results = x;
mysql> SET collation_connection = &#64;&#64;collation_database;
```

Lorsque vous exécutez la commande `SET NAMES` ou `SET CHARACTER SET` , vous changez aussi la collation de la connexion. Cependant, la collation de connexion existe uniquement par cohérence. Généralement sa valeur n'a pas d'importance.

Avec le client `mysql` , il n'est pas nécessaire d'exécuter la commande `SET NAMES` à chaque démarrage. Vous pouvez ajouter l'option `--default-character-set-name` dans la ligne de commande de `mysql` , ou dans le fichier d'options. Par exemple, la ligne suivante est exécutée automatiquement à chaque démarrage de `mysql` :

```
[mysql]
default-character-set=koi8r
```

EXEMPLE : supposez que `column1` est défini par `CHAR(5) CHARACTER SET latin2` . Si vous n'indiquez pas `SET CHARACTER SET` , alors la commande `SELECT column1 FROM t` retournera les valeurs de la colonne `column1` en utilisant le jeu de caractères `latin2` . Si, d'un autre coté, vous utilisez la commande `SET CHARACTER SET latin1` , le serveur va alors convertir le résultat de `latin2` en `latin1` juste avant de l'envoyer. De telles conversion sont lentes.

Si vous ne voulez pas que le serveur fasse des conversions, utilisez la valeur `NULL` à `character_set_results` :

```
mysql> SET character_set_results = NULL;
```

11.3.7 Jeu de caractères et collation des chaînes littérales

Chaque chaîne de caractères littérale a un jeu de caractères et une collation qui ne peut pas être nulle.

Une chaîne de caractères littérale peut avoir un spécificateur optionnel de jeu de caractères optionnel et une clause `COLLATE` :

```
[_character_set_name]'string' [COLLATE collation_name]
```

Exemples :

```
SELECT 'string';
SELECT _latin1'string';
SELECT _latin1'string' COLLATE latin1_danish_ci;
```

La simple commande `SELECT 'string'` utilise le jeu de caractères par défaut de la connexion.

L'expression `_character_set_name` est formellement appelée un spécificateur (`introducer`). Elle indique à l'analyseur : "la chaîne qui va suivre utilise le jeu de caractère `X` ." Cela a été source de confusions par le passé; aussi nous insistons sur le fait qu'un spécificateur ne provoque pas de conversion. C'est strictement une indication, qui ne change pas la valeur de la chaîne. Un spécificateur est aussi autorisé avant des notations littérales hexadécimale et numérique (`x'literal'` et `0xnnnn`), et avant `?` (qui est une substitution de paramètre lorsque l'on utilise des commandes préparées avec une interface de langage de programmation).

Exemples :

```
SELECT _latin1 x'AABBCC';
SELECT _latin1 0xAABBCC;
SELECT _latin1 ?;
```

MySQL détermine le jeu de caractères et la collation de la façon suivante :

- Si, à la fois `_X` et `COLLATE Y` ont été précisés, alors le jeu de caractères est `X` et la collation littérale est `Y` .
- Si `_X` est précisé mais que `COLLATE` ne l'est pas, alors le jeu de caractères de la chaîne littérale

est `x` et la collation est la collation par défaut du jeu de caractères `x` .
- Sinon, on utilise le jeu de caractères et la collation par défaut de la connexion.

Exemples:

- Une chaîne avec le jeu de caractères `latin1` et la collation `latin1_german1_ci` :
```
SELECT _latin1'Müller' COLLATE latin1_german1_ci;
```
- Une chaîne avec le jeu de caractères `latin1` et sa collation par défaut, c'est à dire `latin1_swedish_ci` :
```
SELECT _latin1'Müller';
```
- Une chaîne avec le jeu de caractère et la collation connexion/littérale :
```
SELECT 'Müller';
```

Les introducteurs de jeux de caractères et la clause `COLLATE` sont implémentés selon les spécifications standard SQL.

11.3.8 Clause COLLATE dans différentes parties d'une requête SQL

Avec la clause `COLLATE` , vous pouvez remplacer la collation par défaut, quelle qu'elle soit, par une comparaison. `COLLATE` peut être utilisé dans différentes parties des requêtes SQL. Voici quelques exemples :

- Avec `ORDER BY` :
```
SELECT k
FROM t1
ORDER BY k COLLATE latin1_german2_ci;
```
- Avec `AS` :
```
SELECT k COLLATE latin1_german2_ci AS k1
FROM t1
ORDER BY k1;
```
- Avec `GROUP BY` :
```
SELECT k
FROM t1
GROUP BY k COLLATE latin1_german2_ci;
```
- Avec les fonctions d'agrégation :
```
SELECT MAX(k COLLATE latin1_german2_ci)
FROM t1;
```
- Avec `DISTINCT` :
```
SELECT DISTINCT k COLLATE latin1_german2_ci
FROM t1;
```
- Avec `WHERE` :
```
SELECT *
FROM t1
WHERE _latin1 'Müller' COLLATE latin1_german2_ci = k;
```
- Avec `HAVING` :
```
SELECT k
FROM t1
GROUP BY k
HAVING k = _latin1 'Müller' COLLATE latin1_german2_ci;
```

11.3.9 COLLATE clause de précédence

La clause `COLLATE` a précédence (plus haute que ||), donc les expressions suivantes sont équivalentes :

```
x || y COLLATE z
x || (y COLLATE z)
```

11.3.10 Opérateur BINARY

L'opérateur `BINARY` est un raccourci pour une clause `COLLATE` . Par exemple `BINARY 'x'` est équivalent à `'x' COLLATE y` , où `y` est le nom d'une collation binaire appropriée. Par exemple, si nous supposons que `a` appartient au jeu de caractères `latin1` , les deux requêtes suivantes ont le même effet :

```
SELECT * FROM t1 ORDER BY BINARY a;
SELECT * FROM t1 ORDER BY a COLLATE latin1_bin;
```

Note : Chaque jeu de caractères a une collation binaire.

11.3.11 Quelques cas spéciaux où la détermination de la collation est difficile

Dans la grande majorité des requêtes, la collation utilisée par MySQL pour résoudre une comparaison est évidente. Par exemple, dans les cas suivants, il devrait être clair que la collation sera "la collation de colonne de la colonne `x` " :

```
SELECT x FROM T ORDER BY x;
SELECT x FROM T WHERE x = x;
SELECT DISTINCT x FROM T;
```

Cependant, quand des opérandes multiples sont, en jeu il peut y avoir des ambiguïtés. Par exemple :

```
SELECT x FROM T WHERE x = 'Y';
```

Cette requête devrait elle utiliser la collation de la colonne `x` , ou celle de la chaîne littérale `'Y'` ?

Le standard SQL résout de telles questions en utilisant ce qui portait le nom de règles coercitives (`` `coercibility` ''). L'essence de la question est : puisque `x` et `'Y'` ont tous deux des collations différentes, laquelle a priorité? C'est une question complexe, mais ces règles devraient résoudre la plupart des situations :

- Une clause explicite `COLLATE` a pour priorité 4.
- Une concaténation de deux chaînes avec des collations différentes à pour priorité 3.
- La collation d'une colonne a priorité 2.
- La collation d'une chaîne littérale a pour priorité 1.

Ces règles résolvent les ambiguités de la façon suivante :

- Utiliser la collation qui a la précédence la plus élevée.
- Si les deux opérandes ont une collation de même priorité, alors il y a erreur si les collations ne sont pas les mêmes.

Exemples :

column1 = 'A'	Utilise la collation de la colonne column1
column1 = 'A' COLLATE x	Utilise la collation de 'A'
column1 COLLATE x = 'A' COLLATE y	Erreur

La fonction `COERCIBILITY()` peut être utilisée pour déterminer la coercibilité d'une chaîne :

```
mysql> SELECT COERCIBILITY('A' COLLATE latin1_swedish_ci);
        -> 0
mysql> SELECT COERCIBILITY('A');
        -> 3
```

Fonctions d'informations .

11.3.12 Les collation doivent correspondre au bon jeu de caractères

Souvenez−vous que chaque jeu de caractères a une ou plusieurs collations, et que chaque collation est associée à un et un seul un jeu de caractères. Par conséquent, la commande suivante engendre un message erreur car la collation `latin2_bin` n'est pas autorisée avec le jeu de caractères `latin1` :

```
mysql> SELECT _latin1 'x' COLLATE latin2_bin;
ERROR 1251: COLLATION 'latin2_bin' is not valid
for CHARACTER SET 'latin1'
```

Dans certaines cas, les expressions qui fonctionnaient avant MySQL 4.1 échoueront en MySQL 4.1 si vous ne prenez pas en compte les collations et jeux de caractères. Par exemple, avant la version 4.1, cette commande fonctionnait comme :

```
mysql> SELECT SUBSTRING_INDEX(USER(),'&#64;',1);
+-------------------------------+
| SUBSTRING_INDEX(USER(),'&#64;',1) |
+-------------------------------+
| root                          |
+-------------------------------+
```

Après une mise à jour en MySQL 4.1, la commande échoue :

```
mysql> SELECT SUBSTRING_INDEX(USER(),'&#64;',1);
ERROR 1267 (HY000): Illegal mix of collations
(utf8_general_ci,IMPLICIT) and (latin1_swedish_ci,COERCIBLE)
for operation 'substr_index'
```

La raison à cela est que les noms d'utilisateurs sont stockées en UTF8 (UTF8 pour metadata). Par conséquent, la fonction `USER()` et la chaîne littérale `'@'` ont des jeux de caractères différents et des collations différentes :

```
mysql> SELECT COLLATION(USER()), COLLATION('&#64;');
+-------------------+-------------------+
| COLLATION(USER()) | COLLATION('&#64;')    |
+-------------------+-------------------+
| utf8_general_ci   | latin1_swedish_ci |
+-------------------+-------------------+
```

Un moyen pour corriger cela est de dire à MySQL qu'il doit interpréter les chaînes littérales avec le jeu de caractères s `utf8` :

```
mysql> SELECT SUBSTRING_INDEX(USER(),_utf8'&#64;',1);
+------------------------------------+
| SUBSTRING_INDEX(USER(),_utf8'&#64;',1) |
+------------------------------------+
| root                               |
+------------------------------------+
```

Un autre moyen est de changer le jeu de caractères et la collation de la connexion en `utf8` . Vous pouvez aussi utiliser la commande `SET NAMES 'utf8'` ou les variables système `character_set_connection` et `collation_connection` .

11.3.13 Un exemple de l'effet de collation

Supposons que la colonne `X` dans la table `T` a ces valeurs de colonne `latin1` :

```
Muffler
Müller
MX Systems
MySQL
```

Et supposons que les valeurs de la colonne soient récupérées en utilisant la commande suivante :

```
SELECT X FROM T ORDER BY X COLLATE collation_name;
```

L'ordre résultant des valeurs pour différentes collations est montré dans cette table :

latin1_swedish_ci	latin1_german1_ci	latin1_german2_ci
Muffler	Muffler	Müller
MX Systems	Müller	Muffler
Müller	MX Systems	MX Systems

MySQL	MySQL	MySQL

Cette table est un exemple qui montre quel effet cela aurait si l'on utilisait des collations différentes dans une clause `ORDER BY` . Le caractère qui pose problème dans cet exemple est le U avec deux points dessus. Les allemands l'appellent U−umlaut, mais nous l'appellerons U−tréma.

- La première colonne montre le résultat de `SELECT` en utilisant la collation Suédoise/Finlandaise, qui dit que le u−tréma est trié comme le Y
- La seconde colonne montre le résultat de `SELECT` en utilisant la règle German DIN−1, qui dit que le U−tréma est trié comme le U.
- La troisième colonne montre le résultat de `SELECT` en utilisant la règle German DIN−2 qui dit que le u−tréma est trié comme le UE.

Trois collations différentes engendrent trois résultats différents. MySQL est là pour gérer cela. En utilisant la collation appropriée, vous pouvez choisir l'ordre de tri que vous voulez.

11.4 Opérations affectées par le support de jeux de caractères.

Cette section décrit des opérations qui prennent maintenant en compte des informations sur le jeu de caractères.

11.4.1 Chaînes résultats

MySQL a de nombreux opérateurs et fonctions qui retournent une chaîne. Cette section répond à la question : quels sont le jeu de caractères et la collation d'une telle chaîne?

Pour des fonctions simples qui prennent une chaîne en entrée et renvoient une chaîne comme résultat en sortie, le jeu de caractères du résultat et sa collation sont les mêmes que deux de l'argument principal. Par exemple, `UPPER(X)` renvoie une chaîne dont la chaîne de caractères et la collation sont les mêmes que celles de `X` .

Le même concept s'applique à : `INSTR()` , `LCASE()` , `LOWER()` , `LTRIM()` , `MID()` , `REPEAT()` , `REPLACE()` , `REVERSE()` , `RIGHT()` , `RPAD()` , `RTRIM()` , `SOUNDEX()` , `SUBSTRING()` , `TRIM()` , `UCASE()` , `UPPER()` . Veuillez aussi noter que la fonction `REPLACE()` , contrairement à toutes les autres fonctions, ignore la collation de la chaîne d'entrée et effectue une comparaison insensible à la casse à chaque fois.

Pour des opérations qui combinent des entrées de chaînes multiples et renvoient une chaîne seule en sortie, les "règles d'agrégation" du standard SQL99 s'appliquent. Les règles sont :

- Si `COLLATE X` est explicitement donné, alors on utilise `X` .
- Si `COLLATE X` et `COLLATE Y` sont explicitement donnés, alors erreur.
- Sinon, si toutes les collations sont la même collation `X` , alors utilise `X` .
- Sinon, le résultat n'a pas de collation.

Par exemple, avec `CASE ... WHEN a THEN b WHEN b THEN c COLLATE X END` , la collation résultante est `X` . L'exemple s'applique à : `CONCAT()` , `GREATEST()` , `IF()` , `LEAST()` , `CASE` , `UNION` , `||` , `ELT()` .

Pour des opérations qui convertissent vers des données caractères, le jeu de caractères de la chaîne résultante et sa collation sont dans le `jeu de caractères de connexion` et ont la `collation de connexion` .

Ceci s'applique à : `CHAR()` , `CAST()` , `CONV()` , `FORMAT()` . `HEX()` , `SPACE()` .

11.4.2 CONVERT()

`CONVERT()` fournit une méthode pour convertir des données entre différents jeux de caractères. La syntaxe est :

```
CONVERT(expr USING transcoding_name)
```

Avec MySQL, les nom de conversions sont les mêmes que les noms des jeux caractères correspondants. Exemples :

```
SELECT CONVERT(_latin1'Müller' USING utf8);
INSERT INTO utf8table (utf8column)
   SELECT CONVERT(latin1field USING utf8) FROM latin1table;
```

`CONVERT(... USING ...)` est implémenté selon le standard SQL−99.

11.4.3 CAST()

Vous pouvez aussi utiliser `CAST()` pour convertir une chaîne dans un jeu de caractères différent. Le nouveau format est :

```
CAST ( character_string AS character_data_type
   CHARACTER SET character_set_name )
```

Exemple :

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8);
```

Vous ne pouvez pas utiliser une clause `COLLATE` dans un `CAST()` , mais vous pouvez l'utiliser en dehors. Cela revient à dire que `CAST(... COLLATE ...)` est interdit mais `CAST(...) COLLATE ...` est autorisé.

Exemple :

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin;
```

Si vous utilisez `CAST()` sans préciser `CHARACTER SET` , alors le jeu de caractères et la collation résultants sont le jeu de caractères de la connexion et sa collation par défaut. Si vous utilisez `CAST()` avec `CHARACTER SET X` , alors le jeu de caractères résultant est `X` et la collation résultante est la collation par défaut de `X` .

11.4.4 Commande SHOW

Plusieurs commandes `SHOW` sont nouvelles ou modifiées en MySQL 4.1 pour fournir de nouvelles informations sur les jeux de caractères. `SHOW CHARACTER SET` , `SHOW COLLATION` et `SHOW CREATE DATABASE` sont nouveaux. `SHOW CREATE TABLE` et `SHOW COLUMNS` sont modifiés.

La commande `SHOW CHARACTER SET` affiche tous les jeux de caractères disponibles. Elle accepte la clause optionnelle `LIKE` qui indique quel noms de caractères rechercher. Par exemple :

```
mysql> SHOW CHARACTER SET LIKE 'latin%';
+---------+-----------------------------+-------------------+--------+
| Charset | Description                 | Default collation | Maxlen |
+---------+-----------------------------+-------------------+--------+
| latin1  | ISO 8859-1 West European    | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          | latin7_general_ci |      1 |
+---------+-----------------------------+-------------------+--------+
```

Syntaxe de `SHOW CHARACTER SET` .Le résultat de `SHOW COLLATION` inclut tous les jeux de caractères disponibles.

Elle accepte la clause optionnelle `LIKE` qui indique quels nom de collation rechercher. Par exemple :

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------+---------+
| Collation         | Charset | Id | Default | Compiled | Sortlen |
+-------------------+---------+----+---------+----------+---------+
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       0 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       0 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
| latin1_spanish_ci | latin1  | 94 |         |          |       0 |
+-------------------+---------+----+---------+----------+---------+
```

Syntaxe de `SHOW COLLATION` . `SHOW CREATE DATABASE` affiche la commande `CREATE DATABASE` qui va créer la base de données. Le résultat inclut toutes les options de bases de données. `DEFAULT CHARACTER SET` et `COLLATE` sont supportés. Toutes les options sont stockées dans un fichier texte, appelée `db.opt` qui peut être trouvé dans le fichier de base de données.

```
mysql> SHOW CREATE DATABASE a\G
*************************** 1. row ***************************
       Database: a
Create Database: CREATE DATABASE `a`
                 /*!40100 DEFAULT CHARACTER SET macce */
```

Syntaxe de `SHOW CREATE DATABASE` . `SHOW CREATE TABLE` est similaire, mais affiche la commande `CREATE TABLE` pour créer une table donnée. Les définitions de colonnes incluent maintenant toutes les spécifications de jeu de caractères, et les options de tables aussi.

Syntaxe de `SHOW CREATE TABLE` .

La commande `SHOW COLUMNS` affiche la collation des colonnes d'une table, lorsqu'elle est appelée avec `SHOW FULL COLUMNS` . Les colonnes de type `CHAR` , `VARCHAR` et `TEXT` ont des collations de type non− `NULL` . Les valeurs numériques et non−caractères on des collations `NULL` . Par exemple :

```
mysql> SHOW FULL COLUMNS FROM t;
+-------+---------+------------+------+-----+---------+-------+
| Field | Type    | Collation  | Null | Key | Default | Extra |
+-------+---------+------------+------+-----+---------+-------+
| a     | char(1) | latin1_bin | YES  |     | NULL    |       |
| b     | int(11) | NULL       | YES  |     | NULL    |       |
+-------+---------+------------+------+-----+---------+-------+
```

Le jeu de caractères ne fait pas partie de l'affichage. Le nom du jeu de caractère est lié à la collation. Syntaxe de `SHOW COLUMNS` .

11.5 Support de Unicode

Depuis la version 4.1 de MySQL, il y deux nouveaux jeux de caractères pour stocker des données Unicode :

- `ucs2` , le jeu de caractères Unicode UCS−2.
- `utf8` , l'encodage UTF−8 du jeu de caractères Unicode.

En UCS−2 (`binary Unicode representation`) chaque caractère est représenté par un code Unicode de deux octets avec l'octet le plus significatif en premier. Par exemple : "LATIN CAPITAL LETTER A" a le code 0x0041 et est stocké comme une séquence à deux octets `0x00 0x41` . "CYRILLIC SMALL LETTER YERU" (Unicode 0x044B) est stocké comme une séquence à deux octets `0x04 0x4B` . Pour les caractères Unicode et leurs codes veuillez consulter Unicode Home Page .

Restriction temporaire : UCS−2 ne peut pas (encore) être utilisé comme jeu de caractères client. Cela signifie que `SET NAMES ucs2` ne fonctionnera pas.

Le jeu de caractères UTF8 (`transform Unicode representation`) est une alternative pour stocker les données Unicode. il est implémenté selon la RFC 2279. L'idée du jeu de caractères UTF−8 est que différents caractères Unicode soient représentés par des séquences de différentes longueurs.

- les lettres, chiffres et caractères de ponctuation latins de base utilisent un octet.
- La plupart des lettres européennes et moyen−orientales sont stockées avec une séquence à deux octets : les lettres latines étendues (avec les tildes, macrons, accents graves, aigus et autres accents), cyrilliques, grecques, arméniennes, hébreues, arabes, syriaques et autres.
- Les idéographes coréens, chinois et japonais utilisent des séquences à trois octets.

Actuellement, MySQL UTF8 ne supporte pas les séquences à 4 octets.

Conseil : pour économiser de l'espace avec UTF−8, utilisez `VARCHAR` au lieu de `CHAR` . Sinon, MySQL doit réserver 30 octets pour une colonne `CHAR(10) CHARACTER SET utf8` parce que c'est la longueur maximale à accepter.

11.6 UTF8 pour les meta−données

Les meta−données sont les données sur les données. Tout ce qui décrit la base de données, par opposition au contenu de la base de données, sont des meta−données. Ainsi, les noms de colonnes, les noms de bases de données, les noms d'utilisateur, les noms de version et la plupart des résultats chaînes de `SHOW` , sont des meta−données.

La représentation des meta−données doit satisfaire les contraintes suivantes :

- Toutes les meta−données doivent être dans le même jeu de caractères. Sinon `SHOW` ne fonctionnerait pas correctement, car des lignes différentes de la même colonne seraient dans des jeux de caractères différents.
- Les meta−données doivent inclure tous les caractères de toutes les langues. Sinon, les utilisateurs ne pourraient pas nommer les colonnes et les tables dans leur langue.

Pour atteindre ces deux objectifs, MySQL enregistre les meta−données dans un jeu de caractères Unicode, nommé UTF8. Cela ne causera pas de perturbation si vous n'utilisez jamais de caractères accentués; mais si vous les utilisez, sachez que les meta−données sont en UTF8.

Cela signifie que les fonctions `USER()` et ses synonymes, `SESSION_USER()` et `SYSTEM_USER()`), `CURRENT_USER()` , et `VERSION()` auront le jeu de caractères UTF8 par défaut.

Le serveur assigne la variable système `character_set_system` avec le jeu de caractères des méta−données :

```
mysql> SHOW VARIABLES LIKE 'character_set_system';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| character_set_system | utf8  |
+----------------------+-------+
```

Le stockage des méta−données utilisant Unicode **ne signifie pas** que les entêtes et les résultats de la fonction `DESCRIBE` seront dans le jeu de caractères `character_set_system` . Lorsque vous envoyez la commande `SELECT column1 FROM t` , le nom de la colonne `column1` sera retourné par le serveur vers le client dans le jeu de caractères déterminer par la commande `SET NAMES` . Plus spécifiquement, le jeu de caractères est déterminé par la valeur de la variable système `character_set_results` . Si cette variable vaut `NULL` , aucune conversion n'est effectuée, et le serveur retourne les données dans leur jeu de caractères original (tel qu'indiqué par `character_set_system`).

Si vous voulez que le serveur affiche les résultats des meta−données dans un jeu de caractères autre que UTF8, alors utilisez `SET CHARACTER SET` pour forcer le serveur faire la conversion (Jeux de caractères de connexion et collations), ou configurez le client pour qu'il fasse la conversion. Il est toujours plus efficace de configurer le client pour qu'il fasse la conversion, mais ce choix ne sera pas toujours possible pour de nombreux clients jusqu'à tard dans le cycle des produits MySQL 4.x.

Si vous utilisez seulement, par exemple, la fonction `USER()` pour une comparaison ou une assignation dans une seule commande... ne vous inquiétez pas. MySQL fera des conversions automatiques pour vous.

```
SELECT * FROM Table1 WHERE USER() = latin1_column;
```

Ceci fonctionnera, car le contenu de `latin1_column` est automatiquement converti en UTF8 avant la comparaison.

```
INSERT INTO Table1 (latin1_column) SELECT USER();
```

Ceci fonctionnera car le contenu de `USER()` est automatiquement converti en `latin1` avant l'assignation. La conversion automatique n'est pas encore complètement implémentée, mais devrait fonctionner correctement dans une version ultérieure.

Bien que la conversion automatique ne soit pas un standard SQL, le document de standard SQL dit que chaque jeu de caractères est (en termes de caractères supportés) un "sous−jeu" de l'Unicode.

Comme c'est un principe bien connu que "ce qui s'applique à un super−jeu peut s'appliquer à un sous−jeu", nous croyons que la collation d'Unicode peut s'appliquer à des comparaisons avec des chaînes non−Unicode.

11.7 Compatibilité avec d'autres bases de données

Pour la compatibilité avec SAP DB, les deux commandes suivantes sont identiques :

```
CREATE TABLE t1 (f1 CHAR(n) UNICODE);
CREATE TABLE t1 (f1 CHAR(n) CHARACTER SET ucs2);
```

11.8 Nouveau format de fichier de configuration de jeux de caractères

En MySQL version 4.1, la configuration du jeu de caractères est stockée dans des fichiers XML : un fichier par jeu de caractères. Dans les versions précédentes, cette information était stockée dans les fichiers `.conf` .

11.9 Jeux de caractères national

En MySQL version 4.x et plus ancien, `NCHAR` et `CHAR` étaient synonymes. ANSI définit `NCHAR` ou `NATIONAL CHAR` comme une manière de définir le jeu de caractère par défaut d'une colonne de type `CHAR` . MySQL utilise `utf8` comme jeu de caractère prédéfini. Par exemple, les définitions de ces colonnes sont identiques :

```
CHAR(10) CHARACTER SET utf8
NATIONAL CHARACTER(10)
NCHAR(10)
```

de même que :

```
VARCHAR(10) CHARACTER SET utf8
NATIONAL VARCHAR(10)
NCHAR VARCHAR(10)
NATIONAL CHARACTER VARYING(10)
```

```
NATIONAL CHAR VARYING(10)
```

Vous pouvez utiliser `N'literal'` pour indiquer qu'une chaîne utilise le jeu de caractère national.

Ces deux commandes sont équivalentes :

```
 SELECT N'some text';
SELECT _utf8'some text';
```

11.10 Préparer le passage de version 4.0 en 4.1

Que faut−il faire pour mettre à jour MySQL au niveau des jeux de caractères? MySQL 4.1 est presque totalement compatible avec les versions 4.0 et plus ancien, pour la simple raison que toutes les fonctionnalités sont nouvelles, et ne peuvent entrer en conflit avec une version plus ancienne. Cependant, il y a quelques différences qu'il faut avoir en tête.

Le plus important : Le ``jeu de caractères de MySQL 4.0'' a les caractéristiques des ``jeux de caractères de MySQL 4.1'' et des ``collations MySQL 4.1.'' Vous devez désapprendre cela. Par conséquent, nous n'allons pas rassembler les jeux de caractères et les collations dans le même objet agglomérant.

Il n'y a pas de traitement spéciaux pour les jeux de caractères nationaux de MySQL 4.1. `NCHAR` n'est pas le même que `CHAR` , et les littéraux `N'...'` sont différents des littéraux `'...'` .

Finalement, il y a un format de fichier différent pour stocker les informations sur les jeux de caractères et les collations. Assurez−vous d'avoir installé le dossier `/share/mysql/charsets/` qui contient les nouveaux fichiers de configuration.

Si vous voulez lancer `mysqld` dans une installation 4.1.x avec les données de MySQL 4.0, lancez le serveur avec le même jeu de caractères et collation que lancier serveur. Dans ce cas, vous n'aurez pas besoin de réindexer vos données.

Voici deux moyens de le faire :

```
 shell> ./configure --with-character-set=... --with-collation=...
shell> ./mysqld --default-character-set=... --default-collation=...
```

Si vous utilisez `mysql` avec, par exemple, le jeux de caractères MySQL 4.0 `danois` , vous devez utiliser maintenant le jeu de caractères `latin1` avec la collation `latin1_danish_ci` :

```
 shell> ./configure --with-character-set=latin1
          --with-collation=latin1_danish_ci
shell> ./mysqld --default-character-set=latin1
          --default-collation=latin1_danish_ci
```

Utilisez la table de la section Jeux de caractères de la version 4.0 et correspondance en 4.1 pour retrouver les anciens jeux de caractères 4.0 et leur équivalent 4.1 en jeu de caractère et collation.Si vous avez des données non− `latin1` qui sont stockées dans une table 4.0 `latin1` et que vous voulez convertir la table pour que les définitions reflètent le véritable jeu de caractères, utilisez les instructions de la section Conversion des colonnes de caractères version 4.0 en format 4.1 .

11.10.1 Jeux de caractères 4.0 et binômes de jeux de caractères/collations 4.1 correspondants

ID	Jeu de caractères 4.0	Jeu de caractères 4.1	Collation 4.1
1	big5	big5	big5_chinese_ci

2	czech	latin2	latin2_czech_ci
3	dec8	dec8	dec8_swedish_ci
4	dos	cp850	cp850_general_ci
5	german1	latin1	latin1_german1_ci
6	hp8	hp8	hp8_english_ci
7	koi8_ru	koi8r	koi8r_general_ci
8	latin1	latin1	latin1_swedish_ci
9	latin2	latin2	latin2_general_ci
10	swe7	swe7	swe7_swedish_ci
11	usa7	ascii	ascii_general_ci
12	ujis	ujis	ujis_japanese_ci
13	sjis	sjis	sjis_japanese_ci
14	cp1251	cp1251	cp1251_bulgarian_ci
15	danish	latin1	latin1_danish_ci
16	hebrew	hebrew	hebrew_general_ci
17	win1251	(removed)	(removed)
18	tis620	tis620	tis620_thai_ci
19	euc_kr	euckr	euckr_korean_ci
20	estonia	latin7	latin7_estonian_ci
21	hungarian	latin2	latin2_hungarian_ci
22	koi8_ukr	koi8u	koi8u_ukrainian_ci
23	win1251ukr	cp1251	cp1251_ukrainian_ci
24	gb2312	gb2312	gb2312_chinese_ci
25	greek	greek	greek_general_ci
26	win1250	cp1250	cp1250_general_ci
27	croat	latin2	latin2_croatian_ci
28	gbk	gbk	gbk_chinese_ci
29	cp1257	cp1257	cp1257_lithuanian_ci
30	latin5	latin5	latin5_turkish_ci
31	latin1_de	latin1	latin1_german2_ci

11.10.2 Conversion de colonnes version 4.0 en version 4.1

Normalement, votre serveur utilise le jeu de caractères par défaut `latin1` . Si vous avez stocké des données qui sont dans un autre jeu de caractères que ceux supportés par la version 4.1, vous pouvez convertir la colonne. Cependant, vous devriez éviter de convertir directement les colonnes de `latin1` vers un "vrai" jeu de caractère. Cela peut conduire à des pertes de données. Au lieu de cela, convertissez la colonne en un type de colonne binaire, avec le jeu de caractères voulu. La conversion avec un format binaire conservera les données intactes. Par exemple, supposez que vous avez une table version 4.0, avec trois colonnes qui sont utilisées pour stocker des valeurs avec les jeux de caractères `latin1` , `latin2` et `utf8` :

```
CREATE TABLE t
(
    latin1_col CHAR(50),
    latin2_col CHAR(100),
    utf8_col CHAR(150)
);
```

Après mise à jour en MySQL version 4.1, vous pouvez convertir cette table pour qu'elle laisse intacte la colonne `latin1_col` mais modifie les colonnes `latin2_col` et `utf8_col` pour qu'elles utilisent les jeux de caractères `latin2` et `utf8` . D'abord, sauvez votre table, puis convertissez les colonnes comme ceci :

```
ALTER TABLE t MODIFY latin2_col BINARY(100);
ALTER TABLE t MODIFY utf8_col BINARY(150);
ALTER TABLE t MODIFY latin2_col CHAR(100) CHARACTER SET latin2;
ALTER TABLE t MODIFY utf8_col CHAR(150) CHARACTER SET utf8;
```

La première commande ``supprime'' les informations de jeux de caractères des colonnes `latin2_col` et `utf8_col` . Les deux autres commandes assignent le bon jeu de caractère aux deux colonnes.

Si vous voulez, vous pouvez combiner les deux conversions en une seule commande :

```
ALTER TABLE t
    MODIFY latin2_col BINARY(100),
    MODIFY utf8_col BINARY(150);
ALTER TABLE t
```

```
    MODIFY latin2_col CHAR(100) CHARACTER SET latin2,
    MODIFY utf8_col CHAR(150) CHARACTER SET utf8;
```

11.11 Les jeux de caractères et collation supportés par MySQL.

Voici une liste annotée de jeux de caractères et de collations que MySQL supporte. Puisque les options et configurations d'installation diffèrent, certains sites n'auront pas tous ces éléments dans leur liste, et certains sites auront des éléments qui ne sont pas sur la liste car la définition de nouveaux jeux de caractères ou collations est directe.

MySQL supporte plus de 70 collations pour plus de 30 jeux de caractères.

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
| cp850    | DOS West European           | cp850_general_ci    |      1 |
| hp8      | HP West European            | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | ISO 8859-1 West European    | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| swe7     | 7bit Swedish                | swe7_swedish_ci     |      1 |
| ascii    | US ASCII                    | ascii_general_ci    |      1 |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci    |      3 |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci    |      2 |
| cp1251   | Windows Cyrillic            | cp1251_bulgarian_ci |      1 |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   |      1 |
| tis620   | TIS620 Thai                 | tis620_thai_ci      |      1 |
| euckr    | EUC-KR Korean               | euckr_korean_ci     |      2 |
| koi8u    | KOI8-U Ukrainian            | koi8u_general_ci    |      1 |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci   |      2 |
| greek    | ISO 8859-7 Greek            | greek_general_ci    |      1 |
| cp1250   | Windows Central European    | cp1250_general_ci   |      1 |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci      |      2 |
| latin5   | ISO 8859-9 Turkish          | latin5_turkish_ci   |      1 |
| armscii8 | ARMSCII-8 Armenian          | armscii8_general_ci |      1 |
| utf8     | UTF-8 Unicode               | utf8_general_ci     |      3 |
| ucs2     | UCS-2 Unicode               | ucs2_general_ci     |      2 |
| cp866    | DOS Russian                 | cp866_general_ci    |      1 |
| keybcs2  | DOS Kamenicky Czech-Slovak  | keybcs2_general_ci  |      1 |
| macce    | Mac Central European        | macce_general_ci    |      1 |
| macroman | Mac West European           | macroman_general_ci |      1 |
| cp852    | DOS Central European        | cp852_general_ci    |      1 |
| latin7   | ISO 8859-13 Baltic          | latin7_general_ci   |      1 |
| cp1256   | Windows Arabic              | cp1256_general_ci   |      1 |
| cp1257   | Windows Baltic              | cp1257_general_ci   |      1 |
| binary   | Binary pseudo charset       | binary              |      1 |
+----------+-----------------------------+---------------------+--------+
33 rows in set (0.01 sec)
```

Note : Tous les jeux de caractères sont une collation binaire. Nous n'avons pas inclus cette collation dans les descriptions qui suivent.

11.11.1 Les jeux de caractère Unicode

Il y a deux jeux de caractères Unicode. Vous pouvez stocker des textes dans près de 650 langues en utilisant ces jeux de caractères. Nous n'avons pas encore ajouté un grand nombre de collations pour ces deux nouveaux jeux, mais cela se fera bientôt. Actuellement, ils ont des collations par défaut pour l'insensibilité à la casse et aux accents, plus une collation binaire.

Actuellement, la collation `ucs2_general_uca` ne supporte que partiellement l'algorithme de collations d'Unicode. Certains jeux de caractères ne sont pas supportés.

- Collations `ucs2` (UCS−2 Unicode) :
    - `ucs2_bin`

- `ucs2_general_ci` (par défaut)
- `ucs2_general_uca`

• Collations `utf8` (UTF−8 Unicode) :
- `utf8_bin`
- `utf8_general_ci` (par défaut)

11.11.2 Les jeux de caractères d'Europe de l'Ouest.

Les jeux de caractères de l'Europe de l'ouest recouvrent la plupart des langues d'Europe de l'ouest comme le français, l'espagnol, le catalan, le basque, le portugais, l'italien, l'albanais, le néerlandais, l'allemand le danois, le suédois, le norvégien, le finlandais, le faroe, l'islandais, l'Irlandais, l'écossais, et l'anglais.

- Collations `ascii` (US ASCII) :
  - `ascii_bin`
  - `ascii_general_ci` (par défaut)
- Collations `cp850` (DOS West European) :
  - `cp850_bin`
  - `cp850_general_ci` (par défaut)
- Collations `dec8` (DEC West European) :
  - `dec8_bin`
  - `dec8_swedish_ci` (par défaut)
- Collations `hp8` (HP West European) :
  - `hp8_bin`
  - `hp8_english_ci` (par défaut)
- Collations `latin1` (ISO 8859−1 West European) :
  - `latin1_bin`
  - `latin1_danish_ci`
  - `latin1_general_ci`
  - `latin1_general_cs`
  - `latin1_german1_ci`
  - `latin1_german2_ci`
  - `latin1_spanish_ci`
  - `latin1_swedish_ci` (par défaut)

  Le jeu de caractères `latin1` est le jeu de caractères par défaut. La collation `latin1_swedish_ci` est la collation par défaut qui est probablement utilisée par la majorité des clients de MySQL. Il est souvent dit que ceci est basé sur les règles de collation suédoises/finlandaises mais vous trouverez des suèdois et des finlandais qui ne sont pas d'accord avec cette affirmation.

  Les collations `latin1_german1_ci` et `latin1_german2_ci` sont basées sur les standards DIN−1 and DIN−2, où DIN signifie `Deutsches Institut für Normung` (c'est la contrepartie allemande de ANSI). DIN−1 est appelée la collation dictionnaire et DIN−2 est appelée la collation annuaire.

  - règles `latin1_german1_ci` (dictionnaire) :
```
'Ä' = 'A', 'Ö' = 'O', 'Ü' = 'U', '@ss{'} = 's'
```
  - règles `latin1_german2_ci` (annuaire) :
```
'Ä' = 'AE', 'Ö' = 'OE', 'Ü' = 'UE', '@ss{'} = 'ss'
```

  Dans la collation `latin1_spanish_ci` , `'Ñ'` (N−tilde) est une lettre séparée entre `'N'` et `'O'` .
- Collations `macroman` (Mac West European) :
  - `macroman_bin`
  - `macroman_general_ci` (par défaut)
- Collations `swe7` (7 bits Swedish) :
  - `swe7_bin`
  - `swe7_swedish_ci` (par défaut)

11.11.3 Les jeux de caractères d'Europe Centrale

MySQL supporte les jeux de caractères utilisés en République Tchèque, en Slovakie, Hongrie, Roumanie, SLovénie, Croatie, et Pologne.

- collations `cp1250` (Windows Central European) :
  - `cp1250_bin`
  - `cp1250_czech_ci`
  - `cp1250_general_ci` (par défaut)
- collations `cp852` (DOS Central European) :
  - `cp852_bin`
  - `cp852_general_ci` (par défaut)
- collations `keybcs2` (DOS Kamenicky Czech−Slovak) :
  - `keybcs2_bin`
  - `keybcs2_general_ci` (par défaut)
- collations `latin2` (ISO 8859−2 Central European) :
  - `latin2_bin`
  - `latin2_croatian_ci`
  - `latin2_czech_ci`
  - `latin2_general_ci` (par défaut)
  - `latin2_hungarian_ci`
- collations `macce` (Mac Central European) :
  - `macce_bin`
  - `macce_general_ci` (par défaut)

11.11.4 Jeu de caractères pour l'Europe du Sud et le Moyen−Orient

- Collations `armscii8` (ARMSCII−8 Armenian) :
  - `armscii8_bin`
  - `armscii8_general_ci` (par défaut)
- Collations `cp1256` (Windows Arabic) :
  - `cp1256_bin`
  - `cp1256_general_ci` (par défaut)
- Collations `geostd8` (GEOSTD8 Georgian) :
  - `geostd8_bin`
  - `geostd8_general_ci` (par défaut)
- Collations `greek` (ISO 8859−7 Greek) :
  - `greek_bin`
  - `greek_general_ci` (par défaut)
- Collations `hebrew` (ISO 8859−8 Hebrew) :
  - `hebrew_bin`
  - `hebrew_general_ci` (par défaut)
- Collations `latin5` (ISO 8859−9 Turkish) :
  - `latin5_bin`
  - `latin5_turkish_ci` (par défaut)

11.11.5 Les jeux de caractères baltes

Les jeux de caractères baltes couvrent les langues estonienne, lette et lituanienne. Il y a actuellement deux jeux de caractères baltiques supportés :

- Collations `cp1257` (Windows Baltic) :
  - `cp1257_bin`
  - `cp1257_general_ci` (par défaut)
  - `cp1257_lithuanian_ci`
- Collations `latin7` (ISO 8859−13 Baltic) :
  - `latin7_bin`
  - `latin7_estonian_cs`
  - `latin7_general_ci` (par défaut)
  - `latin7_general_cs`

11.11.6 Les jeux de caractère cyrilliques

Voici les jeux de caractères cyrilliques et les collations à utiliser avec les langues suivantes : biélorusse, bulgare, russe, ukrainien.

- Collations `cp1251` (Windows Cyrillic) :
    - `cp1251_bin`
    - `cp1251_bulgarian_ci`
    - `cp1251_general_ci` (par défaut)
    - `cp1251_general_cs`
    - `cp1251_ukrainian_ci`
- Collations `cp866` (DOS Russian) :
    - `cp866_bin`
    - `cp866_general_ci` (par défaut)
- Collations `koi8r` (KOI8−R Relcom Russian) :
    - `koi8r_bin`
    - `koi8r_general_ci` (par défaut)
- Collations `koi8u` (KOI8−U Ukrainian) :
    - `koi8u_bin`
    - `koi8u_general_ci` (par défaut)

11.11.7 Les jeux de caractères asiatiques

Les jeux de caractères asiatiques que nous supportons incluent le chinois, le japonais, le coréen et le thailandais. Ces jeux peuvent être compliqués. Par exemple, les jeux chinois doivent permettre des milliers de caractères différents.

- Collations `big5` (Big5 Traditional Chinese) :
    - `big5_bin`
    - `big5_chinese_ci` (par défaut)
- Collations `euckr` (EUC−KR Korean) :
    - `euckr_bin`
    - `euckr_korean_ci` (par défaut)
- Collations `gb2312` (GB2312 Simplified Chinese) :
    - `gb2312_bin`
    - `gb2312_chinese_ci` (par défaut)
- Collations `gbk` (GBK Simplified Chinese) :
    - `gbk_bin`
    - `gbk_chinese_ci` (par défaut)
- Collations `sjis` (Shift−JIS Japanese) :
    - `sjis_bin`
    - `sjis_japanese_ci` (par défaut)
- Collations `tis620` (TIS620 Thai) :
    - `tis620_bin`
    - `tis620_thai_ci` (par défaut)
- Collations `ujis` (EUC−JP Japanese) :
    - `ujis_bin`
    - `ujis_japanese_ci` (par défaut)

12 Types de colonnes

MySQL supporte un grand nombre de types de colonnes, qui peuvent être rassemblées en trois catégories : les types numériques, temporels et chaînes. Cette section vous donne un aperçu des types disponibles, et résume les besoin de stockage de chaque colonne, puis fournit une description détaillée des propriétés de chaque type de données. Cette présentation est volontairement courte. Les descriptions détaillées peuvent être consultées pour plus d'informations sur chaque type, comme les formats autorisés.

MySQL 4.1 et plus récente supporte des extensions pour gérer les données géographiques. Des informations sur ces types sont disponibles dans la section Extensions spatiale de MySQL .

Plusieurs définitions de colonnes partagent la même convention :

`M`
: Indique la taille maximale d'affichage. La taille maximale légale est de 255.

`D`
: S'applique aux types à virgule flottante, et indique le nombre de chiffres qui suivent la virgule décimale. Le nombre maximal est de 30, mais ne devrait pas dépasser `M` −2.

`[]`
: Les crochets ('`[`' et '`]`') indiquent des spécifications qui sont optionnelles.

12.1 Introduction aux types de colonnes

12.1.1 Présentation des types numériques of Numeric Types

Un résumé des colonnes numériques suit. Pour plus de détails sur les types numériques, voyez la section Types numériques . La taille des colonnes sont dans la section Taille de stockage des types .

Si vous spécifiez l'option `ZEROFILL` pour une valeur numérique, MySQL va automatiquement ajouter l'attribut `UNSIGNED` à la colonne.

Attention : soyez conscient que lorsque vous utilisez la soustraction entre deux entier, dont l'un est de type `UNSIGNED` , le résultat sera sans signe! Fonctions de transtypage .

`TINYINT[(M)] [UNSIGNED] [ZEROFILL]`
: Un très petit entier. L'intervalle de validité pour les entiers signés est de `-128` à `127` . L'intervalle de validité pour les entiers non−signés est `0` à `255` .

`BIT`
`BOOL`
`BOOLEAN`
: Ce sont des synonymes de `TINYINT(1)` . Le synonyme `BOOLEAN` a été ajouté en version 4.1.0

  Un type booléen complet, qui sera introduit pour être en accord avec la norme SQL−99.

`SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`
: Un petit entier. L'intervalle de validité pour les entiers signés est de `-32768` à `32767` . L'intervalle de validité pour les entiers non−signés est `0` à `65535` .

`MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`
: Un entier. L'intervalle de validité pour les entiers signés est de `-8388608` à `8388607` . L'intervalle de validité pour les entiers non−signés est `0` à `16777215` .

`INT[(M)] [UNSIGNED] [ZEROFILL]`
: Un grand entier. L'intervalle de validité pour les entiers signés est de `-2147483648` à

`2147483647` . L'intervalle de validité pour les entiers non−signés est `0` à `4294967295` .

`INTEGER[(M)] [UNSIGNED] [ZEROFILL]`
: Ceci est un synonyme `INT` .

`BIGINT[(M)] [UNSIGNED] [ZEROFILL]`
: Un très grand entier. L'intervalle de validité pour les entiers signés est de `-9223372036854775808` à `9223372036854775807` . L'intervalle de validité pour les entiers non−signés est `0` à `18446744073709551615` . Quelques conseils à suivre avec les colonnes de type `BIGINT` :
  - Tous les calculs arithmétiques sont fait en utilisant des `BIGINT` signés ou des valeurs `DOUBLE` . Il est donc recommandé de ne pas utiliser de grands entiers non−signés dont la taille dépasse `9223372036854775807` (63 bits), hormis avec les fonctions sur les bits! Si vous faîtes cela, les derniers chiffres du résultats risquent d'être faux, à cause des erreurs d'arrondis lors de la conversion de `BIGINT` en `DOUBLE` .MySQL 4.0 peut gérer des `BIGINT` dans les cas suivants :
    - Utiliser des entiers pour stocker des grandes valeurs entières non signées, dans une colonne de type `BIGINT` .
    - Avec `MIN(big_int_column)` et `MAX(big_int_column)` .
    - Avec les opérateurs (`+` , `-` , `*` , etc.) où tous les opérandes sont des entiers.
  - Vous pouvez toujours stocker une valeur entière exacte `BIGINT` dans une colonne de type chaîne. Dans ce cas, MySQL fera des conversions chaîne / nombre, qui n'utilisera pas de représentation intermédiaire en nombre réels.
  - `'-'` , `'+'` et `'*'` utiliseront l'arithmétique entière des `BIGINT` lorsque les deux arguments sont des entiers. Cela signifie que si vous multipliez deux entiers (ou des résultats de fonctions qui retournent des entiers), vous pourriez rencontrer des résultats inattendus lorsque le résultat est plus grand que `9223372036854775807` .

`FLOAT(precision) [UNSIGNED] [ZEROFILL]`
: Un nombre à virgule flottante. `precision` peut valoir `<=24` pour une précision simple, et entre 25 et 53 pour une précision double. Ces types sont identiques aux types `FLOAT` et `DOUBLE` , décrit ci−dessous. `FLOAT(X)` a le même intervalle de validité que `FLOAT` et `DOUBLE` , mais la taille d'affichage et le nombre de décimales est indéfini.En MySQL version 3.23, c'est un véritable nombre à virgule flottante. Dans les versions antérieures, `FLOAT(precision)` avait toujours 2 décimales.

  Notez qu'utiliser `FLOAT` peut vous donner des résultats inattendus, car tous les calculs de MySQL sont fait en double précision. Résoudre les problèmes des lignes non retournées .

  Cette syntaxe est fournie pour assurer la compatibilité avec ODBC.

  Utiliser des `FLOAT` peut vous donner des résultats inattendus, car les calculs sont fait en précision double. Résoudre les problèmes des lignes non retournées .

`FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]`
: Un petit nombre à virgule flottante, en précision simple. Les valeurs possibles vont de `-3.402823466E+38` à `-1.175494351E-38` , `0` , et `1.175494351E-38` à `3.402823466E+38` . Si `UNSIGNED` est spécifié, les valeurs négatives sont interdites. L'attribut `M` indique la taille de l'affichage, et `D` est le nombre de décimales. `FLOAT` sans argument et `FLOAT(X)` (où `X` est dans l'intervalle 0 à 24) représente les nombres à virgule flottante en précision simple.

`DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]`
: Un nombre à virgule flottante, en précision double. Les valeurs possibles vont de `-1.7976931348623157E+308` à `-2.2250738585072014E-308` , `0` , et `2.2250738585072014E-308` à `1.7976931348623157E+308` . Si `UNSIGNED` est spécifié, les valeurs négatives sont interdites. L'attribut `M` indique la taille de l'affichage, et `D` est le nombre de décimales. `DOUBLE` sans argument et `FLOAT(X)` (où `X` est dans l'intervale 25 to 53) représente les nombres à virgule flottante en précision double.

`DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]`
`REAL[(M,D)] [UNSIGNED] [ZEROFILL]`
: Ce sont des synonymes pour `DOUBLE` .

Exception : si le serveur SQL utilise l'option `REAL_AS_FLOAT` , `REAL` est alors un synonyme de `FLOAT` plutôt que `DOUBLE` .

`DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]`

Un nombre à virgule flottante littéral. Il se comporte comme une colonne de type `CHAR` : ``littéral'' ( ``unpacked'') signifie que le nombre est stocké sous forme de chaîne : chaque caractère représente un chiffre. La virgule décimale et le signe moins `'-'` des nombres négatifs ne sont pas comptés dans `M` (mais de l'espace leur est réservé). Si `D` vaut 0, les valeurs n'auront pas de virgule décimale ou de partie décimale. L'intervale de validité du type `DECIMAL` est le même que `DOUBLE` , mais le vrai intervalle de validité de `DECIMAL` peut être restreint par le choix de la valeur de `M` et `D` . Si `UNSIGNED` est spécifié, les valeurs négatives sont interdites.

Si `D` est omis, la valeur par défaut est 0. Si `M` est omis, la valeur par défaut est 10.

Avant MySQL Version 3.23, l'argument `M` devait inclure l'espace nécessaire pour la virgule et le signe moins.

`DEC[(M[,D])] [UNSIGNED] [ZEROFILL]`
`NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]`
`FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]`

Ce sont des synonymes pour `DECIMAL` .L'alias `FIXED` a été ajouté en version 4.1.0 pour assurer la compatibilité avec les autres serveurs.

12.1.2 Présentation des types de données temporels : dates et heures

Une description succincte des types de données temporel suit. Pour plus d'informations, voyez la section Types temporels . La taille de stockage des valeurs est présenté dans la section Taille de stockage des types .

DATE

Une date. L'intervalle supporté va de `'1000-01-01'` à `'9999-12-31'` . MySQL affiche les valeurs de type `DATE` au format `'YYYY-MM-DD'` , mais vous permet d'assigner des valeurs `DATE` en utilisant plusieurs formats de chaînes et nombres.

DATETIME

Une combinaison de date et heure. L'intervalle de validité va de `'1000-01-01 00:00:00'` à `'9999-12-31 23:59:59'` . MySQL affiche les valeurs de type `DATE` au format `'YYYY-MM-DD HH:MM:SS'` , mais vous permet d'assigner des valeurs `DATE` en utilisant plusieurs formats de chaînes et nombres.

TIMESTAMP[(M)]

Un timestamp. L'intervalle de validité va de `'1970-01-01 00:00:00'` à quelque part durant l'année `2037` .

En MySQL 4.0 et plus récent, les valeurs `TIMESTAMP` sont affichées au format `YYYYMMDDHHMMSS` , `YYMMDDHHMMSS` , `YYYYMMDD` ou `YYMMDD` , suivant que la valeur de `M` est `14` (ou absente), `12` , `8` ou `6` , respectivement, mais vous permet d'assigner des valeurs aux colonnes `TIMESTAMP` en utilisant des nombres ou des chaînes.

Depuis MySQL 4.1, `TIMESTAMP` est retournée comme une chaîne, au format `'YYYY-MM-DD HH:MM:SS'` . Si vous voulez que MySQL vous retourne un nombre, ajoutez +0 à la colonne. Les différentes tailles de timestamp ne sont pas supportées. Depuis la version 4.0.12, l'option `--new` peut être utilisée pour que le serveur adopte le comportement de la version 4.1.

Une colonne `TIMESTAMP` est utile pour enregistrer les dates et heures des opérations `INSERT` et `UPDATE` , car elle prend automatiquement date actuellement si vous ne lui assignez pas de valeur par vous−même. Vous pouvez aussi lui donner la valeur courante en lui donnant la valeur `NULL` .

L'argument `M` affecte l'affichage des colonnes de type `TIMESTAMP` . ses valeurs sont toujours stockées sur 4 octets.

Notez que les colonnes `TIMESTAMP(M)` où `M` vaut 8 ou 14 sont indiquée comme étant des nombres, alors que les colonnes `TIMESTAMP(M)` sont indiquées comme étant des chaînes. Cela est fait pour s'assurer que l'ont peut enregistrer et lire correctement les tables ayant ce type.

TIME

Une heure. L'intervalle va de `'-838:59:59'` à `'838:59:59'` . MySQL affiche les valeurs `TIME` au format `'HH:MM:SS'` , mais vous permet d'assigner des valeurs `TIME` en utilisant des nombres ou des chaînes.

YEAR[(2|4)]

Une année, au format 2 ou 4 chiffres (par défaut, c'est 4 chiffres). Les valeurs possibles vont de `1901` à `2155` plus `0000` pour le format à 4 chiffres, et de `1970` à `2069` si vous utilisez le format à 2 chiffres. MySQL affiche les valeurs `YEAR` au format `YYYY` mais vous permet d'assigner des valeurs en utilisant des nombres ou des chaînes. Le type `YEAR` n'est pas disponible avant la version 3.22.

12.1.3 Présentation des types de chaînes

Voici une présentation sommaire des types chaînes de caractères. Pour plus d'informations, voyez Types chaînes de caractères . Le tailles de stockage des lignes sont donnés dans Taille de stockage des types .

Dans certains cas, MySQL change le type d'une colonne en un autre, lors de l'utilisation des commandes `CREATE TABLE` et `ALTER TABLE` . Changement de type de colonne automatique .

Une modification qui affecte de nombreux types de colonnes est que depuis MySQL version 4.1.1, les définitions de colonnes peuvent inclure l'attribut `CHARACTER SET` pour spécifier le jeu de caractères, et, éventuellement, la collation de la colonne. Cela s'applique à `CHAR` , `VARCHAR` , les types `TEXT` types, `ENUM` et `SET` . Par exemple :

```
 CREATE TABLE t
(
    c1 CHAR(20) CHARACTER SET utf8,
    c2 CHAR(20) CHARACTER SET latin1 COLLATE latin1_bin
);
```

Cette définition de table crée une colonne appelée `c1` dont le jeu de caractères est `utf8` avec la collation par défaut de ce jeu de caractères, et une colonne appelée `c2` qui a le jeu de caractères `latin1` et la collation binaire du jeu de caractères. La collation binaire n'est pas sensible à la casse.Le tri et les comparaisons de colonnes sont basés sur le jeu de caractères de la colonne. Avant MySQL 4.1, les tris et comparaisons étaient fait avec la collation du jeu de caractères du serveur. Pour les colonnes `CHAR` et `VARCHAR` , vous pouvez déclarer la colonne avec l'attribut `BINARY` pour que le tri et la recherche soient insensibles à la casse, utilisant le jeu de caractère sous−jacent, plutôt qu'un ordre lexical.

Pour plus de détails, voyez Jeux de caractères .

De plus, depuis la version 4.1, MySQL interprète les spécifications de taille d'une colonne en terme de nombre de caractères. Les versions précédentes interprétaient les tailles en nombre d'octets.

`[NATIONAL] CHAR(M) [BINARY | ASCII | UNICODE]`

Une chaîne de caractère de taille fixe, toujours complété à droite par des espaces pour remplir l'espace de stockage. L'intervalle de `M` va de 0 à 255 (1 à 255 pour les versions antérieure à la version 3.23). Les espaces terminaux sont supprimés lorsque la valeur est relue. Les valeurs `CHAR` sont triées et comparées sans tenir compte de la casse, en utilisant le jeu de caractères par défaut, à moins que le mot clé `BINARY` ne soit utilisé. **Note** : les espaces terminaux sont supprimées lorsque la valeur est stockée.

Depuis la version 4.1.0, si la valeur `M` est supérieure à 255, Une colonne de type `TEXT` est créée. Ceci est une fonctionnalité de compatibilité.

`NATIONAL CHAR` (sous son équivalent raccourci `NCHAR`) est le nom SQL−99 pour définir une colonne de type `CHAR` qui utilise le jeu de caractère par défaut. C'est le comportement par défaut de MySQL.

`CHAR` est un raccourci pour `CHARACTER` .

Depuis la version 4.1.0, l'attribut `ASCII` peut être spécifiée avec, pour assigner le jeu de caractère `latin1` à une colonne de type `CHAR` .

Depuis la version 4.1.1, l'attribut `UNICODE` peut être spécifié pour assigner le jeu de caractères `ucs2` à une colonne `CHAR` .

MySQL permet la création d'une colonne de type `CHAR(0)` . Ceci est principalement utile dans de vieille application, qui ont besoin de la colonne, mais n'ont pas besoin de la valeur. C'est aussi pratique pour avoir une colonne à deux valeurs : un `CHAR(0)` , qui n'est pas défini comme `NOT NULL` , va occuper un bit, et prendre deux valeurs : `NULL` ou `""` .

`CHAR`
: Ceci est un synonyme de `CHAR(1)` .

`[NATIONAL] VARCHAR(M) [BINARY]`
: Une chaîne de taille dynamique. `M` représente la taille maximale de la valeur dans une colonne. L'intervalle de `M` va de 0 à 255 caractères (1 à 255 avant MySQL 4.0.2).

Note : les espaces terminaux sont supprimées lorsque la valeur est stockée (cela diffère des spécifications de SQL−99).

Depuis la version 4.1.0, si la valeur `M` est supérieure à 255, Une colonne de type `TEXT` est créée. Ceci est une fonctionnalité de compatibilité. Par exemple une colonne `VARCHAR(500)` est convertie en `TEXT` , et `VARCHAR(200000)` est convertie en `MEDIUMTEXT` . Attention, cette conversion affecte la suppression des espaces finaux...

`VARCHAR` est un raccourci pour `CHARACTER VARYING` .

`TINYBLOB`
`TINYTEXT`
: Une colonne `TINYBLOB` ou `TINYTEXT` peut contenir au maximum 255 (2^8 −{} 1) caractères.

`BLOB`
`TEXT`
: Une colonne `TEXT` ou `BLOB` peut contenir au maximum 65535 (2^16 −{} 1) caractères.

`MEDIUMBLOB`
`MEDIUMTEXT`
: Une colonne `MEDIUMTEXT` ou `MEDIUMBLOB` peut contenir au maximum 16777215 (2^24 −{} 1) caractères.

`LONGBLOB`
`LONGTEXT`
: Une colonne `LONGTEXT` ou `LONGBLOB` peut contenir au maximum 4294967295 ou 4 Go (2^32 −{} 1) caractères. Jusqu'en version 3.23 le protocole client/serveur et les tables MyISAM avait une limite de 16 Mo par paquet de communication pour une ligne de table. Depuis les versions 4.x, la taille maximale d'un `LONGTEXT` ou `LONGBLOB` dépend de la taille maximal de paquet de communication pour le protocole de communication, et de la mémoire disponible.

`ENUM('value1','value2',...)`

Une énumération. Un objet chaîne qui peut prendre une valeur, choisie parmi une liste de valeurs `'valeur1'` , `'valeur2'` , `...` , `NULL` ou la valeur spéciale d'erreur `""` . Une valeur `ENUM` peut avoir un maximum de 65535 valeurs distinctes.

`SET('value1','value2',...)`

Un ensemble. Un objet chaîne, qui peut prendre zéro, une ou plusieurs valeurs, choisies parmi une liste de valeurs `'valeur1'` , `'valeur2'` , `...` Une valeur `SET` peut avoir un maximum de 64 membres.

12.2 Types numériques

MySQL supporte tous les types numériques de la norme ANSI/ISO SQL92. Ceux ci représentent les types numériques exacts (`NUMERIC` , `DECIMAL` , `INTEGER` , et `SMALLINT`), ainsi que les types approchés (`FLOAT` , `REAL` , et `DOUBLE PRECISION`). Le mot clef `INT` est un synonyme de `INTEGER` , et le mot clef `DEC` est un synonyme de `DECIMAL` .

Les types `NUMERIC` et `DECIMAL` sont considérés comme identiques par MySQL, comme l'autorise le standard SQL92. Ils sont utilisées par des valeurs dont il est primordial de conserver la précision exacte, comme pour des données financières. Lorsque vous déclarez des colonnes avec l'un de ces types, vous pouvez indiquer la précision et l'échelle comme ceci :

```
salaire DECIMAL(5,2)
```

Dans cet exemple, `5` (`précision`) représente le nombre de décimales signifiantes qui seront stockées pour les valeurs, et `2` (`échelle`) représente le nombre de chiffres qui seront stockés après le point des décimales.

Les valeurs de type `DECIMAL` et `NUMERIC` sont stockées sous forme de chaînes de caractères, plutôt que comme des nombres à virgule flottante, afin de préserver la précision décimale des valeurs. Un caractère est donc nécessaire pour chaque chiffre, plus la virgule (si `scale` > 0), et le signe moins `'-'` (pour les nombres négatifs). Si `scale` vaut 0, les valeurs de type `DECIMAL` et `NUMERIC` ne comporteront pas de valeur décimale, ni de virgule.

Les standards SQL requièrent que la colonne `salary` soit capable de stocker toute valeur de 5 chiffres et 2 décimales. Dans ce cas, l'intervalle de valeur qui peut être stockée dans la colonne `salary` va de `-999.99` et `999.99` . MySQL dévie de cette spécification de deux manières :

- A la limite supérieure de l'intervalle, la colonne peut stocker les nombres jusqu'à `9999.99` . Pour les nombres positifs, MySQL utilise l'octet réservé au signe pour étendre la limite supérieure.
- Les colonnes `DECIMAL` de MySQL avant 3.23 sont stockés differement et ne peuvent pas représenter toutes les valeurs requises par le standard SQL. Ceci est dû au fait que pour `DECIMAL(M,D)` , la valeur de `M` inclut les octets pour le signe et le point décimal. L'intervalle de la colonne `salary` avant MySQL 3.23 serait de `-9.99` à `99.99` .

Avec les standards SQL, la syntaxe `DECIMAL(p)` est équivalente à `DECIMAL(p,0)` . De manière similaire, la syntaxe `DECIMAL` est équivalente à `DECIMAL(p,0)` , où l'implémentation est autorisée à choisir la valeur de `p` . MySQL ne supporte pas actuellement ces formes variantes des types de données `DECIMAL` / `NUMERIC` . Cela ne pose pas généralement de grand problèmes, vu que le principal bénéfice de ces types est de pouvoir contrôler explicitement la précision et l'échelle.

L'intervalle de validité maximal des valeurs de type `DECIMAL` et `NUMERIC` est le même que pour le type `DOUBLE` , mais l'intervalle réel peut être limité par le choix des paramètres `précision` et `scale` . Lorsqu'une valeur ayant trop de décimales est affectée à une colonne, la valeur est arrondie à `scale` décimales. Lorsqu'une valeur est hors des limites de validité de la colonne `DECIMAL` ou `NUMERIC` , MySQL enregistre la plus grande valeur qu'il peut à la place.

En extension de la norme ANSI/ISO SQL92, MySQL supporte aussi les types entiers `TINYINT` , `MEDIUMINT` , et `BIGINT` , comme présenté ci−dessus. Un autre extension supportée par MySQL permet de spécifier optionnellement la taille d'affichage, sous la forme d'une valeur entière entre parenthèses, juste après le mot clé spécifiant le type (par exemple, INT(4)). Cette spécification de taille est utilisée pour remplir à gauche, avec le caractère de remplissage par défaut, les nombres dont la taille est inférieure à celle spécifiée mais uniquement à l'affichage : cela ne réduit pas l'intervalle de validité des valeurs qui peuvent être stockées dans la colonne.

Lorsqu'elle est utilisée avec l'attribut de colonne optionnel `ZEROFILL` , le caractère de remplissage par défaut est remplacé par le caractère zéro. Par exemple, pour une colonne dont le type est `INT(5) ZEROFILL` , la valeur `4` sera lue `00004` .

Notez que si vous stockez des nombres plus grands que la taille maximale d'affichage, vous pouvez rencontrer des problèmes lors de jointures de tables particulièrement compliquées, surtout si MySQL génére des tables temporaires : dans ce cas, MySQL pense que les données étaient limitées par l'affichage.

Tous les types entiers ont un attribut optionnel (non−standard) `UNSIGNED` (non−signé, en français). Les valeurs non−signées peuvent être utilisées pour n'autoriser que des valeurs positives dans une colonne, ou bien pour exploiter un intervalle de validité plus haut. Depuis la version 4.0.2 de MySQL, les nombres à virgule flottante peuvent aussi être `UNSIGNED` Comme avec les types entiers, cet attribut interdit les valeurs négatives dans la colonne, mais n'élève pas l'intervalle de validité.

Le type FLOAT est utilisé pour représenter des données numériques approchées. La norme ANSI/ISO SQL92 permet la spécification optionnelle de la précision (mais pas de l'intervalle de validité) en fournissant le nombre de décimales voulues après la spécification de type, et entre parenthèses. L'implémentation de MySQL supporte aussi le paramétrage de la précision. Si le mot clé `FLOAT` est utilisé pour une colonne sans précision supplémentaire, MySQL utilise quatre octets pour stocker les valeurs. Une syntaxe alternative existe aussi, elle utilise deux paramètre optionnel après le mot clé `FLOAT` . Avec cette option, le premier nombre représente toujours la taille de stockage nécessaire pour la valeur, et le second nombre représente le nombre de chiffres à stocker et afficher, après la virgule décimale (comme pour les types `DECIMAL` et `NUMERIC`). Lorsque MySQL stocke un nombre pour une telle colonne, et que cette valeur a plus de décimale que requis, la valeur est arrondie pour éliminer les chiffres surnuméraires.

Les types `REAL` et `DOUBLE PRECISION` n'acceptent pas de paramétrage de la précision. En extension du standard ANSI/ISO SQL92, MySQL reconnaît `DOUBLE` comme un synonyme du type `DOUBLE PRECISION` . Contrairement à la norme qui requiert que `REAL` soit plus petit que `DOUBLE PRECISION` , MySQL implémente ces deux types comme des nombres à virgule flottante de 8 octets, en double précision (lorsque le mode ``ANSI'' n'est pas activé). Pour une portabilité maximale, les applications réclamant le stockage de nombres approché doivent utiliser les types `FLOAT` ou `DOUBLE PRECISION` sans spécification de précision ou de nombre de décimales.

Lorsque MySQL doit stocker une valeur qui est hors de l'intervalle de validité d'une colonne, il ramène la valeur à la plus proche possible, et stocke cette valeur. Par exemple, l'intervalle de validité d'une colonne d'entiers `INT` va de `-2147483648` à `2147483647` . Si vous essayez d'insérer `-9999999999` dans une colonne de ce type, la valeur sera ramenée à la plus proche possible, c'est à dire `-2147483648` . De même, si vous essayez d'insérer `9999999999` , `2147483647` sera stocké à la place.

Si la colonne `INT` possède l'attribut `UNSIGNED` , l'intervalle de validité est aussi large, mais les valeurs extrêmes se décalent vers `0` et `4294967295` . Si vous essayez de stocker `-9999999999` et `9999999999` dans cette colonne, vous obtiendrez respectivement `0` et `4294967296` .

Les dépassements de capacité entraînant des troncatures sont affichés comme des alertes (``

`warnings` ") lors de l'utilisation des commandes `ALTER TABLE` , `LOAD DATA INFILE` , `UPDATE` , et les insertions `INSERT` multiples.

Type	Octets	De	A
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

12.3 Les types date et heure

Les types dates et heures sont `DATETIME` , `DATE` , `TIMESTAMP` , `TIME` , et `YEAR` . Chacun d'eux à une échelle de valeurs légales, de même que la valeur ``zéro'' quand vous spécifiez une valeur illégale. A noter que MySQL vous permet d'enregistrer certaines dates qui ne sont pas strictement légales, par exemple `1999-11-31` . La raison est que nous pensons que la vérification des dates est à faire niveau application. Pour accélérer les tests, MySQL vérifie juste que le mois est entre 0 et 12 et que le jour est entre 0 et 31. Les intervalles précédentes sont définies de cette façon car MySQL vous permet d'enregistrer dans une colonne `DATE` ou `DATETIME` , des dates où le jour de la semaine ou le jour du mois est zéro. C'est extrêmement utile pour les applications où vous avez besoin d'enregistrer une date d'anniversaire pour laquelle vous n'avez pas la date exacte. Dans ce cas, vous enregistrez simplement la date comme `1999-00-00` ou `1999-01-00` . (Vous ne devez pas vous attendre à obtenir de valeurs correctes de fonctions tel que `DATE_SUB()` ou `DATE_ADD` pour des dates comme cela.)

Voici quelques considérations à garder à l'esprit quand vous manipulerez ce type de champs :

- MySQL extrait les valeurs d'un champ date ou heure dans un format standard, mais essaye d'interpréter une grande variétés de format pour les valeurs que vous donnez (par exemple, quand vous essayez de comparer ou attribuer une valeur à un champ de type date ou heure). Néanmoins, seul les formats décrits dans les sections suivantes sont disponibles. Il est attendu que vous fournissiez des valeurs légales et des erreurs imprévues peuvent survenir si vous utilisez d'autre formats.
- Même si MySQL essaye d'interpreter les valeurs sous différents formats, il s'attend toujours à ce que l'année soit dans la partie gauche de la valeur. Les dates doivent êtres données sous la forme année−mois−jour (exemple : `98-09-04`), au lieu de mois−jour−année ou jour−mois−année qui sont très utilisés ailleurs (comme `09-04-98` ou `'04-09-98'`).
- Les dates représentées par deux chiffres pour les années sont ambigues, car le siècle n'est pas connu. MySQL interprète les années sur deux chiffres suivant les règles suivantes :
    - Si l'année est dans l'intervalle `00-69` , elle est convertie en `2000-2069` .
    - Si l'année est dans l'intervalle `70-99` , elle est convertie en `1970-1999` .
- MySQL convertit automatiquement une date ou heure en nombre si la valeur est utilisée dans un contexte numérique et vice versa.
- Lorsque MySQL rencontre une valeur hors d'intervalle pour un type date ou heure qui est donc illégale pour ce type (voir le début de cette section), il la convertit à la valeur ``zéro'' de ce type. (L'exception est que les valeurs hors intervalles pour les champs `TIME` sont coupées à la limite appropriée des valeurs de `TIME` .) Le tableau suivant présente le format de la valeur ``zéro'' de chaque type :

Column type	valeur du ``zéro''
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	00000000000000 (la longueur dépend de la taille de l'affichage)
TIME	'00:00:00'
YEAR	0000

- La valeur ``zéro'' est spéciale, mais vous pouvez l'enregistrer ou vous y référer explicitement en utilisant les valeurs contenues dans le tableau ci dessus. Vous pouvez aussi le faire en utilisant la valeur `'0'` ou `0` qui est plus facile à manipuler.

- La date ou le temps ``Zéro'' utilisé avec `MyODBC` est automatiquement convertie en `NULL` à partir de la version 2.50.12 de `MyODBC` , car ODBC ne peut manipuler de telles valeurs.

12.3.1 Les types DATETIME , DATE , et TIMESTAMP

Les types `DATETIME` , `DATE` , et `TIMESTAMP` sont liés. Cette section décrit leurs caractéristiques, leur similarités et leurs différences.

Le type `DATETIME` est prévu lorsque vous souhaitez stocker une date et une heure. MySQL affiche les valeurs de type `DATETIME` au format '`AAAA-MM-JJ HH:MM:SS`'. L'intervalle de validité va de '`1000-01-01 00:00:00`' à '`9999-12-31 23:59:59`'. (``validité'' signifie que même si d'autres valeurs plus anciennes peuvent être manipulées, il n'est pas garantit qu'elles le seront).

Le type `DATE` est prévu lorsque vous souhaitez stocker une date. MySQL affiche les valeurs de type `DATE` au format '`AAAA-MM-JJ`'. L'intervalle de validité va de `'1000-01-01'` à `'9999-12-31'` .

La colonne `TIMESTAMP` a vu ses propriétés et comportements évoluer avec les versions de MySQL et le mode SQL du serveur.

Vous pouvez spécifier les valeurs des colonnes `DATETIME` , `DATE` et `TIMESTAMP` , avec les formats communs suivants :

- Une chaîne au format `'AAAA-MM-JJ HH:MM:SS'` ou `'AA-MM-JJ HH:MM:SS'` . Une syntaxe plus souple est permise : tout caractère de ponctuation peut être utilisé comme délimiteur entre les parties de temps ou heure. Par exemple, `'98-12-31 11:30:45'` , `'98.12.31 11+30+45'` , `'98/12/31 11*30*45'` , et `'98@12@31 11^30^45'` sont équivalents.
- Une chaîne au format '`AAAA-MM-JJ`' ou '`AA-MM-JJ`'. Une syntaxe plus flexible est aussi acceptée ici. Par exemple, '`98-12-31`', '`98.12.31`', '`98/12/31`', et '`98@12@31`' sont équivalent.
- Une chaîne sans aucun délimiteurs sous la forme '`AAAAMMJJHHMMSS`' ou '`AAMMJJHHMMSS`', en supposant qu'une telle chaîne ait un sens en terme de date. Par exemple '`19970523091528`' et '`970523091528`' sont interprétés comme '`1997-05-23 09:15:28`', mais '`971122129015`' est invalide (les minutes ne sont pas valides) et devient alors '`0000-00-00 00:00:00`'.
- Une chaîne sans aucun délimiteurs sous la forme '`AAAAMMJJ`' ou '`AAMMJJ`', en supposant qu'une telle chaîne ait un sens en terme de date. Par exemple, '`19970523`' et '`970523`' sont interprétés comme '`1997-05-23`', mais '`971332`' est invalide (les mois ne sont pas valides) et devient alors '`0000-00-00`'.
- Un nombre au format `AAAAMMJJHHMMSS` ou `AAMMJJHHMMSS` , en supposant qu'un tel nombre ait un sens en terme de date. Par exemple, `19830905132800` et `830905132800` sont interprétés comme '`1983-09-05 13:28:00`'.
- Un nombre au format `AAAAMMJJ` ou `AAMMJJ` en supposant qu'un tel nombre ait un sens en terme de date. Par exemple, `19830905` et `830905` sont interprétés comme '`1983-09-05`'.
- Un résultat de fonction qui retourne une valeur acceptable dans une colonne de type `DATETIME` , `DATE` , ou `TIMESTAMP` , tels que `NOW()` ou `CURRENT_DATE` .

Les valeurs invalides `DATETIME` , `DATE` , ou `TIMESTAMP` sont remplacées par la date ``zéro'' du type approprié (respectivement '`0000-00-00 00:00:00`', '`0000-00-00`', ou `00000000000000`).

Pour la valeurs spécifiées sous forme de chaînes avec des délimiteurs de date, il n'est pas nécessaire de spécifier les deux chiffres pour les mois ou les dates qui sont inférieurs à `10` . Par exemple, '`1979-6-9`' est valide et est équivalent à '`1979-06-09`'. Similairement, pour les valeurs spécifiées sous forme de chaîne avec des délimiteurs d'heure, il n'est pas obligatoire de spécifier les deux chiffres des heures, minutes et secondes qui sont inférieures à `10` . '`1979-10-30 1:2:3`' est valide et est équivalent à '`1979-10-30 01:02:03`'.

Les valeurs spécifiées sous forme de nombres doivent avoir 6, 8, 12, ou 14 chiffres de long. Si le nombre a 8 ou 14 chiffres, MySQL suppose que le format est `AAAAMMJJ` ou `AAAAMMJJHHMMSS` (respectivement) et que l'année est représentées par les 4 premiers chiffres. Si le nombre a 6 ou 12 chiffres, MySQL suppose que le format est `AAMMJJ` ou `AAMMJJHHMMSS` (respectivement) et format et que l'année est représentées par les 2 premiers chiffres. Les nombres qui ne sont pas d'une taille valide, sont complétés avec des 0 jusqu'à la taille lisible la plus proche.

Les valeurs spécifiées sous forme de chaînes sans délimiteurs sont interprétés en fonction de leur taille. Si la chaîne à 8 ou 14 caractères de long, l'année est supposée avoir 4 chiffres. Sinon, l'année est supposée avoir 2 chiffres. La chaîne est interprétée de gauche à droite, en lisant successivement l'année, le mois, la date, l'heure, les minutes et les secondes, tant qu'il y a des valeurs dans la chaîne. Cela signifie que vous ne devez pas utiliser de chaînes qui ont moins de 6 caractères. Par exemple, si vous spécifiez '`9903`', en pensant qu'il représente Mars 1999, vous vous apercevrez que MySQL insère à la place la date ``zéro'' dans votre table. Cela est dû au fait que si l'année et le mois sont 99 et 03, la date est 0, ce qui en fait une date invalide, qui est rejetée par MySQL.

Dans une certaines mesure, vous pouvez assigner des valeurs d'une colonne à une autre colonne d'un autre type. Cependant, vous devez vous attendre à quelques altération ou pertes de valeurs durant la conversion :

- Si vous assignez une valeur `DATE` à une colonne de type `DATETIME` ou `TIMESTAMP` , la partie représentant les heures vaudra '`00:00:00`', car les colonnes de type `DATE` ne contiennent pas d'information d'heure.
- Si vous assignez une valeur `DATETIME` ou `TIMESTAMP` à une colonne de type `DATE` , la composante heure sera perdue, car les colonnes de type `DATE` ne contiennent pas d'information d'heure.
- N'oubliez pas que même si les valeurs `DATETIME` , `DATE` , et `TIMESTAMP` peuvent être spécifiée avec différents formats, ces types n'ont pas les mêmes intervalle de validité. Par exemple, les valeurs de type `TIMESTAMP` ne peuvent pas prendre de valeur antérieure à 1970 ou postérieure à 2037. Cela signifie qu'une date telle que '`1968-01-01`', est légale dans les colonnes de type `DATETIME` , mais n'est pas valide pour les `TIMESTAMP` , et sera convertie en date zéro (`0`) si elle est assignée à une telle colonne.

Attention à certains pièges concernant les spécifications de dates :

- La syntaxe à délimiteur libre peut être une source de problème. Par exemple, une valeur telle que '`10:11:12`' ressemble à une heure, à cause du délimiteur ` '`:`' ', mais avec une colonne de date, elle sera interprétée comme la date '`2010-11-12`'. La valeur '`10:45:15`' sera convertie en '`0000-00-00`' car '`45`' n'est pas un mois valide.
- Le serveur MySQL effectue seulement la vérification de base la validité d'une date : jours `00-31` , mois `00-12` , années `1000-9999` . N'importe quelle date qui n'est pas dans cette marge retournera `0000-00-00` . Veuillez noter que ceci vous permet toujours de stocker les dates inadmissibles telles que `2002-04-31` . Il permet à des applications web de stocker des données d'une forme sans vérifier plus loin. Pour s'assurer qu'une date est valide, vous devrez effectuer un test dans votre application.
- Les années spécifiée avec deux chiffres seulement sont ambiguÎs, car il manque le siècle. MySQL interprète les années à deux chiffres suivant ces règles :
    - Les années de l'intervalle `00-69` sont converties en `2000-2069` .
    - Les années de l'intervalle `70-99` sont converties en `1970-1999` .

12.3.1.1 Comportement de TIMESTAMP avant MySQL 4.1

Le type `TIMESTAMP` est prévu pour stocker automatiquement l'heure courante lors d'une commande `INSERT` ou `UPDATE` . Si vous avez plusieurs colonnes de type `TIMESTAMP` , seule la première colonne sera mise à jour automatiquement.

La modification automatique de la première colonne de type `TIMESTAMP` survient si l'une des conditions suivantes est remplie :

- Vous insérez explicitement la valeur `NULL` dans la colonne.
- La colonne n'est pas spécifiée explicitement dans la commande `INSERT` ou `LOAD DATA INFILE` .
- La colonne n'est pas spécifiée explicitement dans la commande `UPDATE` et d'autres colonnes changent de valeurs (Notez qu'une commande `UPDATE` qui affecte une valeur qui est déjà celle de la colonne sera ignorée, et la colonne `TIMESTAMP` ne sera pas modifiée, car la ligne n'est pas à proprement parlée modifiée. MySQL ignore alors ces modifications pour des raisons d'efficacité).

Les autres colonnes de type `TIMESTAMP` , hormis la première, peuvent aussi prendre la valeur courante. Affectez−lui alors la valeur `NULL` ou la fonction `NOW()` .

Vous pouvez affecter à n'importe quelle colonne de type `TIMESTAMP` une valeur différente de l'heure et la date courant en fournissant une valeur explicite. Cela s'applique aussi à la première colonne de type `TIMESTAMP` . Par exemple, si vous voulez affecter la date de création d'une ligne à une colonne de type `TIMESTAMP` , mais ne plus y toucher ultérieurement :

- Laissez MySQL donner la valeur de la colonne lors de la création de la ligne. Cela va initialiser la colonne à la date et heure courante.
- Lorsque vous faites des modifications ultérieures, affectez explicitement à la colonne `TIMESTAMP` sa propre valeur.

```
UPDATE tbl_name
   SET timestamp_col = timestamp_col,
       other_col1 = new_value1,
       other_col2 = new_value2, ...
```

D'un autre coté, vous pouvez aussi facilement initialiser la colonne `TIMESTAMP` avec `NOW()` lors de sa création, puis ne plus la modifier ultérieurement.

L'intervalle de validité des valeurs `TIMESTAMP` va du début de l'année 1970 jusque quelque part durant l'année 2037, avec une précision d'une seconde. Les valeurs sont affichés comme des nombres entiers.

Le format d'affichage des valeurs `TIMESTAMP` dépend de la taille d'affichage, comme illustré ci−dessous. Le format total `TIMESTAMP` a 14 chiffres, mais les colonnes `TIMESTAMP` peuvent être créées avec des formats plus courts :

Type de colonne	Format d'affichage
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Toutes les colonnes de type `TIMESTAMP` ont la même taille de stockage, indépendamment de la taille d'affichage. Les formats les plus courants sont 6, 8, 12, et 14. Vous pouvez spécifier une taille arbitraire lors de la création de la table, mais 0 et les valeurs supérieures à 14 sont ramenées à 14. Les valeurs impaires sont aussi ramenées au nombre pair supérieur.

Les colonnes `TIMESTAMP` stockent une date valide, en utilisant la totalité de l'espace de stockage, quelque soit la valeur de l'affichage. Cela a les implication suivantes :

- Spécifiez toujours l'année, le mois et le jour, même si le type de colonne est `TIMESTAMP(4)` ou `TIMESTAMP(2)` . Sinon, la valeur ne sera pas légale et `0` sera stockée.
- Si vous utilisez la commande `ALTER TABLE` pour réduire la largeur d'une colonne `TIMESTAMP` , les informations qui étaient affichées sont désormais ``cachées'', mais pas détruites.
- Similairement, réduire une colonne de type `TIMESTAMP` ne cause aucune perte d'information, en dehors du fait que ces informations ne sont plus affichées.
- Bien que les valeurs `TIMESTAMP` soient stockées avec une précision d'une seconde, la seule fonction qui travaille directement avec ces valeurs est la fonction `UNIX_TIMESTAMP()` . Les autres fonctions opèrent sur des valeurs lues et formatées. Cela signifie que vous ne pouvez pas utiliser de fonctions telles que `HOUR()` ou `SECOND()` a moins que le format d'affichage de la valeur `TIMESTAMP` ne présente cette valeur. Par exemple, les heures ne sont jamais affichées dans une colonne de type `TIMESTAMP` à moins que la taille d'affichage de la colonne ne soit d'au moins 10. L'utilisation de la fonction `HOUR()` sur une valeur ayant un format d'affichage plus court que `10` retournera un résultat inutilisable.

12.3.1.2 Propriétés de TIMESTAMP depuis la version 4.1

Depuis MySQL 4.1.0, les propriétés des colonnes `TIMESTAMP` diffèrent des versions prédécentes de MySQL :

- Les colonnes `TIMESTAMP` sont affichées dans le même format que les valeurs des colonnes `DATETIME` .
- Les tailles d'affichage ne sont plus supportées comme décrit dans la section précédente. En d'autres termes, vous ne pouvez pas utiliser `TIMESTAMP(2)` , `TIMESTAMP(4)` , etc.

De plus, si le serveur MySQL est en mode `MAXDB` , `TIMESTAMP` est identique à `DATETIME` . C'est à dire que si le serveur fonctionne en mode `MAXDB` au moment où la table est créée, toutes les colonnes `TIMESTAMP` créées sont en fait de type `DATETIME` . En conséquence, ces colonnes utilisent le format d'affichage `DATETIME` , ont le même intervalle de validité et aucune mise à jour automatique n'intervient.

MySQL peut fonctionner en mode `MAXDB` depuis la version 4.1.1. Pour activer ce mode, lancez le serveur avec le mode `MAXDB` au démarrage avec l'option `--sql-mode=MAXDB` , ou en modifiant la variable `sql_mode` durant l'exécution :

```
mysql> SET GLOBAL sql_mode=MAXDB;
```

Un client peut mettre le serveur en mode `MAXDB` pour sa propre connexion avec la commande suivante :

```
mysql> SET SESSION sql_mode=MAXDB;
```

12.3.2 Le type TIME

MySQL lit et affiche les colonnes de type `TIME` au format `'HH:MM:SS'` (ou `'HHH:MM:SS'` pour les grandes quantités d'heures). Les valeurs de `TIME` vont de `'-838:59:59'` à `'838:59:59'` . La raison de cet intervalle de validité si large est que les colonnes de type `TIME` peuvent être utilisés pour représenter non seulement des heures du jour, mais aussi des durées entre deux événements (ce qui peut dépasser largement les 24 heures, ou même, être négatif).

Vous pouvez spécifier une valeur de type `TIME` avec différents formats :

- Une chaîne au format `'D HH:MM:SS.fraction'` . (Notez que MySQL ne stockera pas la fraction d'une valeur `TIME` .)

  Vous pouvez aussi utiliser l'une des syntaxes alternatives suivantes : `HH:MM:SS.fraction` , `HH:MM:SS` , `HH:MM` , `D HH:MM:SS` , `D HH:MM` , `D HH` ou `SS` . Ici, `D` peut prendre des valeurs entre 0 et 33.
- Une chaîne sans délimiteur au format `'HHMMSS'` , en supposant que cela puisse avoir un sens en terme de date. Par exemple, `'101112'` est interprété comme `'10:11:12'` , mais `'109712'` est invalide (le nombre de minutes n'a pas de sens), et devient la date zéro : `'00:00:00'` .
- Un nombre au format `HHMMSS` , en supposant que cela puisse avoir un sens en terme de date. Par exemple, `101112` est interprété comme `'10:11:12'` . Les formats alternatifs sont aussi compris : `SS` , `MMSS` , `HHMMSS` et `HHMMSS.fraction` . Notez que MySQL ne stocke pas encore les fractions de secondes.
- Le résultat d'une fonction qui retourne une valeur acceptable dans un contexte de valeurs `TIME` , comme `CURRENT_TIME` .

Pour les valeurs `TIME` spécifiées avec des délimiteurs, il n'est pas nécessaire de préciser deux chiffres pour les valeurs inférieurs à `10` pour les heures, minutes et secondes. `'8:3:2'` est la même chose que `'08:03:02'` .

Soyez soigneux lors de l'utilisation de valeurs ``courtes'' à une colonne de type `TIME` . MySQL interprète les valeurs en supposant que les chiffres de droite représentent les secondes (MySQL interprète les valeurs `TIME` comme des durées et non comme des heures d'une journée). Par exemple, vous pouvez penser que les valeurs `'11:12'` , `'1112'` et `1112` représentent `'11:12:00'` (12 minutes après 11 heures), mais MySQL les interprétera comme `'00:11:12'` (11 minutes, 12 secondes). Similairement, `'12'` et `12` représentent `'00:00:12'` . Les valeurs de `TIME` déclarées avec des `:` , au contraire, sont toujours traités comme des heures de journée. `'11:12'` signifiera `'11:12:00'` et non pas `00:11:12`

Les valeurs hors de l'intervalle de validité de `TIME` mais qui sont valides sont ramenées à la valeur maximale stockable la plus proche. Par exemple, `'-850:00:00'` et `'850:00:00'` sont respectivement converties en `'-838:59:59'` et `'838:59:59'` .

Les valeurs `TIME` non valides sont transformées en date zéro `'00:00:00'` . Notez que comme `'00:00:00'` est elle−même une valeur `TIME` valide, vous n'aurez pas le moyen de faire la différence entre une valeur `'00:00:00'` stockée en connaissance de cause, et `'00:00:00'` stockée à cause d'une erreur.

12.3.3 Le type YEAR

Le type `YEAR` est un type d'1 octet utilisé pour représenter les années.

MySQL extrait et affiche la valeur de `YEAR` au format `YYYY` . L'échelle va de `1901` à `2155` .

Vous pouvez spécifier la valeur de `YEAR` en plusieurs formats :

- Une chaîne de quatre chiffres entre `'1901'` et `'2155'` .
- Un nombre à quatre chiffres entre `1901` et `2155` .
- Une chaîne de deux chiffres entre `'00'` et `'99'` . Les valeurs entre `'00'` et `'69'` et entre `'70'` et `'99'` sont respectivement converties en valeurs `YEAR` comprises entre `2000` et `2069` d'une part, et `1970` et `1999` de l'autre.
- Une nombre de deux chiffres entre `1` et `99` . Les valeurs entre `1` et `69` et entre `70` et `99` sont respectivement converties en valeurs `YEAR` comprises entre `2001` et `2069` d'une part, et `1970` et `1999` d'autre part. Notez que le rang de valeurs pour les nombres à deux chiffres est totalement différent du rang pour les chaînes à deux chiffres parce que vous ne pouvez pas spécifier deux zéro directement en tant que nombre et le faire interpréter en tant que `2000` . Vous devez

le spécifier comme chaîne `'0'` ou `'00'` sinon il sera interprété comme `0000` .
- En tant que résultat d'une fonction retournant une valeur acceptable dans le contexte de `YEAR` , comme as `NOW()` .

Les valeurs illégales pour `YEAR` sont converties en `0000` .

12.3.4 An 2000 et les types date

MySQL lui même est compatible an 2000. (Compatibilité an 2000), mais les valeurs manipulées par MySQL peuvent ne pas l'être. N'importe quelle valeur n'ayant que deux chiffres pour représenter l'année est ambigu, car le siècle n'est pas précisé. Ces valeurs doivent être interprétées comme des valeurs à 4 chiffres, car MySQL stocke les années en interne en utilisant 4 chiffres.

Pour les types `DATETIME` , `DATE` , `TIMESTAMP` , et `YEAR` , MySQL interprète les dates ambigus en se basant sur les règles suivantes :

- Les valeurs d'années comprises dans l'intervalle `00-69` sont converties en `2000-2069` .
- Les valeurs d'années comprises dans l'intervalle `70-99` sont converties en `1970-1999` .

Gardez bien à l'esprit que ces règles ne sont que la meilleure approximation possible d'une valeur. Si l'heuristique proposée par MySQL ne fournit pas les valeurs attendues, vous devrez fournir une valeur sans ambiguïté. (à 4 chiffres)

`ORDER BY` ordonnera correctement les types `YEAR/DATE/DATETIME` à deux chiffres.

Notez aussi que quelques fonctions comme `MIN()` et `MAX()` convertiront un `TIMESTAMP/DATE` en nombre. Cela signifie qu'un timestamp avec une année à deux chiffres ne donneront pas de résultats corrects avec ces fonctions. Une solution dans ce cas est de convertir le `TIMESTAMP/DATE` en une année à 4 chiffres ou d'utiliser quelque chose comme `MIN(DATE_ADD(timestamp,INTERVAL 0 DAYS))` .

12.4 Les types chaînes

Les types chaînes sont `CHAR` , `VARCHAR` , `BLOB` , `TEXT` , `ENUM` , et `SET` . Cette section décrit comment ces types fonctionnent, leur besoins en espace et comment les utiliser dans vos requêtes.

12.4.1 Les types CHAR et VARCHAR

Les types `CHAR` et `VARCHAR` sont similaires, mais diffèrent dans la manière dont ils sont stockés et récupérés.

La longueur d'une colonne `CHAR` est fixée à la longueur que vous avez défini lors de la création de la table. La longueur peut être n'importe quelle valeur entre 1 et 255. (Dans la version 3.23 de MySQL, la longueur est comprise entre 0 et 255.) Quand une valeur `CHAR` est enregistrée, elle est complété à droite avec des espaces jusqu'à atteindre la valeur fixée. Quand une valeur de `CHAR` est lue, les espaces en trop sont retirés.

Les valeurs contenues dans les colonnes de type `VARCHAR` sont de tailles variables. Vous pouvez déclarer une colonne `VARCHAR` pour que sa taille soit comprise entre 1 et 255, exactement comme pour les colonnes `CHAR` . Par contre, contrairement à `CHAR` , les valeurs de `VARCHAR` sont stockées en utilisant autant de caractères que nécessaire, plus un octet pour mémoriser la longueur. Les valeurs ne sont pas complétées. Au contraire, les espaces finaux sont supprimés avant stockage (ce qui ne fait pas partie des spécifications ANSI SQL).

Si vous assignez une chaîne de caractères qui dépasse la capacité de la colonne `CHAR` ou `VARCHAR` , celle ci est tronquée jusqu'à la taille maximale du champ.

Le tableau suivant illustre les différences entre les deux types de colonnes en montrant les différences entre l'enregistrement dans une colonne `CHAR(4)` ou `VARCHAR(4)` :

Valeur	CHAR(4)	**Espace requis**	VARCHAR(4)	**Espace requis**
''	' '	4 octets	''	1 octet
'ab'	'ab '	4 octets	'ab'	3 octets
'abcd'	'abcd'	4 octets	'abcd'	5 octets
'abcdefgh'	'abcd'	4 octets	'abcd'	5 octets

Les valeurs lues dans les colonnes de type `CHAR(4)` et `VARCHAR(4)` seront les mêmes dans tous les cas, car les espaces finaux sont retirés des valeurs issues de colonnes de type `CHAR` lors de la lecture.

Les valeurs dans les colonnes `CHAR` et `VARCHAR` sont classées et comparées sans tenir compte de la casse, à moins que l'attribut `BINARY` n'ai été spécifié lors de la création de la table. L'attribut `BINARY` signifie que les valeurs sont classées et triées en tenant compte de la casse, suivant l'ordre des caractères ASCII de la machine ou est installé le serveur MySQL. `BINARY` n'affecte pas les méthodes de lecture et de stockage des valeurs.

L'attribut `BINARY` se propage dans une expression : il suffit qu'une seule colonne, utilisée dans une expression, ait l'attribut `BINARY` pour que toute l'expression ne tienne plus compte de la casse.

MySQL peut changer automatiquement le type d'une colonne `CHAR` ou `VARCHAR` lors de la création de la table. Changement de type de colonne automatique .

12.4.2 Les types BLOB et TEXT

Une valeur de type `BLOB` est un objet binaire de grande taille, qui peut contenir une quantité variable de données. Les quatre types `BLOB` (`TINYBLOB` , `BLOB` , `MEDIUMBLOB` , et `LONGBLOB`) ne différent que par la taille maximale de données qu'ils peuvent stocker. Taille de stockage des types .

Les quatre types `TEXT` (`TINYTEXT` , `TEXT` , `MEDIUMTEXT` , et `LONGTEXT` correspondent aux types `BLOB` équivalents, et ont les mêmes contraintes de stockage. Les seules différences entre les colonnes de type `BLOB` et celles de type `TEXT` se situent aux niveau des tris et comparaisons : Les tris, faits sur les `BLOB` , contrairement à ceux faits sur les `TEXT` , tiennent compte de la casse. En d'autres termes, une valeur `TEXT` est une valeur `BLOB` insensible à la casse.

Si vous assignez une valeur trop grande à une colonne de type `BLOB` ou `TEXT` , la valeur sera tronquée à la taille maximale possible.

Dans la majorité des cas, vous pouvez considérer une colonne de type `TEXT` comme une colonne de type `VARCHAR` , aussi grande que vous le souhaitez. De même, vous pouvez considérer une colonne de type `BLOB` comme une colonne de type `VARCHAR BINARY` . Les seules différences sont :

- Vous pouvez indexer les colonnes de type `BLOB` ou `TEXT` à partir de la version 3.23.2 de MySQL. Les versions plus anciennes ne peuvent pas indexer ces colonnes.
- Pour les index des colonnes `BLOB` et `TEXT` , vous devez spécifier une taille d'index. Pour les colonnes de type `CHAR` et `VARCHAR` , la taille du préfixe est optionnelle.
- Il n'y a pas de suppression des espaces finaux lors du stockage de valeur dans des colonnes de type `BLOB` et `TEXT` , ce qui est le cas dans pour les colonnes de type `VARCHAR` .

- Les colonnes `BLOB` et `TEXT` ne peuvent avoir de valeur par défaut. (`DEFAULT`)

`MyODBC` considère les valeurs `BLOB` comme des `LONGVARBINARY` et les valeurs `TEXT` comme des `LONGVARCHAR` .

Vous pouvez rencontrer les problèmes suivants, à cause de la grande taille des colonnes de type `BLOB` et `TEXT` , lors de leur utilisation :

- Si vous voulez utiliser les commandes `GROUP BY` ou `ORDER BY` sur une colonne de type `BLOB` ou `TEXT` , vous devez d'abord la convertir en un objet de taille fixe. Le meilleur moyen est d'utiliser la fonction `SUBSTRING` . Par exemple :

```
mysql> SELECT comment FROM nom_de_table,SUBSTRING(comment,20) AS substr
    ->                ORDER BY substr;
```

  Si vous le ne faites pas, seuls les `max_sort_length` premiers octets de la colonne seront utilisés pour le tri. La valeur par défaut de `max_sort_length` est 1024. Cette valeur peut être modifiée en utilisant l'option `-O` au démarrage du serveur `mysqld` . Vous pouvez utiliser la commande `GROUP BY` sur une colonne de type `BLOB` ou `TEXT` en spécifiant la position de la colonne, ou avec un alias :

```
mysql> SELECT id,SUBSTRING(blob_col,1,100) FROM nom_de_table GROUP BY 2;
mysql> SELECT id,SUBSTRING(blob_col,1,100) AS b FROM nom_de_table GROUP BY b;
```

- La taille maximale d'un objet `BLOB` ou `TEXT` est déterminée par son type, mais la valeur la plus grande que vous pouvez transmettre au programme client est déterminée par la quantité de mémoire disponible sur le serveur et par les tailles des buffers de communication. Vous pouvez changer la taille des buffers de communication, mais vous devez le faire sur le serveur et le client en même temps. Choix des paramètres du serveur .

  Par exemple, `mysql` et `mysqldump` vous autorises tous les deux à modifier la valeur cliente de `max_allowed_packet` . Voyez Choix des paramètres du serveur , L'outil de ligne de commande et `mysqldump` , exporter les structures de tables et les données .

Notez que chaque valeur `BLOB` ou `TEXT` est représentée en interne par un objet alloué séparément, contrairement à tous les autres types de colonne, pour lesquels la place de stockage est allouée une fois pour chaque colonne, lorsque la table est ouverte.

12.4.3 Le type ENUM

Une énumération `ENUM` est une chaîne dont la valeur est choisie parmi une liste de valeurs autorisées lors de la création de la table.

Cette chaîne peut aussi être la chaîne vide (`""`) ou `NULL` dans certaines circonstances :

- Si vous insérez une valeur illégale dans une énumération `ENUM` (c'est à dire, une chaîne qui n'est pas dans la liste de valeurs autorisées), la chaîne vide est insérée pour représenter une erreur. Cette chaîne peut être distinguée d'une chaîne vide 'normale' par le fait que cette chaîne à la valeur numérique 0. Nous reviendrons sur ce point plus tard.
- Si une colonne d'énumération est déclarée `NULL` , `NULL` devient aussi une valeur autorisée, et la valeur par défaut est alors `NULL` . Si une colonne d'énumération est déclarée `NOT NULL` , la valeur par défaut est le premier élément de la liste des valeurs autorisées.

Chaque élément de l'énumération dispose d'un index :

- Les valeurs de la liste des valeurs autorisées sont indexées à partir de 1.
- L'index de la chaîne vide (cas d'erreur) est 0. Cela signifie que vous pouvez utiliser la sélection suivante pour repérer les valeurs d'énumération invalides :

```
mysql> SELECT * FROM nom_de_table WHERE enum_col=0;
```

- L'index de la valeur `NULL` est `NULL` .

Par exemple, une colonne créée comme `ENUM("un", "deux", "trois")` peut prendre n'importe quelle valeur ci−dessous. L'index de chaque valeur est aussi présenté :

Valeur	**Index**
NULL	NULL
""	0
"un"	1
"deux"	2
"trois"	3

Une énumération peut avoir un maximum de 65535 éléments.

A partir de la version 3.23.51, les espaces en début et fin de chaîne sont automatiquement supprimés des éléments de l'énumération `ENUM` lorsque la table est créée.

La casse des lettres est sans importance lors de l'assignation de valeurs dans une énumération. Cependant, les valeurs lues dans la base auront la même casse que celle spécifiée lors de la création de la table.

Si vous lisez le contenu d'une énumération dans un contexte numérique, l'index de la valeur `ENUM` sera retournée. Par exemple, vous pouvez lire des valeurs numériques comme ceci :

```
mysql> SELECT enum_col+0 FROM nom_de_table;
```

Si vous stockez un nombre dans une colonne de type `ENUM` , le nombre sera traité comme un index, et la valeur stockée sera celle de l'élément ayant cet index (Attention, cela ne fonctionnera pas avec les commandes `LOAD DATA` , car cette dernière traite toutes les valeurs comme des chaînes). Il est déconseillé de stocker des valeurs numériques dans un `ENUM` car cela engendre des confusions. Par exemple, la colonne suivante est une énumération de chaînes contenant les valeurs `'0'` , `'1'` et `'2'` , mais leur valeur numérique est `1` , `2` et `3` :

```
numbers ENUM('0','1','2')
```

Les valeurs de type `ENUM` sont triées en fonction de l'ordre des éléments, fixé à la création de la table (en d'autres termes, les valeurs `ENUM` sont stockées en fonction de leur index). Par exemple, `"a"` précède `"b"` dans l'énumération `ENUM("a", "b")` , mais `"b"` précède `"a"` dans l'énumération `ENUM("b", "a")` . La chaîne vide précède toujours les chaînes non vides, et `NULL` précède toutes les valeurs. Si vous voulez connaître toutes les valeurs possibles d'une colonne de type `ENUM` , pensez à utiliser cette commande : `SHOW COLUMNS FROM nom_de_table LIKE enum_column_name` , puis analysez la définition de la colonne de type `ENUM` (deuxième colonne dans le résultat).

12.4.4 Le type SET

Un `SET` est une chaîne qui peut avoir zéro ou plusieurs valeurs, chacune doit être choisie dans une liste de valeurs définies lors de la création de la table. Les valeurs des colonnes `SET` composées de plusieurs membres sont définies en séparant celles−ci avec des virgules (`','`). Ce qui fait que la valeur d'un membre de `SET` ne peut contenir lui même de virgule.

Par exemple, une colonne définie en tant que `SET("un", "deux") NOT NULL` peut avoir l'une de ces valeurs :

```
 ""
"un"
"deux"
"un,deux"
```

Un `SET` peut avoir au plus 64 membres.

A partir de la version 3.23.51, les espaces en trop sont automatiquement effacés des membres de `SET` lorsque la table est créée.

MySQL enregistre les valeurs de SET numériquement. Le bit de poids faible de la valeur correspond alors au premier élément de la liste. Si vous utilisez une valeur SET dans un contexte numérique, les bits des éléments dans cet ensemble seront mis à un, et les autres à zéro. Par exemple, vous pouvez obtenir un entier à partir d'un ensemble comme ceci :

```
mysql> SELECT col_set+0 FROM nom_de_table;
```

Si un nombre est enregistré dans une colonne SET , les bits un à un de ce nombre représenteront les éléments placés dans cet ensemble. Supposons qu'une colonne est spécifiée en tant que `SET("a","b","c","d")` , les membres ont alors les valeurs suivantes :

SET **membre**	**Valeur décimale**	**Valeur binaire**
a	1	0001
b	2	0010
c	4	0100
d	8	1000

Si vous assignez 9 à cette colonne, cela donne 1001 en binaire, ce qui fait que les valeurs du premier et quatrième membres "a" et "d" sont sélectionnés et la valeur résultante est "a,d" .

Pour les valeurs se composant de plus d'un membre du SET , l'ordre des membres n'a pas d'importance lors des insertions. Le nombre d'occurrence d'un élément n'importe pas non plus. Lorsque la valeur sera lue ultérieurement, chaque élément n'apparaîtra qu'une seule fois, et dans l'ordre donné à la déclaration de la colonne. Par exemple, si une colonne est spécifiée comme `SET("a","b","c","d")` , alors "a,d" , "d,a" , et "d,a,a,d,d" seront tous représentés par "a,d" .

Si vous spécifiez une valeur incorrecte dans une colonne SET , la valeur sera ignorée.

Les valeurs de SET sont triées numériquement. La valeur NULL précède toutes les autres.

Normalement, vous exécuterez un SELECT sur une colonne SET en utilisant l'opérateur LIKE ou la fonction `FIND_IN_SET()` :

```
mysql> SELECT * FROM nom_de_table WHERE set_col LIKE '%value%';
mysql> SELECT * FROM nom_de_table WHERE FIND_IN_SET('value',set_col)>0;
```

Mais ce qui suit fonctionnera aussi :

```
mysql> SELECT * FROM nom_de_table WHERE set_col = 'val1,val2';
mysql> SELECT * FROM nom_de_table WHERE set_col & 1;
```

La première requête cherche les lignes qui correspondent exactement. La seconde ne cherche que les lignes contenant le premier membre du set.Si vous voulez connaître toutes les valeurs possible d'une colonne SET , vous devez utiliser : `SHOW COLUMNS FROM nom_de_table LIKE nom_colonne_set` et étudier la définition du SET dans la seconde colonne.

12.5 Capacités des colonnes

Les capacités de stockage de chaque type de colonnes de MySQL sont listés par catégories.

Capacités de stockage des colonnes numériques

Type de colonne	**Espace requis**
TINYINT	1 octet
SMALLINT	2 octets
MEDIUMINT	3 octets
INT , INTEGER	4 octets
BIGINT	8 octets
FLOAT(p)	4 if X <= 24 or 8 if 25 <= X <= 53
FLOAT	4 octets
DOUBLE PRECISION , REAL	8 octets

DECIMAL(M,D)	M+2 octets si D > 0, M+1 octets si D = 0 (D +2, si M < D)

Capacités de stockage des colonnes de temporelles

Type de colonne	Espace requis
DATE	3 octets
DATETIME	8 octets
TIMESTAMP	4 octets
TIME	3 octets
YEAR	1 octet

Capacités de stockage des colonnes de texte

Type de colonne	Espace requis
CHAR(M)	M octets, 1 <= M <= 255
VARCHAR(M)	L +1 octets, avec L <= M et 1 <= M <= 255
TINYBLOB , TINYTEXT	L +1 octets, avec L < 2^8
BLOB , TEXT	L +2 octets, avec L < 2^16
MEDIUMBLOB , MEDIUMTEXT	L +3 octets, avec L < 2^24
LONGBLOB , LONGTEXT	L +4 octets, avec L < 2^32
ENUM('valeur1','valeur2',...)	1 ou 2 octets, suivant le nombre d'éléments de l'énumération (65535 au maximum)
SET('valeur1','valeur2',...)	1, 2, 3, 4 ou 8 octets, suivant le nombre de membres de l'ensemble (64 au maximum)

Les types `VARCHAR` , `BLOB` et `TEXT` sont de longueur variable, et l'espace disque requis dépend de la taille réelle de la valeur présente dans la colonne, (taille représentée par L dans le tableau précédent) et non pas de la taille maximale de la colonne. Par exemple une colonne `VARCHAR(10)` peut contenir une chaîne de 10 caractères. L'espace requis est dans ce cas là la longueur de la chaîne (`L`), plus 1 octet pour enregistrer la longueur de celle ci. Pour la chaîne `'abcd'` , `L` est égal à 4 et l'espace requis est de 5 octets.

Les types `BLOB` et `TEXT` requièrent 1, 2, 3, ou 4 octets pour mémoriser la taille de la valeur dans la colonne, suivant la longueur maximale du type. Les types `BLOB` et `TEXT` .

Si une table inclus au moins une colonne de taille variable, la ligne sera de taille variable. Notez que lorsqu'une table est créée, MySQL peut, dans certaines circonstances, changer automatiquement une colonne de taille variable en colonne à taille fixe (et vice−versa). Changement de type de colonne automatique .

La taille d'un `ENUM` est déterminée par le nombre d'éléments de l'énumération. Un octet est nécessaire pour les énumérations ayant jusqu'à 255 valeurs possibles. Deux octets sont nécessaires pour les énumérations ayant jusqu'à 65535 valeurs possibles. Le type `ENUM` .

La taille d'un `SET` est déterminé par le nombre de ses membres. Si il y en a `N` , l'objet occupe `(N+7)/8` octets, arrondis à 1, 2, 3, 4, or 8 octets. Un `SET` peut avoir au plus 64 membres. Le type `SET` .

La taille maximale d'une ligne dans une table `MyISAM` est de 65534 octets. Les colonnes `BLOB` et `TEXT` acceptent jusqu'à 5−9 octets en dessous de cette taille.

12.6 Choisir le bon type de colonne

Pour une utilisation optimale des capacités de stockage, essayez d'utiliser le type le plus optimal dans chaque cas. Par exemple, si une colonnes du type entier sera utilisée pour des valeurs entre `1` et `99999` , le type `MEDIUMINT UNSIGNED` sera le plus approprié.

La représentation des valeurs monétaires est un problème commun. Avec MySQL, vous devrez utiliser le type `DECIMAL` . Il est sauvegardé en tant que chaîne, aucune perte de précision ne devrait avoir lieu. Si la précision n'est pas très importante, vous pouvez utiliser le type `DOUBLE` .

Pour une haute précision, vous pouvez toujours transcrire en nombre décimaux, et les enregistrer dans des `BIGINT` . Cela vous permettra d'effectuer tout vos calculs avec des entiers et de convertir à nouveau en nombre décimaux au besoin.

12.7 Utilisation des types de données issues d'autres SGBDR

Pou faciliter l'importation de code SQL issu d'autres systèmes de gestion de bases de données, MySQL convertit les types de colonnes comme le montre le tableau suivant. Cette conversion facilite l'import de structures de tables :

Autre dénomination	**Type MySQL**
BINARY(M)	CHAR(M) BINARY
CHAR VARYING(M)	VARCHAR(M)
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT (depuis MySQL 4.1.0)
MIDDLEINT	MEDIUMINT
VARBINARY(M)	VARCHAR(M) BINARY

La conversion du type de colonnes s'effectue lors de la création. Si vous créez une table avec des types issus d'un autre SGBDR puis que vous exécutez la commande `DESCRIBE nom_de_table` , MySQL fournira la structure de la table en utilisant les types équivalents.

13 Fonctions à utiliser dans les clauses SELECT et WHERE

Les expressions peuvent être utilisées en différents endroits des requêtes SQL, comme dans les clauses `ORDER BY` et `HAVING` des commandes `SELECT` , dans les clauses `WHERE` de `SELECT` , `DELETE` et `UPDATE` , ou dans les commandes `SET` . Les expressions peuvent contenir des valeurs littérales, des noms de colonnes, la valeur `NULL` , des fonctions et des opérateurs. Ce chapitre décrit les fonctions et opérateurs qui sont autorisés pour écrire une expression avec MySQL.

Une expression contenant `NULL` produira toujours la valeur `NULL` comme résultat. (Sauf contre−indication dans le manuel)

Note : Il ne doit pas y avoir d'espace entre le nom d'une fonction et la parenthèse ouvrante la suivant. Cela aide l'analyseur MySQL à distinguer les appels à ces fonction des références aux tables ou colonnes ayant le même nom qu'une fonction. Les espaces autour des arguments sont autorisés.

Vous pouvez forcer MySQL à accepter les espaces après les nom de fonctions grâce à l'option `--ansi` de `mysqld` , ou en utilisant l'option `CLIENT_IGNORE_SPACE` avec `mysql_connect` . Dans ce cas, toutes les fonctions définies deviendront des mots strictement réservés. Exécuter MySQL en mode ANSI .

Dans un soucis de simplicité, les affichages des résultats de `mysql` sont fournis sous forme abrégée. Par exemple :

```
 mysql> SELECT MOD(29,9);
1 rows in set (0.00 sec)

 +-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
```

est affiché comme ceci :

```
 mysql> SELECT MOD(29,9);
        -> 2
```

13.1 Opérateurs et fonctions tous types

13.1.1 Parenthèses

(...)

Utilisez les parenthèses pour forcer l'ordre des évaluations dans une expression. Par exemple :

```
 mysql> SELECT 1+2*3;
        -> 7
mysql> SELECT (1+2)*3;
        -> 9
```

13.1.2 Opérateurs de comparaison

Les opérations de comparaison donnent comme résultats `1` (TRUE), `0` (FALSE), ou `NULL` . Ces fonctions fonctionnent pour les nombres comme pour les chaînes. Les nombres sont automatiquement transformés en chaînes et les chaînes en nombres si besoin en est. (comme en

Perl)

MySQL effectue les comparaisons suivant les règles suivantes :

- Si l'un ou les deux arguments sont `NULL` , le résultat de la comparaison est `NULL` , exception faite pour l'opérateur `<=>` .
- Si les deux arguments de la comparaison sont des chaînes, ils seront comparés en tant que chaînes.
- Si les deux arguments sont des entiers, ils sont comparés en tant qu'entiers.
- Les valeurs hexadécimales sont traitées en tant que chaînes binaires si elles ne sont pas comparées à un nombre.
- Si l'un des arguments est une colonne de type `TIMESTAMP` ou `DATETIME` et que l'autre est une constante, celle ci est convertie en timestamp avant que la comparaison ne s'opère. Cela est fait pour être mieux compatible avec ODBC.
- Dans tous les autres cas, les arguments sont comparés en tant que nombres à décimale flottante. (réels)

Par défaut, la comparaison des chaînes s'effectue d'une façon insensible à la casse en utilisant le jeu de caractères courant (ISO−8859−1 Latin1 par défaut, qui fonctionne aussi très bien pour l'anglais).

Si vous comparez des chaînes insensibles à la chasse, avec les opérateurs standards (`=` , `<>` ..., mais pas avec `LIKE`) les espaces terminaux seront ignorés (espaces, tabulations et nouvelles lignes).

```
mysql> SELECT "a" ="A \n";
        -> 1
```

Les exemples suivants, montrent la conversion des chaînes en nombres pour les opérations de comparaison :

```
mysql> SELECT 1 > '6x';
         -> 0
mysql> SELECT 7 > '6x';
         -> 1
mysql> SELECT 0 > 'x6';
         -> 0
mysql> SELECT 0 = 'x6';
         -> 1
```

`=`

Egal :

```
mysql> SELECT 1 = 0;
        -> 0
mysql> SELECT '0' = 0;
        -> 1
mysql> SELECT '0.0' = 0;
        -> 1
mysql> SELECT '0.01' = 0;
        -> 0
mysql> SELECT '.01' = 0.01;
        -> 1
```

`<=>`

Comparaison compatible avec `NULL` . Cet opérateur fait une comparaison d'égalité comme l'opérateur `=` , mais retourne `1` plutôt que `NULL` si les deux opérandes sont `NULL` , et `0` plutôt que `NULL` si un opérande est `NULL` .

```
mysql> SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
        -> 1, 1, 0
mysql> SELECT 1 = 1, NULL = NULL, 1 = NULL;
        -> 1, NULL, NULL
```

`<=>` a été ajouté en MySQL 3.23.0.

`<>`

!=

Différent :

```
mysql> SELECT '.01' <> '0.01';
        -> 1
mysql> SELECT .01 <> '0.01';
```

```
        -> 0
mysql> SELECT 'zapp' <> 'zappp';
        -> 1
```

`<=`

Inférieur ou égal :

```
mysql> SELECT 0.1 <= 2;
        -> 1
```

`<`

Strictement inférieur :

```
mysql> SELECT 2 < 2;
        -> 0
```

`>=`

Supérieur ou égal :

```
mysql> SELECT 2 >= 2;
        -> 1
```

`>`

Strictement supérieur :

```
mysql> SELECT 2 > 2;
        -> 0
```

`IS NULL`

IS NOT NULL

Tester si une valeur est ou n'est pas `NULL` :

```
mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
        -> 0 0 1
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
        -> 1 1 0
```

Pour être compatible avec les autres programmes, MySQL gère les appels qui utilisent `IS NULL` de la façon suivante :

Vous pouvez trouver le dernier enregistrement inséré en utilisant :

```
SELECT * FROM nom_de_table WHERE auto_col IS NULL
```

Cela peut être interdit en mettant `SQL_AUTO_IS_NULL=0` . Syntaxe de `SET` .

Pour les colonnes `NOT NULL DATE` et `DATETIME` , vous pouvez sélectionner lignes ayant la date spéciale `0000-00-00` avec :

```
SELECT * FROM nom_de_table WHERE date_column IS NULL
```

C'est une fonctionnalité nécessaire pour que certaines applications ODBC fonctionnent (car ODBC ne supporte pas les dates `0000-00-00`)

`expression BETWEEN min AND max`

Si `expression` est supérieure ou égale à `min` et `expression` est inférieure ou égale à `max` , `BETWEEN` retourne `1` , sinon `0` . Ceci est équivalent à l'expression `(min <= expression AND expression <= max)` si tous les arguments sont du même type. Dans tous les autres cas, la conversion de type prends place, selon les règles suivantes, mais appliquée aux trois arguments. **Notez** que avant la 4.0.5, les arguments étaient convertis au type de `expr` .

```
mysql> SELECT 1 BETWEEN 2 AND 3;
        -> 0
mysql> SELECT 'b' BETWEEN 'a' AND 'c';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND '3';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND 'x-3';
        -> 0
```

`expr NOT BETWEEN min AND max`

Même chose que `NOT (expr BETWEEN min AND max)` .

`COALESCE(list)`

Retourne le premier élément non− `NULL` de la liste :

```
mysql> SELECT COALESCE(NULL,1);
        -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
        -> NULL
```

`GREATEST(value1,value2,...)`

Avec deux ou plusieurs arguments, retourne la valeur la plus grande. Les arguments sont comparés en utilisant les mêmes règles que pour `LEAST()` .

```
mysql> SELECT GREATEST(2,0);
        -> 2
mysql> SELECT GREATEST(34.0,3.0,5.0,767.0);
        -> 767.0
mysql> SELECT GREATEST('B','A','C');
        -> 'C'
```

Avant MySQL 3.22.5, vous pouvez utiliser `MAX()` au lieu de `GREATEST()` .

`expr IN (valeur,...)`

Retourne `1` si `expr` est l'une des valeurs dans la liste `IN` , sinon retourne `0` . Si toutes les valeurs sont des constantes, toutes les valeurs sont évaluées avec le type de `expr` et triées. La recherche de l'élément est alors faite en utilisant la recherche binaire. Cela signifie que `IN` est très rapide si les valeurs contenues dans la liste `IN` sont toutes des constantes. Si `expr` est une chaîne sensible à la casse, la comparaison est faite dans un contexte sensible à la casse :

```
mysql> SELECT 2 IN (0,3,5,'wefwf');
        -> 0
mysql> SELECT 'wefwf' IN (0,3,5,'wefwf');
        -> 1
```

Depuis MySQL version 4.1, une clause IN() peut aussi contenir une sous−requête. Sous−requêtes avec `ANY` , `IN` et `SOME` .

`expr NOT IN (value,...)`

Même chose que `NOT (expr IN (valeur,...))` .

`ISNULL(expr)`

Si `expr` est `NULL` , `ISNULL()` retourne `1` , sinon il retourne `0` :

```
mysql> SELECT ISNULL(1+1);
        -> 0
mysql> SELECT ISNULL(1/0);
        -> 1
```

Notez que la comparaison de deux valeurs `NULL` en utilisant `=` donnera toujours false !

`INTERVAL(N,N1,N2,N3,...)`

Retourne `0` si `N` < `N1` , `1` si `N` < `N2` etc... Tous les arguments sont traités en tant qu'entiers. Il est requis que `N1` < `N2` < `N3` < `...` < `Nn` pour que cette fonction fonctionne correctement. Cela est due à la recherche binaire utilisée (très rapide) :

```
mysql> SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);
        -> 3
mysql> SELECT INTERVAL(10, 1, 10, 100, 1000);
        -> 2
mysql> SELECT INTERVAL(22, 23, 30, 44, 200);
        -> 0
```

`LEAST(value1,value2,...)`

Avec deux arguments ou plus, retourne la plus petite valeur. Les arguments sont comparés avec les règles suivantes :

- Si la valeur retournée est utilisée dans un contexte `INTEGER` ou que tous les arguments sont des entiers, ils sont comparés comme des entiers.
- Si la valeur retournée est utilisée dans un contexte `REAL` ou que tous les arguments sont des entiers, ils sont comparés comme des entiers.
- Si un des arguments est une chaîne sensible à la casse, les arguments sont comparés comme des chaînes sensibles à la casse.
- Dans les autres cas, les arguments sont comparés comme des chaînes insensibles à la casse.

```
mysql> SELECT LEAST(2,0);
        -> 0
mysql> SELECT LEAST(34.0,3.0,5.0,767.0);
        -> 3.0
mysql> SELECT LEAST('B','A','C');
        -> 'A'
```

Avant MySQL 3.22.5, vous pouvez utiliser `MIN()` au lieu de `LEAST()` .

Notez que les conversions précédentes peuvent produire des résultats étranges dans certains cas limites :

```
mysql> SELECT CAST(LEAST(3600, 9223372036854775808.0) as SIGNED);
        -> -9223372036854775808
```

Cela arrive parce que MySQL lit `9223372036854775808.0` dans un contexte d'entier. La représentation entière n'est pas suffisante pour contenir la valeur, alors elle est transformée en entier signé.

13.1.3 Opérateurs logiques

En SQL, tous les opérateurs logiques évaluent à `TRUE` , `FALSE` ou `NULL` (INCONNU). En MySQL, c'est implémenté en `1` (TRUE), `0` (FALSE), et `NULL` . La plupart de ce qui suit est est commun entre les différents bases de données SQL, pourtant, certains système pourraient retourner une valeur non nulle pour TRUE (pas obligatoirement 1).

`NOT`
`!`

NOT (NON) logique. Evalue à `1` si l'opérande est `0` , à `0` si l'opérande est non nulle, et `NOT NULL` retourne `NULL` .

```
mysql> SELECT NOT 10;
        -> 0
mysql> SELECT NOT 0;
        -> 1
mysql> SELECT NOT NULL;
        -> NULL
mysql> SELECT ! (1+1);
        -> 0
mysql> SELECT ! 1+1;
        -> 1
```

Le dernier exemple donne `1` car l'expression est évaluée comme `(!1)+1` .

`AND`

`AND` (ET) logique. Evalue à `1` si toutes les opérandes sont différentes de zéro et de `NULL` , à `0` si l'une des opérandes est `0` , dans les autres cas, `NULL` est retourné.

```
mysql> SELECT 1 && 1;
        -> 1
mysql> SELECT 1 && 0;
        -> 0
mysql> SELECT 1 && NULL;
        -> NULL
mysql> SELECT 0 && NULL;
        -> 0
mysql> SELECT NULL && 0;
        -> 0
```

Notez que pour les versions antérieures à la 4.0.5 l'évaluation est interrompue lorsque `NULL` est rencontré, au lieu de continuer à tester une éventuelle existence de `0` . Cela signifie que dans ces versions, `SELECT (NULL AND 0)` retourne `NULL` au lieu de `0` . En 4.0.5 le code a été revu pour que le résultat réponde toujours au normes ANSI tout en optimisant le plus possible.

`OR`
`||`

OR (OU inclusif) logique. Evalue à `1` si aucune opérande n'est nulle, à `NULL` si l'une des opérandes est `NULL` , sinon `0` est retourné.

```
mysql> SELECT 1 || 1;
        -> 1
mysql> SELECT 1 || 0;
        -> 1
mysql> SELECT 0 || 0;
        -> 0
mysql> SELECT 0 || NULL;
        -> NULL
mysql> SELECT 1 || NULL;
        -> 1
```

`XOR`

XOR (OU exclusif) logique. Retourne `NULL` si l'une des opérandes est `NULL` . Pour les opérandes non− `NULL` , évalue à `1` si un nombre pair d'opérandes est non−nul, sinon `0` est retourné.

```
mysql> SELECT 1 XOR 1;
        -> 0
mysql> SELECT 1 XOR 0;
        -> 1
mysql> SELECT 1 XOR NULL;
        -> NULL
mysql> SELECT 1 XOR 1 XOR 1;
        -> 1
```

`a XOR b` est mathématiquement égal à `(a AND (NOT b)) OR ((NOT a) and b)` .

13.1.4 Sensibilité à la casse

`BINARY`

L'opérateur `BINARY` modifie la chaîne qui le suit en une chaîne binaire. C'est une solution simple pour forcer la comparaison de colonnes à être sensible à la casse même si la colonne n'est pas définie comme étant de type `BINARY` ou `BLOB` :

```
mysql> SELECT "a" = "A";
        -> 1
mysql> SELECT BINARY "a" = "A";
```

```
        -> 0
```

`BINARY string` est un raccourci pour `CAST(string AS BINARY)` . Fonctions de transtypage . `BINARY` a été introduit dans MySQL à partir de la version 3.23.0.

Notez que dans quelques cas, MySQL n'est pas capable d'utiliser l'index efficacement lorsque vous modifiez une colonne indexée en `BINARY` .

Si vous voulez comparer un champ de type `BLOB` d'une manière insensible à la casse, vous pouvez suivre cette procédure :

- Avant MySQL 4.1.1, utilisez la fonction `UPPER()` pour convertir le `BLOB` en majuscules avant la comparaison :

  ```
  SELECT 'A' LIKE UPPER(blob_col) FROM tbl_name;
  ```

  Si la valeur de comparaison est en minuscules, convertisses le `BLOB` en minuscules avec la fonction `LOWER()` .
- Depuis MySQL 4.1.1, les colonnes `BLOB` ont un jeu de caractère de type `binary` , ce qui signifie qu'il n'y a pas de concept de minuscules. Pour faire une comparaison sans la casse, utilisez la fonction `CONVERT()` pour convertir une valeur `BLOB` en un jeu de caractères qui n'est pas insensible à la casse. Le résultat est une chaîne non−binaires, et l'opérateur `LIKE` n'est pas sensible à la casse :

  ```
  SELECT 'A' LIKE CONVERT(blob_col USING latin1) FROM tbl_name;
  ```

  Pour utiliser d'autres jeux de caractères, remplacez le nom de `latin1` dans l'exemple précédent.

`CONVERT()` peut aussi être utilisée pour comparer des chaînes qui sont représentées dans d'autres jeux de caractères.

13.2 Les fonctions de contrôle

IFNULL(expr1,expr2)

Si l'argument `expr1` n'est pas `NULL` , la fonction `IFNULL()` retournera l'argument `expr1` , sinon elle retournera l'argument `expr2` . La fonction `IFNULL()` retourne une valeur numérique ou une chaîne de caractères, suivant le contexte d'utilisation :

```
mysql> SELECT IFNULL(1,0);
        -> 1
mysql> SELECT IFNULL(NULL,10);
        -> 10
mysql> SELECT IFNULL(1/0,10);
        -> 10
mysql> SELECT IFNULL(1/0,'oui');
        -> 'oui'
```

En version 4.0.6 et plus récent, le résultat par défaut de `IFNULL(expr1,expr2)` est le plus "général" des deux expressions, dans l'ordre de type `STRING` , `REAL` ou `INTEGER` . La différence avec les anciennes versions de MySQL ne seront notables que si vous créez une table basée sur des expressions, ou si MySQL stocke en interne des valeurs issues de `IFNULL()` dans une table temporaire.

```
CREATE TABLE foo SELECT IFNULL(1,"test") as test;
```

En MySQL 4.0.6, le type de la colonne `test` est `CHAR(4)` tandis que dans les versions plus anciennes, vous auriez obtenu un `BIGINT` .

NULLIF(expr1,expr2)

Si l'expression `expr1 = expr2` est vrai, la fonction retourne `NULL` sinon elle retourne `expr1` . Cela revient à faire `CASE WHEN x = y THEN NULL ELSE x END` :

```
mysql> SELECT NULLIF(1,1);
        -> NULL
mysql> SELECT NULLIF(1,2);
        -> 1
```

Notez que l'argument `expr1` est évalué deux fois dans MySQL si les arguments sont égaux.

IF(expr1,expr2,expr3)

Si l'argument `expr1` vaut TRUE (`expr1 <> 0` et `expr1 <> NULL`) alors la fonction `IF()` retourne l'argument `expr2` , sinon, elle retourne l'argument `expr3` . La fonction `IF()` retourne une valeur numérique ou une chaîne de caractères, suivant le contexte d'utilisation :

```
 mysql> SELECT IF(1>2,2,3);
        -> 3
mysql> SELECT IF(1<2,'oui','non');
        -> 'oui'
mysql> SELECT IF(STRCMP('test','test1'),'non','oui');
        -> 'non'
```

Si l'argument `expr2` ou `expr3` est explicitement `NULL` alors le type du résultat de la fonction `IF()` est le type de la colonne non `NULL` . (Ce comportement est nouveau dans MySQL 4.0.3).

L'argument `expr1` est évalué comme un entier, cela signifie que si vous testez un nombre à virgule flottante ou une chaîne de caractères, vous devez utiliser une opération de comparaison :

```
 mysql> SELECT IF(0.1,1,0);
        -> 0
mysql> SELECT IF(0.1<>0,1,0);
        -> 1
```

Dans le premier exemple ci−dessus, `IF(0.1)` retourne `0` parce que `0.1` est converti en une chaîne de caractères, ce qui revient à tester `IF(0)` . Ce n'est certainement pas ce que vous désireriez. Dans le second exemple, la comparaison teste si le nombre à virgule flottante est différent de zéro. Le résultat de cette comparaison sera un entier.

Le type de la fonction `IF()` (ce qui peut être important s'il est stocké dans une table temporaire) est calculé, dans la Version 3.23 de MySQL, comme suit :

Expression	**Valeur retournée**
expr2 ou expr3 retourne une chaîne	chaîne
expr2 ou expr3 retourne un nombre à virgule	nombre à virgule
expr2 ou expr3 retourne un entier	entier

Si expr2 et expr3 sont des chaînes de caractères, alors le résultat est insensible à la casse si les deux chaînes de caractères sont insensibles à la casse. (A partir de la version 3.23.51 de MySQL)

`CASE valeur WHEN [compare-value] THEN résultat [WHEN [compare-value] THEN résultat ...] [ELSE résultat] END`
`CASE WHEN [condition] THEN résultat [WHEN [condition] THEN résultat ...] [ELSE résultat] END`

La première version retourne `résultat` si `valeur=compare-value` . La seconde version retourne le résultat de la première condition qui se réalise. Si aucune des conditions n'est réalisé, alors le résultat de la clause `ELSE` est retourné. Si il n'y a pas de clause `ELSE` alors `NULL` est retourné :

```
 mysql> SELECT CASE 1 WHEN 1 THEN "un"
          WHEN 2 THEN "deux" ELSE "plus" END;
       -> "un"
mysql> SELECT CASE WHEN 1>0 THEN "vrai" ELSE "faux" END;
       -> "vrai"
mysql> SELECT CASE BINARY "B" WHEN "a" THEN 1 WHEN "b" THEN 2 END;
       -> NULL
```

Le type de la valeur retournée (`INTEGER` , `DOUBLE` ou `STRING`) est de même type que la première valeur retournée (l'expression après le premier `THEN`).

13.3 Fonctions de chaînes de caractères

Les fonctions qui traitent les chaînes de caractères retournent `NULL` si la longueur du résultat finit par dépasser la taille maximale du paramètre `max_allowed_packet` , défini dans la configuration du serveur. Choix des paramètres du serveur .

Pour les fonctions qui opèrent sur des positions à l'intérieur d'une chaîne, la position initiale est 0.

`ASCII(str)`

Retourne le code ASCII du premier caractère de la chaîne de caractères `str` . Retourne `0` si la chaîne de caractère `str` est vide. Retourne `NULL` si la chaîne de caractères `str` est `NULL` . `ASCII()` fonctionne avec des valeurs numériques entre `0` et `255` .

```
mysql> SELECT ASCII('2');
        -> 50
mysql> SELECT ASCII(2);
        -> 50
mysql> SELECT ASCII('dx');
        -> 100
```

Voir aussi la fonction `ORD()` .

`BIN(N)`

Retourne une chaîne de caractères représentant la valeur binaire de l'argument `N` , où l'argument `N` est un nombre de type `BIGINT` . Cette fonction est un équivalent de `CONV(N,10,2)` . Retourne `NULL` si l'argument `N` est `NULL` .

```
mysql> SELECT BIN(12);
        -> '1100'
```

`BIT_LENGTH(str)`

Retourne le nombre de bits de la chaîne de caractères `str` .

```
mysql> SELECT BIT_LENGTH('text');
        -> 32
```

`BIT_LENGTH()` a été ajouté en MySQL 4.0.2.

`CHAR(N,...)`

La fonction `CHAR()` interprète les arguments comme des entiers et retourne une chaîne de caractères, constituée des caractères, identifiés par leur code ASCII. Les valeurs `NULL` sont ignorées :

```
mysql> SELECT CHAR(77,121,83,81,'76');
        -> 'MySQL'
mysql> SELECT CHAR(77,77.3,'77.3');
        -> 'MMM'
```

`CHAR_LENGTH(str)`

Retourne le nombre de caractères de la chaîne `str` : Un caractère multi−octets compte comme un seul caractère. Cela signifie que pour une chaîne contenant 5 caractères de 2 octets, `LENGTH()` retournera `10` , alors que `CHAR_LENGTH()` retournera `5` .

`CHARACTER_LENGTH(str)`

`CHARACTER_LENGTH()` est un synonyme de `CHAR_LENGTH()` .

`COMPRESS(string_to_compress)`

Compresse une chaîne. Cette fonction requiert la présence de la bibliothèque `zlib` . Sinon, la valeur retournée sera toujours `NULL` .

```
mysql> SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
        -> 21
mysql> SELECT LENGTH(COMPRESS(''));
        -> 0
mysql> SELECT LENGTH(COMPRESS('a'));
        -> 13
mysql> SELECT LENGTH(COMPRESS(REPEAT('a',16)));
        -> 15
```

La chaîne compressée est stockée de cette manière :

Les chaînes vides sont stockées comme des chaînes vides.

Les chaînes non−vides sont stockées avec 4 octets de plus, indiquant la taille de la chaîne non compressée, suivie de la chaîne compressée. Si la chaîne se termine avec des espaces, un point supplémentaire `'.'` est ajouté, pour éviter que les espaces terminaux soient supprimés de la chaîne. N'utilisez pas les types `CHAR` ou `VARCHAR` pour stocker des chaînes compressée. Il est mieux d'utiliser un type `BLOB` .

`COMPRESS()` a été ajouté en MySQL 4.1.1.

`CONCAT(str1,str2,...)`

Retourne une chaîne représentant la concaténation des arguments. Retourne `NULL` si un des arguments est `NULL` . Cette fonction peut prendre plus de 2 arguments. Si un argument est un nombre, il sera converti en son équivalent sous forme de chaîne de caractères :

```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
mysql> SELECT CONCAT(14.3);
```

```
        -> '14.3'
```

CONCAT_WS(separator, str1, str2,...)

La fonction `CONCAT_WS()` signifie `CONCAT With Separator` , c'est−à−dire "concaténation avec séparateur. Le premier argument est le séparateur utilisé pour le reste des arguments. Le séparateur peut être une chaîne de caractères, tout comme le reste des arguments. Si le séparateur est `NULL` , le résultat sera `NULL` . Cette fonction ignorera tous les arguments de valeur `NULL` et vides, hormis le séparateur. Le séparateur sera ajouté entre tous les arguments à concaténer :

```
mysql> SELECT CONCAT_WS(",","Premier nom","Deuxième nom","Dernier nom");
       -> 'Premier nom,Deuxième nom,Dernier nom'
mysql> SELECT CONCAT_WS(",","Premier nom",NULL,"Dernier nom");
       -> 'Premier nom,Dernier nom'
```

CONV(N,from_base,to_base)

Convertit des nombres entre différentes bases. Retourne une chaîne de caractères représentant le nombre `N` , convertit de la base `from_base` vers la base `to_base` . La fonction retourne `NULL` si un des arguments est `NULL` . L'argument `N` est interprété comme un entier, mais peut être spécifié comme un entier ou une chaîne de caractères. Le minimum pour la base est `2` et son maximum est `36` . Si `to_base` est un nombre négatif, `N` sera considéré comme un nombre signé. Dans le cas contraire, `N` sera traité comme un nombre non−signé. La fonction `CONV` travaille avec une précision de 64 bits :

```
mysql> SELECT CONV("a",16,2);
        -> '1010'
mysql> SELECT CONV("6E",18,8);
        -> '172'
mysql> SELECT CONV(-17,10,-18);
        -> '-H'
mysql> SELECT CONV(10+"10"+'10'+0xa,10,10);
        -> '40'
```

ELT(N,str1,str2,str3,...)

Retourne `str1` si `N` = `1` , `str2` si `N` = `2` , et ainsi de suite. Retourne `NULL` si `N` est plus petit que `1` ou plus grand que le nombre d'arguments. La fonction `ELT()` est un complément de la fonction `FIELD()` :

```
mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
        -> 'ej'
mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
        -> 'foo'
```

EXPORT_SET(bits,on,off,[séparateur,[nombre_de_bits]])

Retourne une chaîne dont tous les bits à 1 dans "bit" sont représentés par la chaîne "on", et dont tous les bits à 0 sont représentés par la chaîne "off". Chaque chaîne est séparée par 'séparateur' (par défaut, une virgule `'','''`) et seul "nombre_de_bits" (par défaut, 64) "bits" est utilisé :

```
mysql> SELECT EXPORT_SET(5,'Y','N',',',4)
        -> Y,N,Y,N
```

FIELD(str,str1,str2,str3,...)

Retourne l'index de la chaîne `str` dans la liste `str1` , `str2` , `str3` , `...` . Retourne `0` si `str` n'est pas trouvé. La fonction `FIELD()` est un complément de la fonction `ELT()` :

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 2
mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 0
```

FIND_IN_SET(str,strlist)

Retourne une valeur de `1` à `N` si la chaîne `str` se trouve dans la liste `strlist` constituée de `N` chaînes. Une liste de chaîne est une chaîne composée de sous−chaînes séparées par une virgule `','` . Si le premier argument est une chaîne constante et le second, une colonne de type `SET` , la fonction `FIND_IN_SET()` est optimisée pour utiliser une recherche binaire très rapide. Retourne `0` si `str` n'est pas trouvé dans la liste `strlist` ou si la liste `strlist` est une chaîne vide. Retourne `NULL` si l'un des arguments est `NULL` . Cette fonction ne fonctionne pas correctement si le premier argument contient une virgule `','` :

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
        -> 2
```

HEX(N_or_S)

Si l'argument N_OR_S est un nombre, cette fonction retournera une chaîne de caractère représentant la valeur hexadécimale de l'argument `N` , où l'argument `N` est de type `BIGINT` . Cette fonction est un équivalent de `CONV(N,10,16)` .Si N_OR_S est une chaîne de caractères, cette fonction retournera une chaîne de caractères hexadécimale de N_OR_S où chaque caractère de N_OR_S est converti en 2 chiffres hexadécimaux. C'est l'inverse de la chaîne `0xff` .

```
mysql> SELECT HEX(255);
        -> 'FF'
mysql> SELECT HEX("abc");
        -> 616263
mysql> SELECT 0x616263;
        -> "abc"
```

`INSERT(str,pos,len,newstr)`

Retourne une chaîne de caractères `str` , après avoir remplacé la portion de chaîne commençant à la position `pos` et de longueur `len` caractères, par la chaîne `newstr` :

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
        -> 'QuWhattic'
```

Cette fonction gère les caractères multi−octets.

`INSTR(str,substr)`

Retourne la position de la première occurrence de la chaîne `substr` dans la chaîne de caractères `str` . Cette fonction est exactement la même que la fonction `LOCATE()` , à la différence que ces arguments sont inversés :

```
mysql> SELECT INSTR('foobarbar', 'bar');
        -> 4
mysql> SELECT INSTR('xbar', 'foobar');
        -> 0
```

Cette fonction gère les caractères multi−octets. Dans la version 3.23 de MySQL, cette fonction est sensible à la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible à la casse si l'argument est une chaîne de caractères binaire.

`LCASE(str)`

`LCASE()` est un synonyme de `LOWER()` .

`LEFT(str,len)`

Retourne les `len` caractères les plus à gauche de la chaîne de caractères `str` :

```
mysql> SELECT LEFT('foobarbar', 5);
        -> 'fooba'
```

Cette fonction gère les caractères multi−octets.

`LENGTH(str)`

Retourne la taille de la chaîne `str` , mesurée en octets. Un caractère multi−octets compte comme un seul caractère. Cela signifie que pour une chaîne contenant 5 caractères de 2 octets, `LENGTH()` retournera `10` , alors que `CHAR_LENGTH()` retournera `5` .

```
mysql> SELECT LENGTH('text');
        -> 4
```

`LOAD_FILE(file_name)`

Lit le fichier `file_name` et retourne son contenu sous la forme d'une chaîne de caractères. Le fichier doit se trouver sur le serveur qui exécute MySQL, vous devez spécifier le chemin absolu du fichier et vous devez avoir les droits en lecture sur celui−ci. Le fichier doit pouvoir être lisible par tous et doit être plus petit que `max_allowed_packet` .Si ce fichier n'existe pas ou ne peut pas être lu pour différentes raisons, la fonction retourne `NULL` :

```
mysql> UPDATE tbl_name
           SET blob_column=LOAD_FILE("/tmp/picture")
           WHERE id=1;
```

Si vous n'utilisez pas la version 3.23 de MySQL, vous devez lire le fichier depuis votre application et créer ainsi votre requête `INSERT` vous−même, pour mettre à jour la base de données avec le contenu de ce fichier. Une des possibilités pour réaliser ceci, si vous utilisez la bibliothèque MySQL++, peut être trouvée à http://www.mysql.com/documentation/mysql++/mysql++−examples.html .

`LOCATE(substr,str)`
`LOCATE(substr,str,pos)`

Retourne la position de la première occurrence de la chaîne `substr` dans la chaîne de caractères `str` . Retourne `0` si `substr` ne se trouve pas dans la chaîne de caractères `str` :

```
mysql> SELECT LOCATE('bar', 'foobarbar');
        -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
        -> 0
mysql> SELECT LOCATE('bar', 'foobarbar',5);
        -> 7
```

Cette fonction gère les caractères multi−octets. Dans la version 3.23 de MySQL, cette fonction est sensible à la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible à la casse si l'argument est une chaîne de caractères binaire.

`LOWER(str)`

Retourne la chaîne `str` avec tous les caractères en minuscules, en fonction du jeu de caractères courant (par défaut, c'est le jeu `ISO−8859−1 Latin1`) :

```
mysql> SELECT LOWER('QUADRATIQUE');
       −> 'quadratique'
```

Cette fonction gère les caractères multi−octets.

`LPAD(str,len,padstr)`

Retourne la chaîne de caractères `str` , complétée à gauche par la chaîne de caractères `padstr` jusqu'à ce que la chaîne de caractères `str` atteigne `len` caractères de long. Si la chaîne de caractères `str` est plus longue que `len'` caractères, elle sera raccourcie de `len` caractères.

```
mysql> SELECT LPAD('hi',4,'??');
       −> '??hi'
```

`LTRIM(str)`

Retourne la chaîne de caractères `str` sans les espaces initiaux :

```
mysql> SELECT LTRIM('  barbar');
       −> 'barbar'
```

`MAKE_SET(bits,str1,str2,...)`

Retourne une liste (une chaîne contenant des sous−chaînes séparées par une virgule `','`) constituée de chaînes qui ont le bit correspondant dans la liste `bits` . `str1` correspond au bit 0, `str2` au bit 1, etc... Les chaînes `NULL` dans les listes `str1` , `str2` , `...` sont ignorées :

```
mysql> SELECT MAKE_SET(1,'a','b','c');
       −> 'a'
mysql> SELECT MAKE_SET(1 | 4,'hello','nice','world');
       −> 'hello,world'
mysql> SELECT MAKE_SET(0,'a','b','c');
       −> ''
```

`MID(str,pos,len)`

`MID(str,pos,len)` est un synonyme de `SUBSTRING(str,pos,len)` .

`OCT(N)`

Retourne une chaîne de caractères représentant la valeur octal de l'argument `N` , où l'argument `N` est un nombre de type `BIGINT` . Cette fonction est un équivalent de `CONV(N,10,8)` . Retourne `NULL` si l'argument `N` est `NULL` :

```
mysql> SELECT OCT(12);
       −> '14'
```

`OCTET_LENGTH(str)`

`OCTET_LENGTH()` est un synonyme de `LENGTH()` .

`ORD(str)`

Si le premier caractère de la chaîne `str` est un caractère multi−octets, la fonction retourne le code de ce caractère, calculé à partir du code ASCII retourné par cette formule :

```
   (1st octet * 256)
+ (2nd octet * 256^2)
+ (3rd octet * 256^3) ...
```

Si le premier caractère n'est pas un caractère multi−octet, la fonction retournera la même valeur que la fonction `ASCII()` :

```
mysql> SELECT ORD('2');
       −> 50
```

`POSITION(substr IN str)`

`POSITION(substr IN str)` est un synonyme de `LOCATE(substr,str)` .

`QUOTE(str)`

Echappe les caractères d'une chaîne pour produire un résultat qui sera exploitable dans une requête SQL. Les caractères suivants seront précédés d'un anti−slash dans la chaîne retournée : le guillemet simple (`'''`), l'anti−slash (`'\'`), ASCII NUL, et le Contrôle−Z. Si l'argument vaut `NULL` , la valeur retournée sera le mot `` `NULL` '' sans les guillemets simples. La fonction `QUOTE` a été ajoutée en MySQL version 4.0.3.

```
mysql> SELECT QUOTE("Don't");
       −> 'Don\'t!'
mysql> SELECT QUOTE(NULL);
       −> NULL
```

`REPEAT(str,count)`

Retourne une chaîne de caractères constituée de la répétition de `count` fois la chaîne `str` . Si `count <= 0` , retourne une chaîne vide. Retourne `NULL` si `str` ou `count` sont `NULL` :

```
mysql> SELECT REPEAT('MySQL', 3);
       −> 'MySQLMySQLMySQL'
```

`REPLACE(str,from_str,to_str)`

Retourne une chaîne de caractères `str` dont toutes les occurrences de la chaîne `from_str` sont remplacées par la chaîne `to_str` :

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
       -> 'WwWwWw.mysql.com'
```

Cette fonction gère les caractères multi−octets.

`REVERSE(str)`

Retourne une chaîne dont l'ordre des caractères est l'inverse de la chaîne `str` :

```
mysql> SELECT REVERSE('abc');
       -> 'cba'
```

Cette fonction gère les caractères multi−octets.

`RIGHT(str,len)`

Retourne les `len` caractères les plus à droite de la chaîne de caractères `str` :

```
mysql> SELECT RIGHT('foobarbar', 4);
       -> 'rbar'
```

Cette fonction gère les caractères multi−octets.

`RPAD(str,len,padstr)`

Retourne la chaîne de caractères `str` , complétée à droite par la chaîne de caractères `padstr` jusqu'à ce que la chaîne de caractères `str` atteigne `len` caractères de long. Si la chaîne de caractères `str` est plus longue que `len'` caractères, elle sera raccourcie de `len` caractères.

```
mysql> SELECT RPAD('hi',5,'?');
       -> 'hi???'
```

`RTRIM(str)`

Retourne la chaîne de caractères `str` sans les espaces finaux :

```
mysql> SELECT RTRIM('barbar   ');
       -> 'barbar'
```

Cette fonction gère les caractères multi−octets.

`SOUNDEX(str)`

Retourne la valeur Soundex de la chaîne de caractères `str` . Deux chaînes qui ont des sonorités proches auront des valeurs soundex proches. Une chaîne Soundex standard possède 4 caractères, mais la fonction `SOUNDEX()` retourne une chaîne de longueur arbitraire. Vous pouvez utiliser la fonction `SUBSTRING()` sur ce résultat pour obtenir une chaîne Soundex standard. Tout caractère non alpha−numérique sera ignoré. Tous les caractères internationaux qui ne font pas partie de l'alphabet de base (A−Z) seront considérés comme des voyelles :

```
mysql> SELECT SOUNDEX('Hello');
       -> 'H400'
mysql> SELECT SOUNDEX('Quadratically');
       -> 'Q36324'
```

Note : cette fonction implémente l'algorithme soundex original, et non la version populaire améliorée (aussi décrite par D. Knuth). La différence est que la version originale supprime les voyelles, puis les doublons, alors que la version améliorée supprime les doublons d'abord, et ensuite, les voyelles.

`expr1 SOUNDS LIKE expr2`

Identique à `SOUNDEX(expr1)=SOUNDEX(expr2)` (disponible depuis la version 4.1).

`SPACE(N)`

Retourne une chaîne constituée de `N` espaces :

```
mysql> SELECT SPACE(6);
       -> '      '
```

`SUBSTRING(str,pos)`
`SUBSTRING(str FROM pos)`
`SUBSTRING(str,pos,len)`
`SUBSTRING(str FROM pos FOR len)`

Retourne une chaîne de `len` caractères de long de la chaîne `str` , à partir de la position `pos` . La syntaxe ANSI SQL92 utilise une variante de la fonction `FROM` :

```
mysql> SELECT SUBSTRING('Quadratically',5);
       -> 'ratically'
mysql> SELECT SUBSTRING('foobarbar' FROM 4);
       -> 'barbar'
mysql> SELECT SUBSTRING('Quadratically',5,6);
       -> 'ratica'
```

Cette fonction gère les caractères multi−octets.

`SUBSTRING_INDEX(str,delim,count)`

Retourne une portion de la chaîne de caractères `str` , située avant `count` occurrences du délimiteur `delim` . Si l'argument `count` est positif, tout ce qui précède le délimiteur final sera retourné. Si l'argument `count` est négatif, tout ce qui suit le délimiteur final sera retourné :

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
       -> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
       -> 'mysql.com'
```

Cette fonction gère les caractères multi−octets.

`TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)`

Retourne la chaîne de caractères `str` dont tous les préfixes et/ou suffixes `remstr` ont été supprimés. Si aucun des spécificateurs `BOTH` , `LEADING` ou `TRAILING` sont fournis, `BOTH` est utilisé comme valeur par défaut. Si `remstr` n'est pas spécifié, les espaces sont supprimés :

```
mysql> SELECT TRIM('  bar   ');
       -> 'bar'
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
       -> 'barxxx'
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
       -> 'bar'
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
       -> 'barx'
```

Cette fonction gère les caractères multi−octets.

`UCASE(str)`

`UCASE()` est un synonyme de `UPPER()` .

`UNCOMPRESS(string_to_uncompress)`

Décompresse une chaîne compressée avec `COMPRESS()` . Si l'argument n'est pas une valeur compressée, le résultat est `NULL` . Cette fonction requiert la bibliothèque `zlib` . Sinon, la valeur retournée est toujours `NULL` .

```
mysql> SELECT UNCOMPRESS(COMPRESS('any string'));
       -> 'any string'
mysql> SELECT UNCOMPRESS('any string');
       -> NULL
```

`UNCOMPRESS()` a été ajoutée en MySQL 4.1.1.

`UNCOMPRESSED_LENGTH(compressed_string)`

Retourne la taille de la chaîne avant compression.

```
mysql> SELECT UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30)));
       -> 30
```

`UNCOMPRESSED_LENGTH()` a été ajoutée en MySQL 4.1.1.

`UNHEX(str)`

Le contraire de `HEX(string)` . C'est à dire, chaque pair de chiffres hexadécimaux sont interprétées comme des nombres, et sont convertis en un caractère représenté par le nombre. Le résultat est retournée sous forme de chaîne binaire.

```
mysql> SELECT UNHEX('4D7953514C');
       -> 'MySQL'
mysql> SELECT 0x4D7953514C;
       -> 'MySQL'
mysql> SELECT UNHEX(HEX('string'));
       -> 'string'
mysql> SELECT HEX(UNHEX('1267'));
       -> '1267'
```

`UNHEX()` a été ajoutée en MySQL 4.1.2.

`UPPER(str)`

Retourne la chaîne `str` en majuscules, en fonction du jeu de caractères courant. Par défaut, c'est le jeu `ISO-8859-1 Latin1` :

```
mysql> SELECT UPPER('Hey');
       -> 'HEY'
```

Cette fonction gère les caractères multi−octets.

13.3.1 Opérateurs de comparaison pour les chaînes de caractères

MySQL convertit automatiquement les nombres en chaînes et et vice−versa :

```
mysql> SELECT 1+'1';
       -> 2
mysql> SELECT CONCAT(2,' test');
       -> '2 test'
```

Si vous devez convertir explicitement un nombre en chaîne, passez−le en argument de la fonction `CONCAT()` .

```
mysql> SELECT 38.8, CAST(38.8 AS CHAR);
       -> 38.8, '38.8'
```

```
mysql> SELECT 38.8, CONCAT(38.8);
        -> 38.8, '38.8'
```

`CAST()` est recommandée, mais elle a été ajoutée en MySQL 4.0.2.Si une fonction de chaîne de caractères est donnée comme chaîne binaire dans un argument d'une autre fonction, le résultat sera aussi une chaîne binaire. Les nombres convertis en chaînes sont traités comme des chaînes binaires. Cela affecte les comparaisons.

Normalement, si l'une des expressions dans une comparaison de chaîne est sensible à la casse, la comparaison est exécutée en tenant compte de la casse.

`expr LIKE pat [ESCAPE 'escape-char']`

La réalisation d'expression utilisant les expressions régulières simples de comparaison de SQL. Retourne `1` (TRUE) ou `0` (FALSE). Avec `LIKE` , vous pouvez utiliser les deux jokers suivants :

Char	Description
%	Remplace n'importe quel nombre de caractères, y compris aucun
_	Remplace exactement un caractère

```
mysql> SELECT 'David!' LIKE 'David_';
        -> 1
mysql> SELECT 'David!' LIKE '%D%v%';
        -> 1
```

Pour tester la présence littérale d'un joker, précédez−le d'un caractère d'échappement. Si vous ne spécifiez pas le caractère d'échappement `ESCAPE` , le caractère `'\'` sera utilisé :

String	Description
\%	Remplace le caractère littéral '%'
_	Remplace le caractère littéral '_'

```
mysql> SELECT 'David!' LIKE 'David\_';
        -> 0
mysql> SELECT 'David_' LIKE 'David\_';
        -> 1
```

Pour spécifier un caractère d'échappement différent, utilisez la clause `ESCAPE` :

```
mysql> SELECT 'David_' LIKE 'David|_' ESCAPE '|';
        -> 1
```

Les deux exemples suivants illustrent le fait que les comparaisons de chaînes de caractères ne sont pas sensibles à la casse à moins qu'une des opérandes soit une chaîne binaire.

```
mysql> SELECT 'abc' LIKE 'ABC';
        -> 1
mysql> SELECT 'abc' LIKE BINARY 'ABC';
        -> 0
```

`LIKE` est également autorisé pour les expressions numériques. (C'est une extension MySQL à la norme ANSI SQL `LIKE` .)

```
mysql> SELECT 10 LIKE '1%';
        -> 1
```

Note : Comme MySQL utilise la syntaxe d'échappement de caractères du langage C dans les chaînes (par exemple, `'\n'`), vous devez doubler tous les slash `'\'` que vous utilisez dans les expressions `LIKE` . Par exemple, pour rechercher les nouvelles lignes (`'\n'`), vous devez le spécifier comme cela : `'\\n'` . Pour rechercher un anti−slash (`'\'`), vous devez le spécifier comme cela : `'\\\\'` (les anti−slash sont supprimés une première fois pas l'analyseur syntaxique, puis une deuxième fois par le moteur d'expression régulières, ce qui ne laisse qu'un seul anti−slash à la fin).

Note : actuellement, `LIKE` n'est pas compatible avec les caractères multi−octets. La comparaison est faîte caractère par caractère.

`expr NOT LIKE pat [ESCAPE 'escape-char']`

Equivalent à `NOT (expr LIKE pat [ESCAPE 'escape-char'])` .

`expr NOT REGEXP pat`
`expr NOT RLIKE pat`

Equivalent à `NOT (expr REGEXP pat)` .

`expr REGEXP pat`
`expr RLIKE pat`

Effectue une recherche de chaîne avec l'expression régulière `pat` . Le masque peut être une expression régulière étendue. Voir la section Expressions régulières de MySQL . Retourne `1` si `expr` correspond au masque `pat` , sinon, retourne `0` . `RLIKE` est un synonyme de `REGEXP` , fourni pour assurer la compatibilité avec `mSQL` . Note : Comme MySQL utilise la syntaxe d'échappement de caractères du langage C dans les chaînes (par exemple, `'\n'`), vous devez doubler tous les anti−slash `'\'` que vous utilisez dans les expressions `REGEXP` . A partir de la version 3.23.4 de MySQL, `REGEXP` est insensible à la casse pour les comparaisons de chaînes normales (non binaires) :

```
mysql> SELECT 'Monty!' REGEXP 'm%y%%';
        -> 0
mysql> SELECT 'Monty!' REGEXP '.*';
        -> 1
mysql> SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
        -> 1
mysql> SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';
        -> 1  0
mysql> SELECT 'a' REGEXP '^[a-d]';
        -> 1
```

`STRCMP(expr1,expr2)`

`STRCMP()` retourne `0` si les chaînes sont identiques, `-1` si la première chaîne est plus petite que la seconde et `1` dans les autres cas :

```
mysql> SELECT STRCMP('text', 'text2');
        -> -1
mysql> SELECT STRCMP('text2', 'text');
        -> 1
mysql> SELECT STRCMP('text', 'text');
        -> 0
```

Depuis MySQL 4.0, `STRCMP()` utilise le jeu de caractères courant pour effectuer des comparaisons. Cela fait que le comportement par défaut est la comparaison insensible à la casse, à moins que l'un des deux opérandes soient une chaîne binaire. Avant MySQL 4.0, `STRCMP()` était sensible à la casse.

13.4 Fonctions numériques

13.4.1 Opérations arithmétiques

Les opérateurs arithmétiques usuels sont disponibles. Notez que dans le cas de `'-'` , `'+'` et `'*'` , le résultat est calculé avec en `BIGINT` avec une précision de 64 bits si les deux arguments sont des entiers ! Si l'un des arguments est un entier non signé, et que l'autre argument est aussi un entier, le résultat sera un entier non signé. Fonctions de transtypage .

`+`

Addition :

```
mysql> SELECT 3+5;
        -> 8
```

`−`

Soustraction :

```
mysql> SELECT 3-5;
        -> -2
```

`−`

Moins unaire. Change le signe de l'argument.

```
mysql> SELECT - 2;
        -> -2
```

Notez que si cet opérateur est utilisé avec un `BIGINT` , la valeur retournée est un `BIGINT` ! Cela signifie que vous devez éviter d'utiliser `-` sur des entiers qui peuvent avoir une valeur de `-2^63` !

`*`

Multiplication :

```
mysql> SELECT 3*5;
        -> 15
mysql> SELECT 18014398509481984*18014398509481984.0;
        -> 324518553658426726783156020576256.0
mysql> SELECT 18014398509481984*18014398509481984;
        -> 0
```

Le résultat du dernier calcul est incorrect car le résultat de la multiplication des deux entiers à dépassé la capacité de calcul de `BIGINT` (64 bits).

`/`

Division :

```
mysql> SELECT 3/5;
        -> 0.60
```

La division par zéro produit un résultat `NULL` :

```
mysql> SELECT 102/(1-1);
        -> NULL
```

Une division sera calculée en `BIGINT` seulement si elle est effectuée dans un contexte où le résultat est transformé en entier.

`DIV`

Division entière. Similaire à `FLOOR()` mais compatible avec les valeurs `BIGINT` .

```
mysql> SELECT 5 DIV 2;
        -> 2
```

`DIV` a été ajouté en MySQL 4.1.0.

13.4.2 Fonctions mathématiques

Toutes les fonctions mathématiques retournent `NULL` en cas d'erreur.

`ABS(X)`

Retourne la valeur absolue de `X` .

```
mysql> SELECT ABS(2);
        -> 2
mysql> SELECT ABS(-32);
        -> 32
```

Cette fonction est utilisable avec les valeurs issues des champs `BIGINT` .

`ACOS(X)`

Retourne l'arccosinus de `X` , c'est à dire, la valeur de l'angle dont `X` est la cosinus. Retourne `NULL` si `X` n'est pas dans l'intervalle `-1` − `1` .

```
mysql> SELECT ACOS(1);
        -> 0.000000
mysql> SELECT ACOS(1.0001);
        -> NULL
mysql> SELECT ACOS(0);
        -> 1.570796
```

`ASIN(X)`

Retourne l'arcsinus de `X` , c'est à dire, la valeur de l'angle dont le sinus est `X` . Retourne `NULL` si `X` n'est pas dans l'intervalle `-1` − `1` :

```
mysql> SELECT ASIN(0.2);
        -> 0.201358
mysql> SELECT ASIN('foo');
        -> 0.000000
```

`ATAN(X)`

Retourne l'arctangente de `X` , c'est à dire, la valeur de l'angle dont la tangente est `X` .

```
mysql> SELECT ATAN(2);
        -> 1.107149
mysql> SELECT ATAN(-2);
        -> -1.107149
```

`ATAN(Y,X)`
`ATAN2(Y,X)`

Retourne l'arctangente des variables `X` et `Y` . Cela revient à calculer l'arctangente de `Y / X` , excepté que les signes des deux arguments servent à déterminer le quadrant du résultat :

```
mysql> SELECT ATAN(-2,2);
        -> -0.785398
mysql> SELECT ATAN2(PI(),0);
        -> 1.570796
```

`CEILING(X)`
`CEIL(X)`

Retourne la valeur entière supérieure de `X` .

```
mysql> SELECT CEILING(1.23);
        -> 2
mysql> SELECT CEILING(-1.23);
        -> -1
```

Notez que la valeur retournée sera de type `BIGINT` !

`COS(X)`

Retourne le cosinus de `X` , où `X` est donné en radians.

```
mysql> SELECT COS(PI());
        -> -1.000000
```

`COT(X)`

Retourne la cotangente de `X` .

```
mysql> SELECT COT(12);
        -> -1.57267341
mysql> SELECT COT(0);
        -> NULL
```

`CRC32(expr)`

Calcule la somme de contrôle et retourne un entier 32 bits non−signé. Le résultat est la valeur `NULL` si l'argument est `NULL` . L'argument attendu est une chaîne, et sera traité comme une chaîne s'il n'est pas du bon type.

```
mysql> SELECT CRC32('MySQL');
        -> 3259397556
```

`CRC32()` est disponible en MySQL 4.1.0.

`DEGREES(X)`

Retourne l'argument `X` , convertit de radians en degrés.

```
mysql> SELECT DEGREES(PI());
        -> 180.000000
```

`EXP(X)`

Retourne la valeur de `e` (la base des logarithmes naturels) élevé à la puissance `X` .

```
mysql> SELECT EXP(2);
        -> 7.389056
mysql> SELECT EXP(-2);
        -> 0.135335
```

`FLOOR(X)`

Retourne la valeur entière inférieure de `X` .

```
mysql> SELECT FLOOR(1.23);
        -> 1
mysql> SELECT FLOOR(-1.23);
        -> -2
```

Notez que la valeur retournée sera de type `BIGINT` !

`LN(X)`

Retourne le logarithme naturel de `X` (népérien).

```
mysql> SELECT LN(2);
        -> 0.693147
mysql> SELECT LN(-2);
        -> NULL
```

Cette fonction a été ajoutée à MySQL à partir de la version 4.0.3. C'est un synonyme de la fonction `LOG(X)` .

`LOG(X)`
`LOG(B,X)`

Appelée avec un seul paramètre, cette fonction retourne le logarithme naturel (népérien) de `X` .

```
mysql> SELECT LOG(2);
        -> 0.693147
mysql> SELECT LOG(-2);
        -> NULL
```

Appelée avec deux paramètres, cette fonction retourne le logarithme naturel de `X` pour une base `B` arbitraire :

```
mysql> SELECT LOG(2,65536);
        -> 16.000000
mysql> SELECT LOG(1,100);
        -> NULL
```

Cette base arbitraire a été ajoutée à MySQL à partir de la version 4.0.3. `LOG(B,X)` est l'équivalent de `LOG(X)/LOG(B)` .

`LOG2(X)`

Retourne le logarithme en base 2 de `X` .

```
mysql> SELECT LOG2(65536);
        -> 16.000000
mysql> SELECT LOG2(-100);
        -> NULL
```

`LOG2()` est utile pour trouver combien de bits sont nécessaires pour stocker un nombre. Cette fonction a été ajoutée à MySQL à partir de la version 4.0.3. Dans les versions antérieures, vous pouvez utiliser `LOG(X)/LOG(2)` en remplacement.

`LOG10(X)`

Retourne le logarithme en base 10 de `X` .

```
mysql> SELECT LOG10(2);
        -> 0.301030
mysql> SELECT LOG10(100);
        -> 2.000000
mysql> SELECT LOG10(-100);
        -> NULL
```

`MOD(N,M)`
`N % M`
`N MOD M`

Modulo (équivalent de l'opérateur `%` dans le langage C). Retourne le reste de la division de `N` par `M` .

```
mysql> SELECT MOD(234, 10);
        -> 4
mysql> SELECT 253 % 7;
        -> 1
mysql> SELECT MOD(29,9);
        -> 2
```

Cette fonction ne pose pas de problèmes avec les `BIGINT` .

`PI()`

Retourne la valeur de pi. Par défaut, 5 décimales sont retournées, mais MySQL utilise la double précision pour pi.

```
mysql> SELECT PI();
        -> 3.141593
mysql> SELECT PI()+0.000000000000000000;
        -> 3.141592653589793116
```

`POW(X,Y)`
`POWER(X,Y)`

Retourne la valeur de `X` élevée à la puissance `Y` :

```
mysql> SELECT POW(2,2);
        -> 4.000000
mysql> SELECT POW(2,-2);
        -> 0.250000
```

`RADIANS(X)`

Retourne l'argument `X` , converti de degrés en radians.

```
mysql> SELECT RADIANS(90);
        -> 1.570796
```

`RAND()`
`RAND(N)`

Retourne un nombre aléatoire à virgule flottante compris dans l'intervalle `0` − `1.0` . Si l'argument entier `N` est spécifié, il est utilisé comme initialisation du générateur de nombres aléatoires.

```
mysql> SELECT RAND();
        -> 0.9233482386203
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND();
        -> 0.63553050033332
mysql> SELECT RAND();
        -> 0.70100469486881
```

Vous ne pouvez pas utiliser une colonne de valeur `RAND()` dans une clause `ORDER BY` , parce que `ORDER BY` va évaluer la colonne plusieurs fois. Dans la version 3.23 de MySQL, vous pouvez, tout de même, faire ceci :

```
mysql> SELECT * FROM tbl_name ORDER BY RAND();
```

Cette syntaxe est très pratique pour faire une sélection aléatoire de lignes :

```
mysql> SELECT * FROM table1, table2 WHERE a=b AND c<d
    -> ORDER BY RAND() LIMIT 1000;
```

Notez que la fonction `RAND()` dans une clause `WHERE` sera réévaluée à chaque fois que `WHERE` sera exécuté.

`RAND()` n'est pas un générateur parfait de nombres aléatoires, mais reste une manière rapide de produire des nombres aléatoires portables selon les différentes plates−formes pour une même version de MySQL.

`ROUND(X)`
`ROUND(X,D)`

Retourne l'argument `X` , arrondi à un nombre à `D` décimales. Avec deux arguments, la valeur est arrondie avec `D` décimales.

Si `D` vaut `0` , le résultat n'aura ni de partie décimale, ni de séparateur de décimal.

```
mysql> SELECT ROUND(-1.23);
        -> -1
mysql> SELECT ROUND(-1.58);
        -> -2
mysql> SELECT ROUND(1.58);
        -> 2
mysql> SELECT ROUND(1.298, 1);
        -> 1.3
mysql> SELECT ROUND(1.298, 0);
        -> 1
mysql> SELECT ROUND(23.298, -1);
```

```
        -> 20
```

Notez que le comportement de l'opérateur `ROUND()` , lorsque l'argument est exactement entre deux entiers, dépend de la bibliothèque C active. Certaines arrondissent toujours à l'entier pair le plus proche, toujours vers le haut, toujours vers le bas, ou toujours vers zéro. Si vous avez besoin d'un certain type d'arrondissement, vous devez utiliser une fonction bien définie comme `TRUNCATE()` ou `FLOOR()` .

`SIGN(X)`

Retourne le signe de l'argument sous la forme `-1` , `0` , ou `1` , selon que `X` est négatif, zéro, ou positif.

```
mysql> SELECT SIGN(-32);
        -> -1
mysql> SELECT SIGN(0);
        -> 0
mysql> SELECT SIGN(234);
        -> 1
```

`SIN(X)`

Retourne le sinus de `X` , où `X` est donné en radians.

```
mysql> SELECT SIN(PI());
        -> 0.000000
```

`SQRT(X)`

Retourne la racine carrée de `X` .

```
mysql> SELECT SQRT(4);
        -> 2.000000
mysql> SELECT SQRT(20);
        -> 4.472136
```

`TAN(X)`

Retourne la tangente de `X` , où `X` est donné en radians.

```
mysql> SELECT TAN(PI()+1);
        -> 1.557408
```

`TRUNCATE(X,D)`

Retourne l'argument `X` , tronqué à `D` décimales. Si `D` vaut `0` , le résultat n'aura ni séparateur décimal, ni partie décimale.

```
mysql> SELECT TRUNCATE(1.223,1);
        -> 1.2
mysql> SELECT TRUNCATE(1.999,1);
        -> 1.9
mysql> SELECT TRUNCATE(1.999,0);
        -> 1
mysql> SELECT TRUNCATE(-1.999,1);
        -> -1.9
```

A partir de MySQL 3.23.51 tous les nombres sont arrondis vers zéro.

Notez que les nombres décimaux ne sont pas stockés exactement comme les nombres entiers , mais comme des valeurs doubles. Vous pouvez être dupés par le résultat suivant :

```
mysql> SELECT TRUNCATE(10.28*100,0);
       -> 1027
```

Ce résultat est normal car 10.28 est actuellement stocké comme cela 10.2799999999999999.

13.5 Fonctions de dates et d'heures

Cette section décrit les fonctions qui peuvent être utilisées pour manipuler les valeurs temporelles. Voir Types temporels pour une description détaillée des intervalles de validité de chaque type, ainsi que les formats valides de spécifications des dates et heures.

Voici un exemple d'utilisation des fonctions de date. La requête suivante sélectionne toutes les lignes dont la colonne `date_col` représente une date de moins de 30 jours :

```
mysql> SELECT quelquechose FROM nom_de_table
          WHERE TO_DAYS(NOW()) - TO_DAYS(date_col) <= 30;
```

Notez que cette requête va aussi sélectionner des lignes dont les dates sont dans le futur.Les fonctions qui utilisent des valeurs de date acceptent les valeurs de type `DATETIME` et ignore la partie horaire. Les fonctions qui attendent des heures acceptent les valeurs littérales et ignorent la partie de date.

Les fonctions qui retourne la date ou l'heure courante sont évaluées une fois par requête, tout au début. Cela signifie que des références multiples à la fonction `NOW()` dans une même requête produiront toujours le même résultat. Ce principe s'applique aussi à `CURDATE()` , `CURTIME()` , `UTC_DATE()` , `UTC_TIME()` , `UTC_TIMESTAMP()` , et leurs synonymes.

Les intervalles de valeurs de retour des fonctions suivantes s'appliquent aux dates complètes. Si une date est une valeur ``zéro'' ou une date incomplète, comme `'2001-11-00'` , les fonctions qui extraient une partie d'une date retourneront `0` . Par exemple, `DAYOFMONTH('2001-11-00')` retourne `0` .

`ADDDATE(date,INTERVAL expr type)`
`ADDDATE(expr,days)`

Lorsqu'elle est utilisée avec la forme `INTERVAL` , `ADDDATE()` est un synonyme de `DATE_ADD()` . La fonction complémentaire `SUBDATE()` est un synonyme `DATE_SUB()` .

```
mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY);
       -> '1998-02-02'
mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY);
       -> '1998-02-02'
```

Depuis MySQL 4.1.1, la seconde syntaxe est utilisée si `expr` est une expression de type `DATE` ou `DATETIME` , et que `days` est un nombre de jour à ajouter à `expr` .

```
mysql> SELECT ADDDATE('1998-01-02', 31);
       -> '1998-02-02'
```

`ADDTIME(expr,expr2)`

`ADDTIME()` ajoute `expr2` à `expr` et retourne le résultat. `expr` est une expression de type `DATE` ou `DATETIME` , et `expr2` est une expression de type `TIME` .

```
mysql> SELECT ADDTIME("1997-12-31 23:59:59.999999", "1 1:1:1.000002");
       -> '1998-01-02 01:01:01.000001'
mysql> SELECT ADDTIME("01:00:00.999999", "02:00:00.999998");
       -> '03:00:01.999997'
```

`ADDTIME()` a été ajouté en MySQL 4.1.1.

`CURDATE()`
`CURRENT_DATE`

Retourne la date courante au format `'YYYY-MM-DD'` ou `YYYYMMDD` , suivant le contexte numérique ou chaîne :

```
mysql> SELECT CURDATE();
       -> '1997-12-15'
mysql> SELECT CURDATE() + 0;
       -> 19971215
```

`CURRENT_DATE`
`CURRENT_DATE()`

`CURRENT_DATE` et `CURRENT_DATE()` sont synonymes de `CURDATE()` .

`CURTIME()`

Retourne l'heure courante au format `'HH:MM:SS'` or `HHMMSS` suivant le contexte numérique ou chaîne :

```
mysql> SELECT CURTIME();
       -> '23:50:26'
mysql> SELECT CURTIME() + 0;
       -> 235026
```

`CURRENT_TIME`
`CURRENT_TIME()`

`CURRENT_TIME` et `CURRENT_TIME()` sont synonymes de `CURTIME()` .

`CURRENT_TIMESTAMP`
`CURRENT_TIMESTAMP()`

`CURRENT_TIMESTAMP` et `CURRENT_TIMESTAMP()` sont synonymes de `NOW()` .

`DATE(expr)`

Extrait la partie date de l'expression `expr` de type `DATE` ou `DATETIME` .

```
mysql> SELECT DATE('2003-12-31 01:02:03');
       -> '2003-12-31'
```

`DATE()` est disponible depuis MySQL 4.1.1.

`DATEDIFF(expr,expr2)`

`DATEDIFF()` retourne le nombre de jours entre la date de début `expr` et la date de fin `expr2` . `expr` et `expr2` sont des expressions de type `DATE` ou `DATETIME` . Seule la partie `DATE` est utilisée dans le calcul.

```
mysql> SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');
        -> 1
mysql> SELECT DATEDIFF('1997-11-31 23:59:59','1997-12-31');
        -> -30
```

`DATEDIFF()` est disponible depuis MySQL 4.1.1.

`DATE_ADD(date,INTERVAL expr type)`
`DATE_SUB(date,INTERVAL expr type)`

Ces fonctions effectuent des calculs arithmétiques sur les dates.Depuis MySQL 3.23, `INTERVAL expr type` est autorisé des deux cotés de l'opérateur `+` si l'expression de l'autre coté est de type `DATE` ou `DATETIME` . Pour l'opérateur `-` , `INTERVAL expr type` est autorisé uniquement du coté droit, car on ne peut pas soustraire une date d'un intervalle (voir les exemples ci−dessous).

`date` est une valeur de type `DATETIME` ou `DATE` qui spécifie la date de début. `expr` est une expression qui spécifie une valeur d'intervalle à ajouter ou soustraire de la date initiale. `expr` est une chaîne : elle peut commencer avec `'-'` pour les intervalles négatifs. `type` est un mot−clé, indiquant comment l'expression doit être interprétée.

La table suivante indique la signification des arguments `type` et `expr` :

`type` **Valeur**	**Attendue** `expr` **Format**
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS.MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

Les valeurs de `type` `DAY_MICROSECOND` , `HOUR_MICROSECOND` , `MINUTE_MICROSECOND` , `SECOND_MICROSECOND` , et `MICROSECOND` ont été ajoutés en MySQL 4.1.1. Les valeurs `QUARTER` et `WEEK` sont disponibles depuis MySQL 5.0.0.

MySQL autorise tous les signes de ponctuation, comme délimiteur dans le format de `expr` . Ceux qui sont affichés dans la table sont des suggestions. Si l'argument `date` est une valeur `DATE` et que vos calculs impliquent des parties `YEAR` , `MONTH` et `DAY` (c'est à dire, sans partie horaire), le résultat sera de type `DATE` . Sinon, le résultat est de type `DATETIME` :

```
mysql> SELECT '1997-12-31 23:59:59' + INTERVAL 1 SECOND;
        -> '1998-01-01 00:00:00'
mysql> SELECT INTERVAL 1 DAY + '1997-12-31';
        -> '1998-01-01'
mysql> SELECT '1998-01-01' - INTERVAL 1 SECOND;
        -> '1997-12-31 23:59:59'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
    ->                 INTERVAL 1 SECOND);
        -> '1998-01-01 00:00:00'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
    ->                 INTERVAL 1 DAY);
        -> '1998-01-01 23:59:59'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
    ->                 INTERVAL '1:1' MINUTE_SECOND);
        -> '1998-01-01 00:01:00'
mysql> SELECT DATE_SUB('1998-01-01 00:00:00',
    ->                 INTERVAL '1 1:1:1' DAY_SECOND);
```

```
        -> '1997-12-30 22:58:59'
mysql> SELECT DATE_ADD('1998-01-01 00:00:00',
    ->                 INTERVAL '-1 10' DAY_HOUR);
        -> '1997-12-30 14:00:00'
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
        -> '1997-12-02'
mysql> SELECT DATE_ADD('1992-12-31 23:59:59.000002',
    ->                 INTERVAL '1.999999' SECOND_MICROSECOND);
        -> '1993-01-01 00:00:01.000001'
```

Si vous spécifiez un intervalle qui est trop court (il n'inclut pas toutes les parties d'intervalle attendues par `type`), MySQL suppose que vous avez omis les valeurs de gauche. Par exemple, si vous spécifiez un type `type` de `DAY_SECOND` , la valeur `expr` devrait contenir des jours, heures, minutes et secondes. Si vous fournissez une valeur de la forme `'1:10'` , MySQL suppose que les jours et heures manquent, et que la valeur représente des minutes et secondes. En d'autres termes, `'1:10' DAY_SECOND` est interprété comme `'1:10' MINUTE_SECOND` . C'est similaire au comportement de MySQL avec les valeurs de type `TIME` , qui représente des durées plutôt que des horaires.

Notez que si vous ajoutez ou soustrayez à une valeur de type `DATE` des horaires, le résultat sera automatiquement au format `DATETIME` :

```
 mysql> SELECT DATE_ADD('1999-01-01', INTERVAL 1 DAY);
        -> '1999-01-02'
mysql> SELECT DATE_ADD('1999-01-01', INTERVAL 1 HOUR);
        -> '1999-01-01 01:00:00'
```

Si vous utilisez des dates malformées, le résultat sera `NULL` . Si vous ajoutez des `MONTH` , `YEAR_MONTH` ou `YEAR` , et que le résultat a un jour du mois qui est au−delà de ce qui est possible dans le mois, le jour sera adapté au plus grand jour possible du mois. Par exemple :

```
 mysql> SELECT DATE_ADD('1998-01-30', interval 1 month);
        -> '1998-02-28'
```

Notez que dans l'exemple précédent, le mot clé `INTERVAL` et le spécificateur `type` sont insensibles à la casse.

`DATE_FORMAT(date,format)`

Formate la date `date` avec le format `format` . Les spécificateurs suivants peuvent être utilisé dans la chaîne `format` :

Option	Description
%%	Un signe pourcentage littéral '%' .
%a	Nom du jour de la semaine, en abrégé et en anglais (Sun .. Sat)
%b	Nom du mois, en abrégé et en anglais (Jan .. Dec)
%c	Mois, au format numérique (1 .. 12)
%d	Jour du mois, au format numérique (00 .. 31)
%D	Jour du mois, avec un suffixe anglais (1st , 2nd , 3rd , etc.)
%e	Jour du mois, au format numérique (0 .. 31)
%f	Microsecondes (000000 .. 999999)
%H	Heure (00 .. 23)
%h	Heure (01 .. 12)
%I	Heure (01 .. 12)
%i	Minutes, au format numérique (00 .. 59)
%j	Jour de l'année (001 .. 366)
%k	Heure (0 .. 23)
%l	Heure (1 .. 12)
%m	Mois, au format numérique (01 .. 12)
%M	Nom du mois (January .. December)
%p	AM ou PM
%r	Heures, au format 12 heures (hh:mm:ss [AP]M)
%s	Secondes (00 .. 59)
%S	Secondes (00 .. 59)
%T	Heures, au format 24 heures (hh:mm:ss)
%U	Numéro de la semaine (00 .. 53), où Dimanche est le premier jour de la semaine
%u	Numéro de la semaine (00 .. 53), où Lundi est le premier jour de la semaine
%V	Numéro de la semaine (01 .. 53), où Dimanche est le premier jour de la semaine, utilisé avec '%X'
%v	Numéro de la semaine (01 .. 53), où Lundi est le premier jour de la semaine, utilisé avec '%x'
%W	Nom du jour de la semaine (Sunday .. Saturday)

%w	Numéro du jour de la semaine (0 =Sunday.. 6 =Saturday)
%X	Année, pour les semaines qui commencent le Dimanche, au format numérique, sur 4 chiffres, utilisé avec '%V'
%x	Année, pour les semaines qui commencent le Lundi, au format numérique, sur 4 chiffres, utilisé avec '%v'
%y	Année, au format numérique, sur 2 chiffres
%Y	Année, au format numérique, sur 4 chiffres

Tous les autres caractères sont simplement copiés dans le résultat sans interprétation:

Le format `%f` est disponible depuis MySQL 4.1.1.

Depuis MySQL version 3.23, le caractère `'%'` est requis devant les caractères de format. Dans les versions antérieures de MySQL, `'%'` était optionnel.

La raison qui fait que les intervalles de mois et de jours commencent avec zéro est que MySQL autorise les dates incomplètes comme `'2004-00-00'` , depuis MySQL 3.23.

```
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');
        -> 'Saturday October 1997'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
        -> '22:23:00'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%D %y %a %d %m %b %j');
        -> '4th 97 Sat 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%H %k %I %r %T %S %w');
        -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
        -> '1998 52'
```

`DAY(date)`

`DAY()` est un synonyme de `DAYOFMONTH()` . Cette fonction est disponible depuis MySQL 4.1.1.

`DAYNAME(date)`

Retourne le nom du jour de la semaine de `date` :

```
mysql> SELECT DAYNAME('1998-02-05');
        -> 'Thursday'
```

`DAYOFMONTH(date)`

Retourne le jour de la date `date` , dans un intervalle de `1` à `31` :

```
mysql> SELECT DAYOFMONTH('1998-02-03');
        -> 3
```

`DAYOFWEEK(date)`

Retourne l'index du jour de la semaine : pour `date` (`1` = Dimanche, `2` = Lundi, ... `7` = Samedi). Ces index correspondent au standard ODBC :

```
mysql> SELECT DAYOFWEEK('1998-02-03');
        -> 3
```

`DAYOFYEAR(date)`

Retourne le jour de la date `date` , dans un intervalle de `1` à `366` :

```
mysql> SELECT DAYOFYEAR('1998-02-03');
        -> 34
```

`EXTRACT(type FROM date)`

La fonction `EXTRACT()` utilise les mêmes types d'intervalles que la fonction `DATE_ADD()` ou la fonction `DATE_SUB()` , mais extrait des parties de date plutôt que des opérations de date.

```
mysql> SELECT EXTRACT(YEAR FROM "1999-07-02");
       -> 1999
mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03");
       -> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
       -> 20102
mysql> SELECT EXTRACT(MICROSECOND FROM "2003-01-02 10:30:00.00123");
        -> 123
```

`FROM_DAYS(N)`

Retourne la date correspondant au nombre de jours (`N`) depuis la date 0 :

```
mysql> SELECT FROM_DAYS(729669);
        -> '1997-10-07'
```

`FROM_DAYS()` n'est pas fait pour travailler avec des dates qui précèdent l'avènement du calendrier Grégorien (1582), car elle ne prend pas en compte les jours perdus lors du changement de calendrier.

`FROM_UNIXTIME(unix_timestamp)`

Retourne une représentation de l'argument `unix_timestamp` sous la forme `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS` , suivant si la fonction est utilisé dans un contexte numérique ou de chaîne.

```
mysql> SELECT FROM_UNIXTIME(875996580);
        -> '1997-10-04 22:23:00'
mysql> SELECT FROM_UNIXTIME(875996580) + 0;
        -> 19971004222300
```

Si `format` est donné, le résultat est formaté en fonction de la chaîne `format` . `format` peut contenir les mêmes options de format que celles utilisées par `DATE_FORMAT()` :

```
mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(),
    ->                      '%Y %D %M %h:%i:%s %x');
        -> '2003 6th August 06:22:58 2003'
```

`GET_FORMAT(DATE | TIME | TIMESTAMP, 'EUR' | 'USA' | 'JIS' | 'ISO' | 'INTERNAL')`

Retourne une chaîne de format. Cette fonction est pratique lorsqu'elle est utilisée avec les fonctions `DATE_FORMAT()` et `STR_TO_DATE()` . Les trois valeurs possibles pour le premier argument, et les cinq valeurs possible pour le second argument donnent 15 formats d'affichage (pour les options utilisées, voyez la table de la fonction `DATE_FORMAT()`) :

Appel fonction	**Résultat**
GET_FORMAT(DATE,'USA')	'%m.%d.%Y'
GET_FORMAT(DATE,'JIS')	'%Y-%m-%d'
GET_FORMAT(DATE,'ISO')	'%Y-%m-%d'
GET_FORMAT(DATE,'EUR')	'%d.%m.%Y'
GET_FORMAT(DATE,'INTERNAL')	'%Y%m%d'
GET_FORMAT(TIMESTAMP,'USA')	'%Y-%m-%d-%H.%i.%s'
GET_FORMAT(TIMESTAMP,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(TIMESTAMP,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(TIMESTAMP,'EUR')	'%Y-%m-%d-%H.%i.%s'
GET_FORMAT(TIMESTAMP,'INTERNAL')	'%Y%m%d%H%i%s'
GET_FORMAT(TIME,'USA')	'%h:%i:%s %p'
GET_FORMAT(TIME,'JIS')	'%H:%i:%s'
GET_FORMAT(TIME,'ISO')	'%H:%i:%s'
GET_FORMAT(TIME,'EUR')	'%H.%i.%S'
GET_FORMAT(TIME,'INTERNAL')	'%H%i%s'

Le format ISO est le format ISO 9075, et non ISO 8601.

```
mysql> SELECT DATE_FORMAT('2003-10-03', GET_FORMAT(DATE, 'EUR')
        -> '03.10.2003'
mysql> SELECT STR_TO_DATE('10.31.2003', GET_FORMAT(DATE, 'USA'))
        -> 2003-10-31
```

`GET_FORMAT()` est disponible depuis MySQL 4.1.1. Voyez Syntaxe de `SET` .

`HOUR(time)`

Retourne le nombre d'heures pour l'heure `time` , dans un intervalle de `0` à `23` :

```
mysql> SELECT HOUR('10:05:03');
        -> 10
```

Cependant, l'intervalle des valeurs `TIME` est bien plus grand, et donc, `HOUR` peut retourner des valeurs plus grandes que `23` :

```
mysql> SELECT HOUR('272:59:59');
        -> 272
```

`LAST_DAY(date)`

Prend une valeur de format `DATE` ou `DATETIME` , et retourne le dernier jour du mois correspondant. Retourne `NULL` si l'argument est invalide.

```
mysql> SELECT LAST_DAY('2003-02-05'), LAST_DAY('2004-02-05');
        -> '2003-02-28', '2004-02-29'
mysql> SELECT LAST_DAY('2004-01-01 01:01:01');
        -> '2004-01-31'
mysql> SELECT LAST_DAY('2003-03-32');
        -> NULL
```

`LAST_DAY()` est disponible depuis MySQL 4.1.1.

`LOCALTIME`
`LOCALTIME()`

`LOCALTIME` et `LOCALTIME()` sont synonymes de `NOW()` .

`LOCALTIMESTAMP`
`LOCALTIMESTAMP()`

`LOCALTIMESTAMP` et `LOCALTIMESTAMP()` sont synonymes de `NOW()` .

`MAKEDATE(year,dayofyear)`

Retourne une valeur de format `DATE` , à partir d'une année et du numéro de jour. `dayofyear` doit être plus grand que 0 ou le résultat sera `NULL` .

```
mysql> SELECT MAKEDATE(2001,31), MAKEDATE(2001,32);
        -> '2001-01-31', '2001-02-01'
mysql> SELECT MAKEDATE(2001,365), MAKEDATE(2004,365);
        -> '2001-12-31', '2004-12-30'
mysql> SELECT MAKEDATE(2001,0);
        -> NULL
```

`MAKEDATE()` est disponible depuis MySQL 4.1.1.

`MAKETIME(hour,minute,second)`

Retourne une valeur de format `TIME` , calculée à partir des arguments `hour` , `minute` et `second` .

```
mysql> SELECT MAKETIME(12,15,30);
       -> '12:15:30'
```

`MAKETIME()` est disponible depuis MySQL 4.1.1.

`MICROSECOND(expr)`

Retourne le nombre de microsecondes dans l'expression de type `TIME` ou `DATETIME expr` , sous la forme d'un nombre entre `0` et `999999` .

```
mysql> SELECT MICROSECOND('12:00:00.123456');
       -> 123456
mysql> SELECT MICROSECOND('1997-12-31 23:59:59.000010');
       -> 10
```

`MICROSECOND()` est disponible depuis MySQL 4.1.1.

est disponible depuis

`MINUTE(time)`

Retourne le nombre de minutes pour l'heure `time` , dans un intervalle de `0` à `59` :

```
mysql> SELECT MINUTE('98-02-03 10:05:03');
       -> 5
```

`MONTH(date)`

Retourne le numéro du mois de la date `date` , dans un intervalle de `1` à `12` :

```
mysql> SELECT MONTH('1998-02-03');
       -> 2
```

`MONTHNAME(date)`

Retourne le nom du mois de la date `date` :

```
mysql> SELECT MONTHNAME("1998-02-05");
       -> 'February'
```

`NOW()`

Retourne la date courante au format `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS` , suivant le contexte numérique ou chaîne :

```
mysql> SELECT NOW();
       -> '1997-12-15 23:50:26'
mysql> SELECT NOW() + 0;
       -> 19971215235026
```

`PERIOD_ADD(P,N)`

Ajoute `N` mois à la période `P` (au format `YYMM` ou `YYYYMM`). Retourne une valeur dans le format `YYYYMM` .

Notez que l'argument `P` **n'est pas** de type date :

```
mysql> SELECT PERIOD_ADD(9801,2);
       -> 199803
```

`PERIOD_DIFF(P1,P2)`

Retourne le nombre de mois entre les périodes `P1` et `P2` . `P1` et `P2` doivent être au format `YYMM` ou `YYYYMM` .Notez que les arguments `P1` et `P2` **ne sont pas** de type date :

```
mysql> SELECT PERIOD_DIFF(9802,199703);
       -> 11
```

`QUARTER(date)`

Retourne le numéro du trimestre de la date `date` , dans un intervalle de `1` à `4` :

```
mysql> SELECT QUARTER('98-04-01');
       -> 2
```

`SECOND(time)`

Retourne le nombre de secondes pour l'heure `time` , dans un intervalle de `0` à `59` :

```
mysql> SELECT SECOND('10:05:03');
       -> 3
```

`SEC_TO_TIME(seconds)`

Retourne l'argument `seconds` , convertit en heures, minutes et secondes au format `'HH:MM:SS'` ou `HHMMSS` , suivant le contexte numérique ou chaîne :

```
mysql> SELECT SEC_TO_TIME(2378);
       -> '00:39:38'
mysql> SELECT SEC_TO_TIME(2378) + 0;
       -> 3938
```

`STR_TO_DATE(str,format)`

Cette fonction est l'inverse de la fonction `DATE_FORMAT()` . Elle prend la chaîne `str` , et une chaîne de format `format` , puis retourne une valeur `DATETIME` . Les valeurs de type `DATE` , `TIME` ou `DATETIME` contenues dans la chaîne `str` doivent être au format `format` . Pour les options qui sont utilisables dans la chaîne `format` , voyez la table dans la description de la fonction `DATE_FORMAT()` . Tous les autres caractères sont utilisés littéralement, et ne seront pas interprétés. Si `str` contient une valeur illégale, `STR_TO_DATE()` retourne `NULL` .

```
mysql> SELECT STR_TO_DATE('03.10.2003 09.20', '%d.%m.%Y %H.%i')
       -> 2003-10-03 09:20:00
mysql> SELECT STR_TO_DATE('10rap', '%crap')
       -> 0000-10-00 00:00:00
mysql> SELECT STR_TO_DATE('2003-15-10 00:00:00', '%Y-%m-%d %H:%i:%s')
       -> NULL
```

`STR_TO_DATE()` est disponible depuis MySQL 4.1.1.

`SUBDATE(date,INTERVAL expr type)`
`SUBDATE(expr,days)`

Lorsqu'elle est utilisée avec la forme `INTERVAL` du second argument, `SUBDATE()` est synonyme `DATE_SUB()` .

```
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
       -> '1997-12-02'
mysql> SELECT SUBDATE('1998-01-02', INTERVAL 31 DAY);
       -> '1997-12-02'
```

Depuis MySQL 4.1.1, la seconde syntaxe est autorisée, où `expr` est une expression de type `DATE` ou `DATETIME` et `days` est le nombre de jour à soustraire de l'expression `expr` .

```
mysql> SELECT SUBDATE('1998-01-02 12:00:00', 31);
       -> '1997-12-02 12:00:00'
```

`SUBTIME(expr,expr2)`

`SUBTIME()` soustrait `expr2` de `expr` et retourne le résultat. `expr` est une expression de format `DATE` ou `DATETIME` et `expr2` est une expression de type `TIME` .

```
mysql> SELECT SUBTIME("1997-12-31 23:59:59.999999", "1 1:1:1.000002");
       -> '1997-12-30 22:58:58.999997'
mysql> SELECT SUBTIME("01:00:00.999999", "02:00:00.999998");
       -> '-00:59:59.999999'
```

`SUBTIME()` a été ajoutée en MySQL 4.1.1.

`SYSDATE()`

`SYSDATE()` est un synonyme de `NOW()` .

`TIME(expr)`

Extrait la partie horaire de l'expression `expr` , de type `TIME` ou `DATETIME` .

```
mysql> SELECT TIME('2003-12-31 01:02:03');
       -> '01:02:03'
mysql> SELECT TIME('2003-12-31 01:02:03.000123');
       -> '01:02:03.000123'
```

`TIME()` a été ajoutée en MySQL 4.1.1.

`TIMEDIFF(expr,expr2)`

`TIMEDIFF()` retourne la durée entre l'heure de début `expr` et l'heure de fin `expr2` . `expr` et `expr2` sont des expressions de type `TIME` ou `DATETIME` , et doivent être de même type.

```
mysql> SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');
       -> '-00:00:00.000001'
mysql> SELECT TIMEDIFF('1997-12-31 23:59:59.000001','1997-12-30 01:01:01.000002');
       -> '46:58:57.999999'
```

`TIMEDIFF()` a été ajoutée en MySQL 4.1.1.

`TIMESTAMP(expr)`
`TIMESTAMP(expr,expr2)`

Avec un seul argument, retourne l'expression `expr` de type `DATE` ou `DATETIME` sous la forme d'une valeur `DATETIME` . Avec deux arguments, ajouter l'expression `expr2` à l'expression `expr` et retourne le résultat au format `DATETIME` .

```
mysql> SELECT TIMESTAMP('2003-12-31');
       -> '2003-12-31 00:00:00'
mysql> SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00');
       -> '2004-01-01 00:00:00'
```

`TIMESTAMP()` a été ajoutée en MySQL 4.1.1.

`TIMESTAMPADD(interval,int_expr,datetime_expr)`

Ajoute l'expression entière `int_expr` à l'expression `datetime_expr` au format `DATE` ou `DATETIME` . L'unité de `int_expr` est donnée avec l'argument `interval` , qui peut être l'une des valeurs suivantes : `FRAC_SECOND` , `SECOND` , `MINUTE` , `HOUR` , `DAY` , `WEEK` , `MONTH` , `QUARTER` , ou `YEAR` .La valeur `interval` peut être spécifiée, en utilisant un des mots−clé cités, ou avec le préfixe `SQL_TSI_` . Par exemple, `DAY` et `SQL_TSI_DAY` sont tous les deux valides.

```
mysql> SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');
       -> '2003-01-02 00:01:00'
mysql> SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
       -> '2003-01-09'
```

`TIMESTAMPADD()` a été ajoutée en MySQL 5.0.0.

`TIMESTAMPDIFF(interval,datetime_expr1,datetime_expr2)`

Retourne la différence entière entre les expressions `datetime_expr1` et `datetime_expr2` , de format `DATE` et `DATETIME` . L'unité du résultat est donné par l'argument `interval` . Les valeurs légales de `interval` sont les mêmes que pour la fonction `TIMESTAMPADD()` .

```
mysql> SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');
        -> 3
mysql> SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');
        -> -1
```

`TIMESTAMPDIFF()` a été ajoutée en MySQL 5.0.0.

`TIME_FORMAT(time,format)`

Cette fonction est utilisée exactement comme la fonction `DATE_FORMAT()` ci−dessus, mais la chaîne `format` ne doit utiliser que des spécificateurs d'heures, qui gèrent les heures, minutes et secondes. Les autres spécificateurs génèreront la valeur `NULL` ou `0` .

Si la valeur `time` contient une valeur d'heure qui est plus grande que `23` , les formats `%H` et `%k` produiront une valeur qui est hors de l'intervalle `0..23` . L'autre format d'heure produira une heure modulo 12 :

```
mysql> SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');
        -> '100 100 04 04 4'
```

`TIME_TO_SEC(time)`

Retourne l'argument `time` , convertit en secondes :

```
mysql> SELECT TIME_TO_SEC('22:23:00');
        -> 80580
mysql> SELECT TIME_TO_SEC('00:39:38');
        -> 2378
```

`TO_DAYS(date)`

Retourne le nombre de jours depuis la date 0 jusqu'à la date `date` :

```
mysql> SELECT TO_DAYS(950501);
        -> 728779
mysql> SELECT TO_DAYS('1997-10-07');
        -> 729669
```

`TO_DAYS()` n'est pas fait pour travailler avec des dates qui précèdent l'avènement du calendrier Grégorien (1582), car elle ne prend pas en compte les jours perdus lors du changement de calendrier.

N'oubliez pas que MySQL convertit les années représentées sur deux chiffres en dates à quatre chiffres, en utilisant les règles de la section Types temporels . Par exemple, `'1997-10-07'` et `'97-10-07'` sont identiques :

```
mysql> SELECT TO_DAYS('1997-10-07'), TO_DAYS('97-10-07');
        -> 729669, 729669
```

Pour les dates antérieures à 1582, les résultats sont indéfinis.

`UNIX_TIMESTAMP()`
`UNIX_TIMESTAMP(date)`

Lorsqu'elle est appelé sans argument, cette fonction retourne un timestamp Unix (nombre de secondes depuis `'1970-01-01 00:00:00'` GMT). Si `UNIX_TIMESTAMP()` est appelé avec un argument `date` , elle retourne le timestamp correspondant à cette date. `date` peut être une chaîne de type `DATE` , `DATETIME` , `TIMESTAMP` , ou un nombre au format `YYMMDD` ou `YYYYMMDD` , en horaire local :

```
mysql> SELECT UNIX_TIMESTAMP();
        -> 882226357
mysql> SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00');
        -> 875996580
```

Lorsque `UNIX_TIMESTAMP` est utilisé sur une colonne de type `TIMESTAMP` , la fonction reçoit directement la valeur, sans conversion explicite. Si vous donnez à `UNIX_TIMESTAMP()` une date hors de son intervalle de validité, elle retourne 0.Si vous voulez soustraire une colonne de type `UNIX_TIMESTAMP()` , vous devez sûrement vouloir un résultat de type entier signé. Fonctions de transtypage .

`UTC_DATE`
`UTC_DATE()`

Retourne la date UTC courante au format `'YYYY-MM-DD'` ou `YYYYMMDD` suivant le contexte numérique ou chaîne :

```
mysql> SELECT UTC_DATE(), UTC_DATE() + 0;
        -> '2003-08-14', 20030814
```

`UTC_DATE()` est disponible depuis MySQL 4.1.1.

`UTC_TIME`
`UTC_TIME()`

Retourne l'heure UTC courante au format `'HH:MM:SS'` or `HHMMSS` suivant le contexte numérique ou chaîne :

```
mysql> SELECT UTC_TIME(), UTC_TIME() + 0;
       -> '18:07:53', 180753
```

`UTC_TIME()` est disponible depuis MySQL 4.1.1.

`UTC_TIMESTAMP`
`UTC_TIMESTAMP()`

Retourne l'heure et la date UTC courante au format `'YYYY-MM-DD HH:MM:SS'` or `YYYYMMDDHHMMSS` suivant le contexte numérique ou chaîne :

```
mysql> SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;
       -> '2003-08-14 18:08:04', 20030814180804
```

`UTC_TIMESTAMP()` est disponible depuis MySQL 4.1.1.

`WEEK(date [,mode])`

Avec un seul argument, retourne le numéro de la semaine dans l'année de la date `date` , dans un intervalle de `0` à `53` (oui, il peut y avoir un début de semaine numéro 53), en considérant que Dimanche est le premier jour de la semaine. Avec deux arguments, la fonction `WEEK()` vous permet de spécifier si les semaines commencent le Dimanche ou le Lundi et la valeur retournée sera dans l'intervalle `0-53` ou bien `1-52` . Lorsque l'argument `mode` est omis, la valeur de la variable `default_week_format` (ou 0 en MySQL 4.0 ou plus ancien) est utilisé. Syntaxe de `SET` .Voici un tableau explicatif sur le fonctionnement du second argument :

Valeur	Signification
0	La semaine commence le Sunday;l'intervalle de valeur de retour va de 0 à !2; la semaine 1 est la première semaine de l'année
1	La semaine commence le Monday;l'intervalle de valeur de retour va de 0 à !2; la semaine 1 est la première semaine de l'année qui a plus de trois jours
2	La semaine commence le Sunday;l'intervalle de valeur de retour va de 1 à !2; la semaine 1 est la première semaine de l'année
3	La semaine commence le Monday;l'intervalle de valeur de retour va de 1 à !2; la semaine 1 est la première semaine de l'année qui a plus de trois jours
4	La semaine commence le Sunday;l'intervalle de valeur de retour va de 0 à !2; la semaine 1 est la première semaine de l'année qui a plus de trois jours
5	La semaine commence le Monday;l'intervalle de valeur de retour va de 0 à !2; la semaine 1 est la première semaine de l'année
6	La semaine commence le Sunday;l'intervalle de valeur de retour va de 1 à !2; la semaine 1 est la première semaine de l'année qui a plus de trois jours
7	La semaine commence le Monday;l'intervalle de valeur de retour va de 1 à !2; la semaine 1 est la première semaine de l'année

Le `mode` 3 est disponible depuis MySQL 4.0.5. Le `mode` 4 est disponible depuis MySQL 4.0.17.

```
mysql> SELECT WEEK('1998-02-20');
       -> 7
mysql> SELECT WEEK('1998-02-20',0);
       -> 7
mysql> SELECT WEEK('1998-02-20',1);
       -> 8
mysql> SELECT WEEK('1998-12-31',1);
       -> 53
```

Note : en version 4.0, `WEEK(date,0)` a été modifiée pour correspondre au système calendaire des USA. Avant cela, `WEEK()` était calculé incorrectement, pour des dates américaines : en effet, `WEEK(date)` et `WEEK(date,0)` étaient incorrects.Si vous préférez que le résultat soit calculé en fonction de l'année qui contient le premier jour de la semaine de la date utilisée en argument, vous devriez utiliser les valeurs 2, 3, 6, or 7 de l'argument `mode` .

```
mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);
       -> 2000, 0
mysql> SELECT WEEK('2000-01-01',2);
       -> 52
```

Alternativement, utilisez la fonction `YEARWEEK()` :

```
mysql> SELECT YEARWEEK('2000-01-01');
       -> 199952
mysql> SELECT MID(YEARWEEK('2000-01-01'),5,2);
       -> '52'
```

`WEEKDAY(date)`

Retourne l'index du jour de la semaine, avec la conversion suivante : `date` (`0` = Lundi, `1` = Mardi, ... `6` = Dimanche).

```
mysql> SELECT WEEKDAY('1997-10-04 22:23:00');
       -> 5
mysql> SELECT WEEKDAY('1997-11-05');
       -> 2
```

`WEEKOFYEAR(date)`

Retourne le numéro de semaine dans l'année, sous forme d'un nombre compris entre `1` et `53` .

```
mysql> SELECT WEEKOFYEAR('1998-02-20');
       -> 8
```

`WEEKOFYEAR()` est disponible depuis MySQL 4.1.1.

`YEAR(date)`

Retourne l'année de la date `date` , dans un intervalle de `1000` à `9999` :

```
mysql> SELECT YEAR('98-02-03');
       -> 1998
mysql> SELECT YEAR('98-02-03');
       -> 1998
```

`YEARWEEK(date)`
`YEARWEEK(date,start)`

Retourne l'année et la semaine d'une date. L'argument `start` fonctionne exactement comme l'argument `start` de la fonction `WEEK()` . Notez que l'année dans le résultat peut être différente de l'année passée en argument, pour la première et la dernière semaine de l'année.

```
mysql> SELECT YEARWEEK('1987-01-01');
       -> 198653
```

Notez que le numéro de semaine est différent de celui que la fonction `WEEK()` retourne (`0`) pour les arguments optionnels `0` ou `1` , comme `WEEK()` puis retourne la semaine dans le contexte de l'année.

13.6 Recherche en texte intégral (Full−text) dans MySQL

`MATCH (col1,col2,...) AGAINST (expr [IN BOOLEAN MODE | WITH QUERY EXPANSION])`

Depuis la version 3.23.23, `MySQL` propose l'indexation et la recherche sur l'ensemble d'un champ `TEXT` (`full−text`). Les index en texte intégral de `MySQL` sont des index de type `FULLTEXT` . Les index `FULLTEXT` sont utilisés avec les tables `MyISAM` et peuvent être créés depuis des colonnes de types `CHAR` , `VARCHAR` , ou `TEXT` au moment de `CREATE TABLE` ou plus tard avec `ALTER TABLE` ou `CREATE INDEX` . Pour les enregistrements les plus grands, il sera plus rapide de charger les donnés dans une table qui n'a pas d'index `FULLTEXT` , et ensuite de créer l'index avec `ALTER TABLE` (ou `CREATE INDEX`). L'enregistrement de données dans une table qui a déjà des index `FULLTEXT` sera plus lent.

Les contraintes sur la recherche en texte intégral sont listées dans la section Restrictions sur les recherches en texte intégral .

La recherche en texte intégral est effectuée par la fonction `MATCH()` .

```
 mysql> CREATE TABLE articles (
    ->   id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   title VARCHAR(200),
    ->   body TEXT,
    ->   FULLTEXT (title,body)
    -> );
Query OK, 0 rows affected (0.00 sec)

 mysql> INSERT INTO articles VALUES
    -> (NULL,'MySQL Tutorial', 'DBMS stands for DataBase ...'),
    -> (NULL,'How To Use MySQL Efficiently', 'After you went through a ...'),
    -> (NULL,'Optimising MySQL','In this tutorial we will show ...'),
    -> (NULL,'1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    -> (NULL,'MySQL vs. YourSQL', 'In the following database comparison ...'),
    -> (NULL,'MySQL Security', 'When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

 mysql> SELECT * FROM articles
    ->          WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)
```

La fonction `MATCH()` effectue la recherche d'une chaîne de caractères dans une liste de textes (et dans un groupe d'une ou plusieurs colonnes utilisées pour l'index `FULLTEXT`). La chaîne recherchée est donnée en argument à `AGAINST()` . La recherche est sans distinguer les majuscules des minuscules. Pour chaque ligne de la table, `MATCH()` retourne une valeur de pertinence, qui est une mesure de la ressemblance entre le chaîne recherchée et le texte de la ligne dans le colonne donnée dans la liste de `MATCH()` .Quand `MATCH()` est utilisé comme condition de `WHERE` (voir l'exemple suivant) les lignes retournées sont automatiquement organisées avec la pertinence la plus élevé en premier. La pertinence est un nombre décimal positif. La pertinence de zéro signifie qu'il n'y a pas de similarité. La pertinence est calculé en fonction du nombre de mots dans la ligne, du nombre de mots uniques dans cette ligne, du nombre total de mots dans la liste, et du nombre de documents (lignes) qui contiennent un mot en particulier.

Pour les recherches en texte intégral et en langage naturel, la technique impose que les colonnes utilisées avec la fonction `MATCH()` doivent être les mêmes que les colonnes utilisées dans un index `FULLTEXT` . Dans la requête précédente, notez que les colonnes nommées dans la fonction `MATCH()` (`title` et `body`) sont les mêmes que celles de la définition de la table `article` et son index `FULLTEXT` . Si vous voulez rechercher le titre `title` ou le corps `body` séparément, vous devrez créer un index `FULLTEXT` pour chaque colonne.

Il est aussi possible d'exécuter une recherche en mode booléen. Ceci est décrit dans les sections Opérateurs booléens de recherche en texte intégral et Recherches en texte intégral avec extension .

L'exemple précédent est une illustration élémentaire qui montre comment on utilise la fonction `MATCH()` . Les lignes sont retournées par ordre décroissant de pertinence. L'exemple suivant montre comment récupérer la valeur de pertinence explicitement. Comme il n'y a pas de condition `WHERE` ni de condition `ORDER BY` les lignes retournées ne sont pas ordonnées.

```
 mysql> SELECT id,MATCH (title,body) AGAINST ('Tutorial') FROM articles;
+----+-----------------------------------------+
| id | MATCH (title,body) AGAINST ('Tutorial') |
+----+-----------------------------------------+
|  1 |                        0.64840710366884 |
|  2 |                                       0 |
|  3 |                        0.66266459031789 |
|  4 |                                       0 |
|  5 |                                       0 |
|  6 |                                       0 |
+----+-----------------------------------------+
6 rows in set (0.00 sec)
```

L'exemple suivant est plus complexe. La requête retourne la valeur de pertinence et organise les lignes par ordre décroissant de pertinence. Pour obtenir ce résultat, il faut spécifier `MATCH()` deux fois. Cela ne cause pas de surcharge car l'optimiseur de `MySQL` remarquera que les deux appels à `MATCH()` sont identiques et appellent le code de recherche sur texte intégral une seule fois.

```
 mysql> SELECT id, body, MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root') AS score
    -> FROM articles WHERE MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root');
+----+-------------------------------------+-----------------+
| id | body                                | score           |
+----+-------------------------------------+-----------------+
|  4 | 1. Never run mysqld as root. 2. ... | 1.5055546709332 |
|  6 | When configured properly, MySQL ... |   1.31140957288 |
+----+-------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

`MySQL` utilise un filtre très simple pour séparer le texte en mots. Un "mot" est n'importe quelle chaîne de caractères constituée de lettres, chiffres, `'''` et `'-'` . Tout "mot" présent dans la liste des mots à ignorés ou qui est trop court (3 caractères ou moins) est ignoré.

- Un mot trop court est ignoré. La taille minimale pour un mot dans les recherches est de 4 lettres.

- Les mots de la liste sont ignorés. Un mot banni est par exemple `` `the` '' ou `` `some` '', ``un'' or ``les'' en français, qui sont considérés comme trop communs pour avoir une valeur intrinsèque. Il y a une liste de mots bannis en anglais par défaut.

La taille minimale des mots et la liste de mots à ignorer sont décrites dans la section Optimisation des recherches en texte intégral .

Tous les mots corrects de la liste et de la requête sont pondérés en fonction de leur importance dans la liste ou la requête. De cette façon, un mot présent dans de nombreuses lignes aura un poids faible (et peut être même un poids nul), car il a peu d'importance dans cette requête particulière. Au contraire, si le mot est rare, il recevra un poids fort. Le poids des mots sont alors rassemblés pour calculer la pertinence de la ligne.

Une telle technique fonctionne plus efficacement sur de grands volumes de données (en fait, elle est optimisée pour cela). Avec les toutes petites tables, la distribution des mots ne reflète par correctement leur valeur sémantique et ce modèle peut parfois produire des résultats étranges.

```
mysql> SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('MySQL');
Empty set (0.00 sec)
```

La recherche du mot `MySQL` ne donne aucun résultat dans l'exemple précédent, car il est présent dans plus de la moitié des lignes. Ainsi, il est considéré comme un mot à ignorer (un mot avec une valeur sémantique nulle). C'est le comportement le plus optimal : un langage de requêtes ne doit pas retourner chaque ligne d'une table de 1 Go.

Un mot qui est trouvé dans la moitié des enregistrements d'une table n'est pas efficace pour trouver les document appropriés. En fait, il trouvera sûrement beaucoup de documents inappropriés à la recherche. On sait tous que cela arrive souvent lorsqu'on recherche quelque chose sur internet en utilisant un moteur de recherche. C'est en suivant ce raisonnement que ces lignes se sont vues attribuer une valeur sémantique très basse dans **ce cas particulier** .

Le seuil de 50% a un impact significatif lorsque vous commencez à comprendre comment fonctionne l'index : si vous créez une table et insérez une ou deux lignes, chaque mot apparaîtra dans 50% des lignes. Résultat, la recherche ne trouvera rien. Assurez−vous d'insérer au moins trois lignes, et même plus.

13.6.1 Booléens de recherches en texte intégral

Depuis la version 4.0.1, MySQL peut aussi effectuer des recherchez en texte intégral avec l'option `IN BOOLEAN MODE` .

```
mysql> SELECT * FROM articles WHERE MATCH (title,body)
    -> AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+-----------------------+-------------------------------------+
| id | title                 | body                                |
+----+-----------------------+-------------------------------------+
|  1 | MySQL Tutorial        | DBMS stands for DataBase ...        |
|  2 | How To Use MySQL Well | After you went through a ...        |
|  3 | Optimizing MySQL      | In this tutorial we will show ...   |
|  4 | 1001 MySQL Tricks     | 1. Never run mysqld as root. 2. ... |
|  6 | MySQL Security        | When configured properly, MySQL ... |
+----+-----------------------+-------------------------------------+
```

Cette requête recherche toute les lignes qui contiennent le mot `` `MySQL` '', mais qui **ne contient pas** le mot `` `YourSQL` ''.Les recherches booléennes en texte intégral ont les caractéristiques suivantes :

- Elle n'utilise pas le seuil de 50%.

- Elles ne trie pas automatiquement les lignes par ordre de pertinence décroissante. Vous pouvez le voir dans l'exemple précédent : la ligne ayant la plus grande pertinence est celle qui contient `` `MySQL` '' deux fois, mais elle est listée en dernier.
- Elles peuvent fonctionner dans l'index `FULLTEXT` , même si c'est particulièrement **lent** .

Les recherches booléenne en texte intégral supporte les opérateurs suivants :

`+`
: A signe + initial indique que le mot **doit être** présent dans la ligne retournée.

`-`
: Un signe − initial indique que le mot **ne doit pas** être présent dans la ligne retournée.

`(pas d'opérateur)`
: Par défaut, lorsque ni `+` , ni `-` n'est spécifié), le mot est optionnel, mais les lignes qui le contiennent seront mieux cotées. Cela imite le comportement de `MATCH() ... AGAINST()` sans l'option `IN BOOLEAN MODE` .

`> <`
: Ces deux opérateurs servent à changer la contribution d'un mot à la pertinence. L'opérateur `>` accroît la contribution, et l'opérateur `<` la décroît. Voir un exemple ci−dessous.

`( )`
: Les parenthèses servent à grouper des mots en sous−expressions. Les groupes de parenthèses peuvent être imbriquées.

`~`
: Un signe tilde initial marque la négation, et fait que la contribution du mot à la pertinence sera négative. Cet opérateur est pratique pour marquer les mots ambigus. Une ligne qui contient un tel mot sera classée bien plus bas, mais elle ne sera pas exclue, comme ce serait le cas avec `-` .

`*`
: Un astérisque est l'opérateur de troncature. Contrairement aux autres opérateurs, il doit être en **suffixe** et non pas en préfixe.

`"`
: Une phrase entre guillemets double (`'"'`) est recherchée littéralement, **telle qu'elle a été saisie** .

Les exemples ci−dessous illustrent quelques résultats de chaînes de recherche avec les opérateurs :

`'pomme banane'`
: Recherche les lignes qui contiennent au moins un de ces mots.

`'+pomme +jus'`
: Recherche les lignes qui contiennent ces deux mots.

`'+pomme macintosh'`
: Recherche les lignes qui contiennent le mot `` `pomme` '', mais classe plus haut les lignes qui contiennent aussi `` `macintosh` ''.

`'+pomme -macintosh'`
: Recherche les lignes qui contiennent `` `pomme` '' mais pas `` `macintosh` ''.

`'+pomme +(>tatin <strudel)'`
: Recherche les lignes qui contiennent les mots `` `pomme` '' et `` `tatin` '', ou `` `pomme` '' et `` `strudel` '' (dans n'importe quel ordre), mais classe `` `pomme tatin` '' plus haut que `` `pomme strudel` ''.

`'pomm*'`
: Trouve les lignes qui contiennent des mots tels que `` `pomme` '', `` `pommes` '', `` `pommier` '', ou `` `pommeau` ''.

`'"deux mots"'`
: Recherche les lignes qui contiennent exactement la phrase `` `deux mots` '' (par exemple, les lignes qui contiennent `` `deux mots d'amour` '' mais pas `` `le mot deux` ''). Notez que les caractères `'"'` qui entourent la phrase délimitent la phrase. Ils ne délimitent pas la chaîne.

13.6.2 Recherche en texte intégral avec extension de requête

Depuis la version 4.1.1, la recherche en texte intégral supporte l'extension de requête (en particulier la variable dite ``extension aveugle''). C'est généralement utile lorsque la phrase de recherche est trop courte, ce qui signifie que l'utilisateur sous−entend des informations. Par exemple, un utilisateur qui recherche `` database '' peut en fait rechercher `` MySQL8 '', `` Oracle '', `` DB2 '' ou encore `` RDBMS '' : ce sont des solutions qui doivent être liées à `` databases '' et être retournée. C'est de l'information implicite.

L'extension de requête aveugle (dite aussi, pertinence automatique), fonctionne en faisant la même recherche 2 fois : la seconde fois, la recherche est complétée avec les mots les plus fréquents des premiers résultats. Par conséquent, si un de ces document contenait `` databases '' et `` MySQL '', la seconde recherche va rechercher les documents qui contiennent `` MySQL '' mais pas `` database ''. L'exemple suivant illustre la différence :

```
mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body)
    -> AGAINST ('database' WITH QUERY EXPANSION);
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  3 | Optimizing MySQL  | In this tutorial we will show ...        |
+----+-------------------+------------------------------------------+
3 rows in set (0.00 sec)
```

Un autre exemple est la recherche de livres de Georges Simenon, de la série Maigret, alors que l'utilisateur ne sais pas trop comment écrire `` Maigret ''. Alors que la recherche de `` Megre and the reluctant witnesses '' ne conduit qu'à `` Maigret and the Reluctant Witnesses '' sans l'extension aveugle, la version avec extension aveugle va sortir la collection complète des livres avec le mot `` Maigret ''.

Note : comme l'extension aveugle augmente le niveau de bruit, en retournant des documents sans rapport, elle n'est utile que si la phrase de recherche est courte.

13.6.3 Restrictions avec la recherche en texte intégral

- La recherche en texte intégral n'est supportée que par les tables MyISAM .
- Depuis MySQL 4.1.1, les recherches en texte plein peuvent être utilisées avec la plupart des jeux de caractères. L'exception est pour Unicode, le jeu de caractères utf8 peut être utilisé, mais pas ucs2 .
- Depuis MySQL 4.1, l'utilisation de jeux de caractères multiples dans une table est supportée. Cependant, toutes les colonnes dans un index FULLTEXT doivent avoir le même jeu de caractères et collation.
- Les arguments de MATCH() doivent correspondre exactement à la liste de colonnes de certaines définitions d'index FULLTEXT pour la table, sauf si MATCH() est utilisé dans un contexte BOOLEAN .
- L'argument de AGAINST() doit être une chaîne constante.

13.6.4 Paramétrage précis de la recherche en text intégral de MySQL

La recherche sur texte entier n'a malheureusement pas encore beaucoup de paramètres modifiables par l'utilisateur, même si l'ajout de certains apparaît très haut dans la liste de tâches. Si vous utilisez MySQL depuis les sources (Installer MySQL à partir des sources .), vous pouvez mieux contrôler le fonctionnement de la recherche sur texte entier.

La recherche sur texte entier a été paramétrée pour une efficacité de recherche maximale. La modification du comportement par défaut ne fera généralement que diminuer la qualité des résultats des recherches. Il ne faut pas modifier les sources de MySQL sans savoir précisément ce qu'on fait.

- La taille minimale des mots à indexer est définie dans la variable `ft_min_word_len` de MySQL. Syntaxe de `SHOW VARIABLES` . Vous pouvez modifier cette valeur pour celle que vous préférez, puis reconstruire les index `FULLTEXT` . (Cette variable n'existe que pour la version 4.0 de MySQL) La valeur par défaut de cette option est de 4 caractères. Modifiez la, puis recompilez les index `FULLTEXT` . Par exemple, si vous souhaitez pouvoir rechercher des mots de 3 caractères, vous pouvez donner à cette variable la valeur suivante dans le fichier d'options :
  ```
  [mysqld]
  ft_min_word_len=3
  ```
  Puis, relancez le serveur et recompilez vos index `FULLTEXT` .
- La liste des mots rejetés est définie dans la variable `ft_stopword_file` . Syntaxe de `SHOW VARIABLES` . Modifiez le selon vos goûts, reconstruisez vos index `FULLTEXT` .
- Le taux de 50% est déterminé par la méthode de pondération choisie. Pour le désactiver, il faut changer la ligne suivante dans `myisam/ftdefs.h` :
  ```
  #define GWS_IN_USE GWS_PROB
  ```
  Par la ligne:
  ```
  #define GWS_IN_USE GWS_FREQ
  ```
  Puis recompiler MySQL. Il n'est pas nécessaire de reconstruire les index dans ce cas. **Note** : en faisant ces modifications, vous diminuez **énormément** les capacités de MySQL à fournir des valeurs pertinentes pour la fonction `MATCH()` . Si vous avez réellement besoin de faire des recherches avec ces mots courants, il est préférable de rechercher `EN MODE BOOLEEN` , lequel ne respecte pas le taux de 50%.
- Pour changer les opérateurs utilisés pour les recherches booléennes, modifiez la variable système `ft_boolean_syntax` (disponible depuis MySQL 4.0.1). La variable peut aussi être modifiée durant le fonctionnement du serveur, mais vous devez avoir les droits de `SUPER` . La recompilation des index n'est pas possible. Variables sytème du serveur décrit les règles de définition de cette variable.

Si vous modifiez des variables d'indexation de textes qui affectent les index (les variables (`ft_min_word_len` , `ft_max_word_len` et `ft_stopword_file`), vous devez reconstruire les index `FULLTEXT` après avoir faire les modifications et relancé le serveur. Pour reconstruire les index, il est suffisant de faire une réparation `QUICK` :

```
mysql> REPAIR TABLE tbl_name QUICK;
```

Si vous utilisez spécifiquement les fonctionnalités `IN BOOLEAN MODE` , si vous mettez à jour depuis MySQL 3.23 vers 4.0 ou plus récent, il est nécessaire de remplacer aussi les entêtes des index. Pour cela, utilisez l'opération de réparation `USE_FRM` :

```
mysql> REPAIR TABLE nom_de_table USE_FRM;
```

C'est nécessaire, car les recherches booléennes en texte plein requièrent une option dans l'entête qui n'était pas présente en MySQL en version 3.23, et elle n'est pas ajoutée si vous faites une réparation de type `QUICK` . Si vous tentez une recherche booléenne sans reconstruire l'index comme ceci, la recherche retournera des résultats incorrects.

Notez que si vous utilisez `myisamchk` pour effectuer une opération qui modifie les index de la table, pour une réparation ou une analyse, les index `FULLTEXT` sont reconstruits en utilisant la configuration par défaut des index en texte plein, à moins que vous ne les spécifiez autrement. Cela peut conduire à des requêtes qui échouent.

Le problème survient car les valeurs de cette configuration n'est connue que du serveur. Elles ne sont pas stockées dans les fichiers d'index `MyISAM` . Pour éviter ce problème si vous avez modifié la taille minimale ou maximale des mots, ou encore le fichier de mots interdits, spécifiez les options `ft_min_word_len` , `ft_max_word_len` et `ft_stopword_file` à `myisamchk` , en donnant les mêmes valeurs que pour `mysqld` . Par exemple, si vous avez donnez une taille minimale de mot de 3, vous pouvez réparer votre table avec `myisamchk` :

```
shell> myisamchk --recover --ft_min_word_len=3 tbl_name.MYI
```

Pour vous assurer que le serveur et `myisamchk` utilisent les mêmes valeurs pour les index, vous pouvez les placer dan s les sections `[mysqld]` et `[myisamchk]` du fichier d'options :

```
 [mysqld]
ft_min_word_len=3

 [myisamchk]
ft_min_word_len=3
```

Une alternative à l'utilisation de `myisamchk` est l'utilisation de `REPAIR TABLE` , `ANALYZE TABLE` , `OPTIMIZE TABLE` ou `ALTER TABLE` . Ces commandes sont effectuées par le serveur, qui connaît la configuration des index en texte plein.

13.6.5 A faire dans la recherche Full−text

- Rendre toutes les opérations avec l'index `FULLTEXT` plus rapides.
- Opérateurs de proximité
- Support de listes de mots à toujours indexer (" `always-index words` "). Ceux ci pourraient être n'importe quelle chaîne de caractères que l'utilisateur voudrais traiter comme des mots : par exemple " `C++` ", " `AS/400` ", " `TCP/IP` ", etc.
- Support de la recherche `full-text` sur les tables `MERGE` .
- Support des jeux de caractères multi−octets.
- Rendre la liste des mots ignorés dépendante de la langue des données.
- `Stemming` (dépendante de la langue des données, bien sûr).
- Preprocesseur générique pour les `UDF` fournies par l'utilisateur.
- Rendre le modèle plus flexible (en ajoutant des valeurs paramétrables pour `FULLTEXT` dans `CREATE/ALTER TABLE`).

13.7 Fonctions de transtypage

```
CAST(expr AS type)
CONVERT(expr,type)
CONVERT(expr USING transcoding_name)
```

Les fonctions `CAST()` et `CONVERT()` peuvent être utilisées pour convertir une donnée d'un type en un autre. Leurs syntaxes sont :

La valeur de `type` peut être l'une des suivantes :

```
BINARY
CHAR
DATE
DATETIME
SIGNED {INTEGER}
TIME
UNSIGNED {INTEGER}
```

`CAST()` et `CONVERT()` sont disponibles depuis MySQL 4.0.2. La conversion de type `CHAR` est disponible depuis la version 4.0.6. La forme `USING` de `CONVERT()` est disponible depuis la version 4.1.0.

`CAST()` et `CONVERT(... USING ...)` sont des syntaxes SQL−99. La forme sans `USING` de `CONVERT()` est une syntaxe ODBC.

`CONVERT()` avec la clause `USING` sert à convertir des données entre différent jeux de caractères. Avec MySQL, les noms d'encodage sont les mêmes que les noms des jeux de caractères. Par exemple, cette commande converti la chaîne `'abc'` depuis le jeu de caractères par défaut du serveur vers `utf8` :

```
SELECT CONVERT('abc' USING utf8);
```

La fonction de transtypage est très pratique lorsque vous voulez créer une colonne avec un type spécifique dans une requête `CREATE ... SELECT` :

```
CREATE TABLE nouvelle_table SELECT CAST('2000−01−01' AS DATE);
```

Les fonctions peuvent aussi être utilisée pour trier des colonnes de type `ENUM` en ordre lexical. Normalement, le tri sur les colonnes `ENUM` est fait avec les valeurs numériques internes. Pour trier les valeurs avec l'ordre lexical `CHAR` :

```
SELECT enum_col FROM tbl_name ORDER BY CAST(enum_col AS CHAR);
```

`CAST(string AS BINARY)` est l'équivalent de `BINARY string` . `CAST(expr AS CHAR)` traite l'expression comme une chaîne, avec le jeu de caractères par défaut.

Note : en MySQL 4.0 le `CAST()` en `DATE` , `DATETIME` ou `TIME` ne fait que marquer la colonne comme étant du type indiqué, mais n'en change pas la valeur.

En MySQL 4.1.0, la valeur est convertie dans le type de colonne demandé, puis il est envoyé à l'utilisateur. Cette fonctionnalité est une nouveauté du protocole 4.1, qui envoie les données au client :

```
mysql> SELECT CAST(NOW() AS DATE);
       −> 2003−05−26
```

Dans les prochaines versions de MySQL (probablement 4.1.2 ou 5.0) nous allons corriger `CAST` pour qu'elle modifie le résultat si vous l'utilisez comme une partie d'une expression plus complexe, comme `CONCAT("Date: ",CAST(NOW() AS DATE))` .N'utilisez pas `CAST()` pour extraire des données dans différents formats, mais utilisez plutôt `LEFT` ou `EXTRACT()` . Fonctions de date et heure .

Pour transformer une chaîne de caractères en une valeur numérique, vous ne devez rien faire de particulier ; juste utiliser la valeur de la chaîne en lieu et place de la valeur numérique :

```
mysql> SELECT 1+'1';
       −> 2
```

Si vous utilisez un nombre dans un contexte de chaîne, le nombre sera automatiquement converti en une chaîne binaire.

```
mysql> SELECT concat("salut toi ",2);
       −>  "salut toi 2"
```

Si vous utilisez un nombre dans un contexte de chaîne, le nombre sera automatiquement converti en chaîne binaire (`BINARY`).

```
mysql> SELECT CONCAT("Salut vous ",2);
       −>  "Salut vous 2"
```

MySQL supporte l'arithmétique avec les valeurs 64 bits signées et non signées. Si vous utilisez une opération numérique (comme le signe `+`) et qu'un des opérandes est de type `unsigned integer` , alors, le résultat sera une valeur non signé. Vous pouvez corriger cela en utilisant les opérateurs de transtypages `SIGNED` et `UNSIGNED` , qui transformeront l'opération respectivement en un entier signé sur 64 bits et un entier non signé sur 64 bits.

```
mysql> SELECT CAST(1−2 AS UNSIGNED)
        −> 18446744073709551615
mysql> SELECT CAST(CAST(1−2 AS UNSIGNED) AS SIGNED);
        −> −1
```

Notez que si l'une ou l'autre opération est une valeur à virgule flottante (Dans ce contexte, `DECIMAL()` est considéré comme une valeur à virgule flottante) le résultat devrait être une valeur à virgule flottante et ne sera pas affecté par la règle ci−dessus.

```
mysql> SELECT CAST(1 AS UNSIGNED) -2.0
       -> -1.0
```

Si vous utilisez une chaîne dans une opération arithmétique, elle sera converti en un nombre à virgule flottante.

Les fonctions `CAST()` et `CONVERT()` ont été ajoutées dans la version 4.0.2 de MySQL.

L'affichage des valeurs non signées a été modifié dans la version 4.0 de MySQL pour pouvoir supporter correctement les valeurs de type `BIGINT` . Si vous voulez utiliser du code fonctionnant dans la version 4.0 et la version 3.23 de MySQL (dans ce cas, vous ne pouvez probablement pas utiliser les fonctions de transtypage), vous pouvez utiliser l'astuce suivante pour avoir un résultat signé lorsque vous soustrayez deux colonnes d'entier non signé :

```
SELECT (unsigned_column_1+0.0)-(unsigned_column_2+0.0);
```

L'idée est que les colonnes sont convertis en un point mobile avant de faire la soustraction.Si vous rencontrez un problème avec les colonnes `UNSIGNED` dans vos anciennes applications MySQL lorsque vous effectuez le port sous la version 4.0 de MySQL , vous pouvez utiliser l'option `--sql-mode=NO_UNSIGNED_SUBTRACTION` lorsque vous lancez `mysqld` . Notez cependant qu'aussi longtemps que vous employez ceci, vous ne serez pas capable d'utiliser efficacement les colonnes de type `UNSIGNED BIGINT` .

13.8 Autres fonctions

13.8.1 Fonctions sur les bits

MySQL utilise l'arithmétique des BIGINT (64−bits) pour les opérations sur les bits. Ces opérateurs travaillent donc sur 64 bits.

|
: OU bit−à−bit (OR)

```
mysql> SELECT 29 | 15;
       -> 31
```

Le résultat est un entier de 64 bits non signé.

&
: ET bit−à−bit (AND)

```
mysql> SELECT 29 & 15;
       -> 13
```

Le résultat est un entier de 64 bits non signé.

^
: XOR bit−à−bit

```
mysql> SELECT 1 ^ 1;
       -> 0
mysql> SELECT 1 ^ 0;
       -> 1
mysql> SELECT 11 ^ 3;
       -> 8
```

Le résultat est un entier de 64 bits non signé.

<<
: Décale les bits de l'entier (`BIGINT`) sur la gauche :

```
mysql> SELECT 1 << 2;
       -> 4
```

Le résultat est un entier de 64 bits non signé.

>>
: Décale les bits de l'entier (`BIGINT`) sur la droite :

```
mysql> SELECT 4 >> 2;
       -> 1
```

Le résultat est un entier de 64 bits non signé.

~

Inverse tous les bits :

```
mysql> SELECT 5 & ~1;
       -> 4
```

Le résultat est un entier de 64 bits non signé.

BIT_COUNT(N)

Retourne le nombre de bits non nuls de l'argument `N` :

```
mysql> SELECT BIT_COUNT(29);
       -> 4
```

13.8.2 Fonctions de chiffrements

Les fonctions de cette section chiffrent et déchiffrent des valeurs. Si vous voulez stocker le résultat d'un chiffrement qui peut contenir des valeur arbitraires, vous devez utiliser une colonne `BLOB` plutôt que `CHAR` ou `VARCHAR` , afin d'éviter les problèmes potentiels de suppression d'espaces terminaux, qui corrompraient les valeurs.

`AES_ENCRYPT(str,key_str)`
`AES_DECRYPT(crypt_str,key_str)`

Ces fonctions permettent le chiffrement/déchiffrement de données utilisant l'algorithme AES (`Advanced Encryption Standard`), anciennement connu sous le nom de Rijndael. Une clé de 128 bits est utilisé pour le chiffrement, mais vous pouvez l'étendre à 256 bits en modifiant les sources. Nous avons choisi 128 bits parce que c'est plus rapide et suffisamment sécurisé.

Les arguments peuvent être de n'importe quelle taille. Si l'un des arguments est `NULL` , le résultat de cette fonction sera `NULL` .

Vu que AES est un algorithme de niveau bloc, le capitonnage est utilisé pour chiffrer des chaînes de longueur inégales et donc, la longueur de la chaîne résultante peut être calculée comme ceci : `16*(trunc(string_length/16)+1)` .

Si la fonction `AES_DECRYPT()` détecte des données invalides ou un capitonnage incorrect, elle retournera `NULL` . Il est également possible que la fonction `AES_DECRYPT()` retourne une valeur différente de `NULL` (valeur incohérente) si l'entrée de données ou la clé est invalide.

Vous pouvez utiliser les fonctions AES pour stocker des données sous une forme chiffrée en modifiant vos requêtes:

```
INSERT INTO t VALUES (1,AES_ENCRYPT("text","password"));
```

Vous pouvez obtenir encore plus de sécurité en évitant de transférer la clé pour chaque requête, en la stockant dans une variable sur le serveur au moment de la connexion :

```
SELECT &#64;password:="my password";
INSERT INTO t VALUES (1,AES_ENCRYPT("text",&#64;password));
```

Les fonctions `AES_ENCRYPT()` et `AES_DECRYPT()` ont été ajoutées dans la version 4.0.2 de MySQL et peuvent être considérées comme étant les fonctions de cryptographie les plus sûres disponibles actuellement dans MySQL.

`DECODE(crypt_str,pass_str)`

Déchiffre la chaîne chiffrée `crypt_str` en utilisant la clé `pass_str` . `crypt_str` doit être une chaîne qui a été renvoyée par la fonction `ENCODE()` .

`ENCODE(str,pass_str)`

Chiffre la chaîne `str` en utilisant la clé `pass_str` . Pour déchiffrer le résultat, utilisez la fonction `DECODE()` .

Le résultat est une chaîne binaire de la même longueur que `string` . Si vous voulez sauvegarder le résultat dans une colonne, utilisez une colonne de type `BLOB` .

`DES_DECRYPT(crypt_str[,key_str])`

Déchiffre une chaîne chiffrée à l'aide de la fonction `DES_ENCRYPT()` .

Notez que cette fonction fonctionne uniquement si vous avez configuré MySQL avec le support SSL. Utilisation des connexions sécurisées .

Si l'argument `key_string` n'est pas donné, la fonction `DES_DECRYPT()` examine le premier bit de la chaîne chiffrée pour déterminer le numéro de clé DES utilisé pour chiffrer la chaîne originale, alors la clé est lu dans le fichier `des-key-file` pour déchiffrer le message. Pour pouvoir utiliser cela, l'utilisateur doit avoir le privilège `SUPER` .

Si vous passé l'argument `key_string` à cette fonction, cette chaîne est utilisée comme clé pour déchiffrer le message.

Si la chaîne `string_to_decrypt` ne semble pas être une chaîne chiffrée, MySQL retournera la chaîne `string_to_decrypt` .

Si une erreur survient, cette fonction retourne `NULL` .

`DES_ENCRYPT(str[,(key_num|key_str)])`

Chiffre la chaîne avec la clé donnée en utilisant l'algorithme DES.

Notez que cette fonction fonctionne uniquement si vous avez configuré MySQL avec le support SSL. Utilisation des connexions sécurisées .

La clé de hachage utilisée est choisie en suivant les recommandations suivantes :

Argument	Description
Un seul argument	La première clé de des-key-file est utilisée.
Un numéro de clé	Le numéro de la clé donnée (0-9) de des-key-file est utilisée.
Une chaîne	La chaîne donnée key_string doit être utilisé pour chiffrer string_to_encrypt .

La chaîne retournée doit être une chaîne binaire où le premier caractère doit être `CHAR(128 | key_number)` .

Le nombre 128 a été ajouté pour reconnaître facilement une clé de hachage. Si vous utilisez une chaîne comme clé, `key_number` doit être 127.

Si une erreur survient, la fonction retournera `NULL` .

La longueur de la chaîne de résultat doit être : `new_length= org_length + (8-(org_length % 8))+1` .

`des-key-file` a le format suivant :

```
 key_number des_key_string
key_number des_key_string
```

Chaque `key_number` doit être un nombre dans l'intervalle 0 à 9. Les lignes dans le fichier peuvent être dans n'importe quel ordre. `des_key_string` est la chaîne qui permettra le chiffrage du message. Entre le nombre et la clé, il doit y avoir au moins un espace. La première clé est la clé par défaut qui sera utilisé si vous ne spécifiez pas d'autres clés en arguments de la fonction `DES_ENCRYPT()` .

Vous pouvez demander à MySQL de lire de nouvelles valeurs de clé dans le fichier de clés avec la commande `FLUSH DES_KEY_FILE` . Cela requière le privilège `Reload_priv` .

Un des bénéfices d'avoir une liste de clés par défaut est que cela donne aux applications la possibilité de regarder l'existence de la valeur chiffrée de la colonne, sans pour autant donner la possibilité à l'utilisateur final de déchiffrer ces valeurs.

```
mysql> SELECT customer_address FROM customer_table WHERE
      crypted_credit_card = DES_ENCRYPT("credit_card_number");
```

`ENCRYPT(str[,salt])`

Chiffre la chaîne `str` en utilisant la fonction `crypt()` . L'argument `salt` doit être une chaîne de deux caractères. (A partir de la version 3.22.16, l'argument `salt` peut être plus long que deux caractères.) :

```
mysql> SELECT ENCRYPT("hello");
       -> 'VxuFAJXVARROc'
```

Si la fonction `crypt()` n'est pas disponible sur votre système, la fonction `ENCRYPT()` retournera toujours `NULL` .La fonction `ENCRYPT()` conserve uniquement les 8 premiers caractères de la chaîne `str` , au moins, sur certains système. Le comportement exact est directement déterminé par la fonction système `crypt()` sous−jacente.

`MD5(str)`

Calcul la somme de vérification MD5 de la chaîne `string` . La valeur retournée est un entier hexadécimal de 32 caractères qui peut être utilisé, par exemple, comme clé de hachage :

```
mysql> SELECT MD5("testing");
       -> 'ae2b1fca515949e5d54fb22b8ed95575'
```

C'est l'algorithme RSA (" `RSA Data Security, Inc. MD5 Message-Digest Algorithm` ").

`OLD_PASSWORD(str)`

`OLD_PASSWORD()` est disponible depuis MySQL 4.1, lorsque l'implémentation de la fonction `PASSWORD()` a été modifiée pour améliorer la sécurité. `OLD_PASSWORD()` retourne la valeur pre−4.1 de `PASSWORD()` . Chiffrement des mots de passe en MySQL 4.1 .

`PASSWORD(str)`

Calcule un mot de passe chiffré à partir de la chaîne `str` . C'est cette fonction qui est utilisé pour chiffrer les mots de passes MySQL pour être stockés dans une colonne de type `Password` de la table `user` :

```
mysql> SELECT PASSWORD('badpwd');
       -> '7f84554057dd964b'
```

Le chiffrage par `PASSWORD()` n'est pas réversible.

`PASSWORD()` n'est pas un chiffrage comparable à la fonction de chiffrage Unix. Voir `ENCRYPT()` .

Note : La fonction `PASSWORD()` est utilisée durant l'identification au serveur MYSQL. Il est recommandé de **ne pas l'utiliser** pour vos applications. Utilisez plutôt `MD5()` ou `SHA1()` . Voyez aussi `RFC-2195` pour plus d'informations sur comment gérer les mots de passe et l'identification de votre système.

`SHA1(str)`
`SHA(str)`

Calcule la somme de vérification SHA1 160 bits de la chaîne `string` , comme décrit dans la RFC 3174 (`Secure Hash Algorithm`). La valeur retournée est un entier hexadécimal de 40 caractères, ou bien `NULL` dans le cas où l'argument vaut `NULL` . Une des possibilités d'utilisation de cette fonction est le hachage de clé. Vous pouvez aussi l'utilisé comme fonction de cryptographie sûre pour stocker les mots de passe.

```
mysql> SELECT SHA1("abc");
       -> 'a9993e364706816aba3e25717850c26c9cd0d89d'
```

La fonction `SHA1()` a été ajoutée dans la version 4.0.2 de MySQL et peut être considérée comme une méthode de cryptographie plus sûre que la fonction `MD5()` . La fonction `SHA()` est un alias de la fonction `SHA1()` .

13.8.3 Fonctions d'informations

BENCHMARK(count,expr)

La fonction `BENCHMARK()` exécute l'expression `expr` de manière répétée `count` fois. Elle permet de tester la vélocité de MySQL lors du traitement d'une requête. Le résultat est toujours `0` . L'objectif de cette fonction ne se voit que du côté client, qui permet à ce dernier d'afficher la durée d'exécution de la requête :

```
mysql> SELECT BENCHMARK(1000000,ENCODE("bonjour","au revoir"));
+--------------------------------------------------+
| BENCHMARK(1000000,ENCODE("bonjour","au revoir")) |
+--------------------------------------------------+
|                                                0 |
+--------------------------------------------------+
1 row in set (4.74 sec)
```

Le temps affiché est le temps côté client, et non pas les ressources processeurs consommées. il est conseillé d'utiliser `BENCHMARK()` plusieurs fois de suite pour interpréter un résultat, en dehors de charges ponctuelles sur le serveur.

CHARSET(str)

Retourne le jeu de caractères de la chaîne argument.

```
mysql> SELECT CHARSET('abc');
        -> 'latin1'
mysql> SELECT CHARSET(CONVERT('abc' USING utf8));
        -> 'utf8'
mysql> SELECT CHARSET(USER());
        -> 'utf8'
```

`CHARSET()` a été ajouté en MySQL version 4.1.0.

COERCIBILITY(str)

Retourne la coercibilité de la collation de la chaîne argument.

```
mysql> SELECT COERCIBILITY('abc' COLLATE latin1_swedish_ci);
        -> 0
mysql> SELECT COERCIBILITY('abc');
        -> 3
mysql> SELECT COERCIBILITY(USER());
        -> 2
```

Les valeurs retournées possibles sont :

Coercibilité	**Signification**
0	Collation explicite
1	Par de collation
2	Collation implicite
3	Coercible

Les valeurs les plus faibles ont la plus haute priorité.

`COERCIBILITY()` a été ajouté en MySQL version 4.1.1.

COLLATION(str)

Retourne la collation du jeu de caractères de la chaîne argument.

```
mysql> SELECT COLLATION('abc');
        -> 'latin1_swedish_ci'
mysql> SELECT COLLATION(_utf8'abc');
        -> 'utf8_general_ci'
```

`COLLATION()` a été ajouté en MySQL version 4.1.0.

CONNECTION_ID()

Retourne l'identifiant de connexion courant (`thread_id`). Chaque connexion a son propre identifiant unique.

```
mysql> SELECT CONNECTION_ID();
        -> 23786
```

`CONNECTION_ID()` a été ajouté en MySQL version 3.23.14.

CURRENT_USER()

Retourne le nom d'utilisateur et le nom d'hôte de la session courante. Cette valeur correspond au compte qui a été utilisé durant l'identification auprès du serveur. Cela peut être différent des valeurs de `USER()` .

```
mysql> SELECT USER();
        -> 'davida&#64;localhost'
mysql> SELECT * FROM mysql.user;
ERROR 1044: Access denied for user: '&#64;localhost' to
database 'mysql'
mysql> SELECT CURRENT_USER();
        -> '&#64;localhost'
```

Cet exemple montre que même si le client a indiqué le nom d'utilisateur `davida` (comme mentionné par la fonction `USER()`), le serveur a identifié le client comme un utilisateur anonyme (comme indiqué par la fonction `CURRENT_USER()`). Une situation qui arrive s'il n'y a aucun compte de listé dans les tables de droits pour `davida` . `CURRENT_USER()` a été ajouté en MySQL version 4.0.6.

DATABASE()

Retourne le nom de la base de données courante :

```
mysql> SELECT DATABASE();
       -> 'test'
```

Si aucune base de données n'a été sélectionnée, `DATABASE()` retourne une chaîne vide. A partir de la version 4.1.1, elle retourne `NULL` .

FOUND_ROWS()

Une commande `SELECT` peut inclure une clause `LIMIT` pour restreindre le nombre de lignes qui sera retournée par le client. Dans certains cas, il est mieux de savoir combien de lignes une commande aurait retourné, sans la clause `LIMIT` , mais sans lancer à nouveau le calcul. Pour cela, ajoutez l'option `SQL_CALC_FOUND_ROWS` dans la commande `SELECT` , puis appelez `FOUND_ROWS()` après :

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name
    -> WHERE id > 100 LIMIT 10;
mysql> SELECT FOUND_ROWS();
```

Le second `SELECT` retourne un nombre indiquant combien de lignes le premier `SELECT` aurait retourné s'il n'avait pas été écrit avec une clause `LIMIT` .Notez que si vous utilisez `SELECT SQL_CALC_FOUND_ROWS ...` , MySQL calcule toutes les lignes dans la liste des résultats. Ainsi, c'est plus rapide si vous n'utilisez pas de clause `LIMIT` et que la liste des résultats n'a pas besoin d'être envoyée au client. Si la commande `SELECT` précédente n'inclut pas l'option `SQL_CALC_FOUND_ROWS` , alors `FOUND_ROWS()` pourrait retourner une valeur différente suivant que `LIMIT` est utilisé ou pas.

`SQL_CALC_FOUND_ROWS` et `FOUND_ROWS()` peuvent être pratiques dans des situations où vous devez limiter le nombre de lignes que la requête retourne, mais que vous devez tout de même connaître le nombre de ligne total, sans exécuter une seconde requête. Un exemple classique est un script web qui présente des résultats de recherche. En utilisant `FOUND_ROWS()` , vous connaîtrez facilement le nombre de lignes de résultat.

L'utilisation de `SQL_CALC_FOUND_ROWS` et `FOUND_ROWS()` est plus complexe pour les requêtes `UNION` que pour les commandes `SELECT` simples, car `LIMIT` peut intervenir plusieurs fois dans une commande `UNION` . Elle sera appliquée à différentes commandes `SELECT` de la commande `UNION` , ou globalement à l' `UNION` .

Le but de `SQL_CALC_FOUND_ROWS` pour `UNION` est deretourner le nombre de ligne qui aurait été retourné sans la clause globale `LIMIT` . Les conditions d'utilisation de `SQL_CALC_FOUND_ROWS` avec `UNION` sont :

- Le mot clé `SQL_CALC_FOUND_ROWS` doit apparaitre dans le premier `SELECT` de l' `UNION` .
- La valeur de `FOUND_ROWS()` est exactement la même que si `UNION ALL` était utilisé. Si `UNION` sans `ALL` est utilisé, des réductions de doublons surviendront, et la valeur de `FOUND_ROWS()` sera approximative.
- Si aucune clause `LIMIT` n'est présente dans `UNION` , `SQL_CALC_FOUND_ROWS` est ignoré et retourne le nombre de lignes dans la table temporaire créé durant le traitement de l' `UNION` .

`SQL_CALC_FOUND_ROWS` et `FOUND_ROWS()` sont disponibles à partir de la version 4.0.0 de MySQL.

LAST_INSERT_ID()
LAST_INSERT_ID(expr)

Returns the last automatically generated value that was inserted into an `AUTO_INCREMENT` column.

```
mysql> SELECT LAST_INSERT_ID();
       -> 195
```

Le dernier ID généré est conservé par le serveur pour chaque connexion. Un autre client ne la modifiera donc pas, même s'ils génèrent une autre valeur `AUTO_INCREMENT` de leur coté. Ce comportement permet de s'assurer que les actions des autres clients ne perturbe pas les actions du client en cours.

La valeur de `LAST_INSERT_ID()` ne sera pas modifiée non plus si vous modifiez directement la valeur d'une colonne `AUTO_INCREMENT` avec une valeur simple (c'est à dire, une valeur qui n'est ni `NULL` , ni 0).

Si vous insérez plusieurs lignes au même moment avec une requête `INSERT` , `LAST_INSERT_ID()` retourne la valeur de la première ligne insérée. La raison à cela est que cela rend possible la reprodruction facilement la même requête `INSERT` sur d'autres serveurs.

Si vous utilisez une commande `INSERT IGNORE` et que la ligne est ignorée, le compteur `AUTO_INCREMENT` sera malgré tout incrémenté, et `LAST_INSERT_ID()` retournera une nouvelle valeur.

Si `expr` est donnée en argument à la fonction `LAST_INSERT_ID()` , alors la valeur de l'argument sera retourné par la fonction et sera enregistré comme étant la prochaine valeur retournée par `LAST_INSERT_ID()` . Cela peut être utilisé pour simuler des séquences :

Commencez par créer la table suivante :

```
 mysql> CREATE TABLE sequence (id INT NOT NULL);
mysql> INSERT INTO sequence VALUES (0);
```

Utilisez cette table pour générer des séquences de nombre comme ceci :

```
 mysql> UPDATE sequence SET id=LAST_INSERT_ID(id+1);
mysql> SELECT LAST_INSERT_ID();
```

La commande `UPDATE` incrémente le compteur de séquence, et fait que le prochain appel à `LAST_INSERT_ID()` va retourner une valeur différente. La commande `SELECT` lit cette valeur. La fonction C `mysql_insert_id()` peut aussi être utilisée pour lire la valeur. `mysql_insert_id()` .

Vous pouvez générer des séquences sans appeler la fonction `LAST_INSERT_ID()` , mais l'utilité d'utiliser cette fonction cette fois si est que la valeur ID est gérée par le serveur comme étant la dernière valeur générée automatiquement. (sécurité multi−utilisateur). Vous pouvez retrouver la nouvelle ID tout comme vous pouvez lire n'importe quelle valeur `AUTO_INCREMENT` dans MySQL. Par exemple, la fonction `LAST_INSERT_ID()` (sans argument) devrait retourner la nouvelle ID. La fonction C de l'API `mysql_insert_id()` peut être également utilisée pour trouver cette valeur.

Notez que la fonction `mysql_insert_id()` est incrémentée uniquement après des requêtes `INSERT` et `UPDATE` , donc, vous ne pouvez pas utiliser la fonction C de l'API pour trouver la valeur de `LAST_INSERT_ID(expr)` après avoir exécuté d'autres types de requêtes, comme `SELECT` ou bien `SET` .

SESSION_USER()
: `SESSION_USER()` est un synonyme de `USER()` .

SYSTEM_USER()
: `SYSTEM_USER()` est un synonyme de `USER()` .

USER()
: Retourne le nom d'utilisateur et le nom d'hôte courant MySQL :

```
 mysql> SELECT USER();
        -> 'davida&#64;localhost'
```

La valeur indique le nom d'utilisateur qui a été spécifié lors de l'identification avec le serveur MySQL, et l'hôte client avec lequel il est connecté. Avant la version 3.22.11, la fonction ne retournait pas le nom d'hôte. Vous pouvez extraire le nom d'utilisateur sans l'hôte avec la commande suivante :

```
 mysql> SELECT SUBSTRING_INDEX(USER(),'&#64;',1);
        -> 'davida'
```

Depuis MySQL version 4.1, `USER()` retourne la valeur au format `utf8` . Assurez vous que la chaîne `'@'` est bien interprétée dans ce jeu de caractères :

```
mysql> SELECT SUBSTRING_INDEX(USER(),_utf8'&#64;',1);
        -> 'davida'
```

VERSION()

Retourne une chaîne indiquant la version courante du serveur MySQL :

```
mysql> SELECT VERSION();
        -> '4.1.2-alpha-log'
```

Notez que si votre version se termine par `-log` , cela signifie que le système d'historique est actif.

13.8.4 Fonctions diverses

`FORMAT(X,D)`

Formate l'argument `X` en un format comme `'#,###,###.##'` , arrondi à `D` décimales. Si `D` vaut `0` , le résultat n'aura ni séparateur décimal, ni partie décimale :

```
mysql> SELECT FORMAT(12332.123456, 4);
        -> '12,332.1235'
mysql> SELECT FORMAT(12332.1,4);
        -> '12,332.1000'
mysql> SELECT FORMAT(12332.2,0);
        -> '12,332'
```

`GET_LOCK(str,timeout)`

Tente de poser un verrou nommé `str` , avec un délai d'expiration (`timeout`) exprimé en seconde. Retourne `1` si le verrou a été posé avec succès, `0` si il n'a pas pu être posé avant l'expiration du délai et `NULL` si une erreur est survenu (comme par exemple un manque de mémoire, ou la mort du thread lui−même, par `mysqladmin kill`). Un verrou sera levé lorsque vous exécuterez la commande `RELEASE_LOCK()` , `GET_LOCK()` ou si le thread se termine. Cette fonction peut être utilisée pour implémenter des verrous applicatifs ou pour simuler des verrous de lignes. Les requêtes concurrentes des autres clients de même nom seront bloquées ; les clients qui s'entendent sur un nom de verrou peuvent les utiliser pour effectuer des verrouillages coopératifs :

```
mysql> SELECT GET_LOCK("lock1",10);
        -> 1
mysql> SELECT IS_FREE_LOCK("lock2");
        -> 1
mysql> SELECT GET_LOCK("lock2",10);
        -> 1
mysql> SELECT RELEASE_LOCK("lock2");
        -> 1
mysql> SELECT RELEASE_LOCK("lock1");
        -> NULL
```

Notez que le deuxième appel à `RELEASE_LOCK()` retourne `NULL` car le verrou `"lock1"` a été automatiquement libéré par le deuxième appel à `GET_LOCK()` .

`INET_ATON(expr)`

Retourne un entier qui représente l'expression numérique de l'adresse réseau. Les adresses peuvent être des entiers de 4 ou 8 octets.

```
mysql> SELECT INET_ATON("209.207.224.40");
       ->  3520061480
```

Le nombre généré est toujours dans l'ordre des octets réseau ; par exemple, le nombre précédent est calculé comme ceci : `209*256^3 + 207*256^2 + 224*256 +40` .Depuis MySQL 4.1.2, `INET_ATON()` comprend aussi les IP courtes :

```
mysql> SELECT INET_ATON('127.0.0.1'), INET_ATON('127.1');
        -> 2130706433, 2130706433
```

`INET_ATON()` a été ajouté en MySQL 3.23.15.

`INET_NTOA(expr)`

Retourne l'adresse réseau (4 ou 8 octets), de l'expression numérique `exp` :

```
mysql> SELECT INET_NTOA(3520061480);
       ->  "209.207.224.40"
```

`IS_FREE_LOCK(str)`

Regarde si le verrou nommé `str` peut être librement utilisé (i.e., non verrouillé). Retourne `1` si le verrou est libre (personne ne l'utilise), `0` si le verrou est actuellement utilisé et `NULL` si une erreur survient (comme un argument incorrect).

`IS_USED_LOCK(str)`

Vérifie si le verrou appelé `str` est actuellement posé ou pas. Si c'est le cas, la fonction retourne l'identifiant de connexion qui a le verrou. Sinon, elle retourne `NULL` .
`IS_USED_LOCK()` a été ajouté en MySQL version 4.1.0.

`MASTER_POS_WAIT(log_name, log_pos)`

Bloque le maître jusqu'à ce que l'esclave atteigne une position donnée dans le fichier d'historique principal, durant une réplication. Si l'historique principal n'est pas initialisé, retourne `NULL` . Si l'esclave n'est pas démarré, le maître restera bloqué jusqu'à ce que l'esclave soit démarré et ai atteint la position demandée. Si l'esclave a déjà dépassé cette position, la fonction se termine immédiatement. La valeur retournée est le nombre d'évènements qui a du être traité pour atteindre la position demandée, ou `NULL` en cas d'erreur. Cette fonction est très utile pour contrôler la synchronisation maître−esclave, mais elle a été initialement écrite pour faciliter les tests de réplications.

`RELEASE_LOCK(str)`

Libère le verrou nommé `str` , obtenu par la fonction `GET_LOCK()` . Retourne `1` si le verrou a bien été libéré, `0` si le verrou n'a pas été libéré par le thread (dans ce cas, le verrou reste posé) et `NULL` si le nom du verrou n'existe pas. Le verrou n'existe pas si il n'a pas été obtenu par la fonction `GET_LOCK()` ou si il a déjà été libéré.

La commande `DO` est utilisable avec `RELEASE_LOCK()` . Syntaxe de `DO` .

`UUID()`

Retourne un `Universal Unique Identifier (UUID)` généré grâce à ``` ``DCE 1.1: Remote Procedure Call'' (Appendix A) CAE (Common Applications Environment) Specifications ``` , publié par le `The Open Group` en octobre 1997 (Document numéro `C706`).

Un UUID est conçu comme un numéro qui est globalement unique dans l'espace, et le temps. Deux appels à `UUID()` sont supposés générer deux valeurs différentes, même si ces appels sont faits sur deux ordinateurs séparés, qui ne sont pas connectés ensembles.

Un UUID est un nombre de 128 bits, représenté par une chaîne de 5 nombres hexadécimaux, au format `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee` :

- Les trois premiers nombres sont générés à partir d'un timestamp.
- Le quatrième nombre préserver l'unicité temporelle si le timestamp perd sa monotonie (par exemple, à cause du changement d'heure d'hiver/été).
- Le cinquième nombre est un nombre IEEE 802 qui fournit l'unicité. Un nombre aléatoire est utilisé la si ce dernier n'est pas disponible (par exemple, comme l'hôte n'a pas de carte Ethernet, nous ne savons pas comment trouver une adresse matériel sur le système d'exploitation). Dans ce cas, l'unicité spatiale ne peut être garantie. Néanmoins, une collision aura une **très** faible propriété.Actuellement, l'adresse MAC est une interface utilisée sur FreeBSD et Linux. Sur les autres systèmes d'exploitation, MySQL génère un nombre aléatoire de 48 bits.

```
mysql> SELECT UUID();
        -> '6ccd780c-baba-1026-9564-0040f4311e29'
```

Notez que `UUID()` ne fonctionne pas encore avec la réplication.

`UUID()` a été ajoutée en MySQL 4.1.2.

13.9 Fonctions et options à utiliser dans les clauses GROUP BY

13.9.1 Fonctions avec GROUP BY

Si vous utilisez les fonctions de groupement avec une requête ne contenant pas de clause `GROUP BY` , cela revient à grouper toutes les lignes.

`AVG(expr)`

Retourne la moyenne de l'expression `expr` :

```
mysql> SELECT student_name, AVG(test_score)
    ->        FROM student
    ->        GROUP BY student_name;
```

`BIT_AND(expr)`

Retourne la combinaison `AND` bit à bit de `expr` . Le calcul est fait en précision de 64 bits (`BIGINT`).

Depuis MySQL 4.0.17, cette fonction retourne 18446744073709551615 s'il n'y avait pas de lignes. (C'est un entier `BIGINT` non−signé, dont tous les bits sont à 1.) Avant 4.0.17, la fonction retournait −1 s'il n'y avait pas de ligne trouvées.

`BIT_OR(expr)`

Retourne la combinaison `OR` bit à bit de `expr` . Le calcul est fait en précision de 64 bits (`BIGINT`).Cette fonction retourne 0 s'il n'y a pas de ligne à traiter.

`BIT_XOR(expr)`

Retourne la combinaison `XOR` bit à bit de `expr` . Le calcul est fait en précision de 64 bits (`BIGINT`).

Cette fonction retourne 0 s'il n'y a pas de ligne à traiter.

Cette fonction est disponible depuis MySQL 4.1.1.

`COUNT(expr)`

Retourne le nombre de valeurs non− `NULL` dans les lignes lues par la commande `SELECT` :

```
mysql> SELECT student.student_name,COUNT(*)
    ->        FROM student,course
    ->        WHERE student.student_id=course.student_id
    ->        GROUP BY student_name;
```

`COUNT(*)` est un peu différente dans son action, car elle retourne le nombre de lignes, même si elles contiennent `NULL` . `COUNT(*)` est optimisée pour retourner très rapidement un résultat si `SELECT` travaille sur une table, qu'aucune autre colonne n'est lue, et qu'il n'y a pas de clause `WHERE` . Par exemple :

```
mysql> SELECT COUNT(*) FROM student;
```

Cette optimisation s'applique uniquement pour les tables `MyISAM` et `ISAM` , car un compte exact du nombre de lignes est stocké pour ces types de tables, et il peut être lu très rapidement. Pour les moteurs de tables transactionnels, (`InnodB` , `BDB`), le stockage de cette valeur est plus problématique, car plusieurs transactions peuvent survenir en même temps, et affecter toutes ce compte.

`COUNT(DISTINCT expr,[expr...])`

Retourne le nombre de valeurs non− `NULL` distinctes :

```
mysql> SELECT COUNT(DISTINCT results) FROM student;
```

Avec MySQL, vous pouvez lire le nombre d'expression distinctes qui ne contiennent pas `NULL` , en plaçant ici une liste d'expression. Avec SQL−99, vous devriez faire une concaténation de toutes les expressions dans `COUNT(DISTINCT ...)` .

`GROUP_CONCAT(expr)`

Syntaxe complète :

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
            [ORDER BY &#123;unsigned_integer | col_name | formula&#125; [ASC | DESC] [,col ...]]
            [SEPARATOR str_val])
```

Cette fonction a été ajoutée en MySQL version 4.1. Elle retourne la chaîne résultant de la concaténation de toutes les valeurs du groupe :

```
mysql> SELECT student_name,
    ->        GROUP_CONCAT(test_score)
    ->        FROM student
     ->        GROUP BY student_name;
```

ou :

```
mysql> SELECT student_name,
    ->        GROUP_CONCAT(DISTINCT test_score
    ->                     ORDER BY test_score DESC SEPARATOR " ")
```

```
    ->        FROM student
    ->        GROUP BY student_name;
```

Avec MySQL, vous pouvez obtenir la concaténation d'une série d'expressions. Vous pouvez éliminer les doublons en utilisant `DISTINCT` . Si vous voulez trier les valeurs du résultat, il faut utiliser `ORDER BY` . Pour trier en ordre inverse, ajoutez le mot clé `DESC` (descendant) au nom de la colonne que vous triez dans la clause `ORDER BY` . Par défaut, l'ordre est ascendant. Cela peut être spécifié explicitement avec le mot clé `ASC` . `SEPARATOR` est une chaîne qui sera insérée entre chaque valeur du résultat. La valeur par défaut est une virgule `","` . vous pouvez supprimer le séparateur en spécifiant la chaîne vide `SEPARATOR ""` .Vous pouvez donner une taille maximale à la variable `group_concat_max_len` de votre configuration. La syntaxe pour faire cela durant l'exécution est :

```
SET [SESSION | GLOBAL] group_concat_max_len = unsigned_integer;
```

Si une taille maximale a été atteinte, le résultat sera tronqué à cette taille maximale. Note : il y a encore de petites limitations pour `GROUP_CONCAT()` lorsqu'il faut utiliser des valeurs `DISTINCT` avec `ORDER BY` et et en utilisant les valeurs `BLOB` . Voyez Bugs ouverts / limitations de MySQL .

`GROUP_CONCAT()` a été ajoutée en MySQL 4.1.

`MIN(expr)`
`MAX(expr)`

Retourne le minimum ou le maximum de `expr` . `MIN()` et `MAX()` peuvent prendre des chaînes comme argument : dans ce cas, elles retournent la valeur minimale ou maximale de la valeur de la chaîne. Comment MySQL utilise les index .

```
mysql> SELECT student_name, MIN(test_score), MAX(test_score)
    ->        FROM student
    ->        GROUP BY student_name;
```

Actuellement, `MIN()` , `MAX()` et d'autres fonctions d'agrégation MySQL, le serveur compare les valeurs de type `ENUM` et `SET` avec leur valeur de chaîne, et non pas leur position relative dans l'ensemble. Ce sera corrigé.à

`STD(expr)`
`STDDEV(expr)`

Retourne la déviation standard de `expr` (la racine carrée de la `VARIANCE()` . Ceci est une extension au standard SQL 99. La forme `STDDEV()` de cette fonction est fournie pour assurer la compatibilité Oracle.

`SUM(expr)`

Retourne la somme de `expr` . Notez que si le résultat ne contient pas de ligne, cette fonction retournera `NULL` .

`VARIANCE(expr)`

Retourne la variance standard de l'expression `expr` (en considérant que les lignes forment une population totale, et non pas un échantillon. Le nombre de ligne est le dénominateur. C'est une extension à la norme SQL−99 (disponible en version version 4.1 ou plus récent).

13.9.2 Options de GROUP BY

Depuis MySQL 4.1.1, la clause `GROUP BY` permet l'utilisation de l'option `WITH ROLLUP` qui fait que des lignes supplémentaires seront ajoutées lors de regroupements. Ces lignes représentent des regroupements de haut niveau (ou des super−agrégats). `ROLLUP` vous permet de répondre simultanément à plusieurs niveaux d'analyse avec une seule requête. Il peut être utilisée, par exemple, pour supporter des opérations `OLAP` (`Online Analytical Processing`).

Voici une illustration. Supposons que vous ayez une table de ventes `sales` , avec des colonnes pour l'année `year` , le pays `country` , le produit `product` et le profit `profit` :

```
CREATE TABLE sales
(
    year    INT NOT NULL,
    country VARCHAR(20) NOT NULL,
    product VARCHAR(32) NOT NULL,
```

```
   profit  INT
);
```

Le contenu de cette table peut être agrégé par année avec la clause `GROUP BY` :

```
mysql> SELECT year, SUM(profit) FROM sales GROUP BY year;
+------+-------------+
| year | SUM(profit) |
+------+-------------+
| 2000 |        4525 |
| 2001 |        3010 |
+------+-------------+
```

Cette requête affiche le profit par année, mais si vous voulez déterminer le profit total de toutes les années, vous devez ajouter ces valeurs vous−mêmes, ou faire une autre requête.

Ou alors, vous pouvez utiliser la clause `ROLLUP` , qui fournit les deux niveaux d'analyse dans la même requête. En ajoutant l'option `WITH ROLLUP` à la clause `GROUP BY` , la requête va produire une autre ligne, avec le grand total de toutes les années :

```
mysql> SELECT year, SUM(profit) FROM sales GROUP BY year WITH ROLLUP;
+------+-------------+
| year | SUM(profit) |
+------+-------------+
| 2000 |        4525 |
| 2001 |        3010 |
| NULL |        7535 |
+------+-------------+
```

La ligne du grand total est identifiée par la valeur `NULL` dans la colonne `year` .

`ROLLUP` a des effets plus complexes lorsqu'il y a plusieurs colonnes dans la clause `GROUP BY` . Dans ce cas, il a y un changement de valeur pour toutes sauf la dernière colonne de groupement, et la requête va produire les super−agrégats.

Par exemple, sans la clause `ROLLUP` , le résumé des ventes de la table `sales` basé sur l'année `year` , le pays `country` et le produit `product` peut ressembler à ceci :

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
| 2000 | USA     | Calculator |          75 |
| 2000 | USA     | Computer   |        1500 |
| 2001 | Finland | Phone      |          10 |
| 2001 | USA     | Calculator |          50 |
| 2001 | USA     | Computer   |        2700 |
| 2001 | USA     | TV         |         250 |
+------+---------+------------+-------------+
```

Le résultat indique les valeurs résumées pour chaque triplet année/pays/produit. Si nous ajoutons la clause `ROLLUP` , la requête produit plusieurs nouvelles lignes :

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product WITH ROLLUP;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | Finland | NULL       |        1600 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
| 2000 | India   | NULL       |        1350 |
| 2000 | USA     | Calculator |          75 |
| 2000 | USA     | Computer   |        1500 |
| 2000 | USA     | NULL       |        1575 |
| 2000 | NULL    | NULL       |        4525 |
| 2001 | Finland | Phone      |          10 |
| 2001 | Finland | NULL       |          10 |
| 2001 | USA     | Calculator |          50 |
| 2001 | USA     | Computer   |        2700 |
| 2001 | USA     | TV         |         250 |
| 2001 | USA     | NULL       |        3000 |
| 2001 | NULL    | NULL       |        3010 |
```

```
| NULL | NULL    | NULL       |        7535 |
+------+---------+------------+-------------+
```

Pour cette requête, ajouter `ROLLUP` fait que la requête ajoute les résumés de quatre niveaux d'analyse, et non pas un seul. Voici comment interpréter le résultat de la clause `ROLLUP` :

- Après chaque jeu de ligne sur les produits, pour une année et un pays donnée, un résumé est ajouté, indiquant le total de tous les produits. Ces lignes voient leur colonne `product` contenir la valeur `NULL` .
- Après chaque jeu de ligne couvrant une année particulière, une nouvelle ligne est ajoutée pour afficher le total de tous les pays et produits, pour cette année la. Ces lignes voient leurs colonnes `country` et `products` contenir `NULL` .
- Finalement, suivant toutes les autres lignes, un résumé général est produit, avec le grand total de toutes les années, pays et produits. Cette ligne contient la valeur `NULL` pour toutes les colonnes `year` , `country` et `products` .

Autres considérations avec `ROLLUP`

Voici quelques comportements spécifiques de MySQL et son implémentation de `ROLLUP` :

Lorsque vous utilisez `ROLLUP` , vous ne pouvez pas utiliser de clause `ORDER BY` pour trier les résultats. En d'autres termes, `ROLLUP` et `ORDER BY` sont mutuellement exclusives. Toutefois, vous avec toujours le contrôle sur l'ordre de tri avec la clause `GROUP BY` . Vous pouvez utiliser explicitement les mots `ASC` et `DESC` avec les colonnes listées dans `GROUP BY` pour spécifier les ordres de tri des colonnes individuelles. Les lignes de résumés de `ROLLUP` apparaissent toujours après les lignes pour lesquelles ils sont calculés, quelque soit le tri.

La clause `LIMIT` peut être utilisée pour restreindre le nombre de lignes retournées au client. `LIMIT` s'applique après `ROLLUP` , et la limite s'appliquera aux lignes ajoutées par `ROLLUP` . Par exemple :

```
 mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product WITH ROLLUP
    -> LIMIT 5;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | Finland | NULL       |        1600 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
+------+---------+------------+-------------+
```

Notez qu'utiliser `LIMIT` avec `ROLLUP` peut conduire à des résultats plus difficiles à interpréter, car vous avez moins de contexte pour comprendre les résumés.Les indicateurs `NULL` de chaque super−agrégat sont produits lorsque la ligne est envoyée au client. Le serveur recherche les colonnes citées dans la clause `GROUP BY` , en les prenant la plus à gauche, dont la valeur change. Toute colonne du jeu de résultat dont le nom ne correspond pas lexicalement à un de ces noms, verra sa valeur être `NULL` . Si vous spécifiez un groupement par numéro de colonne, le serveur identifiera aussi les colonnes qui devront recevoir `NULL` .

Comme les valeurs `NULL` des résumés sont placées dans le résultat aussi tard durant le traitement de la requête, nous ne pouvons pas les tester comme étant des valeurs `NULL` provenant de la requête elle−même. Par exemple, vous ne pourrez pas ajouter `HAVING product IS NULL` pour éliminer certains résumés qui ne vous intéressent pas.

D'un autre coté, les valeurs `NULL` apparaissent comme des valeurs `NULL` du coté du client, et peuvent être repérées en tant que telles par le client MySQL.

13.9.3 GROUP BY avec les champs cachés

MySQL a étendu l'utilisation de la clause `GROUP BY` . Vous pouvez utiliser des colonnes ou des calculs de l'expression `SELECT` qui n'apparaissent pas dans la clause `GROUP BY` . Cela se dit **n'import quelle valeur pour ce groupe** . Vous pouvez utiliser cela pour améliorer les performances en évitant les tris ou les regroupements inutiles de valeurs. Par exemple, vous n'avez pas besoin de faire un regroupement par nom de client `customer.name` dans la requête suivante :

```
mysql> SELECT order.custid,customer.name,MAX(payments)
    ->        FROM order,customer
    ->        WHERE order.custid = customer.custid
    ->        GROUP BY order.custid;
```

En SQL standard, vous devriez ajouter la colonne `customer.name` à la clause `GROUP BY` . Avec MySQL, ce nom est redondant si vous n'utilisez pas le mode ANSI. **N'utilisez pas cette fonctionnalité** si les colonnes que vous omettez dans la clause `GROUP BY` ne sont pas unique dans le groupe!! Vous auriez des résultats inattendus!

Dans certains cas, vous pouvez utiliser `MIN()` et `MAX()` pour obtenir une valeur spécifique d'une colonne, même si cette valeur n'est pas unique. L'exemple suivant donne la valeur de la colonne `column` issue de la ligne contenant la plus petit valeur de la colonne `sort` :

```
SUBSTR(MIN(CONCAT(RPAD(sort,6,' '),column)),7)
```

La ligne contenant la plus grande valeur d'un certain champ par rapport à un groupe .

Notez que si vous utilisez MySQL version 3.22 ou plus ancien, ou si vous essayez de suivre la norme SQL−99, vous ne pouvez pas utiliser les expressions dans `GROUP BY` ou `ORDER BY` . Vous pouvez contourner cette limitation en utilisant un alias pour l'expression :

```
mysql> SELECT id,FLOOR(value/100) AS val FROM tbl_name
    ->        GROUP BY id,val ORDER BY val;
```

En MySQL version 3.23, vous pouvez faire :

```
mysql> SELECT id,FLOOR(value/100) FROM tbl_name ORDER BY RAND();
```

14 Syntaxe des commandes SQL

Ce chapitre décrit la syntaxe des commandes que MySQL supporte.

14.1 Manipulation de données : SELECT , INSERT , UPDATE , DELETE

14.1.1 Syntaxe de DELETE

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM table_name
       [WHERE where_definition]
       [ORDER BY ...]
       [LIMIT row_count]
```

Syntaxe multi−tables :

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] table_name[.*] [, table_name[.*] ...]
      FROM table-references
       [WHERE where_definition]
```

ou :

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
      FROM table_name[.*] [, table_name[.*] ...]
      USING table-references
       [WHERE where_definition]
```

`DELETE` efface les enregistrements de `nom_de_table` qui satisfont la condition donnée par `clause_where` , et retourne le nombre d'enregistrements effacés.Si vous exécutez un `DELETE` sans clause `WHERE` , tous les enregistrements sont effacés. Si vous le faites en mode `AUTOCOMMIT` cela aura le même effet qu'un `TRUNCATE` . Syntaxe de `TRUNCATE` .

Avec MySQL 3.23, `DELETE` sans clause `WHERE` retournera zéro comme nombre d'enregistrements affectés.

Si vous voulez vraiment savoir combien d'enregistrements ont été effacés quand vous videz une table, et que vous êtes prêts à souffrir d'un léger ralentissement, vous pouvez utiliser une requête `DELETE` de ce genre :

```
mysql> DELETE FROM nom_de_table WHERE 1>0;
```

Notez que c'est plus lent que `DELETE FROM nom_de_table` sans clause `WHERE` , parce que cela efface un enregistrement à la fois.Si vous effacez des lignes contenant la valeur maximum d'une colonne `AUTO_INCREMENT` , la valeur sera réutilisée pour par une table `ISAM` ou `BDB` , mais pas pour une table `MyISAM` ou `InnoDB` . Si vous effacez toutes les lignes dans une table avec une commande `DELETE FROM tbl_name` (avec une clause `WHERE`) avec le mode `AUTOCOMMIT` , la séquence redémarrer à zéro pour tous les types de table sauf `InnoDB` et, depuis MySQL 4.0, `MyISAM` . Il y a des exceptions à ce comportement pour les tables `InnoDB` , qui sont présentées dans la section Comment les colonnes `AUTO_INCREMENT` fonctionnent avec `InnoDB` .

Pour les tables `MyISAM` et `BDB` , vous pouvez spécifier une autre colonne `AUTO_INCREMENT` dans une clé multi−colonnes. Dans ce cas, la réutilisation des clés à partir de la fin de la séquence se fait aussi pour les tables `MyISAM` . Utiliser `AUTO_INCREMENT` .

La commande `DELETE` supporte les clauses suivantes :

- Si vous spécifiez le mot clé `LOW_PRIORITY` , l'exécution de la commande `DELETE` est repoussée jusqu'à ce qu'aucun client ne soit en train de lire la table.

- Pour les tables `MyISAM` , si vous spécifiez l'option `QUICK` , le moteur de stockage ne compacte pas les index durant l'effacement, ce qui peut accélérer certains effacements.
- L'option `IGNORE` fait que MySQL ignore les erreurs durant le traitement des lignes. Les erreurs rencontrées durant la phase d'analyse sont traitées comme d'habitude. Les erreurs qui sont ignorées grâce à cette options sont listées comme des alertes. Cette option a été ajoutée en MySQL 4.1.1.

La vitesse d'exécution des opérations de suppressions peut être affectées par les facteurs présentés dans la section Vitesse des requêtes `DELETE` .

Dans les tables de type `MyISAM` , les enregistrements effacés sont maintenus dans une liste liée et les requêtes `INSERT` suivantes réutilisent les vieux emplacements. Pour recouvrir l'espace inutilisé ou réduire la taille des fichiers, utilisez la commande `OPTIMIZE TABLE` ou l'utilitaire `myisamchk` pour réorganiser les tables. `OPTIMIZE TABLE` est plus simple, mais `myisamchk` est plus rapide. Voyez Syntaxe de `OPTIMIZE TABLE` et Optimisation de table .

La clause spécifique MySQL `LIMIT row_count` de la commande `DELETE` indique au serveur le nombre maximal de ligne à supprimer avant de rendre le contrôle au client. Elle peut être utilisée pour s'assurer qu'une commande `DELETE` ne prend pas trop de temps. Vous pouvez simplement répéter la commande `DELETE` jusqu'à ce que le nombre de lignes effacées est inférieure à la valeur de `LIMIT` .

Si la commande `DELETE` inclut la clause `ORDER BY` , les lignes sont effacées dans l'ordre spécifiée par cette clause. Elle n'est vraiment utilise que lorsqu'elle est couplée avec la clause `LIMIT` . Par exemple, la commande suivante applique la condition `WHERE` , trie les lignes avec la colonne `timestamp` , et efface la ligne la plus ancienne :

```
 DELETE FROM somelog
WHERE user = 'jcole'
ORDER BY timestamp
LIMIT 1
```

`ORDER BY` peut être utilisée avec `DELETE` depuis MySQL version 4.0.0.Depuis MySQL version 4.0, vous pouvez spécifier plusieurs tables dans la commande `DELETE` , pour effacer des lignes dans plusieurs tables, en fonction d'une condition de liaison. Cependant, vous ne pouvez pas utiliser les clauses `ORDER BY` et `LIMIT` dans une suppression `DELETE` multi−tables.

La première syntaxe de `DELETE` multi−table est supportée depuis MySQL 4.0.0. La deuxième syntaxe de `DELETE` multi−table est supportée depuis MySQL 4.0.2. La partie `table_references` liste les tables impliquées dans la jointure. Sa syntaxe est décrite dans la section Syntaxe de `JOIN` .

L'idée est que seul les lignes concordante dans les tables énumérées avant le `FROM` ou avant la clause `USING` sont effacés. Le but est de pouvoir effacer des lignes de plusieurs tables en même temps tout en ayant d'autres tables pour les recherches.

Le code `.*` après les noms de tables n'est présent que pour assurer la compatibilité avec `Access` :

```
DELETE t1,t2 FROM t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
```

ou :

```
DELETE FROM t1,t2 USING t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
```

Dans les cas précédents, nous n'avons supprimé les lignes correspondantes que dans les tables `t1` et `t2` .Les exemples ci−dessus présente des jointures internes, en utilisant l'opérateur virgule, mais les `DELETE` multi−tables peuvent utiliser n'importe quel type de jointure qu'une commande `SELECT` accepte, comme un `LEFT JOIN` .

La syntaxe autorise `.*` après le nom de la table pour assurer la compatibilité avec `Access` .

Si vous utilisez une commande `DELETE` multi−tables avec des tables `InnoDB` pour lesquelles il y a des contraintes de clés étrangères, l'optimiseur MySQL risque de traiter les tables dans un ordre qui diffère de celui des relations parent/enfant de la clé. Dans ce cas, la commande échouera, et s'annulera. Pour résoudre ce problème, effacez les lignes tables par table, et utilisez les fonctionnalités `ON DELETE` que `InnoDB` fournit pour que les autres tables soient correctement traitées.

Note : en MySQL 4.0, vous devez utiliser le véritable nom de table. En MySQL 4.1, vous devez utiliser l'alias éventuel, lorsque vous nommez la table :

En MySQL 4.0 :

```
DELETE test FROM test AS t1, test2 WHERE ...
```

En MySQL 4.1 :

```
DELETE t1 FROM test AS t1, test2 WHERE ...
```

La raison qui nous a poussé à ne pas faire ce changement en version 4.0, est la compatibilité ascendante avec les vieilles applications 4.0, qui utilisent la vieille syntaxe.

14.1.2 Syntaxe de DO

```
DO expression, [expression, ...]
```

Exécute l'expression mais ne retourne aucun résultat. C'est un alias de `SELECT expression, expression` , mais il a l'avantage d'être plus rapide quand on n'a pas besoin du résultat.

Cela s'avère très utile avec les fonctions qui ont des effets secondaires, comme `RELEASE_LOCK` .

14.1.3 Syntaxe de HANDLER

```
 HANDLER tbl_name OPEN [ AS alias ]
HANDLER tbl_name READ index_name &#123; = | >= | <= | < &#125; (value1,value2,...)
    [ WHERE ... ] [LIMIT ... ]
HANDLER tbl_name READ index_name &#123; FIRST | NEXT | PREV | LAST &#125;
    [ WHERE ... ] [LIMIT ... ]
HANDLER tbl_name READ &#123; FIRST | NEXT &#125;
    [ WHERE ... ] [LIMIT ... ]
HANDLER tbl_name CLOSE
```

La commande `HANDLER` fournit un accès direct à l'interface de gestion de la table `MyISAM` .

La première forme de `HANDLER` ouvre la table, la rendant accessible via la requête `HANDLER ... READ` qui la suit. Cette objet table n'est pas partagé par les autres threads et ne sera refermé que si le thread appelle `HANDLER nom_de_table CLOSE` ou que celui ci se termine.

La seconde forme récupère une ligne (ou plus, à spécifier dans la clause `LIMIT`) où l'index spécifié remplit les conditions et où la clause `WHERE` est répondue. Si l'index se compose de plusieurs parties, (s'étend sur plusieurs colonnes) les valeurs sont spécifiées dans une liste séparée par des virgules, fournir des valeurs pour quelques premières colonnes est possible. Par exemple :

```
 HANDLER ... index_name = (col_a_val,col_b_val,col_c_val) ...
HANDLER ... index_name = (col_a_val,col_b_val) ...
HANDLER ... index_name = (col_a_val) ...
```

La troisième forme récupère une ligne (ou plus, à spécifier dans la clause `LIMIT`) de la table dans l'ordre de l'index, qui répond à la clause `WHERE` .La quatrième forme (sans spécifications relatives à l'index) récupère une ligne (ou plus, à spécifier dans la clause `LIMIT`) de la table dans un ordre naturel des lignes (comme stocké dans le fichier de données) qui correspond à la condition `WHERE` . C'est plus rapide que `HANDLER nom_de_table READ nom_index` quand une lecture entière de la table est requise. Syntaxe de `SELECT` .

HANDLER ... CLOSE ferme une table qui a été ouverte avec HANDLER ... OPEN .

Note : pour utiliser l'interface HANDLER avec la clé primaire d'une table PRIMARY KEY , utilisez l'identifiant entre guillemets obliques `PRIMARY` :

```
HANDLER tbl_name READ `PRIMARY` > (...);
```

HANDLER est en quelque sorte une commande bas−niveau. Par exemple, elle ne propose pas de consistance. En clair, HANDLER ... OPEN ne se base **pas** sur une image de la table, et ne verrouille **pas** la table. Cela signifie qu'après l'exécution d'une requête HANDLER ... OPEN , les données de la table peuvent être modifiées (par ce ou un autre thread) et ces modifications peuvent apparaître partiellement dans les lectures de HANDLER ... NEXT ou HANDLER ... PREV .Les raisons d'utiliser cette interface plutôt que les commandes MySQL usuelles sont :

- Plus rapide qu'un SELECT car :
    - Un pointeur sur table dédié est alloué au thread dans HANDLER open .
    - Il y a moins de traitements.
    - Pas de pertes de temps en optimisation ou vérifications de requêtes.
    - La table utilisée n'a pas besoin d'être verrouillée entre deux requêtes de gestion.
    - L'interface de gestion n'a pas à fournir une vue consistante des données (par exemple, les lectures corrompues sont autorisées), ce qui permet au gestionnaire d'effectuer des optimisations que SQL ne permet pas.
- Cela facilite le port des applications qui utilisent l'interface ISAM pour MySQL.
- Cela permet de traverse plus facilement la base de données qu'avec SQL (dans certains cas, cette opération est impossible avec SQL). L'interface de gestion amène une façon plus naturelle de manipuler les données lorsque vous travaillez avec des applications qui proposent une interface interactive entre l'utilisateur et la base de données.

14.1.4 Syntaxe de INSERT

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES (&#123;expr | DEFAULT&#125;,...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

ou :

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name
    SET col_name=&#123;expr | DEFAULT&#125;, ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

ou :

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

INSERT insère une nouvelle ligne dans une table existante. La syntaxe INSERT ... VALUES insère une ligne à partir de valeurs explicitement fournies. La syntaxe INSERT ... SELECT insère des valeurs à partir d'une autre table. La syntaxe INSERT ... VALUES avec plusieurs valeurs est supportées à partir de MySQL Version 3.22.5 ou supérieure. la syntaxe nom_colonne=expression est supportée à partir de la version 3.22.10 de MySQL.

INSERT ... SELECT est présenté plus en détails : Syntaxe de INSERT ... SELECT .

nom_de_table est le nom de la table dans laquelle les valeurs seront insérées. La liste de noms de colonne ou la clause SET indiquent les colonnes qui seront affectées:

- La liste des noms de colonnes ou la clause SET indique explicitement les colonnes utilisées.
- Si vous ne spécifiez pas de liste de colonnes avec INSERT ... VALUES ou INSERT ... SELECT , les valeurs

pour toutes les colonnes doivent être fournies dans la clause `VALUES()` ou par la commande `SELECT` . Si vous ne connaissez pas l'ordre des colonnes, utilisez la commande `DESCRIBE nom_de_table` pour le connaître.

Les valeurs des colonnes peuvent être spécifiées de plusieurs façons :

- A chaque fois qu'on ne donne pas explicitement une valeur pour une colonne, celle prend la valeur par défaut. Par exemple, si on définit une liste de colonnes qui ne compte pas toutes les colonnes de la tables, toutes les colonnes qui ne sont pas nommées prendront leur valeur par défaut. La définition de la valeur par défaut se fait avec Syntaxe de `CREATE TABLE` .MySQL a toujours une valeur par défaut pour chaque champs. C'est obligatoire pour MySQL pour pouvoir fonctionner aussi bien avec des tables supportant les transactions qu'avec des tables ne les supportant pas.

  Nous pensons que le contrôle du contenu des champs devrait être fait pas l'application et non par le serveur de base de données.

  Note : si vous voulez que les commandes `INSERT` génèrent une erreur si vous ne spécifiez pas explicitement de valeur pour toutes les colonnes qui requièrent des valeurs non− `NULL` , vous pouvez auss configurer MySQL avec l'option `DONT_USE_DEFAULT_FIELDS` . Ce comportement n'est pas disponible si vous compilez MySQL depuis le source. Les options habituelles de `configure` .
- Vous pouvez utiliser le mot clé `DEFAULT` pour donner explicitement à une colonne sa valeur par défaut. Cette fonctionnalité a été ajoutée en MySQL version 4.0.3. Cela rend plus simple l'écriture de commandes `INSERT` lors de l'assignation de quelques colonnes, sans écrire de valeurs `VALUES` incomplètes. Sinon, il faut écrire la liste des colonnes utilisées pour chaque valeur de la liste `VALUES` .
- Si la liste de colonnes et de valeurs `VALUES` sont vides, `INSERT` crée une ligne pour chaque colonne avec sa valeur par défaut :

```
mysql> INSERT INTO tbl_name () VALUES();
```
- Une `expression` peut faire référence à n'importe quelle colonne qui a été définie précédemment dans une liste de valeurs. Par exemple, on peut dire ceci :

```
mysql> INSERT INTO nom_de_table (col1,col2) VALUES(15,col1*2);
```

  Mais vous ne pouvez pas faire cela, car la valeur de `col1` fait référence à `col2` , qui est assigné après `col1` :

```
mysql> INSERT INTO nom_de_table (col1,col2) VALUES(col2*2,15);
```

Les commandes `INSERT` supportent les options suivantes :

- Si vous spécifiez l'option `DELAYED` , le serveur met la ligne ou les lignes à insérer dans un buffer, et le client qui a émis la commande `INSERT DELAYED` est immédiatement libéré. Si la table est occupée, le serveur conserve les lignes. Lorsque la table se libère, il va insérer les lignes, tout en vérifiant périodiquement s'il n'y a pas de lectures dans la table. Si une lecture arrive, l'insertion est suspendue jusqu'à la prochaine libération. Syntaxe de `INSERT DELAYED` .
- Si on spécifie le mot `LOW_PRIORITY` , l'exécution de `INSERT` sera retardé jusqu'à ce qu'il n'y ait plus de clients qui lisent la table. Dans ce cas le client doit attendre jusqu'à la fin de l'opération d'insertion, ce qui peut prendre beaucoup de temps si la table est fréquemment accédée. C'est la grande différence avec `INSERT DELAYED` , qui laisse le client continuer tout de suite. Syntaxe de `INSERT DELAYED` . On peut remarquer que `LOW_PRIORITY` ne devrait en principe pas être utiliser avec des tables de type `MyISAM` , étant donné que celles−ci n'autorisent pas les insertions simultanées. Tables `MyISAM` .
- Si on spécifie le mot `IGNORE` dans un `INSERT` avec les valeurs de plusieurs lignes, chaque ligne qui qui ferait doublon avec une clé `PRIMARY` ou `UNIQUE` existante dans la table sera ignoré et ne sera pas insérée. Si on ne spécifie pas `IGNORE` , l'insert est abandonné si quelque ligne que ce soit fait doublon avec une clé existante. La fonction `mysql_info()` de l'API C permet de savoir combien de lignes ont été insérées dans la table.

Si vous spécifiez la clause `ON DUPLICATE KEY UPDATE` (nouveau en MySQL 4.1.0), et qu'une ligne insérée engendre un doublon pour une clé `PRIMARY` ou `UNIQUE` , une commande `UPDATE` sera faite à la place de l'insertion. Par exemple, les commandes ont le même effet :

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3)
   -> ON DUPLICATE KEY UPDATE c=c+1;

mysql> UPDATE table SET c=c+1 WHERE a=1;
```

Note : si la colonne `b` est aussi unique, la commande `UPDATE` sera réécrite telle que

```
mysql> UPDATE table SET c=c+1 WHERE a=1 OR b=2 LIMIT 1;
```

Si `a=1 OR b=2` trouve plusieurs lignes, uniquement **une** ligne sera mise à jour! En général, il faut éviter d'utiliser la clause `ON DUPLICATE KEY` sur des tables avec des clés `UNIQUE` multiples.Depuis MySQL version 4.1.1, on peut utiliser la fonction `VALUES(col_name)` pour faire référence à la valeur de la colonne dans la clause `INSERT` d'une commande `INSERT ... UPDATE` : c'est la valeur qui serat inserée s'il n'y a pas de conflit de clé. Cette valeur est particulièrement utilise dans les commandes `INSERT ... UPDATE` et retourne `NULL` sinon.

Exemple :

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
   -> ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

La commande ci−dessus est identique à :

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3)
   -> ON DUPLICATE KEY UPDATE c=3;
mysql> INSERT INTO table (a,b,c) VALUES (4,5,6)
   -> ON DUPLICATE KEY UPDATE c=9;
```

Lors de l'utilisation de `ON DUPLICATE KEY UPDATE` , l'option `DELAYED` est ignorée.

Vous pouvez trouver la valeur utilisée pour une colonne `AUTO_INCREMENT` en utilisant la fonction `LAST_INSERT_ID()` . Depuis l'interface C, utilisez la fonction `mysql_insert_id()` . Cependant, notez que les deux fonctions ne se comportement pas de la la même façons dans toutes les circonstances. Le comportement des commandes `INSERT` avec les colonnes `AUTO_INCREMENT` sont décrites dans la section Fonctions d'informations et `mysql_insert_id()` .

Si vous utilisez une commande `INSERT ... VALUES` avec plusieurs listes de valeurs ou `INSERT ... SELECT` , la commande retourne les informations sous ce format :

```
Records: 100 Duplicates: 0 Warnings: 0
```

`Records` indique le nombre de ligne qui ont été traitées par cette commande. Ce n'est pas forcément le nombre de ligne insérées. `Duplicates` peut être non−nulle. `Duplicates` indique le nombre de lignes qui n'ont pas peut être insérées pour cause de conflit avec une clé unique existante. `Warnings` indique le nombre de tentatives d'inserts de valeurs dans une colonne qui ont généré des problèmes. Les `Warnings` peuvent apparaître dans les conditions suivantes:

- Insertion de `NULL` dans une colonne déclarée `NOT NULL` . Pour les commandes d'insertions multiples `INSERT` ou les commandes `INSERT ... SELECT` , la colonne prend la valeur par défaut adaptée au type de colonne. C'est `0` pour les types numériques, la chaîne vide pour les textes et la valeur ``zéro'' pour les types temporels
- Enregistrement dans une colonne numérique d'une valeur qui déborde de la taille de la colonnes. Cette valeur a été tronquée à l'extrémité la plus adaptée de la colonne.
- Attribution à une colonne numérique d'une valeur telle que `'10.34 a'` . Celle valeur refusée est séparée, et la partie numérique résultante est insérée. Si cette valeur n'a pas une valeur numérique sensée, la valeur `0` est insérée.
- L'insertion d'une chaîne dans une colonne `CHAR` , `VARCHAR` , `TEXT` , ou `BLOB` qui dépasse la taille

maximale de la colonne. La valeur est tronquée à la taille maximale de la colonne.
- L'insertion d'une valeur illégale pour une colonne de type date ou time. La colonne est alors enregistrée avec la valeur de zero appropriée pour le type.

Si vous utilisez l'interface C, la chaîne d'information peut être obtenue en invoquant la fonction `mysql_info()` . `mysql_info()` .

14.1.4.1 Syntaxe de INSERT ... SELECT

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] nom_de_la_table [(liste des colonnes)] SELECT ...
```

La requête `INSERT ... SELECT` permet de rapidement insérer dans une table un grand nombre de lignes d'une ou plusieurs autres tables.

```
 INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID FROM tblTemp1 WHERE
tblTemp1.fldOrder_ID > 100;
```

Les conditions suivantes s'appliquent à la requête `INSERT ... SELECT` :

- Avant MySQL version 4.0.1, `INSERT ... SELECT` opérait implicitement en mode `IGNORE` . Depuis MySQL version 4.0.1, vous devez spécifier le mode `IGNORE` explicitement, pour ignorer les lignes qui causeront des erreurs de doublons pour les index uniques.
- N'utilisez pas `DELAYED` avec `INSERT ... SELECT` .
- Avant MySQL version 4.0.14, la table de destination de la requête `INSERT` ne peut apparaître dans la clause `FROM` de la partie `SELECT` de la requête car il est interdit par le ANSI SQL de lire la table dans laquelle on est en train de faire un insert. (le problème est que le `SELECT` pourrait trouver des enregistrements qui aurait été insérés auparavant dans la même exécution. L'utilisation de "subselect" peut rendre la situation confuse !)
- Les colonnes `AUTO_INCREMENT` fonctionnent comme d'habitude.
- Pour s'assurer que les journaux des modifications ou les journaux binaires puissent être utilisés pour re−créer les tables originales, MySQL n'autorise pas les inserts concurrents pendant `INSERT ... SELECT` .

Il est bien sûr possible d'utiliser `REPLACE` à la place de `INSERT` pour remplacer les anciennes lignes.

14.1.4.2 Syntaxe de INSERT DELAYED

```
INSERT DELAYED ...
```

L'option `DELAYED` de la commande `INSERT` est une option spécifique à MySQL très utile si vos clients ne peuvent pas attendre que `INSERT` se termine. C'est un problème fréquent quand on utilise MySQL pour des logs, mais aussi quand on utilise souvent des commandes `SELECT` ou `UPDATE` qui prennent beaucoup de temps. `DELAYED` a été ajouté à MySQL dans la version 3.22.15. C'est une extension de MySQL au ANSI SQL 92.En utilisant `INSERT DELAYED` , le client reçoit immédiatement un aquitement, et la ligne sera insérée quand la table ne sera plus utilisée par un autre thread.

Un autre avantage de `INSERT DELAYED` est que les insertions des clients sont regroupés, et écrits d'un seul bloc. C'est beaucoup plus rapide que de faire des insertions séparés.

Il y a quelques contraintes à l'utilisation de `DELAYED` :

- `INSERT DELAYED` ne fonctionne qu'avec les tables `MyISAM` et `ISAM` . Pour les tables `MyISAM` , s'il n'y a plus de blocs libres au milieu du fichier de données, les `SELECT` et `INSERT` simultanés sont supportés. Dans ces circonstances, vous n'aurez que très rarement besoin de `INSERT DELAYED` avec `MyISAM` .

Tables `MyISAM` .
- `INSERT DELAYED` doit être utilisé uniquement avec les commandes `INSERT` qui spécifie une liste de valeur. C'est le cas depuis MySQL 4.0.18. Le serveur ignore `DELAYED` pour les commandes `INSERT DELAYED ... SELECT` .
- Le serveur ignore `DELAYED` dans les commandes `INSERT DELAYED ... ON DUPLICATE UPDATE` .
- Comme la commande s'exécute immédiatement, sans que la ligne ne soit insére, vous ne pouvez pas utiliser `LAST_INSERT_ID()` pour lire la valeur que la colonne `AUTO_INCREMENT` va générer.
- Les lignes `DELAYED` ne sont visibles par les commandes `SELECT` que lorsqu'elles ont été réellement insérées.

Actuellement, les lignes en attente sont uniquement stockées en mémoire tant qu'elle ne sont pas insérées dans la table. Cela signifie que si on tue `mysqld` violemment, (`kill -9`) ou si `mysqld` meurt accidentellement, toutes les lignes en attente qui n'auront pas été écrites sur le disque seront perdues !

Les paragraphes suivants décrivent en détail ce qu'il se passe quand on utilise l'option `DELAYED` dans une requête `INSERT` ou `REPLACE` . Dans cette description, ``thread'' est un thread qui reçoit une commande `INSERT DELAYED` ans ``handler'' est un thread qui gère toutes les opérations de `INSERT DELAYED` pour une table donnée.

- Quand un thread exécute une opération `DELAYED` sur une table, un thread de gestion est créé pour exécuter toutes les opérations `DELAYED` pour cette table − si ce thread de gestion n'existe pas.
- Le thread vérifie que a déjà reçu un verrou `DELAYED` ; sinon, il dit au thread de gestion de le faire. le verrou `DELAYED` peut être obtenu même si d'autres threads ont des verrous `READ` ou `WRITE` sur la table. Cependant le gestionnaire attendra que tous les verrous `ALTER TABLE` ou `FLUSH TABLES` soient finis pour s'assurer que la structure de la table est à jour.
- Le thread exécute une opération `INSERT` , mais plutôt que d'écrire la ligne dans la table, il va placer une copie de la ligne finale dans une file d'attente gérée par le thread de gestion. Le programme client est avertit de toutes les erreurs de syntaxe.
- Le client ne peut pas faire de rapport sur le nombre de duplicata ou sur la valeur de `AUTO_INCREMENT` de la ligne enregistrée; il ne peut pas les obtenir du serveur, car le `INSERT` est validé avant que l'opération d'insert n'ait été effectuée. Si vous utilisez l' API C, la fonction `mysql_info()` ne retourne pas de valeur intéressante, pour la même raison.
- Le journal de modification est mis à jour par le thread de gestion au moment où la ligne est insérée dans la table. Si plusieurs lignes sont insérées en même temps, le journal des modifications est mis à jour quand la première ligne est insérée.
- Une fois que toutes les lignes `delayed_insert_limit` sont écrites, le gestionnaire vérifie si des requêtes `SELECT` sont en attente, et si c'est le cas, il leur permet de s'exécuter avant de continuer.
- Quand le thread de gestion n'a plus de ligne dans sa file, la table est déverrouillée. Si aucun `INSERT DELAYED` n'est reçu avant `delayed_insert_timeout` secondes, le gestionnaire s'arrête.
- Si plus de `delayed_queue_size` lignes sont déjà en attente d'un gestionnaire de file donné, le thread qui demande le `INSERT DELAYED` doit attendre qu'il y ait une place dans la file. Cela permet d'être sûr que `mysqld` n'utilisera pas toute la mémoire pour la mémoire des files d'attente d'insertions retardés.
- Le thread de gestion apparaîtra dans la liste des processus de MySQL avec `delayed_insert` dans la colonne `Command` . Il sera tué si on exécute une commande `FLUSH TABLES` ou si on le tue avec `KILL thread_id` . Cependant, il commencera par stocker toutes les lignes en attente dans la table avant de sortir. Pendant ce temps, il n'acceptera aucune commande `INSERT` d'aucun autre thread. Si on exécute une commande `INSERT DELAYED` après cela, un nouveau thread de gestion sera créé.Il faut noter que les commandes `INSERT DELAYED` ont une plus grande priorité que les commandes `INSERT` normales si un gestionnaire de `INSERT DELAYED` existe déjà! les autres commandes de modification devront attendre que la file d'attente de `INSERT DELAYED` soit vide,

que quelqu'un tue le gestionnaire (avec KILL thread_id), ou que quelqu'un exécute FLUSH TABLES ..
- Les variables suivantes fournissent des informations relatives à la commande INSERT DELAYED :

Variable	Signification
Delayed_insert_threads	Nombre de threads de gestion
Delayed_writes	Nombre de lignes écrites avec INSERT DELAYED
Not_flushed_delayed_rows	Nombre de lignes en attente d'être écrites.

On peut voir ces variables avec la commande SHOW STATUS ou en exécutant la commande mysqladmin extended-status .

Il faut noter que INSERT DELAYED est plus lent qu'un INSERT normal si la table n'est pas utilisée. L'utilisation d'un thread de gestion séparé pour chaque table sur lesquelles on utilise INSERT DELAYED rajoute également une surcharge au serveur. Ce qui signifie qu'il vaut mieux utiliser INSERT DELAYED uniquement quand c'est vraiment nécessaire!

14.1.5 Syntaxe de LOAD DATA INFILE

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
   [REPLACE | IGNORE]
   INTO TABLE tbl_name
   [FIELDS
       [TERMINATED BY '\t']
       [[OPTIONALLY] ENCLOSED BY '']
       [ESCAPED BY '\\' ]
   ]
   [LINES
        [STARTING BY '']
       [TERMINATED BY '\n']
   ]
   [IGNORE number LINES]
   [(col_name,...)]
```

La commande LOAD DATA INFILE lit les lignes dans un fichier texte et les insère à très grande vitesse. Pour plus d'informations sur l'efficacité des commandes INSERT comparativement à LOAD DATA INFILE et pour accélérer les commandes LOAD DATA INFILE , voyez Vitesse des requêtes INSERT . Vous pouvez aussi charger des fichiers de données en utilisant l'utilitaire mysqlimport ; Il opère en envoyant la commande LOAD DATA INFILE au serveur. L'option --local fais que mysqlimport lit les fichiers de données chez le client. Vous pouvez spécifier l'option --compress pour avoir de meilleurs performances avec les connexions lentes si le client et le serveur supportent le protocole compressé. mysqlimport , importer des données depuis des fichiers texte .

Si vous spécifiez le mot clef LOW_PRIORITY , l'exécution de la commande LOAD DATA est ajournée jusqu'à ce qu'aucun client ne lise plus de la table.

Si vous spécifiez le mot clef CONCURRENT avec un table au format MyISAM , les autres threads pourront accéder à la table durant l'exécution de la commande LOAD DATA . L'utilisation de cette option ralentira un peu les performances de LOAD DATA même si aucun thread n'utilise la table en même si aucun autre thread n'accède à la table en même temps.

Si le mot clé LOCAL est spécifié, il est interprété en suivant les règles suivantes :

- Si LOCAL est spécifié, le fichier est lu par le programme client, et envoyé vers l'hôte.
- Si LOCAL n'est pas spécifiée, le fichier doit être sur le serveur hôte, et sera lu directement par le serveur.

LOCAL est disponible depuis MySQL 3.22.6 ou plus récent.

Pour des raisons de sécurité, lorsque les fichiers sont lus sur le serveur, ils doivent se trouver dans le répertoire de la base de données courante, ou bien être lisible par tous. Pour utiliser la commande LOAD DATA INFILE sur des fichiers du serveur, vous devez avoir le droit de FILE sur le serveur. Droits fournis par MySQL .

Utiliser `LOCAL` est plus lent que de laisser le serveur accéder directement aux fichiers, car le contenu du fichier doit être envoyé via le réseau au serveur. D'un autre coté, vous n'aurez pas besoin de droits de `FILE` pour faire un chargement local.

Depuis MySQL 3.23.49 et MySQL 4.0.2 (4.0.13 sur Windows), `LOCAL` fonctionne uniquement si votre serveur et votre client ont été configuré pour. Par exemple, si `mysqld` a été lancé avec `--local-infile=0` , `LOCAL` ne fonctionnera pas. Problèmes de sécurité avec `LOAD DATA LOCAL` .

Si vous avez besoin de lire des données `LOAD DATA` depuis un pipe, vous devez utiliser la technique suivante :

```
 mkfifo /mysql/db/x/x
chmod 666 /mysql/db/x/x
cat < /dev/tcp/10.1.1.12/4711 > /mysql/db/x/x
mysql -e "LOAD DATA INFILE 'x' INTO TABLE x" x
```

Si vous utilisez une version de MySQL plus ancienne que 3.23.25, vous pouvez uniquement utiliser cette technique avec `LOAD DATA LOCAL INFILE` .

Si vous utilisez une version de MySQL antérieure à la 3.23.24 vous ne pouvez lire à partir d'un FIFO avec `LOAD DATA INFILE` . Si vous avez besoin de lire à partir d'un FIFO (par exemple la sortie de gunzip), utilisez `LOAD DATA LOCAL INFILE` .

Lorsque les fichiers de données sont sur le serveur, celui−ci utilise les règles suivantes :

- Si un chemin absolu est fourni, le serveur utilise le chemin tel quel.
- Si un chemin relatif est fourni, avec un ou plusieurs éléments de dossiers, le serveur recherche le fichier relativement à son dossier de données.
- Si le fichier n'a pas d'éléments de dossier, le serveur recherche les données dans le dossier de base de données courante.

Notez que ces règles font qu'un fichier tel que `./myfile.txt` est lu dans le dossier de données du serveur, alors que s'il est nommé `myfile.txt` , il sera lu dans le dossier de base de données courante. Par exemple, la commande `LOAD DATA` suivante lit le fichier `donnees.txt` dans le dossier de la base `db1` car `db1` est la base de données courante, même si la commande charge explicitement le fichier dans la base de données `db2` :

```
 mysql> USE db1;
mysql> LOAD DATA INFILE "donnees.txt" INTO TABLE db2.ma_table;
```

Les mots réservés `REPLACE` et `IGNORE` contrôlent la méthode d'insertion de lignes lorsque des doublons apparaissent pour les clés uniques.Si vous spécifiez `REPLACE` , les nouvelles lignes remplaceront les anciennes. Syntaxe de `REPLACE` .

Si vous spécifiez `IGNORE` , les nouvelles lignes seront ignorées. Si vous ne spécifiez pas cette option, une erreur sera générée à chaque doublon, et le reste du fichier sera ignoré. Avec l'option `LOCAL` , le comportement par défaut est le même que si `IGNORE` est spécifié : ceci est dû au fait que le serveur n'a pas moyen de stopper la transmission du fichier au milieu de l'opération.

Si vous chargez un fichier sur votre machine client avec l'option `LOCAL` , le serveur ne peut pas interrompre la transmission du fichier au milieu de l'opération : par défaut, il utilisera l'option `IGNORE` .

Si vous voulez ignorer les clés étrangères le temps du chargement du fichier, utilisez la commande `SET FOREIGN_KEY_CHECKS=0` avant d'exécuter `LOAD DATA` .

Si vous utilisez `LOAD DATA INFILE` sur une table vide de type `MyISAM` , tous les index non−uniques seront créés dans un processus séparé (tout comme `REPAIR`). Cela rend `LOAD DATA INFILE` beaucoup plus rapide si vous avez plusieurs index. Utilisation de `myisamchk` pour maintenir les tables et recouvrir les données

.

`LOAD DATA INFILE` est le complémentaire de `SELECT ... INTO OUTFILE`. Syntaxe de `SELECT`. Pour écrire des données depuis une table dans un fichier, utilisez `SELECT ... INTO OUTFILE`. Pour lire les données dans la table, utilisez `LOAD DATA INFILE`. La syntaxe des clauses `FIELDS` et `LINES` est la même pour les deux commandes. Ces deux clauses sont optionnelles, mais `FIELDS` doit précéder `LINES`, si les deux sont spécifiées.

Si vous spécifiez la clause `FIELDS`, les sous−clauses `TERMINATED BY`, `[OPTIONALLY] ENCLOSED BY`, et `ESCAPED BY` sont aussi optionnelles, mais vous devez en spécifier au moins une.

Si vous ne spécifiez par de clause `FIELDS`, les valeurs par défaut sont :

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Si vous ne spécifiez par de clause `LINES`, les valeurs par défaut sont :

```
LINES TERMINATED BY '\n'
```

En d'autres termes, les valeurs par défaut font que `LOAD DATA INFILE` lit les données comme suit :

- Recherche des limites de lignes parmi les nouvelles lignes.
- Si `LINES STARTING BY prefix` est utilisé, lit jusqu'au préfixe, et commence à lire après le préfixe. Si la ligne n'inclut pas de préfixe, elle sera ignorée.
- Scinde les lignes en champs avec les tabulations.
- Ne suppose pas que les champs sont entourés de guillemets.
- Interprète les occurrences de tabulation, nouvelle ligne, `` `\' `` précédées par `` `\' `` comme des caractères littéraux qui font partie de la valeur d'un champs.

A l'inverse, les valeurs par défaut font que `SELECT ... INTO OUTFILE` écrit les données comme ceci :

- Ecrivez des tabulations entre les champs.
- N'entourez pas les champs de guillemets.
- Utilisez `'\'` pour échapper les occurrences de tabulation, nouvelle ligne, `'\'` trouvées dans les valeurs.
- Insère une nouvelle ligne entre les lignes.

Notez que pour utiliser `FIELDS ESCAPED BY '\\'`, vous devez spécifier deux anti−slash pour que cette valeur soit interprétée comme un anti−slash simple.

Note : si vous avez généré le fichier sur Windows, vous devrez peut−être utiliser `LINES TERMINATED BY '\r\n'` pour lire le fichier correctement, car les programmes Windows utilisent généralement deux caractères comme fin de ligne. Certains programmes, comme `WordPad`, peuvent utiliser `\r` comme terminateur de ligne lors de l'écriture. Pour lire ces fichiers, utilisez `LINES TERMINATED BY '\r`.

L'option `IGNORE nombre LINES` sert à ignorer une en−tête de fichier, telle que des noms de colonnes, qui débutent parfois un fichier à charger :

```
mysql> LOAD DATA INFILE "/tmp/nom_fichier" INTO TABLE test IGNORE 1 LINES;
```

Lorsque vous utilisez `SELECT ... INTO OUTFILE` conjointement avec `LOAD DATA INFILE` pour écrire des données dans un fichier et les relire dans une table, les options de `FIELDS` et `LINES` doivent être identiques. Sinon, `LOAD DATA INFILE` ne pourra pas interpréter le contenu du fichier correctement. Supposez que la commande `SELECT ... INTO OUTFILE` ait écrit un fichier délimité par des virgules :

```
mysql> SELECT * INTO OUTFILE 'donnees.txt'
    ->          FIELDS TERMINATED BY ','
    ->          FROM ...;
```

Pour lire ce fichier, la commande correcte serait :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE table2
    ->           FIELDS TERMINATED BY ',';
```

Si au contraire, vous essayez de lire le fichier avec la commande ci−dessous, cela ne fonctionnera pas, car la commande `LOAD DATA INFILE` essaie de lire des tabulations entre les champs :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE table2
    ->           FIELDS TERMINATED BY '\t';
```

Il est probable que chaque ligne d'entrée sera interprétée que comme un seul champ.

La commande `LOAD DATA INFILE` peut être utilisée pour lire des données issues d'autres sources. Par exemple, un fichier au format dBASE présente des champs séparés par des virgules, et entourés de guillemets doubles. Si les lignes sont terminées par de nouvelles lignes, la commande ci−dessous illustre la relecture d'un tel fichier avec MySQL :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE nom_de_table
    ->           FIELDS TERMINATED BY ',' ENCLOSED BY '"'
    ->           LINES TERMINATED BY '\n';
```

Les clauses `FIELDS` et `LINES` peuvent prendre des chaînes vides comme valeur. S'il la chaîne n'est pas vide, `FIELDS [OPTIONALLY] ENCLOSED BY` et `FIELDS ESCAPED BY` ne doivent avoir qu'un seul caractère. Les valeurs de `FIELDS TERMINATED BY` et `LINES TERMINATED BY` peuvent avoir plus d'un caractère. Par exemple, pour écrire des lignes terminées par le couple retour chariot/nouvelle ligne, ou pour lire un tel fichier, spécifiez la clause `LINES TERMINATED BY '\r\n'` .

Par exemple, pour charger un fichier de blagues, qui sont séparées par une ligne de `%%` , dans une table vous pouvez faire :

```
 CREATE TABLE blagues (
   a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    blague TEXT NOT NULL
   );
LOAD DATA INFILE "/tmp/blagues.txt" INTO TABLE blagues FIELDS TERMINATED BY ""
LINES TERMINATED BY "\n%%\n" (blague);
```

`FIELDS [OPTIONALLY] ENCLOSED BY` contrôle la mise entre guillemets des champs. Pour l'écriture de fichier (`SELECT ... INTO OUTFILE`), si vous omettez le mot `OPTIONALLY` , tous les champs seront entourés par le caractère spécifié dans la clause `ENCLOSED BY` . Par exemple, si la virgule est utilisée comme séparateur de champs :

```
 "1","une chaîne","100.20"
"2","une chaîne contenant une , virgule","102.20"
"3","une chaîne contenant un \" guillemet","102.20"
"4","une chaîne contenant un \", guillemet et une virgule","102.20"
```

Si vous spécifiez `OPTIONALLY` , le caractère `ENCLOSED BY` n'est utilisé que pour protéger les colonnes de types `CHAR` et `VARCHAR` :

```
 1,"une chaîne",100.20
2,"une chaîne contenant une , virgule",102.20
3,"une chaîne contenant un \" guillemet",102.20
4,"une chaîne contenant un \", guillemet et une virgule",102.20
```

Notez que les occurrences du caractère `ENCLOSED BY` dans un champs sont échappée en les préfixant avec le caractère `ESCAPED BY` . Notez aussi que si vous spécifiez un caractère d'échappement vide, il n'est pas possible de garantir que les champs seront correctement relus par `LOAD DATA INFILE` . Par exemple, l'exemple ci−dessus apparaîtra comme montré ci−dessous. Notez que le second champ de la quatrième ligne comporte une virgule suivant un guillemet qui semble (mais c'est faux) terminer la ligne :

```
 1,"une chaîne",100.20
2,"une chaîne contenant une , virgule",102.20
3,"une chaîne contenant un " guillemet",102.20
4,"une chaîne contenant un ", guillemet et une virgule",102.20
```

Lors des lectures, le caractère `ENCLOSED BY` , s'il est présent, est supprimé des extrémités de la valeur du champ. (ce qui est vrai, qu'il y ait l'option `OPTIONALLY` ou pas). Les occurrences du caractère `ENCLOSED BY` , précédées par le caractère `ESCAPED BY` sont interprétées comme faisant partie de la valeur du champ. Les caractères `ENCLOSED BY` doublées, apparaissant dans la chaîne, sont interprétés comme le caractère `ENCLOSED BY` lui−même. Par exemple, si `ENCLOSED BY '"'` est spécifié, les guillemets sont gérés comme ceci :

```
 "Le ""GRAND"" chef"  -> Le "GRAND" chef
Le "GRAND" chef      -> Le "GRAND" chef
Le ""GRAND"" chef    -> Le ""GRAND"" chef
```

`FIELDS ESCAPED BY` contrôle les caractères spéciaux. Si le caractère `FIELDS ESCAPED BY` n'est pas vide, il est utilisé pour préfixer les caractères suivants en écriture :

- La caractère `FIELDS ESCAPED BY`
- Le caractère `FIELDS [OPTIONALLY] ENCLOSED BY`
- Le premier caractère des valeurs de `FIELDS TERMINATED BY` et `LINES TERMINATED BY`
- ASCII `0` (en fait, ce qui est écrit après le caractère d'échappement est le caractère ASCII `'0'`, et non pas le code ASCII de zéro)

Si le caractère `FIELDS ESCAPED BY` est vide, aucun caractère ne sera échappé. Ce n'est probablement pas une bonne idée de spécifier un caractère d'échappement vide, en particulier si les valeurs dans vos champs risquent d'utiliser l'un des caractères de la liste ci−dessus.

En lecture, si le caractère `FIELDS ESCAPED BY` n'est pas vide, les occurrences de ce caractère sont supprimées, et le caractère suivant est lu littéralement. Les exceptions à cette règle sont `'0'` ou `'N'` (par exemple, `0` ou `\N` si le caractère d'échappement est `'\'`). Ces séquences sont interprétées comme l'octet nul (ASCII `0`) et la valeur `NULL`. Voyez plus bas pour la gestion des valeurs `NULL`.

Pour plus d'informations sur la syntaxe avec les caractères d'échappement `'\'`, consultez Comment écrire les chaînes et les nombres ? .

Dans certains cas, les options de `FIELDS` et `LINES` interfèrent entre elles :

- Si le caractère de `LINES TERMINATED BY` est une chaîne vide et que celui de `FIELDS TERMINATED BY` ne l'est pas, ce dernier sera celui utilisé pour `LINES TERMINATED BY`.
- Si les valeurs `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont vides toutes les deux (`''`), un format à taille de champ fixe est utilisé. Avec ce format, aucun délimiteur n'est utilisé entre les champs. Au lieu de cela, les valeurs des colonnes sont écrites avec leur configuration d'affichage. Par exemple, si une colonne a été déclarée `INT(7)`, la valeur de cette colonne sera écrite avec 7 caractères. Lors de la relecture, la valeur de la colonne sera obtenue en lisant à nouveau 7 caractères. Ce format à taille fixe affecte la gestion de la valeur `NULL` ; voyez plus loin pour cela. Notez que ce format ne fonctionne pas avec les jeux de caractères multi−octets.

La gestion des valeurs `NULL` dépend des options `FIELDS` et `LINES` que vous utilisez :

- Pour les valeurs par défaut de `FIELDS` et `LINES`, `NULL` est écrit `\N` et `\N` est lu `NULL` (en supposant que le caractère d'échappement est `'\'`).
- Si `FIELDS ENCLOSED BY` n'est pas vide, un champ contenant le mot `NULL` comme valeur sera lu comme la valeur `NULL` (ce qui diffère du mot `NULL`, entouré du caractère `FIELDS ENCLOSED BY`, qui sera lu comme le mot `'NULL'`).
- Si `FIELDS ESCAPED BY` est vide, `NULL` est écrit comme le mot `'NULL'`.
- Avec le format à taille fixe (ce qui arrive si `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont tous les deux vides), les valeurs `NULL` sont écrites sous forme de chaîne vide. Notez que cela fait que `NULL` et les chaînes vides seront représentées par une valeur qui ne les distingue pas l'une de l'autre. Si vous avez besoin de différencier entre les deux, n'utilisez par ce format !

Certains cas ne sont pas supportés par `LOAD DATA INFILE` :

- Lignes à tailles fixes (`FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont tous les deux vides) et les types de colonne `BLOB` ou `TEXT`.

- Si vous spécifiez un séparateur qui est le même qu'un autre préfixe, `LOAD DATA INFILE` ne sera pas capable de relire proprement le résultat. Par exemple, la clause `FIELDS` suivante posera sûrement des problèmes :
  ```
  FIELDS TERMINATED BY '"' ENCLOSED BY '"'
  ```
- Si `FIELDS ESCAPED BY` est vide, une valeur de colonne qui contient une occurrence de `FIELDS ENCLOSED BY` ou de `LINES TERMINATED BY` suivi du caractère `FIELDS TERMINATED BY` interrompra la lecture de `LOAD DATA INFILE` trop tôt. Cela est dû au fait que `LOAD DATA INFILE` ne peut pas faire la différence entre la valeur dans le champ et la fin de la ligne.

L'exemple suivant charge toutes les colonnes de la table `persondata` :

```
mysql> LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;
```

Aucun champ n'est spécifié, ce qui fait que `LOAD DATA INFILE` s'attend à ce que les lignes lues contiennent le bon nombre de champs. Les valeurs par défaut de `FIELDS` et `LINES` sont utilisées.

Si vous voulez charger uniquement quelques colonnes dans une table, spécifiez la liste des champs :

```
mysql> LOAD DATA INFILE 'persondata.txt'
    ->           INTO TABLE persondata (col1,col2,...);
```

Vous devez aussi spécifier les champs si l'ordre dans lequel ils seront lus diffère de l'ordre des colonnes dans la table. Sinon, MySQL ne pourra pas savoir à quelle colonne correspond une valeur.

Si une ligne a trop peu de champs, les colonnes omises prendront leur valeur par défaut. Les affectations de valeurs par défaut sont décrites dans Syntaxe de `CREATE TABLE` .

Une valeur de champs vide et un champ manquant ne seront pas interprétés de la même façon :

- Pour les types chaîne, la colonne est remplie avec la chaîne vide.
- Pour les types numériques, la colonne est mise à `0` .
- Pour les types dates et heures, la colonne est mise au zéro approprié pour le type. Types temporels .

Notez que vous obtiendrez le même résultat en assignant à ces différents types de champs la chaîne vide dans une commande `INSERT` ou `UPDATE` .

Les colonnes `TIMESTAMP` prendront la date et l'heure courante uniquement si on leur affecte la valeur `NULL` , ou (pour la première colonne `TIMESTAMP` seulement) si la colonne `TIMESTAMP` est ignorée de la liste des colonnes spécifiée.

Si une ligne d'entrée comporte trop de colonnes, les champs en trop sont ignorés, et le nombre d'alertes est incrémenté.

`LOAD DATA INFILE` considère toutes les valeurs lues comme des chaînes de caractères : vous ne pourrez donc pas utiliser la forme numérique des colonnes `ENUM` ou `SET` , comme d'habitude. Toutes les colonnes `ENUM` et `SET` doivent être spécifiée comme des chaînes ! Si vous utilisez l'API C, vous pouvez obtenir des informations à propos de la requête en utilisant la fonction `mysql_info()` quand `LOAD DATA INFILE` se termine. Le format de la chaîne d'informations est le suivant :

```
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

Les alertes sont générées dans les mêmes circonstances que pour la commande `INSERT` (Syntaxe de `INSERT`), excepté que `LOAD DATA INFILE` génère aussi des alertes s'il y a trop peu ou trop de champs dans une ligne. Les alertes ne sont pas stockées; le nombre d'alertes est la seule indication. Si vous recevez des alertes et vous voulez savoir exactement ce qui s'est passé, exécutez une commande `SELECT ... INTO OUTFILE` dans un autre fichier et comparez le avec le fichier original.

En MySQL version 4.1.1 vous pouvez utiliser `SHOW WARNINGS` pour obtenir la liste des premières `max_error_count` alertes. Syntaxe de `SHOW WARNINGS` .

Pour plus d'informations sur les performances de `INSERT` comparées à `LOAD DATA INFILE` et accélérer `LOAD DATA INFILE` : Vitesse des requêtes `INSERT` .

14.1.6 Syntaxe de REPLACE

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [(col_name,...)]
    VALUES (&#123;expr | DEFAULT&#125;,...),(...),...
```

ou :

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name
    SET col_name=&#123;expr | DEFAULT&#125;, ...
```

ou :

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

`REPLACE` fonctionne exactement comme `INSERT` , sauf que si une vieille ligne dans la table à la même valeur qu'une nouvelle pour un index `UNIQUE` ou une `PRIMARY KEY` , la vielle ligne sera effacée avant que la nouvelle ne soit insérée. Syntaxe de `INSERT` .

En d'autres termes, vous ne pouvez pas accéder aux valeurs de l'ancienne ligne à partir d'une requête `REPLACE` . Dans quelques vieilles versions de MySQL, il apparaît que c'était possible, mais c'etait un dysfonctionnement qui a été corrigé depuis.

Pour utiliser `REPLACE` vous devez avoir les privilèges `INSERT` et `DELETE` sur la table.

Quand vous utilisez une commande `REPLACE` , `mysql_affected_rows()` retournera 2 si une nouvelle ligne en remplace une existante, et cela parce qu'il y aura eu une insertion puis une suppression.

Cela aide à savoir si `REPLACE` a ajouté ou a remplacé une ligne : Testez si le nombre de lignes affectées est égal à 1 (ajout) ou s'il est égal à 2 (remplacement).

Notez que si vous n'utilisez pas un index `UNIQUE` ou une `PRIMARY KEY` , utiliser un `REPLACE` n'a pas de sens vu que cela revient à utiliser un `INSERT` . Il devient équivalent à `INSERT` , car il n'y a pas d'index à utiliser pour déterminer si un nouvelle ligne est un double d'une autre.

Voici quelques détails sur l'algorithme utilisé : Il est aussi utilisé par `LOAD DATA ... REPLACE` .

- Insertion de la ligne dans la table
- Si une erreur de clé dupliqué ou de clé unique ou de clé primaire survient :
    - Annuler les changements de clés
    - Essayer à nouveau d'insérer la clé primaire et unique dans la table

14.1.7 Syntaxe de SELECT

```
SELECT [STRAIGHT_JOIN]
       [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    select_expression,...
    [INTO &#123;OUTFILE | DUMPFILE&#125; 'nom_fichier' export_options]
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY &#123;unsigned_integer | nom_de_colonne | formula&#125; [ASC | DESC], ...
      [HAVING where_definition]
      [ORDER BY &#123;unsigned_integer | nom_de_colonne | formula&#125; [ASC | DESC] ,...]
      [LIMIT [offset,] lignes]
      [PROCEDURE procedure_name(argument_list)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

`SELECT` est utilisé pour obtenir des enregistrements venant d'une ou plusieurs tables. Le support des commandes `UNION` et des sous−requêtes est disponibles depuis MySQL 4.0 et 4.1, respectivement. Voir Syntaxe de `UNION` et Syntaxe des sous−requêtes .

- Chaque `select_expr` indique une colonne à lire.
- `table_references` indique la ou les tables à utiliser. La syntaxe est décrite dans Syntaxe de `JOIN` .
- `where_definition` indique les conditions que les lignes sélectionnées doivent satisfaire.

`SELECT` peut aussi être utilisée pour lire des lignes calculées, sans référence à une table.

Par exemple :

```
mysql> SELECT 1 + 1;
         -> 2
```

Tous les mots−clés utilisés doivent être donnés exactement dans le même ordre que ci−dessus. Par exemple, une clause `HAVING` doit être placée après toute clause `GROUP BY` et avant toute clause `ORDER BY` .

- Une expression `SELECT` peut recevoir un alias en utilisant `AS` . L'alias est utilisé de la même façon que le nom du champ et peut être employé avec des clauses `ORDER BY` ou `HAVING` . Par exemple :
  ```
  mysql> SELECT CONCAT(last_name,', ',first_name) AS full_name
      FROM mytable ORDER BY full_name;
  ```
  Le mot clé `AS` est optionnel lors de la création d'un alias pour une expression `SELECT` . L'exemple précédent aurait pu être écrit comme ceci :
  ```
  mysql> SELECT CONCAT(last_name,', ',first_name) full_name
      FROM mytable ORDER BY full_name;
  ```
  Comme `AS` est optionnel, un problème subtil peut survenir si vous oubliez une virgule entre deux expressions de `SELECT` : MySQL va interpréter la seconde comme un alias de la première. Par exemple, dans la commande suivante, `columnb` est traité comme un nom d'alias :
  ```
  mysql> SELECT columna columnb FROM mytable;
  ```
- Il n'est pas possible d'utiliser un alias de champ dans une clause `WHERE` , car la valeur du champ peut ne pas être définie lorsque la clause `WHERE` est exécutée. Problèmes avec les `alias` .
- La clause `FROM table_references` indique les tables à partir desquelles nous allons obtenir les enregistrements. Si vous indiquez le nom de plusieurs tables, vous faites une jointure. Pour davantage d'informations sur la syntaxe des jointures, consultez Syntaxe de `JOIN` . Pour chaque table spécifiée, vous pouvez éventuellement indiquer un alias.
  ```
  tbl_name [[AS] alias]
      [[USE INDEX (key_list)]
        | [IGNORE INDEX (key_list)]
        | [FORCE INDEX (key_list)]]
  ```
  L'utilisation de `USE INDEX` , `IGNORE INDEX` , `FORCE INDEX` pour donner des conseils d'optimisation à l'optimiseur d'index. Syntaxe de `JOIN` .

En MySQL 4.0.14, vous pouvez utiliser `SET MAX_SEEKS_FOR_KEY=value` comme une alternative pour forcer MySQL à choisir un scan d'index, plutôt qu'un scan de table.

- Vous pouvez faire référence à une table avec `nom_de_table` (au sein de la base de données courante), ou avec `dbname.nom_de_table` pour expliciter le nom de la base de données. Vous pouvez vous référer à un champ avec `nom_de_colonne` , `nom_de_table.nom_de_colonne` , ou `db_name.nom_de_table.nom_de_colonne` . Vous n'êtes pas obligés d'indiquer de préfixe `nom_de_table` ou `db_name.nom_de_table` pour une référence à un champ dans un `SELECT` , à moins que la référence ne soit ambiguÎ. Consultez Noms de base de données, tables, index, colonnes et alias , pour des exemples d'ambiguïtés qui nécessitent des formes plus explicites de référence à des champs.
- Depuis la version 4.1.0, vous êtes autorisés à spécifier `DUAL` comme nom de table, dans les situations ou aucune table n'est référencé. C'est une fonctionnalité pure de compatibilité, car certains autres serveurs requièrent cette syntaxe.

```
mysql> SELECT 1 + 1 FROM DUAL;
        -> 2
```

- Une référence à une table peut être aliasée en utilisant `nom_de_table [AS] alias_name` :

```
mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
    ->        WHERE t1.name = t2.name;
mysql> SELECT t1.name, t2.salary FROM employee t1, info t2
    ->        WHERE t1.name = t2.name;
```

- Dans la clause `WHERE` , vous pouvez utiliser toutes les fonctions que MySQL supporte, hormis les fonctions d'agrégation. Fonctions et opérateurs .
- Les colonnes sélectionnées dans le résultat peuvent être nommées dans les clauses `ORDER BY` et `GROUP BY` en utilisant leur nom de colonne, les alias ou leur position de colonne. Les positions commencent à 1 :

```
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY region, seed;
mysql> SELECT college, region AS r, seed AS s FROM tournament
    ->        ORDER BY r, s;
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY 2, 3;
```

Pour trier dans l'ordre inverse, ajoutez le mot−clé `DESC` (descendant) au nom du champ dans la clause `ORDER BY` qui vous permet de trier. Par défaut, l'ordre ascendant est utilisé; ceci peut être indiqué de façon explicite en utilisant le mot−clé `ASC` .L'utilisation des positions de colonnes est obsolète, car la syntaxe a été supprimée du SQL standard.

- Si vous utilisez `GROUP BY` , les lignes sont triées en fonction des colonnes `GROUP BY` comme si on avait ajouté la clause `ORDER BY` pour ces colonnes. MySQL a amélioré la clause `GROUP BY` depuis la version 3.23.34 pour que vous puissiez aussi spécifier `ASC` et `DESC` après le nom de la colonne :

```
SELECT a, COUNT(b) FROM test_table GROUP BY a DESC
```

- MySQL améliore l'utilisation de `GROUP BY` en vous autorisant à l'utiliser avec des champs qui ne sont pas mentionnés dans la clause `GROUP BY` . Si vous n'obtenez pas les résultats que vous attendiez, lisez la description de `GROUP BY` . Fonctions à utiliser dans les définitions `GROUP BY` .
- Depuis MySQL 4.1.1, `GROUP BY` dispose de l'option `WITH ROLLUP` . Options de `GROUP BY` .
- La clause `HAVING` peut faire référence à n'importe quel champs ou alias défini dans `select_expr` . C'est évalué en dernier lieu, juste avant que les éléments ne soient envoyés au client, sans aucune optimisation.
- N'utilisez pas `HAVING` pour des éléments qui devraient être dans la clause `WHERE` . Par exemple, n'écrivez pas ceci :

```
mysql> SELECT nom_de_colonne FROM nom_de_table HAVING nom_de_colonne > 0;
```

Ecrivez plutôt cela :

```
mysql> SELECT nom_de_colonne FROM nom_de_table WHERE nom_de_colonne > 0;
```

Dans les versions 3.22.5 et supérieures de MySQL, vous pouvez aussi écrire des requêtes ainsi :

```
mysql> SELECT user,MAX(salary) FROM users
    ->        GROUP BY user HAVING MAX(salary)>10;
```

Dans des versions plus anciennes de MySQL, vous pouvez écrire à la place :

```
mysql> SELECT user,MAX(salary) AS sum FROM users
    ->        group by user HAVING sum>10;
```

- La clause `HAVING` peut utiliser des fonctions d'agrégation, alors que la clause `WHERE` ne le peut pas :

```
mysql> SELECT user, MAX(salary) FROM users
    ->     GROUP BY user HAVING MAX(salary)>10;
```

Cependant, cela ne fonctionne pas dans les anciennes versions du serveur MySQL, : avant la version 3.22.5. Au lieu de cela, ajoutez un alias de colonne dans la liste de colonnes, et faites référence à cet alias dans la colonne `HAVING` :

```
mysql> SELECT user, MAX(salary) AS max_salary FROM users
    ->     GROUP BY user HAVING max_salary>10;
```

- La clause `LIMIT` peut être utilisée pour limiter le nombre d'enregistrements retournés par la commande `SELECT` . `LIMIT` accepte un ou deux arguments numériques. Ces arguments doivent être des entiers constants.Avec un argument, la valeur spécifie le nombre de lignes à retourner depuis le début du jeu de résultat. Si deux arguments sont donnés, le premier indique le décalage du premier enregistrement à retourner, le second donne le nombre maximum d'enregistrement à retourner. Le décalage du premier enregistrement est 0 (pas 1) :

  Pour être compatible avec PostgreSQL, MySQL supporte aussi la syntaxe : `LIMIT row_count OFFSET offset` .

  ```
  mysql> SELECT * FROM table LIMIT 5,10;  # Retourne les enregistrements 6 à 15
  ```

  Pour obtenir tous les enregistrement d'un certain décalage jusqu'à la fin du résultat, vous pouvez utiliser de grands entier en tant que second paramètre :

  ```
  mysql> SELECT * FROM table LIMIT 95,18446744073709551615; # Retourne les enregistrements de 96 jusqu'au dernier.
  ```

  Si un seul argument est donné, il indique le nombre maximum d'enregistrements à retourner :

  ```
  mysql> SELECT * FROM table LIMIT 5;     # Retourne les 5 premiers enregistrements
  ```

  Autrement dit, `LIMIT n` est équivalent à `LIMIT 0,n` .
- La forme `SELECT ... INTO OUTFILE 'nom_fichier'` de `SELECT` écrit les lignes sélectionnées dans un fichier. Le fichier est crée sur le serveur et ne peut y être déjà présent (cela permet entre autre d'éviter la destruction des tables et de fichiers tel que `/etc/passwd`). Vous devez avoir le droit `FILE` sur le serveur pour utiliser cette forme de `SELECT` . `SELECT ... INTO OUTFILE` à pour but principal de vous permettre de réaliser des dumps rapides des tables sur la machine serveur. Si vous voulez créer le fichier sur une autre machine, vous ne pouvez utiliser `SELECT ... INTO OUTFILE` . Dans ce cas là, vous pouvez utiliser à la place un programme client comme `mysqldump --tab` ou `mysql -e "SELECT ..." > fichier` pour générer le fichier.

  `SELECT ... INTO OUTFILE` est le complément de `LOAD DATA INFILE` ; La syntaxe pour la partie `export_options` de la requête se compose des mêmes clauses `FIELDS` et `LINES` que celles utilisées avec la commande `LOAD DATA INFILE` . Syntaxe de `LOAD DATA INFILE` .

  Dans le fichier résultant, seul les caractères suivants sont protégés par le caractère `ESCAPED BY` :

  - Le caractère `ESCAPED BY`
  - Les premier caractère de `FIELDS TERMINATED BY`
  - Les premier caractère de `LINES TERMINATED BY`
  - `ASCII 0` est convertit en `ESCAPED BY` suivi de 0 (`ASCII 48`).

  Si le caractère `FIELDS ESCAPED BY` est vide, aucun caractère n'est protégé, et `NULL` vaut `NULL` , et non `\N` . Il est probable que ce ne soit pas une bonne idée de spécifier un caractère de protection vide, en particulier sir les valeurs de vos champs peuvent être n'importe quoi.

  La raison de ce qui précède est que vous devez **impérativement** protéger chaque caractère `FIELDS TERMINATED BY` , `ESCAPED BY` , ou `LINES TERMINATED BY` pour assurer une relecture fiable du fichier. Le caractère `ASCII 0` est échappé pour assurer la lisibilité sur certains clients.

  Comme le fichier résultant ne se doit pas d'être syntaxiquement conforme à SQL, vous n'avez besoin d'échapper rien d'autre.

Voila un exemple de relecture de fichier au format utilisé par plusieurs anciens programmes.

```
 SELECT a,b,a+b INTO OUTFILE '/tmp/result.text'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM test_table;
```

- Si vous utilisez `INTO DUMPFILE` au lieu de `INTO OUTFILE` , MySQL n'écrira qu'une seule ligne dans le fichier, sans aucun caractère de fin de ligne ou de colonne, ni d'échappement. Cela est utile lorsque vous voulez enregistrer un `BLOB` dans un fichier.
- **Note :** notez que les fichiers crées par `INTO OUTFILE` et `INTO DUMPFILE` sera lisible par tout les utilisateurs ! La raison est que le serveur MySQL ne peut créer de fichier appartenant à autre que l'utilisateur qui l'a mis en route. (vous devez éviter d'exécuter `mysqld` en tant que root), le fichier doit se composer de mot lisible pour que les données puissent être récupérées.
- Une clause `PROCEDURE` indique une procédure qui doit traiter les lignes du jeu de résultat. Pour un exemple, voyez La procédure Analyse .
- Si vous utilisez la clause `FOR UPDATE` avec un gestionnaire de tables qui gère les verrous de lignes ou de pages, les lignes seront verrouillées.

Après le mot `SELECT` , vous pouvez ajouter certaines options qui affectent le comportement de la commande.

Les options `DISTINCT` , `DISTINCTROW` et `ALL` indiquent quels enregistrements avec doublons doivent être retournés. Par défaut, c'est (`ALL`), retournant ainsi tous les enregistrements. `DISTINCT` et `DISTINCTROW` sont synonymes et indique que les doublons doivent être éliminés du résultat.

`HIGH_PRIORITY` , `STRAIGHT_JOIN` , et les options commençants par `SQL_` sont des extensions MySQL au standard SQL.

- `HIGH_PRIORITY` donne à une commande `SELECT` une plus grande priorité qu'une commande qui modifie une table. Vous devez l'utiliser seulement pour les requêtes qui sont très rapides et qui doivent être effectuées en premier lieu. Une requête `SELECT HIGH_PRIORITY` s'exécutera sur une table verrouillée en lecture même si une commande de mise à jour attend que la table soit libérée. `HIGH_PRIORITY` ne peut être utilisée avec les commandes `SELECT` qui font partie d'une `UNION` .
- `STRAIGHT_JOIN` force l'optimiseur à joindre les tables dans l'ordre dans lequel elles sont listées dans la clause `FROM` . Vous pouvez utiliser cela pour accélérer la requête, si les tables sont réordonnées sub−optimalement par l'optimiseur. Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT`) . `STRAIGHT_JOIN` peut aussi être utilisée dans la liste `table_references` . Syntaxe de `JOIN` .
- `SQL_BIG_RESULT` peut être utilisé avec `GROUP BY` ou `DISTINCT` pour indiquer à l'optimiseur que le résultat comportera beaucoup d'enregistrements. Dans ce cas, MySQL utilisera si besoin directement les bases temporaires stockées sur le disque. MySQL préférera, dans ce cas, trier que d'obtenir une table temporaire avec une clé sur les éléments du `GROUP BY` .
- `SQL_BUFFER_RESULT` forcera le résultat à être stocké dans une table temporaire. Ceci va aider MySQL à libérer plus tôt les verrous des tables et aidera aussi dans les cas ou l'envoi du résultat au client prend un temps assez conséquent.
- `SQL_SMALL_RESULT` , une option spécifique à MySQL, peut être utilisée avec `GROUP BY` ou `DISTINCT` pour indiquer à l'optimiseur que le résultat sera petit. Dans ce cas, MySQL utilise des tables temporaires rapides pour stocker la table résultante plutôt que d'utiliser le tri. Dans MySQL 3.23, ceci n'est normalement pas nécessaire.
- `SQL_CALC_FOUND_ROWS` (version 4.0.0 et supérieure) indique à MySQL de calculer combien d'enregistrements seront dans le jeu de résultats, indépendamment de n'importe quelle clause `LIMIT` . Le nombre d'enregistrements peut alors être obtenu avec `SELECT FOUND_ROWS()` . Fonctions diverses .Avant MySQL 4.1.0, cette option ne fonctionne pas avec `LIMIT 0` , qui est optimisée pour se terminer instantanément (le résultat ne contiendra pas de lignes).

Comment MySQL optimise `LIMIT` .

- `SQL_CACHE` demande à MySQL de ne pas stocker le résultat de la requête si vous utilisez `query_cache_type` avec la valeur `2` ou `DEMAND` . Pour une requête qui utilise `UNION` ou une sous−requête, cette option prend effet si elle est utilisée dans n'importe quelle partie de la requête `SELECT` . Le cache de requêtes MySQL .
- `SQL_CACHE` indique à MySQL de stocker le résultat de la requête dans le cache de requêtes si vous utilisez `QUERY_CACHE_TYPE=2` (`DEMAND`). Le cache de requêtes MySQL . Pour les requêtes qui utilisent `UNION` ou les sous−requêtes, cette option aura un effet sur toutes les parties de la requête `SELECT` .

14.1.7.1 Syntaxe de JOIN

MySQL supporte les syntaxes suivantes de `JOIN` pour une utilisation dans les `SELECT` :

```
 reference_table, reference_table
reference_table [CROSS] JOIN reference_table
reference_table INNER JOIN reference_table condition_jointure
reference_table STRAIGHT_JOIN reference_table
reference_table LEFT [OUTER] JOIN reference_table condition_jointure
reference_table LEFT [OUTER] JOIN reference_table
reference_table NATURAL [LEFT [OUTER]] JOIN reference_table
&#123; OJ reference_table LEFT OUTER JOIN reference_table ON expr_conditionnelle &#125;
reference_table RIGHT [OUTER] JOIN reference_table condition_jointure
reference_table RIGHT [OUTER] JOIN reference_table
reference_table NATURAL [RIGHT [OUTER]] JOIN reference_table
```

où `reference_table` est définie de la manière suivante :

```
 nom_de_table [[AS] alias] [USE INDEX (liste_de_clefs)] [IGNORE INDEX (liste_de_clefs)]
```

et `condition_jointure` est définie comme suit :

```
 ON expr_conditionnelle |
USING (column_list)
```

Généralement, vous ne devez avoir aucune condition, dans la partie `ON` , qui soit utilisée pour spécifier les lignes que vous voulez obtenir en résultat. (il y a des exceptions à cette règle). Si vous voulez restreindre les lignes résultantes, vous devez le faire dans la clause `WHERE` .Notez que dans les versions antérieures à la 3.23.17, `INNER JOIN` ne prenait pas en compte `condition_jointure` !

La dernière syntaxe de `LEFT OUTER JOIN` vue plus haut, n'existe que pour assurer la compatibilité avec ODBC :

- On peut créer un alias sur une référence de table en utilisant `nom_de_table AS alias_name` ou `nom_de_table alias_name` :

  ```
  mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
      ->        WHERE t1.name = t2.name;
  ```
- La condition `ON` est de la même forme qu'une condition pouvant être utilisée dans la clause `WHERE` .
- Si aucune ligne ne correspond dans la table de droite dans la partie `ON` ou `USING` du `LEFT JOIN` , une ligne avec toutes les colonnes mises à `NULL` est utilisé en remplacement. Vous pouvez utiliser ce fait pour trouver les enregistrements dans une table qui n'ont pas de correspondances dans une autre :

  ```
  mysql> SELECT table1.* FROM table1
      ->        LEFT JOIN table2 ON table1.id=table2.id
      ->        WHERE table2.id IS NULL;
  ```

  Cet exemple retourne toutes les lignes trouvées dans `table1` avec une valeur de `id` qui n'est pas présente dans `table2` (autrement dit, toutes les lignes de `table1` sans correspondances dans la table `table2`). Cela demande que `table2.id` soit déclaré `NOT NULL` , bien sur. Comment MySQL optimise `LEFT JOIN` .
- La clause `USING (column_list)` recense la liste des colonnes qui doivent exister dans les deux tables. Les clauses `USING` suivantes sont identiques :

  ```
   a LEFT JOIN b USING (c1,c2,c3)
  a LEFT JOIN b ON a.c1=b.c1 AND a.c2=b.c2 AND a.c3=b.c3
  ```

- La jointure de deux tables avec `NATURAL [LEFT] JOIN` est définie pour être sémantiquement équivalent à un `INNER JOIN` ou un `LEFT JOIN` avec une clause `USING` qui nomme toutes les colonnes qui existent dans les deux tables.
- `INNER JOIN` et `,` (virgule) sont sémantiquement équivalents. Les deux opèrent une jointure totale sur les tables utilisées. Normalement, vous spécifiez les conditions de jointure dans la clause `WHERE` .
- `RIGHT JOIN` fonctionne de façon analogue à `LEFT JOIN` . Pour garder un code facilement portable, il est recommandé d'utiliser les `LEFT JOIN` à la place des `RIGHT JOIN` .
- `STRAIGHT_JOIN` est identique à `JOIN` , sauf que la table de gauche est toujours lues avant celle de droite. Cela peut être utilisé dans les cas (rares) où l'optimiseur des jointures place les tables dans le mauvais ordre.

A partir de la version 3.23.12 de MySQL, vous pouvez donner des indications à propos de l'index à utiliser lors de la lecture d'informations d'une table. C'est utile si `EXPLAIN` montre que MySQL utilise un mauvais index de la liste de ceux disponibles. En spécifiant `USE INDEX (liste_de_clefs)` , vous pouvez forcer MySQL à utiliser un index spécifique pour trouver les enregistrements dans la table. Une alternative réside dans l'utilisation de `IGNORE INDEX (liste_de_clefs)` pour dire à MySQL de ne pas utiliser certains index.

En MySQL 4.0.9, vous pouvez aussi utiliser la clause `FORCE INDEX` . Elle se comporte comme `USE INDEX (key_list)` mais en supposant que les scan de tables seront **très** coûteux. En d'autres termes, les scans de tables seront utilisés que s'il n'y a pas d'autres méthodes pour trouver les lignes.

`USE/IGNORE KEY` sont des synonymes de `USE/IGNORE INDEX` .

Note : `USE INDEX` , `IGNORE INDEX` et `FORCE INDEX` affectent uniquement les index qui sont utilisés lors du choix de la méthode de sélection des lignes dans la table, et comment faire une jointure. Elles n'affectent pas l'utilisation finale de l'index dans les clauses `ORDER BY` ou `GROUP BY` .

Quelques exemples :

```
 mysql> SELECT * FROM table1,table2 WHERE table1.id=table2.id;
mysql> SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id;
mysql> SELECT * FROM table1 LEFT JOIN table2 USING (id);
mysql> SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id
    ->          LEFT JOIN table3 ON table2.id=table3.id;
mysql> SELECT * FROM table1 USE INDEX (key1,key2)
    ->          WHERE key1=1 AND key2=2 AND key3=3;
mysql> SELECT * FROM table1 IGNORE INDEX (key3)
    ->          WHERE key1=1 AND key2=2 AND key3=3;
```

Comment MySQL optimise `LEFT JOIN` .

14.1.7.2 Syntaxe de UNION

```
 SELECT ...
UNION [ALL | DISTINCT]
SELECT ...
  [UNION [ALL | DISTINCT]
   SELECT ...]
```

`UNION` est implémentée en MySQL 4.0.0. `UNION` est utilisé pour combiner le résultat de plusieurs requêtes `SELECT` en un seul résultat.

Les colonnes listées dans la partie `select_expression` du `SELECT` doivent être du même type. Les noms de colonnes utilisés dans le premier `SELECT` seront utilisé comme nom de champs pour les résultats retournés.

Les commandes `SELECT` sont des sélections normales, mais avec les restrictions suivantes :

- Seule la dernière commande `SELECT` peut avoir une clause `INTO OUTFILE` .
- `HIGH_PRIORITY` ne peut être utilisée avec les commandes `SELECT` qui ne font pas partie de l' `UNION` . Si vous la spécifiez pour la première commande `SELECT` , elle n'aura pas d'effet. Si vous la spécifiez pour toute autre commandes `SELECT` suivante, une erreur de syntaxe sera signalée.

Si vous n'utilisez pas le mot clef `ALL` pour l' `UNION` , toutes les lignes retournées seront uniques, comme si vous aviez fait un `DISTINCT` pour l'ensemble du résultat. Si vous spécifiez `ALL` , vous aurez alors tout les résultats retournés par toutes les commandes `SELECT` .

Si vous voulez utiliser un `ORDER BY` pour le résultat final de `UNION` , vous devez utiliser des parenthèses :

```
 (SELECT a FROM nom_de_table WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM nom_de_table WHERE a=11 AND B=2 ORDER BY a LIMIT 10)
ORDER BY a;
```

Note : vous ne pouvez pas mélanger les clauses `UNION ALL` et `UNION DISTINCT` dans la même requête. Si vous utilisez `ALL` dans une des `UNION` , alors elle devra être utilisée partout.

Les types et longueurs des colonnes du jeu de résultat de `UNION` prend en compte les valeurs lues dans tous les `SELECT` . Avant MySQL 4.1.1, une limitation de `UNION` est que seules les valeurs du premier `SELECT` étaient utilisée pour déterminer le type de résultats, et leur taille. Cela peut conduire à un raccourcissement de la valeur si, par exemple, le second `SELECT` trouvait des valeurs plus grandes que le premier `SELECT` :

```
 mysql> SELECT REPEAT('a',1) UNION SELECT REPEAT('b',10);
+---------------+
| REPEAT('a',1) |
+---------------+
| a             |
| b             |
+---------------+
```

Cette limitation a été supprimée en MySQL version 4.1.1 :

```
 mysql> SELECT REPEAT('a',1) UNION SELECT REPEAT('b',10);
+---------------+
| REPEAT('a',1) |
+---------------+
| a             |
| bbbbbbbbbb    |
+---------------+
```

14.1.8 Sous−sélections (SubSELECT)

Une sous−requête est une commande `SELECT` dans une autre commande. Par exemple :

```
 SELECT * FROM table1 WHERE id IN (SELECT id FROM table2);
```

La **requête externe** (ou **commande externe**), et `(SELECT column1 FROM t2)` est la **sous−requête** . Nous disons que la sous−requête est **imbriquée** dans la requête externe, et en fait, il est possible d'imbriquer des requêtes dans des sous−requêtes, avec d'autres commandes. Une sous−requête doit toujours être entre parenthèses.

Depuis la version 4.1, MySQL supporte toutes les formes de sous−requêtes et opérations que le standard SQL requiert, ainsi que quelques fonctionnalités spécifiques. Les avantages des sous−requêtes sont :

- Elles permettent aux requêtes d'être **structuré** pour que chaque partie puisse être isolée.
- Elles fournissent une méthode pour réaliser des opérations qui seraient complexes, et impliqueraient des unions et jointures.

* Elles sont, au dire de nombreuses personnes, lisibles. En fait, c'est les sous−requêtes qui ont donné aux inventeurs le nom original de `SQL ``Structured Query Language''` .

Dans les versions plus anciennes de MySQL, il fallait trouver des palliatifs, et contourner les sous−requêtes. Il est bien plus facile de se mettre aux sous−requêtes.

Voici un exemple de commande qui montre les principaux avantages des sous−requêtes et de leur syntaxe, aussi bien pour celle qui est proposée par le standard, que celle de MySQL.

```
 DELETE FROM t1
WHERE s11 > ANY
 (SELECT COUNT(*) /* no hint */ FROM t2
 WHERE NOT EXISTS
  (SELECT * FROM t3
   WHERE ROW(5*t2.s1,77)=
    (SELECT 50,11*s1 FROM t4 UNION SELECT 50,77 FROM
     (SELECT * FROM t5) AS t5)));
```

Pour les versions de MySQL antérieure à la version 4.1, la plupart des sous requêtes peuvent être réécrites avec des jointures et d'autres méthodes. Se passer des sous−requêtes .

14.1.8.1 Les sous−requêtes comme opérateur scalaire

Dans dans forme la plus simple, une sous−requête **scalaire** , par opposition à une sous−requête de **ligne** ou de **table** qui seront présentées plus loin, est un simple opérande. Vous pouvez l'utilisez à chaque fois qu'une valeur de colonne ou qu'une valeur littérale est valide, et vous pouvez en attendre les mêmes caractéristiques : type de données, taille et indication de nullité, etc. Par exemple :

```
 CREATE TABLE t1 (s1 INT, s2 CHAR(5) NOT NULL);
SELECT (SELECT s2 FROM t1);
```

La sous−requête de la commande `SELECT` ci−dessus est de type `CHAR` , de longueur 5. Son jeu de caractères et sa collation sont ceux fournis par défaut, et elle porte une marque de nullité. En fait, toutes les sous−requêtes peuvent prendre la valeur `NULL` , car si la table est vide, la valeur de la sous−requête sera alors `NULL` . Il y a quelques restrictions :

* Une sous−requête peut être utilisée avec les commandes suivantes : `SELECT` , `INSERT` , `UPDATE` , `DELETE` , `SET` et `DO` .
* Une sous−requête peut contenir les mots−clé et les clauses qu'une commande `SELECT` peut contenir : `DISTINCT` , `GROUP BY` , `ORDER BY` , `LIMIT` , jointures, `UNION` , commentaires, fonctions, etc.

Ainsi, lorsque vous lirez les exemples des sections suivantes qui utilisent la commande spartiate `(SELECT column1 FROM t1)` , imaginez que votre code pourra contenir des commandes bien plus diverses et complexes.

Par exemple, supposons que nous avons ces deux tables :

```
 CREATE TABLE t1 (s1 INT);
INSERT INTO t1 VALUES (1);
CREATE TABLE t2 (s1 INT);
INSERT INTO t2 VALUES (2);
```

Puis, que vous envoyons la commande suivante `SELECT` :

```
 SELECT (SELECT s1 FROM t2) FROM t1;
```

Le résultat sera `2` car il y a une ligne dans `t2` , dont la colonne `s1` a une valeur de 2.

La sous−requête peut faire partie d'une expression. Si c'est un opérande d'une fonction, n'oubliez pas les parenthèses.

Par exemple :

```
SELECT UPPER((SELECT s1 FROM t1)) FROM t2;
```

14.1.8.2 Comparaisons avec les sous−requêtes

L'utilisation la plus répandue des sous−requêtes est celle−ci :

```
<non-subquery operand> <comparison operator> (<subquery>)
```

où `<comparison operator>` est l'un des opérateurs suivants :

```
= > < >= <= <>
```

Par exemple :

```
 ... 'a' = (SELECT column1 FROM t1)
```

Il fut un temps où la seule place possible pour une sous−requête était à la droite de l'opérateur de comparaison, mais vous pourrez rencontrer de vieilles bases qui insisteront sur ce point.Voici un exemple de comparaison classiques, pour lequel vous ne pouvez pas utiliser de jointure : trouvez toutes les valeurs de la table `t1` qui sont égales au maximum de la valeur dans la table `t2` .

```
SELECT column1 FROM t1
      WHERE column1 = (SELECT MAX(column2) FROM t2);
```

Voici un autre exemple, qui est aussi impossible à réaliser avec une jointure, car elle impose l'agrégation de plusieurs tables : trouver toutes les lignes de la table `t1` qui contiennent une valeur qui apparaît deux fois.

```
SELECT * FROM t1
      WHERE 2 = (SELECT COUNT(column1) FROM t1);
```

14.1.8.3 Sous−requêtes avec les clauses ANY , IN et SOME

Syntaxe :

```
 <operand> <comparison operator> ANY (<subquery>)
<operand> IN (<subquery>)
<operand> <comparison operator> SOME (<subquery>)
```

Le mot `ANY` , qui doit suivre immédiatement un opérateur de comparaison, signifie : ``retourne `TRUE` si la comparaison est `TRUE` pour `UNE` des lignes que la sous−requête retourne.'' Par exemple :

```
SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);
```

Supposons qu'il y ait une ligne dans la table `t1` qui contienne {10}. L'expression est `TRUE` si la table `t2` contient {21,14,7} car il y a une valeur de `t2` , 7, qui est inférieure à 10. Cette expression est `FALSE` si la table `t2` contient {20,10}, ou si la table `t2` est vide. L'expression est `UNKNOWN` si la table `t2` contient { `NULL` , `NULL` , `NULL` }.Le mot `IN` est un alias de `= ANY` . Les deux commandes suivantes sont identiques :

```
 SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN    (SELECT s1 FROM t2);
```

Le mot `SOME` est un alias de `ANY` . Les deux commandes suivantes sont identiques :

```
 SELECT s1 FROM t1 WHERE s1 <> ANY  (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);
```

L'utilisation du mot `SOME` est rare, mais les exemples ci−dessus montrent pourquoi il peut être utile. En langage parlé, ``a n'est pas égal à aucun b'' signifie pour la majorité des gens, ``il n'y a pas de b qui est égal à a'' : ce n'est pas la signification de la syntaxe SQL. En utilisant `<> SOME` , vous pouvez vous assurer que tout le monde comprend le véritable sens de la commande.

14.1.8.4 Sous−requêtes avec ALL

Syntaxe :

```
<operand> <comparison operator> ALL (<subquery>)
```

Le mot `ALL` , qui doit suivre immédiatement l'opérateur de comparaison, signifie ``retourne `TRUE` si la comparaison est `TRUE` pour `TOUTES` les lignes que la sous−requête retourne''.Par exemple :

```
SELECT s1 FROM t1 WHERE s1 > ALL (SELECT s1 FROM t2);
```

Supposons qu'il y ait une ligne dans la table `t1` contenant {10}. L'expression est `TRUE` si la table `t2` contient {−5,0,+5} car les trois valeurs de `t2` sont inférieures à 10. L'expression est `FALSE` si la table `t2` contient {12,6, `NULL` ,−100} car il y a une des valeurs de la table `t2` , ici 12, qui est plus grande que 10. L'expression est `UNKNOWN` si la table `t2` contient {0, `NULL` ,1}.Finalement, si la table `t2` est vide, le résultat est `TRUE` . Vous pouvez pensez que le résultat doit être indéterminé (`UNKNOWN`), mais c'est bien `TRUE` . Ce qui fait que, bizarrement,

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT s1 FROM t2);
```

est `TRUE` si la table `t2` est vide, mais

```
SELECT * FROM t1 WHERE 1 > (SELECT s1 FROM t2);
```

est `UNKNOWN` si la table `t2` est vide. De plus,

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT MAX(s1) FROM t2);
```

est `UNKNOWN` si la table `t2` est vide. En général, **les tables avec des** `NULL` et les **tables vides** sont **des cas particuliers** : lorsque vous écrivez vos sous−requêtes, pensez bien à les prendre en compte.

14.1.8.5 Sous−requêtes corrélées

Une **sous−requête corrélée** est une sous−requête qui contient une référence à une colonne qui est aussi dans une requête différente.

Par exemple :

```
SELECT * FROM t1 WHERE column1 = ANY
      (SELECT column1 FROM t2 WHERE t2.column2 = t1.column2);
```

Notez que dans notre exemple, la sous−requête contient une référence à une colonne de la table `t1` , même si la sous−requête de la clause `FROM` ne mentionne pas la table `t1` . MySQL recherche hors de la requête et trouve `t1` dans la requête externe.

Supposez que la table `t1` contienne une ligne où `column1 = 5` et `column2 = 6` ; alors que la table `t2` continue une ligne où `column1 = 5` et `column2 = 7` . l'expression `... WHERE column1 = ANY (SELECT column1 FROM t2)` sera alors `TRUE` , mais dans cet exemple, la clause `WHERE` de la sous−requête est `FALSE` (car 7 <> 5), et donc, toute la sous−requête est `FALSE` .

Règles de contexte : MySQL fait les évaluations de l'intérieur vers l'extérieur. Par exemple :

```
SELECT column1 FROM t1 AS x
 WHERE x.column1 = (SELECT column1 FROM t2 AS x
   WHERE x.column1 = (SELECT column1 FROM t3 WHERE x.column2 = t3.column1));
```

Dans l'exemple ci−dessus, `x.column2` doit être une colonne de la table `t2` car `SELECT column1 FROM t2 AS x ...` prend le nom de `t2` . ce n'est pas une colonne de la table `t1` car `SELECT column1 FROM t1 ...` est une requête externe, qui est **à venir** .Pour les sous−requêtes placées dans des clauses `HAVING` ou `ORDER BY` , MySQL recherche aussi les noms de colonnes dans la liste des sélections externes.

Dans certains cas, les sous−requêtes corrélées sont optimisées. Par exemple :

```
val IN (SELECT key_val FROM tbl_name WHERE correlated_condition)
```

Sinon, elles sont inefficaces et plutôt lentes. Réécrire une requête sous forme de jointure peut améliorer les performances.

14.1.8.6 EXISTS et NOT EXISTS

Si une sous−requête retourne absolument aucune valeur, alors la clause `EXISTS <subquery>` est `TRUE` , et la clause `NOT EXISTS <subquery>` est `FALSE` . Par exemple :

```
SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

Traditionnellement, une sous−requête qui `EXISTS` commence avec `SELECT *` mais elle peut commencer aussi bien avec `SELECT 5` ou `SELECT column1` ou n'importe quoi d'autre encore : MySQL ignore la liste de colonnes du `SELECT` de cette requête, ce qui fait que cela n'a pas d'importance.

Dans l'exemple ci−dessus, si la table `t2` ne contient aucune ligne, même pas de ligne avec uniquement des valeurs `NULL` , alors la condition `EXISTS` est `TRUE` . C'est un exemple plutôt exceptionnel, car il y a presque toujours une sous−requête `[NOT] EXISTS` qui contiendra des corrélations. Voici des exemples plus concrets :

- Quel type de magasin est le plus fréquent dans une ou plusieurs villes?
  ```
  SELECT DISTINCT store_type FROM Stores
   WHERE EXISTS (SELECT * FROM Cities_Stores
                 WHERE Cities_Stores.store_type = Stores.store_type);
  ```
- Quel type de magasin n'est présent dans aucune villes?
  ```
  SELECT DISTINCT store_type FROM Stores
   WHERE NOT EXISTS (SELECT * FROM Cities_Stores
                     WHERE Cities_Stores.store_type = Stores.store_type);
  ```
- Quel type de magasin est présent dans toutes les villes?
  ```
  SELECT DISTINCT store_type FROM Stores S1
   WHERE NOT EXISTS (
     SELECT * FROM Cities WHERE NOT EXISTS (
       SELECT * FROM Cities_Stores
        WHERE Cities_Stores.city = Cities.city
        AND Cities_Stores.store_type = Stores.store_type));
  ```

Le dernier exemple est une double imbrication de requête `NOT EXISTS` : elle possède une clause `NOT EXISTS` à l'intérieur de la clause `NOT EXISTS` . Formellement, elle répond à la question : ``Existe−t−il une ville avec un magasin qui n'est pas dans Stores?''. Mais il est plus facile de dire qu'une clause `NOT EXISTS` imbriquée répond à la question ``est−ce que x est vrai pour tous les y?''.

14.1.8.7 Sous−requêtes de ligne

Jusqu'ici, nous avons étudié les **sous−requêtes scalaires, ou de colonnes** : des sous−requêtes qui retournent une seule valeur dans une ligne. Une **sous−requête de ligne** est une variante qui retourne une seule ligne : elle peut donc retourner plusieurs colonnes. Voici deux exemples :

```
 SELECT * FROM t1 WHERE (1,2) = (SELECT column1, column2 FROM t2);
SELECT * FROM t1 WHERE ROW(1,2) = (SELECT column1, column2 FROM t2);
```

Les requêtes ci−dessus sont toutes les deux `TRUE` si la table `t2` a une ligne où `column1 = 1` et `column2 = 2` .L'expression `(1,2)` est parfois appelée un **constructeur de ligne** et est valide dans d'autres contextes. Par exemple, les deux commandes suivantes sont sémantiquement équivalentes, même si la précédente peut être optimisée :

```
 SELECT * FROM t1 WHERE (column1,column2) = (1,1);
SELECT * FROM t1 WHERE column1 = 1 AND column2 = 1;
```

L'utilisation traditionnelle des constructeurs de ligne est lors des comparaisons avec des sous−requêtes qui retournent plusieurs colonnes. Par exemple, cette requête répond à la question : ``trouve toutes les lignes de la table t1 qui sont dupliquées dans la table t2'':

```
SELECT column1,column2,column3
      FROM t1
      WHERE (column1,column2,column3) IN
            (SELECT column1,column2,column3 FROM t2);
```

14.1.8.8 Sous−requêtes dans la clause FROM

Les sous−requêtes sont valides dans la clause `FROM` d'une commande `SELECT` . Voici une syntaxe que vous allez rencontrer :

```
SELECT ... FROM (<subquery>) AS <name> ...
```

La clause `AS <name>` est obligatoire, car les tables de la clause `FROM` doivent avoir un nom. Toutes les colonnes de la sous−requête `<subquery>` doivent avoir des noms distincts. Vous pourrez trouver cette syntaxe décrite ailleurs dans ce manuel, sous le nom de ``tables dérivées''.Par exemple, supposons que vous avons cette table :

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5), s3 FLOAT);
```

Comment utiliser la fonctionnalité de sous−requêtes dans la clause `FROM` , avec cette table d'exemple :

```
 INSERT INTO t1 VALUES (1,'1',1.0);
INSERT INTO t1 VALUES (2,'2',2.0);
SELECT sb1,sb2,sb3
       FROM (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) AS sb
       WHERE sb1 > 1;
```

Résultat : 2, '2', 4.0.

Voici un autre exemple : supposons que vous voulez connaître la moyenne de la somme pour un groupe de table. Ceci ne fonctionnera pas :

```
SELECT AVG(SUM(column1)) FROM t1 GROUP BY column1;
```

Mais cette requête−ci vous donnera les informations nécessaires :

```
SELECT AVG(sum_column1)
       FROM (SELECT SUM(column1) AS sum_column1
             FROM t1 GROUP BY column1) AS t1;
```

Notez que les colonnes sont nommées à partir de la sous−requête : `(sum_column1)` est reconnue dans la requête externe.Actuellement, les sous−requêtes en clause `FROM` ne peuvent pas être corrélées.

14.1.8.9 Erreurs de sous−requêtes

Il y a de nouvelles erreurs qui ne s'appliquent qu'aux sous−requêtes. Cette section les rassemble, car elles vous aideront à garder en tête certains points importants.

- ```
   ERROR 1235 (ER_NOT_SUPPORTED_YET)
  SQLSTATE = 42000
  Message = "This version of MySQL doesn't yet support
  'LIMIT & IN/ALL/ANY/SOME subquery'"
  ```
  Cela signifie que
  ```
   SELECT * FROM t1 WHERE s1 IN (SELECT s2 FROM t2 ORDER BY s1 LIMIT 1)
  ```
  ne fonctionnera pas, mais uniquement dans certaines versions d'origines, comme MySQL 4.1.1.
- ```
   ERROR 1240 (ER_CARDINALITY_COL)
  SQLSTATE = 21000
  Message = "Operand should contain 1 column(s)"
  ```
  Cette erreur va survient dans des cas comme celui−ci :
  ```
   SELECT (SELECT column1, column2 FROM t2) FROM t1;
  ```
  Il est valide d'utiliser une sous−requête qui utilise plusieurs colonnes, dans le cadre d'une comparaison. Sous−requêtes de ligne . Mais dans d'autres contextes, la sous−requête doit être un opérande scalaire.
- ```
   ERROR 1241 (ER_SUBSELECT_NO_1_ROW)
  SQLSTATE = 21000
  Message = "Subquery returns more than 1 row"
  ```
  Cette erreur survient dans des cas comme celui−ci :
  ```
   SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
  ```
  ```

mais uniquement lorsque plus d'une ligne sont extraites de `t2` . Cela signifie que cette erreur peut survenir dans du code qui fonctionne depuis longtemps : quelqu'un vient de modifier le nombre de ligne que la requête retourne. N'oubliez pas que si votre but est de trouver un nombre arbitraire de lignes, et non pas juste une seule, la commande correcte aurait été :

```
SELECT * FROM t1 WHERE column1 = ANY (SELECT column1 FROM t2);
```

- ```
  Error 1093 (ER_UPDATE_TABLE_USED)
  SQLSTATE = HY000
  Message = "You can't specify target table 'x' for update in FROM clause"
  ```

  Cette erreur survient dans des cas comme celui−ci :

  ```
  UPDATE t1 SET column2 = (SELECT MAX(column1) FROM t1);
  ```

Il est valide d'utiliser une sous−requête lors d'une affectation dans une commande `UPDATE` , car les sous−requêtes sont valides avec les commandes `UPDATE` et `DELETE` , tout comme dans les commandes `SELECT` . Cependant, vous ne pouvez pas les utiliser sur la même table, qui est ici `t1` , car cette table est alors la cible de la clause `FROM` et de la commande `UPDATE` .

Généralement, l'échec d'un sous−requête entraîne l'échec de toute la commande.

14.1.8.10 Optimisation des sous−requêtes

Le développement des sous−requêtes se poursuit, et aucun des conseils d'optimisation ne sera valable longtemps. Voici quelques astuces que vous voulez prendre en compte :

- Utiliser une clause de sous−requête pour affecter le nombre ou l'ordre des lignes dans une sous−requête, par exemple :

  ```
  SELECT * FROM t1 WHERE t1.column1 IN
    (SELECT column1 FROM t2 ORDER BY column1);
  SELECT * FROM t1 WHERE t1.column1 IN
    (SELECT DISTINCT column1 FROM t2);
  SELECT * FROM t1 WHERE EXISTS
    (SELECT * FROM t2 LIMIT 1);
  ```
- Remplacer une jointure par une sous−requête. Par exemple :

  ```
  SELECT DISTINCT column1 FROM t1 WHERE t1.column1 IN (
   SELECT column1 FROM t2);
  ```

  au lieu de

  ```
  SELECT DISTINCT t1.column1 FROM t1, t2
   WHERE t1.column1 = t2.column1;
  ```
- Déplacer une clause `FROM` externe dans une sous−requête, comme ceci :

  ```
  SELECT * FROM t1
   WHERE s1 IN (SELECT s1 FROM t1 UNION ALL SELECT s1 FROM t2);
  ```

  au lieu de

  ```
  SELECT * FROM t1
   WHERE s1 IN (SELECT s1 FROM t1) OR s1 IN (SELECT s1 FROM t2);
  ```

  Un autre exemple :

  ```
  SELECT (SELECT column1 + 5 FROM t1) FROM t2;
  ```

  au lieu de

  ```
  SELECT (SELECT column1 FROM t1) + 5 FROM t2;
  ```
- Utiliser une sous−requête de ligne plutôt qu'une corrélation. Par exemple :

  ```
  SELECT * FROM t1
   WHERE (column1,column2) IN (SELECT column1,column2 FROM t2);
  ```

  au lieu de

  ```
  SELECT * FROM t1
   WHERE EXISTS (SELECT * FROM t2 WHERE t2.column1=t1.column1
   AND t2.column2=t1.column2);
  ```
- Utiliser `NOT (a = ANY (...))` au lieu de `a <> ALL (...)` .
- Utiliser `x = ANY (table containing {1,2})` plutôt que `x=1 OR x=2` .
- Utiliser `= ANY` de préférence à `EXISTS`

Le truc ci−dessus peut accélérer certains programmes, et en ralentir d'autres. En utilisant la fonction utilitaire `BENCHMARK()` , vous pouvez obtenir une idée de votre cas. Ne vous attardez pas trop à transformer vos jointures, sauf si la compatibilité avec les anciennes versions est importante pour vous.
  ```

Quelques optimisation que MySQL va prendre en charge lui−même :

- MySQL va exécuter les sous−requêtes non corrélées une seule fois (utilisez la commande `EXPLAIN` pour vous assurer que les requêtes ne sont pas correllées).
- MySQL va transformer les sous−requêtes `IN` / `ALL` / `ANY` / `SOME` pour essayer de profiter de la possibilité que les colonnes sélectionnées dans la sous−requêtes sont indexées.
- MySQL remplacera les sous−requêtes de la forme
  ```
  ... IN (SELECT indexed_column FROM single_table ...)
  ```
  par une recherche dans un index, que `EXPLAIN` décrira comme une jointure de type spécial.
- MySQL va améliorer les expressions de la forme
  ```
  valeur &#123;ALL|ANY|SOME&#125; &#123;> | < | >= | <=&#125; (sous-requête non-correllée)
  ```
  avec une expression impliquant `MIN` ou `MAX` (à moins d'une valeur `NULL` ou d'ensembles `SET` vides). Par exemple,
  ```
  WHERE 5 > ALL (SELECT x FROM t)
  ```
  revient à
  ```
  WHERE 5 > (SELECT MAX(x) FROM t)
  ```

Il y a un chapitre intitulé ``Comment MySQL adapte les sous−requêtes'' dans les manuels internes de MySQL, que vous pouvez trouver en téléchargeant les sources de MySQL : il est dans un fichier appelé `internals.texi` , dans le dossier `Docs` .

14.1.8.11 Se passer des sous−requêtes avec les premières versions de MySQL

Jusqu'à la version 4.0 inclue, seules les requêtes imbriquées de la forme `INSERT ... SELECT ...` et `REPLACE ... SELECT ...` étaient supportées. La clause `IN()` peut être utilisée dans certains contextes.

Il est souvent possible de réécrire une requête sans sous−requête :

```
SELECT * FROM t1 WHERE id IN (SELECT id FROM t2);
```

Cela peut se réécrire :

```
SELECT t1.* FROM t1,t2 WHERE t1.id=t2.id;
```

Les requêtes :

```
 SELECT * FROM t1 WHERE id NOT IN (SELECT id FROM t2);
SELECT * FROM t1 WHERE NOT EXISTS (SELECT id FROM t2 WHERE t1.id=t2.id);
```

peuvent être réécrites :

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id
                                    WHERE table2.id IS NULL;
```

Une clause `LEFT [OUTER] JOIN` peut être plus rapide qu'une sous−requête équivalent, car le serveur va pouvoir l'optimiser bien mieux : c'est un fait qui n'est pas spécifique à MySQL. Avant SQL−92, les jointures externes n'existaient pas, et les sous−requêtes étaient la seule méthode pour résoudre certains problèmes. Aujourd'hui, le serveur MySQL et d'autres bases de données modernes offrent toute une gamme de jointures externes.Pour les sous−requêtes plus complexes, vous pouvez simplement créer des tables temporaires pour contenir les résultats intermédiaires. Dans certains cas, cela ne sera pas possible. C'est notamment le cas des commandes de type DELETE, pour lesquelles le standard SQL ne supporte pas les jointures, sauf pour les sous−requêtes. Dans ces situations, trois solutions s'offrent à vous :

- Passez en MySQL version 4.1.
- Utilisez un langage de programmation procédural, comme Perl ou PHP, pour envoyer la requête `SELECT` , lire les clés primaires à effacer, et utiliser ces valeurs pour soumettre des requêtes de type `DELETE` (`DELETE FROM ... WHERE ... IN (key1, key2, ...)`).
- La troisième option est d'utiliser le client interactif de mysql pour construire et exécuter une liste de commande `DELETE` automatiquement, avec la fonction `CONCAT()` au lieu de l'opérateur `||` . Par exemple :
  ```
  SELECT CONCAT('DELETE FROM tab1 WHERE pkid = ', "'", tab1.pkid, "'", ';')
   FROM tab1, tab2
  WHERE tab1.col1 = tab2.col2;
  ```

Vous pouvez placer cette requête dans un fichier de script, et rediriger son résultat vers le client en ligne de commande `mysql` , pour que ce dernier lance une seconde instance de l'interprêteur :

```
shell> mysql --skip-column-names mydb < myscript.sql | mysql mydb
```

MySQL 4.0 supporte les commandes `DELETE` multi−tables qui peuvent être utilisées pour effacer des lignes dans une table en fonction d'informations qui sont dans une autre table, ou même effacer des lignes simultanément dans plusieurs tables. Les commandes `UPDATE` multi−tables sont aussi supportés depuis la version 4.0.

14.1.9 Syntaxe de TRUNCATE

```
TRUNCATE TABLE nom_de_table
```

Dans la version 3.23, `TRUNCATE TABLE` est équivalent à `COMMIT ; DELETE FROM nom_de_table` . Syntaxe de `DELETE` .

`TRUNCATE TABLE` diffère de `DELETE FROM ...` des façons suivantes :

- Implémentée comme une destruction/création de table, ce qui accélère la suppression des enregistrements.
- Ne respecte pas les transactions. Vous aurez des erreurs si vous avez une transaction active ou une table protégée en écriture.
- Ne retourne pas le nombre de lignes effacées.
- Tant que le fichier de définition `nom_de_table.frm` est valide, la table peut être recréée, me si les données ou un index a été corrompu.
- Le gestionnaire de table ne se souvient pas de la dernière valeur `AUTO_INCREMENT` utilisée, mais peut commencer à compter depuis le début. C'est vrai pour les tables `MyISAM` , `ISAM` et `BDB` .

`TRUNCATE` est une extension Oracle SQL. Cette commande a été ajoutée en MySQL 3.23.28, même si dans les versions 3.23.28 à 3.23.32, le mot clé `TABLE` devait être omis.

14.1.10 Syntaxe de UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Syntaxe multi−tables :

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name [, tbl_name ...]
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
```

`UPDATE` met à jour des enregistrements dans une tables avec de nouvelles valeurs. La clause `SET` indique les colonnes à modifier et les valeurs à leur donner. La clause `WHERE` , si fournie, spécifie les enregistrements à mettre à jour. Sinon, tous les enregistrements sont mis à jour. Si la clause `ORDER BY` est fournie, les enregistrements seront mis à jour dans l'ordre spécifié.

La commande `UPDATE` accepte les options suivantes :

- Si vous spécifiez le mot clef `LOW_PRIORITY` , l'exécution de l' `UPDATE` sera repoussé jusqu'à ce que aucun client ne lise plus de la table.
- Si vous spécifiez le mot clef `IGNORE` , la mise à jour ne s'interrompra pas même si on rencontre des problèmes d'unicité de clefs durant l'opération. Les enregistrements posant problèmes

ne seront pas mis à jour.

Si vous accédez à une colonne d'une table dans une expression, `UPDATE` utilisera la valeur courante de la colonne. Par exemple, la requête suivante ajoute une année à l'âge actuel de tout le monde :

```
mysql> UPDATE persondata SET age=age+1;
```

Les requêtes `UPDATE` sont évaluées de gauche à droite. Par exemple, la requête suivante double la valeur de la colonnes âge, puis l'incrémente :

```
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

Si vous changez la valeur d'une colonne en lui spécifiant sa valeur actuelle, MySQL s'en aperçoit et ne fait pas la mise à jour. `UPDATE` retourne le nombre d'enregistrements ayant changé. Depuis la version 3.22 de MySQL, la fonction `mysql_info()` de l'API C retourne le nombre de colonnes qui correspondaient, le nombre de colonnes mises à jour et le nombre d'erreurs générées pendant l' `UPDATE` .

Dans la version 3.23 de MySQL, vous pouvez utilisez le code `LIMIT #` pour vous assurer que seul un nombre d'enregistrements bien précis est changé.

- Avant MySQL 4.0.13, `LIMIT` est une restrictions sur le nombre de lignes affectées. Cette clause stoppe dès que `row_count` ont été trouvées par la clause `WHERE` .
- Depuis la version 4.0.13, `LIMIT` est une restriction sur le nombre de lignes trouvées. La commande s'arrête une fois que `row_count` lignes ont été trouvées par la clause `WHERE` , qu'elles ait été changées ou pas.

Ai une clause `ORDER BY` est utilisée (disponible depuis MySQL version 4.0.0), les lignes seront modifiées selon cet ordre. Ce n'est vraiment utile qu'en conjonction avec `LIMIT` .

Depuis MySQL version 4.0.4, vous pouvez aussi faire des opérations de `UPDATE` qui couvrent plusieurs tables :

```
 UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

L'exemple ci−dessus montre une jointure interne, en utilisant la virgule comme séparateur, mais une commande `UPDATE` multi−table peut utiliser n'importe quel type de jointure autorisée dans une commande `SELECT` , tel qu'un `LEFT JOIN` .Note : vous ne pouvez pas utiliser `ORDER BY` ou `LIMIT` avec les `UPDATE` multi−table.

14.2 Définition de données : CREATE , DROP , ALTER

14.2.1 Syntaxe de ALTER DATABASE

```
ALTER DATABASE db_name
   alter_specification [, alter_specification] ...

alter_specification:
   [DEFAULT] CHARACTER SET charset_name
 | [DEFAULT] COLLATE collation_name
```

`ALTER DATABASE` vous permet de modifier les caractéristiques générales d'une base de données. Ces caractéristiques sont stockées dans le fichier `db.opt` du dossier de base. Pour utiliser `ALTER DATABASE` , vous avez besoin des droits de `ALTER` sur la base.

La clause `CHARACTER SET` modifie le jeu de caractères par défaut de la base. La clause `COLLATE` modifie la collation par défaut de la base. Les noms de jeu de caractères et de collation sont présentés dans la section Jeux de caractères .

`ALTER DATABASE` a été ajoutée MySQL 4.1.1.

14.2.2 Syntaxe de ALTER TABLE

```
ALTER [IGNORE] TABLE tbl_name
   alter_specification [, alter_specification] ...

alter_specification:
   ADD [COLUMN] column_definition [FIRST | AFTER col_name ]
  | ADD [COLUMN] (column_definition,...)
  | ADD INDEX [index_name] [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
       PRIMARY KEY [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
       UNIQUE [index_name] [index_type] (index_col_name,...)
  | ADD [FULLTEXT|SPATIAL] [index_name] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
       FOREIGN KEY [index_name] (index_col_name,...)
       [reference_definition]
  | ALTER [COLUMN] col_name &#123;SET DEFAULT literal | DROP DEFAULT&#125;
  | CHANGE [COLUMN] old_col_name column_definition
       [FIRST|AFTER col_name]
  | MODIFY [COLUMN] column_definition [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | table_options
```

`ALTER TABLE` vous permet de changer la structure d'une table existante. Par exemple, vous pouvez ajouter ou supprimer des colonnes, des index, changer le type des colonnes existantes, renommer ces colonnes, ou la table elle−même. Vous pouvez de même changer le commentaire sur la table, ou le type de celle−ci.La syntaxe de nombreuses altérations est similaires aux clauses de la commande `CREATE TABLE` . Syntaxe de `CREATE TABLE` .

Si vous utilisez `ALTER TABLE` pour modifier les spécifications d'une colonne mais que `DESCRIBE nom_de_table` vous indique que cette colonne n'a pas été modifiée, il est possible que MySQL ait ignoré vos modifications pour une des raisons décrite dans Changement de type de colonne automatique . Par exemple, si vous essayez de changer une colonne de type `VARCHAR` en `CHAR` , MySQL continuera d'utiliser `VARCHAR` si la table contient d'autres colonnes de taille variable.

`ALTER TABLE` effectue une copie temporaire de la table originale. Les modifications sont faites sur cette copie, puis l'original est effacée, et enfin la copie est renommée pour remplacer l'originale. Cette méthode permet de rediriger toutes les commandes automatiquement vers la nouvelle table sans pertes. Durant l'exécution de `ALTER TABLE` , la table originale est lisible par d'autres clients. Les modifications et insertions sont reportées jusqu'à ce que la nouvelle table soit prête.

Notez que si vous utilisez une autre option que `RENAME` avec `ALTER TABLE` , MySQL créera toujours une table temporaire, même si les données n'ont pas besoin d'être copiées (comme quand vous changez le nom d'une colonne). Nous avons prévu de corriger cela dans les versions suivantes, mais comme la commande `ALTER TABLE` n'est pas utilisée très souvent, cette correction ne fait pas partie de nos priorités. Pour les tables `MyISAM` , vous pouvez accélérer la réindexation (qui est la partie la plus lente de la modification d'une table) en donnant à la variable système `myisam_sort_buffer_size` une valeur plus grande.

* Pour utiliser `ALTER TABLE`, vous devez avoir les droits `ALTER`, `INSERT`, et `CREATE` sur la table.
* `IGNORE` est une extension MySQL pour ANSI SQL92. Cette option contrôle la façon dont `ALTER TABLE` fonctionne s'il y a des duplications sur une clef unique de la nouvelle table. Si `IGNORE` n'est pas spécifiée, la copie est annulée et la table originale est restaurée. Si `IGNORE` est spécifiée, les lignes contenant les éléments doublons de la table seront effacées, hormis la première, qui sera conservée.
* Vous pouvez effectuer plusieurs opérations de `ADD`, `ALTER`, `DROP`, et `CHANGE` dans une même commande `ALTER TABLE`. C'est une extension de MySQL à la norme ANSI SQL92, qui n'autorise qu'une seule modification par commande `ALTER TABLE`.
* `CHANGE nom_colonne`, `DROP nom_colonne`, et `DROP INDEX` sont des extensions de MySQL à la norme ANSI SQL92.
* `MODIFY` est une extension Oracle à `ALTER TABLE`.
* Le mot optionnel `COLUMN` est purement de la fioriture et peut être ignoré.
* Si vous utilisez `ALTER TABLE nom_de_table RENAME TO nouveau_nom` sans autre option, MySQL va simplement renommer les fichiers qui correspondent à la table `nom_de_table`. Il n'y a pas de création de fichier temporaire. Syntaxe de `RENAME TABLE`.
* La définition `create_definition` utilise la même syntaxe pour les clauses `ADD` et `CHANGE` que dans `CREATE TABLE`. Notez que cette syntaxe inclut le nom de la colonne, et pas seulement son type Syntaxe de `CREATE TABLE`.
* Vous pouvez renommer une colonne avec la syntaxe `CHANGE ancien_nom_de_colonne create_definition`. Pour cela, indiquez l'ancien nom de la colonne, puis le nouveau nom et son type courant. Par exemple, pour renommer une colonne de type `INTEGER`, de `a` en `b`, vous pouvez faire ceci :

```
mysql> ALTER TABLE t1 CHANGE a b INTEGER;
```

  Si vous ne voulez changer que le type de la colonne, avec la clause `CHANGE` vous devrez redonner le nom de la colonne. Par exemple :

```
mysql> ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
```

  Cependant, à partir de la version 3.22.16a de MySQL, vous pouvez aussi utiliser la clause `MODIFY` pour changer le type d'une colonne sans la renommer :

```
mysql> ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
```

* Si vous utilisez les clauses `CHANGE` ou `MODIFY` pour réduire la taille d'une colonne qui comportait un index sur une partie de la colonne (par exemple, si vous aviez un index sur 10 caractères d'une colonne de type `VARCHAR`), vous ne pouvez pas rendre la colonne plus petite que le nombre de caractères indexés.
* Quand vous changez le type d'une colonne avec `CHANGE` ou `MODIFY`, MySQL essaye de convertir les données au niveau type dans la mesure du possible.
* A partir de la version 3.22 de MySQL, vous pouvez utiliser `FIRST` ou `ADD ... AFTER nom_colonne` pour ajouter la colonne à un endroit spécifique dans la table. Par défaut, la colonne est ajoutée à la fin. A partir de la version 4.0.1, vous pouvez aussi utiliser les mots clés `FIRST` et `AFTER` avec `CHANGE` ou `MODIFY`.
* `ALTER COLUMN` spécifie une nouvelle valeur par défaut pour une colonne ou enlève l'ancienne. si l'ancienne valeur est effacée et que la colonne peut être `NULL`, la nouvelle valeur par défaut sera `NULL`. Si la colonne ne peut être `NULL`, MySQL assigne une valeur par défaut, comme défini dans Syntaxe de `CREATE TABLE`.
* `DROP INDEX` supprime un index. C'est une extension MySQL à la norme ANSI SQL92. Syntaxe de `DROP INDEX`.
* Si des colonnes sont effacées d'une table, ces colonnes sont aussi supprimés des index dont elles font partie. Si toutes les colonnes qui forment un index sont effacées, l'index lui même est supprimé.
* Si une table ne comporte qu'une seule colonne, La colonne ne peut être supprimée. Si vous voulez effacer la table, utilisez la commande `DROP TABLE`.
* `DROP PRIMARY KEY` supprime la clef primaire. Si cette clef n'existe pas, cette commande effacera le premier index `UNIQUE` de la table. (MySQL marque la première clef `UNIQUE` en tant que `PRIMARY KEY` si aucune `PRIMARY KEY` n'a été spécifiée explicitement.)

Si vous ajoutez un `UNIQUE INDEX` ou `PRIMARY KEY` à une table, c'est enregistré avant les index non− `UNIQUE` pour que MySQL puisse détecter les valeurs dupliquées aussi vite que possible.
- `ORDER BY` vous permet de créer une nouvelle table tout en ordonnant les lignes par défaut. Notez que cet ordre ne sera pas conservé après les prochaines insertions et modifications. Dans certains cas, cela aide MySQL si les colonnes sont dans l'ordre dans lequel vous allez trier les valeurs. Cette option n'est vraiment utile que si vous savez à l'avance dans quel ordre vous effectuerez les tris : vous y gagnerez alors en performances.
- Si vous utilisez `ALTER TABLE` sur une table `MyISAM` , tous les index non−uniques sont créés par des opérations séparées. (comme dans `REPAIR`). Cela devrait rendre `ALTER TABLE` plus rapide quand vous avez beaucoup d'index.Depuis la version 4.0, la fonctionnalité ci−dessus peut être activée explicitement. `ALTER TABLE ... DISABLE KEYS` force MySQL à ne plus mettre à jour les index non−uniques pour les tables au format `MyISAM` . `ALTER TABLE ... ENABLE KEYS` doit alors être utilisé pour recréer les index manquants. Comme MySQL le fait avec un algorithme spécial qui est plus rapide que le fait d'insérer les clefs une par une, désactiver les clefs peut vous faire gagner en performances.
- Les clauses `FOREIGN KEY` et `REFERENCES` sont supportées par le moteur de tables `InnoDB` , qui implémente les clauses `ADD [CONSTRAINT [symbol]] FOREIGN KEY (...) REFERENCES ... (...)` . Contraintes `FOREIGN KEY` . Pour les autres moteurs de stockages, ces clauses sont lues mais ignorées. La clause `CHECK` est analysée mais ignorée par tous les moteurs de stockage. Syntaxe de `CREATE TABLE` . La raison pour accepter mais ignorer ces clauses est que cela renforce la compatibilité avec le code des autres serveurs SQL, et qu'il est possible de créer des tables avec des références. Différences entre MySQL et ANSI SQL92 .
- Depuis MySQL 4.0.13, `InnoDB` supporte l'utilisation de `ALTER TABLE` pour effacer des clés étrangères :

```
ALTER TABLE yourtablename
   DROP FOREIGN KEY fk_symbol
```

  Pour plus d'informations, voyez Contraintes `FOREIGN KEY` .
- `ALTER TABLE` ignore les options de tables `DATA DIRECTORY` et `INDEX DIRECTORY` .
- Depuis MySQL 4.1.2, si vous voulez changer dans toutes les colonnes de texte (`CHAR` , `VARCHAR` , `TEXT`) le jeu de caractères, vous pouvez utiliser la commande suivante :

```
ALTER TABLE tbl_name CONVERT TO CHARACTER SET charset_name;
```

  C'est pratique lorsque vous passez de MySQL 4.0.x en 4.1.x. Mise à jour depuis MySQL 4.0 .

  Attention : l'opération précédente va convertir les valeurs des colonnes entre les deux jeux de caractères. Ce **n'est pas** ce que vous souhaitez faire si une colonne est de type `latin1` mais que les valeurs sont en fait dans un autre jeu de caractères (comme `utf8`). Dans ce cas, vous devez faire ceci avec une telle colonne :

```
 ALTER TABLE t1 CHANGE c1 c1 BLOB;
ALTER TABLE t1 CHANGE c1 c1 TEXT CHARACTER SET utf8;
```

  La raison est que dans ce cas, il n'y aura pas de conversion lorsque vous passer en type `BLOB` .

  Pour ne changer que le type de caractères par **défaut** , utilisez cette commande :

```
ALTER TABLE tbl_name DEFAULT CHARACTER SET charset_name;
```

  Le mot `DEFAULT` est optionnel. Le jeu de caractères par défaut est utilisé si vous ne spécifiez pas le jeu de caractères de la colonne explicitement, lorsque vous ajoutez une nouvelle colonne : par exemple, avec `ALTER TABLE ... ADD column` .

  Attention : depuis MySQL 4.1.2 et plus récent, `ALTER TABLE ... DEFAULT CHARACTER SET` et `ALTER TABLE ... CHARACTER SET` sont équivalent et ne changent que le jeu de caractères par défaut. Dans les versions antérieures à MySQL 4.1.2, `ALTER TABLE ... DEFAULT CHARACTER SET` changeait le jeu de caractères par défaut, mais `ALTER TABLE ... CHARACTER SET` (sans `DEFAULT`) changeait le jeu de caractères par défaut, et convertissaient les colonnes dans le nouveau jeu.

- Pour une table `InnoDB` qui a été créée avec son propre espace de tables dans un fichier `.ibd` , ce fichier peut être supprimé et importé. Pour supprimer le fichier `.ibd` , utilisez la commande suivante :
  ```
  ALTER TABLE tbl_name DISCARD TABLESPACE;
  ```
  Elle efface le fichier `.ibd` courant, alors assurez vous que vous avez une copie de sauvegarde. Si vous tentez d'accéder à un espace de table sans ce fichier, vous obtiendrez une erreur.Pour importer un fichier de sauvegarde `.ibd` dans la table, copiez le nouveau fichier dans le dossier de la base, et utilisez cette commande :
  ```
  ALTER TABLE tbl_name IMPORT TABLESPACE;
  ```
  Espaces de tables multiples : mettre un fichier par table .
- Avec la fonction `mysql_info()` de l'API C, vous pouvez savoir combien d'enregistrements ont été copiés, et (quand `IGNORE` est spécifié) combien d'enregistrements ont été effacés à cause de la clef unique. `mysql_info()` .

Voilà un exemple qui montre quelques utilisations de `ALTER TABLE` . On commence par une table `t1` créée comme suit :

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Pour renommer la table de `t1` à `t2` :

```
mysql> ALTER TABLE t1 RENAME t2;
```

Pour changer une colonne `a` de `INTEGER` en `TINYINT NOT NULL` (en laissant le même nom), et pour changer une colonne `b` de `CHAR(10)` à `CHAR(20)` et la renommant de `b` en `c` :

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Pour ajouter une nouvelle colonne `TIMESTAMP` nommée `d` :

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Pour ajouter un index sur une colonne `d` , et rendre la colonne `a` la clef primaire :

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Pour effacer la colonne `c` :

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Pour ajouter une nouvelle colonne `AUTO_INCREMENT` nommée `c` :

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
          ADD INDEX (c);
```

Notez que nous avons indexé `c` , car les colonnes `AUTO_INCREMENT` doivent être indexées, et que nous définissons aussi `c` en tant que `NOT NULL` , car les colonnes indexées ne peuvent être `NULL` .Quand vous ajoutez une colonne `AUTO_INCREMENT` , les valeurs de la colonne sont remplies automatiquement pour vous. Vous pouvez choisir la valeur de départ pour l'indexation en utilisant `SET INSERT_ID=#` avant `ALTER TABLE` ou en utilisant l'option `AUTO_INCREMENT = #` de la table. Syntaxe de `SET` .

Avec les tables de type `MyISAM` , si vous ne changez pas la colonne `AUTO_INCREMENT` , l'indice d'auto−incrémentation ne sera pas affecté. Si vous effacez une colonne `AUTO_INCREMENT` puis en ajoutez une autre, l'indexation recommencera à partir de 1.

Problèmes avec `ALTER TABLE` . .

14.2.3 Syntaxe de CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] db_name
   [create_specification [, create_specification] ...]

create_specification:
   [DEFAULT] CHARACTER SET charset_name
 | [DEFAULT] COLLATE collation_name
```

`CREATE DATABASE` crée une base de données avec le nom donné.

Les règles de nommage des bases de donnée sont présentées dans la section Noms de base de données, tables, index, colonnes et alias . Une erreur survient si une base de données de même nom existe déjà, si vous ne spécifiez pas l'option `IF NOT EXISTS` .

Depuis MySQL 4.1.1, les options `create_specification` peuvent être données pour spécifier des caractéristiques de la base. Les caractéristiques de la base sont stockées dans le fichier `db.opt` dans le dossier de la base. La clause `CHARACTER SET` spécifie le jeu de caractères par défaut pour les tables de cette base. La clause `COLLATE` spécifie la collation par défaut de la base de données. Les jeux de caractères et les collations sont présentées dans la section Jeux de caractères .

Les bases de données MySQL sont implémentées comme des répertoires contenant des fichiers qui correspondent aux tables dans les bases de données. Puisqu'il n'y a pas de tables dans une base de données lors de sa création, la requête `CREATE DATABASE` créera seulement le dossier dans le répertoire de données de MySQL (et le fichier `db.opt` , depuis MySQL 4.1.1).

Vous pouvez aussi créer des bases de données avec `mysqladmin` . `mysqladmin` , administrer un serveur MySQL .

14.2.4 Syntaxe de CREATE INDEX

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name [index_type]
    ON tbl_name (index_col_name,...)

index_col_name:
    col_name [(length)] [ASC | DESC]
```

La requête `CREATE INDEX` n'effectue aucune action sur les versions de MySQL antérieures à la version 3.22. Dans les versions 3.22 et supérieures, `CREATE INDEX` est équivalent à une requête `ALTER TABLE` pour créer des index. Syntaxe de `ALTER TABLE` .Normalement, tous les index sont créés en même temps que la table elle même avec `CREATE TABLE` . Syntaxe de `CREATE TABLE` . `CREATE INDEX` permet d'ajouter des index à une table existante.

Une liste de colonnes de la forme `(col1,col2,...)` crée un index multi−colonnes. Les valeurs de l'index sont créées en concaténant la valeur deux colonnes données.

Pour les colonnes `CHAR` et `VARCHAR` , les index peut être créés sur uniquement une partie de la colonne, avec la syntaxe `col_name(length)` . Pour les colonnes `BLOB` et `TEXT` la longueur d'index est **obligatoire** . La requête suivante crée un index en utilisant les 10 premiers caractères de la colonne `name` :

```
mysql> CREATE INDEX part_of_name ON customer (name(10));
```

Comme la plupart des noms ont en général des différences dans les 10 premiers caractères, l'index ne devrait pas être plus lent qu'un index créé à partir de la colonne `name` en entier. Ainsi, en n'utilisant qu'une partie de la colonne pour les index, on peut réduire la taille du fichier d'index, ce qui peut permettre d'économiser beaucoup d'espace disque, et peut aussi accélérer les opérations `INSERT` !Il est important de savoir qu'on peut indexer une colonne qui peut avoir la valeur `NULL` ou une colonne `BLOB` / `TEXT` que si on utilise une version 3.23.2 ou supérieure de MySQL et en utilisant le type `MyISAM` .

Pour plus d'informations à propos de l'utilisation des index dans MySQL, voir Comment MySQL utilise les index .

Les index `FULLTEXT` ne peuvent indexer que des colonnes `VARCHAR` ou `TEXT` , et seulement dans les tables `MyISAM` . Les index `FULLTEXT` sont disponibles dans les versions 3.23.23 et supérieures de MySQL. Fonctions de recherche en texte plein .

Les index `SPATIAL` peuvent indexer les colonnes spatiales, et uniquement avec les tables `MyISAM` . Les index SPATIAL sont disponibles en MySQL 4.1 et plus récent. Les colonnes spatiales sont présentées dans section Extensions spatiale de MySQL .

14.2.5 Syntaxe de CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   [(create_definition,...)]
   [table_options] [select_statement]
```

ou :

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   [(] LIKE old_tbl_name [)];

create_definition:
   column_definition
 | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
 | KEY [index_name] [index_type] (index_col_name,...)
 | INDEX [index_name] [index_type] (index_col_name,...)
 | [CONSTRAINT [symbol]] UNIQUE [INDEX]
       [index_name] [index_type] (index_col_name,...)
 | [FULLTEXT|SPATIAL] [INDEX] [index_name] (index_col_name,...)
 | [CONSTRAINT [symbol]] FOREIGN KEY
       [index_name] (index_col_name,...) [reference_definition]
 | CHECK (expr)

column_definition:
   col_name type [NOT NULL | NULL] [DEFAULT default_value]
       [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
       [reference_definition]

type:
   TINYINT[(length)] [UNSIGNED] [ZEROFILL]
 | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
 | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
 | INT[(length)] [UNSIGNED] [ZEROFILL]
 | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
 | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
 | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
 | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
 | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
 | DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
 | NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
 | DATE
 | TIME
 | TIMESTAMP
 | DATETIME
 | CHAR(length) [BINARY | ASCII | UNICODE]
 | VARCHAR(length) [BINARY]
 | TINYBLOB
 | BLOB
 | MEDIUMBLOB
 | LONGBLOB
 | TINYTEXT
 | TEXT
 | MEDIUMTEXT
 | LONGTEXT
 | ENUM(value1,value2,value3,...)
 | SET(value1,value2,value3,...)
 | spatial_type

index_col_name:
   col_name [(length)] [ASC | DESC]

reference_definition:
   REFERENCES tbl_name [(index_col_name,...)]
              [MATCH FULL | MATCH PARTIAL]
              [ON DELETE reference_option]
              [ON UPDATE reference_option]

reference_option:
   RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_options: table_option [table_option] ...

table_option:
   &#123;ENGINE|TYPE&#125; = &#123;BDB|HEAP|ISAM|InnoDB|MERGE|MRG_MYISAM|MYISAM&#125;
 | AUTO_INCREMENT = value
 | AVG_ROW_LENGTH = value
 | CHECKSUM = &#123;0 | 1&#125;
 | COMMENT = 'string'
 | MAX_ROWS = value
 | MIN_ROWS = value
 | PACK_KEYS = &#123;0 | 1 | DEFAULT&#125;
```

```
  | PASSWORD = 'string'
  | DELAY_KEY_WRITE = &#123;0 | 1&#125;
  | ROW_FORMAT = &#123; DEFAULT | DYNAMIC | FIXED | COMPRESSED &#125;
  | RAID_TYPE = &#123; 1 | STRIPED | RAID0 &#125;
        RAID_CHUNKS = value
        RAID_CHUNKSIZE = value
  | UNION = (tbl_name[,tbl_name]...)
  | INSERT_METHOD = &#123; NO | FIRST | LAST &#125;
  | DATA DIRECTORY = 'absolute path to directory'
  | INDEX DIRECTORY = 'absolute path to directory'
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

`CREATE TABLE` Crée une table avec le nom donné, dans la base de données courante. Vous avez besoin des droits de `CREATE` pour créer une table.

Les règles de nommage des tables sont disponibles dans Noms de base de données, tables, index, colonnes et alias . Par défaut, une table est créée dans la base de données courante. Une erreur est affichée s'il n'y a pas de base courante, si la base de données n'existe pas ou si la table existe déjà.

En MySQL 3.22 et plus récent, le nom de la table peut être spécifié avec la syntaxe `db_name.tbl_name` , pour créer une table dans une base spécifique. Cela fonctionne même s'il n'y a pas de base courante. Si vous utilisez les identifiants protégez, et mettez le nom de la base et de lui de la table entre guillemets, séparément. Par exemple, `` `madb`.`matbl` `` est valide, mais `` `madb.matbl` `` ne l'est pas.

Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot réservé `TEMPORARY` lorsque vous créez une table. Une table temporaire sera immédiatement effacée dès que la connexion se termine. Cela signifie que vous pouvez utiliser le même nom de table temporaire depuis deux connexions différentes sans risque de conflit entre les connexions. Vous pouvez aussi utiliser une table temporaire qui a le même nom qu'une table existante (la table existante est alors cachée tant que dure la table temporaire). En MySQL version 4.0.2 ou plus récent, vous avez juste à avoir le privilège `CREATE TEMPORARY TABLES` pour créer des tables temporaires.

Depuis la version 3.23 de MySQL, vous pouvez utiliser le mot réservé `IF NOT EXISTS` , de façon à ce qu'aucune erreur ne soit affiché si la table que vous essayez de créer existe déjà. Notez qu'il n'y a pas de comparaisons entre les structures de table lors du test d'existence.

MySQL représente chaque table par un fichier de définition de table `.frm` , placé dans le dossier de la base de données. Le moteur de la table peut créer d'autres fichiers complémentaires. Dans le cas des tables `MyISAM` , le moteur de stockage utilise trois fichiers, avec le nom `nom_de_table` :

Fichier	Rôle
nom_de_table.frm	Fichier de définition de la table
nom_de_table.MYD	Fichier de données
nom_de_table.MYI	Fichier d'index

Les fichiers créés par les moteurs de stockages pour représenter les tables sont décrits dans la section Types de tables MySQL .

Pour des informations générales sur les propriétés des différentes colonnes, voyez Types de colonnes . Pour des informations sur les types de données spatiaux, voyez Extensions spatiale de MySQL .

- Si ni `NULL` , ni `NOT NULL` n'est spécifié, une colonne utilisera par défaut l'attribut `NULL` (elle acceptera les valeurs `NULL`).
- Une colonne de nombre entier peut se voir attribuer l'attribut `AUTO_INCREMENT` . Lorsque vous insérez la valeur `NULL` (recommandée) ou `0` dans une colonne `AUTO_INCREMENT` , la colonne prendra automatiquement la valeur de `value+1` , où value est la plus grande valeur positive courante dans cette colonne. La série des valeurs `AUTO_INCREMENT` commence à `1` . `mysql_insert_id()` .

Depuis MySQL 4.1.1, en spécifiant l'option `NO_AUTO_VALUE_ON_ZERO` pour le mode `--sql-mode` ou la variable serveur `sql_mode` permet de stocker la valeur `0` dans les colonnes de type `AUTO_INCREMENT`, au lieu de voir `0` prendre le prochain numéro de séquence. Options de ligne de commande `mysqld`.

Note : Il ne peut y avoir qu'une seule colonne de type `AUTO_INCREMENT` dans une table, et elle doit être indexée. MySQL version 3.23 ne fonctionnera correctement que si cette colonne n'accueille que des valeurs positives. Insérer un nombre négatif sera considéré comme insérer un nombre de très grande taille, mais positif. Ceci est fait pour éviter les problèmes de précision lorsque les nombres passe de positif à négatif lorsqu'ils atteignent leur valeur maximale positive. C'est aussi pour éviter qu'une colonne de type `AUTO_INCREMENT` ne contienne de valeur 0.

Si vous effacez la ligne contenant la valeur maximale dans la colonne `AUTO_INCREMENT`, cette valeur sera réutilisée dans les tables de type `ISAM` mais pas dans les tables de type `MyISAM`. Si vous effacez toutes les lignes dans la table avec la commande `DELETE FROM nom_de_table` (sans la clause `WHERE)` en mode `AUTOCOMMIT`, la série des valeurs `AUTO_INCREMENT` recommencera à `0`.

Avec les tables `MyISAM` et `BDB`, vous pouvez spécifier une colonne secondaire d' `AUTO_INCREMENT` dans une clef multi−colonnes. Utiliser `AUTO_INCREMENT`.

Pour rendre MySQL avec certaines applications `ODBC`, vous pouvez retrouver la valeur de la dernière valeur automatiquement générée avec la requête suivante :

```
SELECT * FROM nom_de_table WHERE auto_col IS NULL
```

- Depuis MySQL 4.1, la définition des colonnes peut inclure un attribut `CHARACTER SET` pour spécifier le jeu de caractères, et éventuellement la collation de la colonne. Pour des détails, voyez Jeux de caractères .

```
CREATE TABLE t (c CHAR(20) CHARACTER SET utf8 COLLATE utf8_bin);
```

  Depuis la version 4.1 aussi, MySQL interprète les spécifications de longueur de colonne en caractères. Les anciennes versions l'interprète comme des octets.
- La valeur `NULL` est traitée différemment dans les colonnes de type `TIMESTAMP`. Vous ne pouvez pas stocker de valeur `NULL` littérale dans une colonne `TIMESTAMP` ; insérer une valeur `NULL` dans une telle colonne revient à insérer la date et l'heure courante. Car les colonnes `TIMESTAMP` ignorent les attributs NULL et `NOT NULL`.

  Cela facilite grandement l'utilisation des colonnes `TIMESTAMP` pour les clients MySQL : le serveur indique que ces colonnes peuvent se voir assigner une valeur `NULL` (ce qui est vrai), même si les colonnes `TIMESTAMP` ne contiendront jamais de valeur `NULL`. Vous pouvez le constater lorsque vous utiliser la commande `DESCRIBE nom_de_table` pour avoir une description de votre table.

  Notez qu'affecter la valeur `0` à une colonne `TIMESTAMP` n'est pas la même chose que lui affecter la valeur `NULL`, car `0` est une valeur `TIMESTAMP` valide.
- Une valeur `DEFAULT` doit être une constante, ça ne peut être une fonction ou une expression. Cela signifie notamment que vous ne pouvez pas donner une valeur par défaut à une colonne de date, le résultat de la fonction `NOW()` ou `CURRENT_DATE`.Si aucune valeur par défaut (attribut `DEFAULT`) n'est spécifiée, MySQL en assigne une automatiquement

  Si la colonne accepte les valeur `NULL`, la valeur par défaut sera la valeur `NULL`.

  Si la colonne est déclarée comme `NOT NULL` (non−nulle), la valeur par défaut dépendra du type de colonne :

  - Pour les types numériques sans l'attribut `AUTO_INCREMENT`, la valeur sera `0`. Pour une colonne `AUTO_INCREMENT`, la valeur par défaut sera la prochaine valeur de la série.

    - Pour les types dates et heures autres que `TIMESTAMP` , la valeur par défaut est la date zéro appropriée. Pour les colonnes `TIMESTAMP` , la valeur par défaut est la date et l'heure courante. Types temporels .
    - Pour les types de chaînes autres que `ENUM` , la valeur par défaut est la chaîne vide. Pour `ENUM` , la valeur par défaut est la première valeur de l'énumération.

  Les colonnes `BLOB` et `TEXT` ne peuvent pas recevoir de valeur par défaut.
- Un commentaire pour une colonne peut être spécifiée avec `COMMENT` . Le commentaire est affiché par la commande `SHOW CREATE TABLE` , et par `SHOW FULL COLUMNS` . Cette option est disponible depuis MySQL 4.1. Il était autorisé, mais ignoré dans les anciennes versions.
- Depuis MySQL version 4.1.0, l'attribut `SERIAL` peut être utilisé comme alias pour les colonnes `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE` . C'est une fonctionnalité de compatibilité.
- `KEY` est un synonyme de `INDEX` . Depuis la version 4.1, l'attribut de clé `PRIMARY KEY` peut aussi être spécifié avec la clause `KEY` . Il a été implémenté pour assurer la compatibilité avec les autres bases.
- Avec MySQL, une clé `UNIQUE` peut avoir uniquement avoir deux valeurs distinctes. Une erreur surviendra si vous essayez d'ajouter une ligne dont la clé correspond à une ligne existante.
- Une clé primaire (`PRIMARY KEY`) est un index `UNIQUE` avec la contrainte supplémentaire que les toutes les colonnes utilisées doit avoir l'attribut `NOT NULL` . En MySQL, cette clé est dite `PRIMARY` . Une table ne peut avoir qu'une seule clé primaire. Si vous n'avez pas de `PRIMARY KEY` et que des applications demandent la `PRIMARY KEY` dans vos tables, MySQL retournera la première clé `UNIQUE` , qui n'a aucune valeur `NULL` .
- Dans une table créée, la clé primaire `PRIMARY KEY` est placée en première, suivie de tous les index `UNIQUE` , et enfin, les index non−unique. Cela permet à l'optimiseur MySQL d'utiliser en priorité les index, et de détecter rapidement les doublons pour les clés `UNIQUE` .
- Une `PRIMARY KEY` peut être multi−colonnes. Cependant, vous ne pouvez pas créer d'index multi−colonnes avec l'attribut `PRIMARY KEY` dans une spécification de colonne. En faisant cela, le seul résultat sera que cette seule colonne sera marquée comme clé primaire. Vous devez absolument utiliser la syntaxe `PRIMARY KEY` (`index_nom_de_colonne` , ...).
- Si une clé primaire (`PRIMARY`) ou unique (`UNIQUE`) est établit sur une seule colonne, et que cette colonne est de type entier, vous pouvez aussi faire référence à cette colonne sous le nom `_rowid` (nouveau en version 3.23.11).
- Avec MySQL, le nom de la clé primaire `PRIMARY KEY` est `PRIMARY` . Si vous ne donnez pas de nom à un index, l'index prendra le nom de la première colonne qui le compose, avec éventuellement un suffixe (`_2` , `_3` , ...) pour le rendre unique. Vous pouvez voir les noms des index avec la commande `SHOW INDEX FROM tbl_name` . Syntaxe de `SHOW DATABASES` .
- Depuis MySQL 4.1.0, certains moteurs de stockage vous permettent de spécifier un type d'index lors de la création d'un index. La syntaxe de `index_type` est `USING type_name` . Les valeurs possibles de `type_name` qui sont supportées par les différents moteurs de stockages sont listés ci−dessous. Lorsque des index multiples sont listés, le premier rencontré est celui par défaut, si aucun `index_type` n'est spécifié.

Moteur de table	**Types d'index**
MyISAM	BTREE
InnoDB	BTREE
MEMORY/HEAP	HASH , BTREE

  Exemple :

  ```
  CREATE TABLE lookup
      (id INT, INDEX USING BTREE (id))
      ENGINE = MEMORY;
  ```

  `TYPE type_name` peut être utilisé comme synonyme de `USING type_name` , pour spécifier un type d'index. Cependant, `USING` est la forme recommandée. De plus, le nom d'index qui précède le type d'index dans la syntaxe de spécification n'est pas optionnelle avec `TYPE` . Ceci est dû au fait que contrairement à `USING` , `TYPE` n'est pas un mot réservé, et donc, il pourrait être interprété comme un nom d'index.Si vous spécifiez un type d'index qui n'est pas légal pour le moteur de stockage, mais qu'il y a un autre type d'index que le moteur peut utiliser sans affecter les résultats de la requête, le moteur utilisera ce type en remplacement.
- Seuls, les formats de table `MyISAM` , `InnoDB` , et `BDB` supportent des index sur des colonnes qui

peuvent contenir des valeurs `NULL` . Dans les autres situations, vous devez déclarer ces colonnes `NOT NULL` ou une erreur sera générée.

- Avec la syntaxe `nom_de_colonne(longueur)` , vous pouvez spécifier un index qui n'utilise qu'une partie de la colonne `CHAR` ou `VARCHAR` . Cela peut réduire la taille des fichiers d'index. Index de colonne .

  Le format de table `MyISAM` , et depuis la version MySQL 4.0.14, `InnoDB` , supportent l'indexation des colonnes `BLOB` et `TEXT` . Lorsque vous ajoutez un index à une colonne `BLOB` ou `TEXT` , vous devez **absolument** spécifier une longueur d'index :

  ```
  CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
  ```

  Le préfixe peut valoir jusqu'à 255 octets de long (où 1000 octets pour les tables `MyISAM` et `InnoDB` depuis MySQL 4.1.2). Notez que le préfixe est mesuré en octets, alors que la longueur du préfixe de `CREATE TABLE` est interprété comme un nombre de caractères. Prenez cela en compte lorsque vous spécifiez une longueur de préfixe pour une colonne dont le jeu de caractères est multi−octets.
- Une spécification `index_col_name` peut se terminer avec `ASC` ou `DESC` . Ces mots clés sont prévus pour des extensions futures qui permettront un stockage dans un ordre donné. Actuellement, ils sont reconnus mais ignorés : les index sont stockés en ordre ascendant.
- Lorsque vous utilisez une clause `ORDER BY` ou `GROUP BY` sur une colonne de type `TEXT` ou `BLOB` , seuls, les `max_sort_longueur` premiers octets seront lus. Les types `BLOB` et `TEXT` .
- En MySQL version 3.23.23 ou plus récent, vous pouvez aussi créer des index spécial `FULLTEXT` . Ils sont utilisés pour faire des recherches en texte plein. Seul, le format de table `MyISAM` supporte les index `FULLTEXT` . Ils peuvent être créés uniquement pour les colonnes de type `VARCHAR` et `TEXT` . L'indexation est alors exécutée sur toute la colonne, et les indexations partielles ne sont pas supportées. Voir Fonctions de recherche en texte plein pour les détails.
- En MySQL version 4.1 ou plus récent, vous pouvez créer les index spéciaux `SPATIAL` pour les colonnes géographiques. Les types spatiaux sont supportés par les tables `MyISAM` , et les colonnes indexées doivent être déclarées comme `NOT NULL` . Voyez Extensions spatiale de MySQL .
- En MySQL version 3.23.44 et plus récent, les tables `InnoDB` supportent la vérification de clé étrangères. Le moteur de tables `InnoDB` . Notez que la syntaxe des clés étrangères `FOREIGN KEY` de `InnoDB` est plus restrictive que la syntaxe présentée ci−dessus. `InnoDB` ne permet pas la spécification d'un `index_name` , et les colonnes de la table référencée doivent être explicitement nommées. Depuis la version 4.0.8, `InnoDB` supporte les clauses `ON DELETE` et `ON UPDATE` avec les clés étrangères. Voyez le manuel `InnoDB` pour la syntaxe précise. Contraintes `FOREIGN KEY` .Pour les autres types de tables, le serveur MySQL n'analyse pas les clauses `FOREIGN KEY` , `CHECK` et `REFERENCES` dans les commandes `CREATE TABLE` , et aucune action n'est réalisée. Clés étrangères .
- Pour les tables `MyISAM` et `ISAM` , chaque colonne `NULL` requiert un bit supplémentaire, arrondi à l'octet supérieur le plus proche.La taille maximale d'enregistrement peut être calculée comme ceci :

  ```
  row longueur = 1
              + (somme des longueurs de colonnes)
              + (nombre de colonnes NULL + 7)/8
              + (nombre de colonnes à taille variable)
  ```

  `delete_flag` vaut 1 pour les tables avec un format fixe. Les tables à format fixe utilisent un bit dans les lignes pour un marqueur, qui indique si la ligne a été effacée. `delete_flag` vaut 0 pour les tables à ligne dynamique, car le marquer est stocké dans l'entête de la ligne.Ces calculs ne s'appliquent pas aux tables `InnoDB` , qui ne font pas la différente entre les colonnes `NULL` et les colonnes `NOT NULL` .

Les options `options_de_table` et `SELECT` ne sont implémentées que dans MySQL version 3.23 et plus récent.

Les options ENGINE et TYPE spécifie le type de moteur de table. ENGINE a été ajouté en MySQL 4.0.18, pour la série des 4.0 et 4.1.2, pour la série des 4.1. C'est le nom d'attribut recommandé pour ces versions et TYPE est maintenant abandonné. TYPE sera supporté dans les séries 4.x, mais abandonnée probablement en MySQL 5.1.

Les différents types de tables sont :

Table type	Description
BDB	Tables avec transactions. Tables BDB ou BerkeleyDB .
BerkeleyDB	Un alias de BDB .
HEAP	Les données de ces tables ne sont stockées qu'en mémoire. Tables HEAP .
ISAM	Le gestionnaire originel de tables. Tables ISAM .
InnoDB	Tables transactionnelles avec verrou de lignes. Le moteur de tables InnoDB .
MEMORY	Un alias de HEAP . (En fait, depuis MySQL 4.1, MEMORY est le terme recommandé.)
MERGE	Un ensemble de tables MyISAM utilisées comme une seule et même table. Tables MERGE .
MRG_MyISAM	Un synonyme pour MERGE les tables.
MyISAM	Le nouveau gestionnaire de table binaire et portable. Tables MyISAM .

Types de tables MySQL .

Si un type de table est demandé, mais que ce type particulier n'est pas disponible, MySQL va choisir le type de table le plus proche de celui qui est spécifié. Par exemple, si TYPE=BDB est spécifié, et que la distribution de MySQL ne supporte pas les tables BDB , la table qui sera créée sera du type MyISAM .

Les autres options de tables sont utilisées pour optimiser le comportement de la table. Dans la plupart des cas, vous n'avez pas à les spécifier. Les options fonctionnent pour tous les types de tables (sauf contre−indication) :

AUTO_INCREMENT
: La prochaine valeur AUTO_INCREMENT de votre table (MyISAM). Ne fonctionne que pour les tables MyISAM . Pour donner la première valeur à une colonne AUTO_INCREMENT InnoDB , insérez une ligne bidon, avec la valeur désirée moins un, puis supprimez la ligne.

AVG_ROW_LENGTH
: La taille moyenne approchée des lignes de votre table. Vous ne devez fournir cette valeur que pour les tables à taille de ligne variable, de très grande taille.

  Lorsque vous créer une table MyISAM , MySQL utilise le produit des options MAX_ROWS et AVG_ROW_LENGTH pour décider de la taille du résultat. Si vous ne spécifiez aucune option, la taille maximale de la table sera de 4 Go (ou 2 Go si votre système d'exploitation ne supporte que les tables de 2 Go). Ceci sert à conserver la taille des pointeurs d'index petite, et rapide, si nous n'avez pas besoin de gros fichiers. Si vous voulez que vos tables dépassent 4 Go de taille, et que vous voulez garder les tables petites taille un peu plus lentes et grosses que nécessaire, vous pouvez augmenter la taille du pointeur d'index en modifiant la variable système globale myisam_data_pointer_size , qui a été ajoutée en MySQL 4.1.2. Variables sytème du serveur .

CHECKSUM
: Passez 1 si vous voulez que MySQL génère une somme de vérification (ce qui facilite la recherche des lignes corrompues, mais ralentit les mises à jour). La commande CHECKSUM TABLE rapporte cette somme. MyISAM uniquement.

COMMENT
: Un commentaire pour votre table (60 caractères).

MAX_ROWS
: Nombre de lignes maximum que vous pensez stocker dans la table.

MIN_ROWS
: Nombre de minimum lignes que vous pensez stocker dans la table.

PACK_KEYS
: Spécifiez 1 si vous voulez un index plus compact. Généralement cela rend les mises à jour plus lentes, mais les lectures plus rapides.

Spécifier la valeur de 0 désactive tout le compactage de clé. Spécifier la valeur `DEFAULT` (MySQL 4.0) indique au moteur de stockage de ne stocker que les colonnes `CHAR` / `VARCHAR` . (`MyISAM` et `ISAM` uniquement)

Si vous n'utilisez pas `PACK_KEYS` , le comportement par défaut est de ne stocker que les chaînes, et non pas les nombres. Si vous utilisez `PACK_KEYS=1` , les nombres seront aussi compactés.

Lors du compactage, MySQL utilise une compression de préfixe :

- Chaque clé requiert un octet de plus pour indiquer combien d'octets sont identiques dans la clé précédente.
- Le pointeur de ligne est stocké au format grand−octet−en−premier, directement après la clé, pour améliorer la compression.

Cela signifie que si vous avez de nombreuses clés proches sur des lignes consécutives, les clés successives ``identiques'' ne prendront généralement que deux octets (incluant le pointeur de ligne). Comparez cela à la situation ordinaire, où les clés successives occupent `taille_de_cle + taille_de_pointeur` (où la taille du pointeur est généralement de 4). En conséquence, vous tirerez le meilleur parti de cette compression si vous avez plusieurs nombres identiques. Si toutes les clés sont totalement différentes, vous utiliserez un octet de plus par clé, si la clé n'accepte pas les valeurs `NULL` . Dans ce cas, la taille de la clé sera stockée dans le même octet que celui qui indique que la clé est `NULL` .)

`PASSWORD`
: Chiffre le fichier `.frm` avec un mot de passe. Cette option ne fait rien du tout pour la version standard de MySQL.

`DELAY_KEY_WRITE`
: Spécifiez 1 si vous voulez attendre la fermeture de la table pour mettre à jour les index. `MyISAM` uniquement.

`ROW_FORMAT`
: Définit la méthode de stockage des lignes (réservé pour le futur). Actuellement, cette option fonctionne uniquement avec des tables `MyISAM` qui supportent le `DYNAMIC` et `FIXED` en format de ligne. Format de table `MyISAM` .

`RAID_TYPE`
: L'option `RAID_TYPE` vous permet de dépasser la limite de 2 Go/4 Go de votre fichier de données `MyISAM` (mais pas le fichier d'index), pour les systèmes d'exploitation qui ne supportent pas les grands fichiers. Cette option n'est pas recommandée pour les systèmes d'exploitation qui supportent les grands fichiers.Vous pouvez réduire les ralentissements d'E/S en plaçant les dossiers `RAID` sur différents disques physiques. Actuellement, le seul type `RAID_TYPE` est `STRIPED` . `1` et `RAID0` sont des alias de `STRIPED` .

Si vous spécifiez l'option `RAID_TYPE` pour une table `MyISAM` , spécifiez les options `RAID_CHUNKS` et `RAID_CHUNKSIZE` en même temps. La valeur maximale de `RAID_CHUNKS` est 255. `MyISAM` va créer `RAID_CHUNKS` sous−dossiers appelés `00` , `01` , `02` , ... `09` , `0a` , `0b` , ... dans le dossier de données. Dans chaque dossier, `MyISAM` va créer un fichier `tbl_name.MYD` . Lors de l'écriture dans le fichier de données, le gestionnaire `RAID` place les `RAID_CHUNKSIZE*1024` premiers octets dans le premier fichier, les seconds `RAID_CHUNKSIZE*1024` octets dans le fichier suivant, etc.

`RAID_TYPE` fonctionne sur tous les systèmes d'exploitation, tant que vous avez compilé MySQL avec `--with-raid` , avec le script `configure` . Pour déterminer si votre serveur a le support des tables `RAID` , utilisez `SHOW VARIABLES LIKE 'have_raid'` pour voir si sa valeur vaut `YES` .

`UNION`
: `UNION` sert lorsque vous voulez que plusieurs tables identiques se comporte comme une seule table. Cela fonctionne avec les tables `MERGE` . Tables `MERGE` .Pour le moment, vous devez avoir les droits de `SELECT` , `UPDATE` et `DELETE` pour les tables intégrées dans la table `MERGE` . Originalement, toutes les tables utilisées devaient être dans la même

base de données, que la table `MERGE` . Cette restriction a été levée depuis MySQL 4.1.1.

INSERT_METHOD

Si vous voulez insérer des données dans une table `MERGE` , vous devez spécifier la table d'insertion avec l'attribut `INSERT_METHOD` . L'option `INSERT_METHOD` est utilisée uniquement avec les tables `MERGE` . Cette option a été introduite en MySQL 4.0.0. Tables `MERGE` .

DATA DIRECTORY

INDEX DIRECTORY

En utilisant `DATA DIRECTORY='directory'` ou `INDEX DIRECTORY='directory'` , vous pouvez spécifier où le moteur de stockage `MyISAM` doit placer les données de la table et le fichier d'index. Notez que vous devez donner un chemin absolu, et non un chemin relatif.

Ces options ne fonctionnent que pour les tables `MyISAM` depuis MySQL 4.0, lorsque vous n'utilisez pas l'option `--skip-symlink` . Votre système d'exploitation doit aussi disposer d'une fonction `realpath()` compatible avec les threads. Utiliser les liens symboliques pour les tables sous Unix .

Depuis MySQL 3.23, vous pouvez créer une table à partir d'une autre, en ajoutant une commande `SELECT` après la commande `CREATE TABLE` :

```
CREATE TABLE new_tbl SELECT * FROM orig_tbl;
```

MySQL va créer une nouvelle colonne pour chaque colonne de résultat de la commande `SELECT` . Par exemple :

```
mysql> CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT,
    ->        PRIMARY KEY (a), KEY(b))
    ->        TYPE=MyISAM SELECT b,c FROM test2;
```

Cela créer une table `MyISAM` avec trois colonnes `a` , `b` , et `c` . Notez que les colonnes de la commande `SELECT` sont ajoutées à droite de la table, et non dans la liste des colonnes. Par exemple :

```
mysql> SELECT * FROM foo;
+---+
| n |
+---+
| 1 |
+---+

mysql> CREATE TABLE bar (m INT) SELECT n FROM foo;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM bar;
+------+---+
| m    | n |
+------+---+
| NULL | 1 |
+------+---+
1 row in set (0.00 sec)
```

Pour chaque ligne de la table `foo` , une ligne est insérée dans la colonne `bar` avec la valeur issue de `foo` et la valeur par défaut pour les nouvelles colonnes.

Si une erreur survient durant la copie de la table, la table est automatiquement effacée.

`CREATE TABLE ... SELECT` ne va pas créer automatiquement les index pour vous. Ceci est fait intentionnellement pour rendre la commande aussi souple que possible. Si vous voulez avoir les mêmes index, vous devez les spécifier dans la commande avant le `SELECT` :

```
mysql> CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;
```

Certaines conversions de type pourraient avoir lieu. Par exemple, l'attribut `AUTO_INCREMENT` n'est pas préservé, et les colonnes `VARCHAR` peuvent devenir des colonnes `CHAR` .Lors de la création de la table avec `CREATE ... SELECT` , assurez vous de mettre un nom d'alias à toutes les fonctions ou expression de la requête. Si vous ne le faîtes pas, la commande `CREATE` peut échouer ou donner des noms de colonnes inattendus.

```
CREATE TABLE artists_and_works
SELECT artist.name, COUNT(work.artist_id) AS number_of_works
FROM artist LEFT JOIN work ON artist.id = work.artist_id
GROUP BY artist.id;
```

Depuis MySQL 4.1, vous pouvez spécifier explicitement le type de colonne généré :

```
CREATE TABLE foo (a TINYINT NOT NULL) SELECT b+1 AS a FROM bar;
```

En MySQL 4.1, vous pouvez aussi utiliser la clause `LIKE` pour créer une table basée sur la définition d'une autre table, y compris les attributs de colonnes et les index originaux :

```
CREATE TABLE new_tbl LIKE orig_tbl;
```

`CREATE TABLE ... LIKE` ne copie pas les options de tables `DATA DIRECTORY` et `INDEX DIRECTORY` qui étaient spécifiées dans la table originale.Vous pouvez faire précéder `SELECT` par `IGNORE` ou `REPLACE` pour indiquer comment gérer les clés doublons. Avec `IGNORE` , les nouvelles lignes qui sont en double seront ignorés. Avec `REPLACE` , les nouvelles lignes remplaceront les lignes précédentes, qui avaient la même valeur d'index. Si ni `IGNORE` , ni `REPLACE` ne sont spécifié, les doublons génèreront une erreur.

Pour s'assurer que le log binaire peut être réutilisé pour recréer la table originale, MySQL ne permettra pas les insertions concurrentes durant une commande `CREATE TABLE ... SELECT` .

14.2.5.1 Modification automatique du type de colonnes

Dans certains cas, MySQL change automatiquement la spécification d'une colonne fournie dans la commande `CREATE TABLE` . (Cela peut aussi arriver avec `ALTER TABLE`) :

- Les colonnes `VARCHAR` avec une taille inférieure à quatre (4) sont changées en `CHAR` .
- Si l'une des colonnes d'une table est de taille variable, toute la ligne est, par conséquent, de taille variable. Ainsi, si une ligne contient une colonne de taille variable (`VARCHAR` , `TEXT` ou `BLOB`) toutes les colonnes `CHAR` de plus de trois caractères sont transformées en `VARCHAR` . Cela ne change en rien la façon dont vous utilisez les colonnes. Pour MySQL, `VARCHAR` est simplement une autre façon de stocker les caractères. MySQL effectue cette conversion car cela économise de la place, et rend les calculs sur les tables plus rapides. Types de tables MySQL .
- Depuis la version 4.1.0, si un champ `CHAR` ou `VARCHAR` est spécifié avec une taille supérieure à 255, il est converti en `TEXT` . C'est une fonctionnalité de compatibilité.
- La taille d'affichage de `TIMESTAMP` doit être un nombre pair et être compris entre 2 et 14. (2, 4, 6, 8, 10, 12 ou 14). Si vous spécifiez une taille plus grande que 14, ou inférieure à 2, celle−ci sera transformée en 14. Les valeurs impaires sont ramenées à la valeur pair supérieure la plus proche.
- Vous ne pouvez pas stocker de valeur littérale `NULL` dans une colonne de type `TIMESTAMP` . Cette valeur sera remplacée par la date et l'heure courante. De ce fait, les attributs `NULL` et `NOT NULL` n'ont pas de sens pour ces colonnes et sont ignorés. `DESCRIBE nom_de_table` indiquera toujours que la colonne `TIMESTAMP` accepte les valeurs `NULL` .
- Les colonnes qui font partie d'une `PRIMARY KEY` ont l'attribut `NOT NULL` même si elles ne sont pas déclarées comme tel.
- Depuis MySQL 3.23.51, les espaces terminaux sont automatiquement supprimés des valeurs `ENUM` et `SET` lors de la création de la table.
- MySQL change certains type de colonnes utilisés par d'autres serveurs SQL en types MySQL. Utiliser les types de colonnes d'autres bases .
- Si vous utilisez une clause `USING` pour spécifier un type d'index qui n'est pas légal pour un moteur de stockage, mais qu'un autre type d'index est disponible pour ce moteur sans affecter les résultats, le moteur utilisera le type disponible.

Si vous voulez voir si MySQL a utilisé un autre type que celui que vous avez spécifié, utilisez la commande `DESCRIBE nom_de_table` , après votre création ou modification de structure de table.

Certain types de colonnes peuvent être modifiés si vous compressez une table en utilisant l'utilitaire `myisampack` . Caractéristiques des tables compressées .

14.2.6 Syntaxe de DROP DATABASE

```
DROP DATABASE [IF EXISTS] db_name
```

`DROP DATABASE` détruit toutes les tables dans la base de données et l'efface elle même. Soyez **très** prudent avec cette commande! Pour utiliser la commande `DROP DATABASE` , vous avez besoin du droit de `DROP` sur cette base. Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot clef `IF EXISTS` pour éviter l'affichage d'erreur si la base n'existe pas.

Si vous utilisez la commande `DROP DATABASE` sur un lien symbolique pointant sur la base de données, le lien et la base seront effacés.

Depuis MySQL 4.1.2, `DROP DATABASE` retourne le nombre de tables qui ont été supprimées. Cela revient à compter le nombre de fichiers `.frm` qui ont été supprimées.

La commande `DROP DATABASE` efface tous les fichiers du dossier de la base de données, qui ont été créés par MySQL lui−même, durant ses opérations normales :

- Tous les fichiers avec les extensions suivantes :

.BAK	.DAT	.HSH	.ISD
.ISM	.ISM	.MRG	.MYD
.MYI	.db	.frm	

- Tous les sous−dossiers qui consistent de 2 chiffres hexadécimaux `00` − `ff` . Ce sont des dossiers `RAID`) qui sont aussi supprimés.
- Le fichier `db.opt` , s'il existe.

Si d'autres fichiers ou dossiers restent dans le dossier de base après que MySQL ait supprimés ceux listés ci−dessus, le dossier de base ne pourra pas être supprimé. Dans ce cas, vous devez supprimer manuellement les fichiers restant, et lancer à nouveau la commande `DROP DATABASE` .

Vous pouvez aussi supprimer des bases de données avec `mysqladmin` . `mysqladmin` , administrer un serveur MySQL .

14.2.7 Syntaxe de DROP INDEX

```
DROP INDEX nom_de_l_index ON nom_de_table
```

`DROP INDEX` supprime l'index nommé `nom_de_l_index` de la table `nom_de_table` . `DROP INDEX` ne fait rien avec la version 3.22 et les précédentes. Depuis cette version, `DROP INDEX` est un alias d' `ALTER TABLE` supprimant l'index. Syntaxe de `ALTER TABLE` .

14.2.8 Syntaxe de DROP TABLE

```
DROP [TEMPORARY] TABLE [IF EXISTS]
   tbl_name [, tbl_name] ...
   [RESTRICT | CASCADE]
```

`DROP TABLE` supprime une ou plusieurs tables. Toutes les données et la structure de la tables sont **perdues** , alors soyez **prudents** avec cette commande !

Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot réservé `IF EXISTS` pour éviter l'affichage des erreurs pour les tables qui n'existent pas. Syntaxe de `SHOW WARNINGS` .

`RESTRICT` et `CASCADE` sont autorisés pour faciliter le port. Pour le moment, elles ne font rien.

Note : `DROP TABLE` va automatiquement valider les transactions actives (hormis si vous utilisez la version 4.1 et le mot clé `TEMPORARY`).

L'option `TEMPORARY` est ignorée en 4.0. En 4.1, cette option fonctionne comme suit :

- Détruit uniquement les tables temporaires.
- Ne termine pas les transactions en cours.
- Aucun droits d'accès n'est vérifié.

`TEMPORARY` est pour le moment ignoré; Dans un futur proche, il servira à s'assurer qu'on efface vraiment une table temporaire.

14.2.9 Syntaxe de RENAME TABLE

```
RENAME TABLE nom_de_table TO nouveau_nom_de_table[, nom_de_table2 TO nouveau_nom_de_table2,...]
```

Le changement de nom se fait atomiquement ce qui signifie qu'aucun autre processus ne peut accéder la table tant que l'opération est en cours. Cela rend possible de remplacer une vielle table avec une table vide :

```
CREATE TABLE nouvelle_table (...);
RENAME TABLE ancienne_table TO backup_table, nouvelle_table TO ancienne_table;
```

L'opération s'effectue de gauche à droite ce qui signifie que si vous voulez échanger deux noms de tables, vous devez :

```
RENAME TABLE ancienne_table    TO backup_table,
             nouvelle_table    TO ancienne_table,
             backup_table TO nouvelle_table;
```

Si les deux bases de données sont sur le même disque, vous pouvez renommer à travers les bases :

```
RENAME TABLE bdd_courante.nom_de_table TO autre_bdd.nom_de_table;
```

Quand vous exécutez `RENAME` , vous ne pouvez avoir aucune transaction active ou une table protégée en mode écriture. Vous devez avoir les privilèges `ALTER` et `DROP` sur l'ancienne table, et les privilèges `CREATE` et `INSERT` sur la nouvelle.

Si MySQL rencontre des erreurs dans un renommage multiple, il remettra les noms changés à leurs valeurs d'origine pour revenir à l'état d'origine.

`RENAME TABLE` a été ajouté à la version 3.23.23 de MySQL.

14.3 Commandes de bases de l'utilisateur de MySQL

14.3.1 Syntaxe de DESCRIBE (obtenir des informations sur les colonnes)

```
&#123;DESCRIBE | DESC&#125; nom_de_table [nom_de_colonne | wild]
```

`DESCRIBE` est un raccourci de `SHOW COLUMNS FROM` . Syntaxe de `SHOW DATABASES` . `DESCRIBE` fournit des informations à propos des colonnes de la table. `nom_de_colonne` peut être le nom d'une colonne ou une chaîne contenant les caractères spéciaux SQL `'%'` et `'_'` .

Si le type de colonne est différent de celui que vous pensiez avoir définit lors du `CREATE TABLE` , notez que MySQL change le type des colonnes de temps en temps. Changement de type de colonne automatique .

Cette instruction est fournie pour une meilleure compatibilité avec Oracle.

L'instruction `SHOW` renvoie les mêmes informations. Syntaxe de `SET` et `SHOW` .

14.3.2 Syntaxe de USE

```
USE db_name
```

La commande `USE db_name` spécifie à MySQL d'utiliser la base `db_name` comme base par défaut pour les requêtes ne les mentionnant pas. La base choisie reste la même jusqu'à la fermeture de la session ou un nouvel appel à `USE` :

```
mysql> USE db1;
mysql> SELECT COUNT(*) FROM ma_table;      # sélectionne à partir de db1.ma_table
mysql> USE db2;
mysql> SELECT COUNT(*) FROM ma_table;      # sélectionne à partir de db2.ma_table
```

Rendre une base de données la base courante (en utilisant `USE`) ne vous interdit pas l'accès à d'autres tables dans d'autres bases. L'exemple suivant accède à la table `author` de la base `db1` et à la table `editor` de la base `db2` :

```
mysql> USE db1;
mysql> SELECT author_name,editor_name FROM author,db2.editor
    ->        WHERE author.editor_id = db2.editor.editor_id;
```

La commande `USE` est fournie pour assurer la compatibilité Sybase.

14.4 Commandes relatives aux verrous et aux transactions

14.4.1 Syntaxes de START TRANSACTION , COMMIT et ROLLBACK

Par défaut, MySQL est lancé en mode `autocommit` . Cela signifie que chaque modification effectué est enregistré immédiatement sur le disque par MySQL.

Si vous utilisez des tables supportant les transactions (comme `InnoDB` , `BDB`), vous pouvez configurer MySQL en mode non− `autocommit` grâce à la commande:

```
SET AUTOCOMMIT=0
```

A partir de là, vous devez utiliser `COMMIT` pour enregistrer les modifications sur le disque ou `ROLLBACK` pour ignorer les modifications apportées depuis le début de la transaction.Si vous souhaitez sortir du mode `AUTOCOMMIT` pour une série d'opérations, vous pouvez utiliser les commandes `BEGIN` ou `BEGIN WORK` :

```
START TRANSACTION;
SELECT &#64;A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summmary=&#64;A WHERE type=1;
COMMIT;
```

`BEGIN` et `BEGIN WORK` peuvent être utilisé à la place de `START TRANSACTION` pour initialiser une transaction. `START TRANSACTION` a été ajouté en MySQL 4.0.11; C'est une syntaxe SQL−99, et il est recommandé de l'utiliser pour lancer une transaction. `BEGIN` et `BEGIN WORK` sont disponibles pour MySQL 3.23.17 et 3.23.19, respectivement.Notez que si vous n'utilisez pas de table transactionnelles, les modifications seront validées automatiquement, indépendamment du mode de validation.

Si vous faites un `ROLLBACK` après avoir modifié une table non transactionnelle, vous obtiendrez (`ER_WARNING_NOT_COMPLETE_ROLLBACK`) comme message d'alerte. Toutes les tables supportant les transactions seront restaurées, mais aucune des autres tables ne changera.

Si vous utilisez `START TRANSACTION` ou `SET AUTOCOMMIT=0` , il est recommandé d'utiliser les " `binary log` " de MySQL à la place des anciens logs de modifications pour les sauvegardes. Les transactions sont stockées dans les logs binaires en un seul bloc, après `COMMIT` , pour être sûr que les transactions qui ont été annulées ne soient pas enregistrées. Le log binaire des mises à jour .

Vous pouvez changer le niveau d'isolation des transactions avec `SET TRANSACTION ISOLATION LEVEL ...` . Syntaxe de `SET TRANSACTION` .

14.4.2 Commandes qui ne peuvent pas être annulées

Certaines commandes ne peuvent pas être annulées. En général, elles incluent le langage de définition des données (DDL), comme les commandes qui créent ou effacent des bases de données, ou celles qui créent, modifient ou effacent des tables de données.

Il est recommandé de concevoir vos transactions pour éviter ces commandes. Si vous soumettez une commande qui ne peut pas être annulée, dès le début de votre transaction, et qu'une commande ultérieure échoue, vous pourrez pas annuler l'ensemble de la transaction avec `ROLLBACK` .

14.4.3 Commandes qui peuvent causer une validation implicite

Les commandes suivantes valident implicitement une transaction, comme si vous aviez émis une commande `COMMIT` après :

ALTER TABLE	BEGIN	CREATE INDEX
DROP DATABASE	DROP INDEX	DROP TABLE
LOAD MASTER DATA	LOCK TABLES	RENAME TABLE
SET AUTOCOMMIT=1	START TRANSACTION	TRUNCATE

`UNLOCK TABLES` termine aussi une transaction si toutes les tables courantes sont verrouillées. Avant MySQL version 4.0.13, `CREATE TABLE` terminait une transaction si le log binaire était activé.

Les transactions ne peuvent pas être imbriquées. C'est la conséquence de cette validation `COMMIT` implicite pour toutes les transactions en cours, lorsque vous émettez une commande `START TRANSACTION` ou équivalent.

14.4.4 Syntaxe de SAVEPOINT et ROLLBACK TO SAVEPOINT

Depuis MySQL 4.0.14 et 4.1.1, `InnoDB` supporte les commandes SQL `SAVEPOINT` et `ROLLBACK TO SAVEPOINT` .

```
SAVEPOINT identifier
```

Cette commande pose un jalon de transaction dont le nom est `identifier` . Si la transaction courante a déjà un jalon de ce nom, l'ancien jalon est effacé, et le nouveau est créé à la place.

Cette commande annule la transaction jusqu'au jalon. Les modifications que cette transaction a fait aux lignes depuis le jalon sont annulées, mais `InnoDB` **ne libère pas** les verrous posés en mémoire après le jalon. Notez que pour une nouvelle ligne insérée, l'information de verrou est conservée par l'identifiant de transaction de la ligne : le verrou n'est pas stocké en mémoire. Dansa ce cas, le verrou sera levé par l'annulation. Les jalons qui ont été posé après celui−ci sont aussi annulés.

Si la commande retourne l'erreur suivante, c'est qu'aucun jalon de ce nom n'a pu être trouvé.

```
ERROR 1181: Got error 153 during ROLLBACK
```

Tous les jalons de la transaction courante sont annulés si vous exécutez les commandes `COMMIT` ou `ROLLBACK` , sans préciser de nom de jalon.

14.4.5 Syntaxe de LOCK TABLES/UNLOCK TABLES

```
LOCK TABLES
    tbl_name [AS alias] &#123;READ [LOCAL] | [LOW_PRIORITY] WRITE&#125;
    [, tbl_name [AS alias] &#123;READ [LOCAL] | [LOW_PRIORITY] WRITE&#125;] ...
UNLOCK TABLES
```

`LOCK TABLES` verrouille une table pour le thread courant. `UNLOCK TABLES` déverrouillera automatiquement tous les verrous posés par le thread courant. Toutes les tables verrouillées par le thread courant sont automatiquement déverrouillées quand ce thread utilise à nouveau `LOCK TABLES` , ou quand la connexion au serveur est perdue. **Note :** `LOCK TABLES` n'est pas compatible avec les transactions, et valide automatiquement toute transaction active avant de verrouiller une table.

L'utilisation de `LOCK TABLES` dans MySQL 4.0.2 nécessite le privilège `LOCK TABLES` global et un privilège de `SELECT` sur les tables impliquées. Dans MySQL 3.23, il faut les privilèges `SELECT` , `INSERT` , `DELETE` et `UPDATE` sur les tables.

Les principales raisons d'utiliser `LOCK TABLES` sont l'émulation de transactions ou l'accélération des processus de modification de tables. Cela sera détaillé plus loin.

Si un thread obtient un verrouillage `READ` sur une table, ce thread (et tous les autres threads) peuvent uniquement accéder a cette table en lecture. Si un thread obtient un verrouillage `WRITE` sur une table, alors seul le thread qui a posé le verrou peut lire ou écrire sur cette table. Tous les autres threads sont bloqués.

La différence entre `READ LOCAL` et `READ` est que `READ LOCAL` autorise des requêtes `INSERT` non−conflictuelles à être exécutées alors que le verrou est posé. Ceci ne peut cependant pas être utilisé si vous souhaitez modifier les fichiers de la base de données en dehors de MySQL pendant que le verrou est posé.

Quand vous utilisez `LOCK TABLES` , vous devez verrouiller toutes les tables que vous allez utiliser, et vous devez utiliser les mêmes alias sur ce que vous utiliserez dans vos requêtes ! Si vous utilisez une table a plusieurs reprises dans une requête (avec des alias), vous devez verrouiller chacun des alias !

Si vos requêtes utilisent un alias pour une table, alors vous devez verrouiller la table avec l'alias. Le verrouillage ne fonctionnera pas si vous verrouillez la table sans spécifier l'alias :

```
mysql> LOCK TABLE t READ;
mysql> SELECT * FROM t AS myalias;
ERROR 1100: Table 'myalias' was not locked with LOCK TABLES
```

De même, lorsque vous verrouillez une table avec un alias, vous devez utiliser le nom de l'alias dans vos requêtes :

```
mysql> LOCK TABLE t AS myalias READ;
mysql> SELECT * FROM t;
ERROR 1100: Table 't' was not locked with LOCK TABLES
mysql> SELECT * FROM t AS myalias;
```

Les verrous `WRITE` ont normalement des priorités supérieures aux verrous `READ` , afin de s'assurer que les modifications sont exécutées au plus vite. Cela signifie que si un thread demande un verrou `READ` et qu'un autre thread demande un verrou `WRITE` , la demande de verrou `READ` attendra que le thread `WRITE` ait abouti pour libérer le verrou. Vous pouvez utiliser le verrou `LOW_PRIORITY WRITE` pour permettre à d'autres threads d'obtenir des verrous `READ` pendant que le thread attend le verrou `WRITE` . Vous ne devriez utiliser les verrous `LOW_PRIORITY WRITE` que si vous êtes sûr qu'il y aura effectivement un moment où aucun thread ne posera de verrou `READ` . `LOCK TABLES` fonctionne de la manière suivante :

- Trie toutes les tables à verrouiller dans un ordre défini par MySQL (l'utilisateur ne définit pas d'ordre).
- Si une table est verrouillée avec un verrou de lecture et un verrou d'écriture, il pose le verrou de lecture avant celui d'écriture.
- Verrouille une table à la fois jusqu'à ce que le thread ait tous ses verrous.

Cette politique garantit le bon verrouillage des tables. Il faut cependant connaître certaines choses sur ce schéma :

Si vous utilisez un verrou `LOW_PRIORITY WRITE` pour une table, cela signifie seulement que MySQL attendra, pour poser ce verrou, qu'aucun autre thread ne réclame de verrou `READ` . Quand le thread aura le verrou `WRITE` et qu'il attendra que les verrous soient posés sur les autres tables de la liste, tous les autres threads attendront que le verrou `WRITE` soit libéré. Si cela devient un problème grave pour votre application, il est conseillé de convertir des tables en tables supportant les transactions.

Vous pouvez terminer un thread attendant un verrouillage de table en toute sécurité avec `KILL` . Syntaxe de `KILL` .

Il est **déconseillé** de verrouiller des tables utilisées avec `INSERT DELAYED` , car, dans ce cas, la requête `INSERT` est exécutée dans un autre thread.

Normalement, vous n'avez pas besoin de verrouiller les tables puisque chaque requête `UPDATE` est atomique : aucun autre thread ne peut interférer avec une autre requête active. Il existe cependant quelques cas où vous aurez besoin de verrouiller les tables :

- Si vous allez exécuter plusieurs requêtes sur plusieurs tables, il est préférable, d'un point de vue rapidité, de verrouiller les tables dont vous aurez besoin. L'inconvénient, bien sur, est que les autres threads ne pourront pas intervenir sur ces tables durant vos opérations, ni en extraire des informations si la table est verrouillée en `WRITE` .La raison pour laquelle les requêtes sont plus rapides avec `LOCK TABLES` est que MySQL ne rafraîchît pas l'index des clés des tables verrouillées tant que `UNLOCK TABLES` n'est pas invoqué (normalement, le cache des clés est rafraîchi après chaque requête SQL). Cela accélère les insertions, les modifications et les suppressions de données dans les tables `MyISAM` .
- Si vous utilisez un type de table dans MySQL qui ne supporte pas les transactions, vous devez utiliser `LOCK TABLES` pour vous assurez qu'aucun autre thread ne s'intercale entre un `SELECT` et un `UPDATE` . L'exemple suivant nécessite `LOCK TABLES` pour s'exécuter en toute sécurité :

```
 mysql> LOCK TABLES trans READ, customer WRITE;
mysql> SELECT SUM(value) FROM trans WHERE customer_id=some_id;
mysql> UPDATE customer SET total_value=sum_from_previous_statement
    ->        WHERE customer_id=some_id;
mysql> UNLOCK TABLES;
```

  Sans `LOCK TABLES` , Il est possible qu'un autre thread ait inséré une nouvelle ligne dans la table `trans` entre l'exécution du `SELECT` et l'exécution de la requête `UPDATE` .

L'utilisation de modifications incrémentales (`UPDATE customer SET value=value+nouvelle_valeur`) ou de la fonction `LAST_INSERT_ID()` permet de se passer de `LOCK TABLES` dans de nombreuses situations. Transactions et opérations atomiques .

Il est aussi possible de résoudre de nombreux cas en utilisant un verrou utilisateur, avec les fonctions `GET_LOCK()` et `RELEASE_LOCK()` . Ces verrous sont stockés dans une table de hashage dans le serveur et utilisent les fonctions `pthread_mutex_lock()` et `pthread_mutex_unlock()` pour plus de vitesse. Fonctions diverses .

Voir Comment MySQL verrouille les tables pour plus de détails.

Il est possible de verrouiller tous les tables de toutes les bases avec la commande `FLUSH TABLES WITH READ LOCK` .

Syntaxe de `FLUSH` . C'est une méthode très pratique pour effectuer des sauvegardes si vous utilisez un système de fichiers qui, comme Veritas, permet de créer des instantanés.

Note : `LOCK TABLES` ne fonctionne pas avec les transactions et validera automatiquement toutes les transactions actives avant de poser verrouiller la table. Problèmes avec `ALTER TABLE` . .

14.4.6 Syntaxe de SET TRANSACTION

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
&#123; READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE &#125;
```

Configuration du niveau d'isolation des transactions en général, pour la totalité de la session, ou pour la prochaine transaction.Le comportement par défaut est de configurer le niveau d'isolation pour la transaction suivante (qui n'a pas encore été commencée) En utilisant le paramètre `GLOBAL` , on configure le niveau par défaut global pour toutes les nouvelles connections. Cette commande requiert les privilèges `SUPER` . En utilisant le paramètre `SESSION` , on configure le niveau par défaut pour toutes les prochaines transactions effectuées durant la session actuelle.

Pour une description de chaque niveau d'isolation de transaction `InnoDB` , voyez InnoDB et `SET ... TRANSACTION ISOLATION LEVEL ...` . `InnoDB` supporte chacun des niveaux depuis MySQL 4.0.5. Le niveau par défaut est `REPEATABLE READ` .

On peut configurer le niveau d'isolation global des transactions pour `mysqld` avec `--transaction-isolation=...` . Utiliser les options de ligne de commande .

14.5 Référence de langage d'administration de la base de données

14.5.1 Commande de gestions des comptes utilisateurs

14.5.1.1 Effacer des utilisateurs MySQL

```
DROP USER user_name
```

Cette commande a été ajoutée en MySQL 4.1.1.

Elle efface un utilisateur qui n'a aucun droits.

Pour effacer un utilisateur dans MySQL, vous devriez utiliser l'une des procédures suivantes, dans cet ordre :

- Vérifiez les droits de l'utilisateur avec la commande `SHOW PRIVILEGES` . Syntaxe de `SHOW PRIVILEGES` .

- Effacez tous les droits de l'utilisateur avec la commande `REVOKE` . Syntaxe de `GRANT` et `REVOKE` .
- Effacez l'utilisateur avec `DROP USER` .

Si vous utilisez une vieille version de MySQL, vous devriez commencer par effacer les droits, puis l'utilisateur avec :

```
 DELETE FROM mysql.user WHERE user='username' and host='hostname';
FLUSH PRIVILEGES;
```

14.5.1.2 Syntaxe de GRANT et REVOKE

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
   ON &#123;tbl_name | * | *.* | db_name.*&#125;
   TO user [IDENTIFIED BY [PASSWORD] 'password']
       [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
   [REQUIRE
       NONE |
       [&#123;SSL| X509&#125;]
       [CIPHER cipher [AND]]
       [ISSUER issuer [AND]]
       [SUBJECT subject]]
   [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR count |
                         MAX_UPDATES_PER_HOUR count |
                         MAX_CONNECTIONS_PER_HOUR count]]
```

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
   ON &#123;tbl_name | * | *.* | db_name.*&#125;
   FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

Les commandes `GRANT` et `REVOKE` permettent à l'administrateur système de créer et supprimer des comptes utilisateur et de leur donner ou retirer des droits. `GRANT` et `REVOKE` sont implémentées en MySQL 3.22.11 ou plus récent. Pour les anciennes versions de MySQL, ces commandes ne font rien.

Les informations sur les comptes MySQL sont stockés dans la base `mysql` . Cette base et son accès sont présentés en détails dans la section Administration de la base .

Les droits sont donnés à 4 niveaux :

Niveau global
: Les droits globaux s'appliquent à toutes les bases de données d'un serveur. Ces droits sont stockés dans la table `mysql.user` . `REVOKE ALL ON *.*` retirera seulement les privilèges globaux.

Niveau base de données
: Les droits de niveau de base de données s'appliquent à toutes les tables d'une base de données. Ces droits sont stockés dans les tables `mysql.db` et `mysql.host` . `REVOKE ALL ON db.*` retirera seulement les privilèges de base de données.

Niveau table
: Les droits de table s'appliquent à toutes les colonnes d'une table. Ces droits sont stockés dans la table `mysql.tables_priv` . `REVOKE ALL ON db.table` retirera seulement les privilèges de table.

Niveau colonne
: Les droits de niveau de colonnes s'appliquent à des colonnes dans une table. Ces droits sont stockés dans la table `mysql.columns_priv` . Quand vous utilisez `REVOKE` vous devez spécifier les mêmes colonnes qui s'étaient vues accorder des privilèges.

Pour faciliter la suppression de tous les droits d'un utilisateur, MySQL 4.1.2 a ajouté la syntaxe suivante, qui efface tous les droits de base, table et colonne pour un utilisateur donné :

```
REVOKE ALL PRIVILEGES,GRANT FROM user_name [, user_name ...]
```

Avant MySQL 4.1.2, tous les droits ne peuvent pas être effacés d'un coup. Il faut deux commandes pour cela :

```
 REVOKE ALL PRIVILEGES FROM user [, user] ...
REVOKE GRANT OPTION FROM user [, user] ...
```

Pour les commandes `GRANT` et `REVOKE` , la clause `priv_type` peut être spécifiée par les constantes suivantes :

Droit	Signification
ALL [PRIVILEGES]	Tous les droits sauf WITH GRANT OPTION .
ALTER	Autorise l'utilisation de ALTER TABLE .
CREATE	Autorise l'utilisation de CREATE TABLE .
CREATE TEMPORARY TABLES	Autorise l'utilisation de CREATE TEMPORARY TABLE .
DELETE	Autorise l'utilisation de DELETE .
DROP	Autorise l'utilisation de DROP TABLE .
EXECUTE	Autorise l'utilisateur à exécuter des procédures stockées (pour MySQL 5.0).
FILE	Autorise l'utilisation de SELECT ... INTO OUTFILE et LOAD DATA INFILE .
INDEX	Autorise l'utilisation de CREATE INDEX et DROP INDEX .
INSERT	Autorise l'utilisation de INSERT .
LOCK TABLES	Autorise l'utilisation de LOCK TABLES sur les tables pour lesquelles l'utilisateur a les droits de SELECT .
PROCESS	Autorise l'utilisation de SHOW FULL PROCESSLIST .
REFERENCES	Réservé pour le futur.
RELOAD	Autorise l'utilisation de FLUSH .
REPLICATION CLIENT	Donne le droit à l'utilisateur de savoir où sont les maîtres et esclaves.
REPLICATION SLAVE	Nécessaire pour les esclaves de réplication (pour lire les historiques binaires du maître).
SELECT	Autorise l'utilisation de SELECT .
SHOW DATABASES	SHOW DATABASES affiche toutes les bases de données.
SHUTDOWN	Autorise l'utilisation de mysqladmin shutdown .
SUPER	Autorise une connexion unique même si max_connections est atteint, et l'exécution des commandes CHANGE MASTER , KILL thread , mysqladmin debug , PURGE MASTER LOGS et SET GLOBAL .
UPDATE	Autorise l'utilisation de UPDATE .
USAGE	Synonyme de ``pas de droits''.
GRANT OPTION	Synonyme pour WITH GRANT OPTION

`USAGE` peut être utilisé lorsque vous voulez créer un utilisateur sans aucun droit.

Les droits de `CREATE TEMPORARY TABLES` , `EXECUTE` , `LOCK TABLES` , `REPLICATION ...` , `SHOW DATABASES` et `SUPER` sont nouveaux en version 4.0.2. Pour utiliser ces droits après mise à jour en 4.0.2, vous devez exécuter le script `mysql_fix_privilege_tables` . Mise à jour des tables de droits .

Dans les anciennes versions de MySQL, le droit de `PROCESS` donnait les mêmes droits que le nouveau droit `SUPER` .

Vous pouvez donner des droits globaux en utilisant la syntaxe `ON *.*` . Vous pouvez donner des droits de base en utilisant la syntaxe `ON nom_base.*` . Si vous spécifiez `ON *` et que vous avez une base de données qui est déjà sélectionnée, vous allez donner des droits pour la base de données courante. **Attention** : si vous spécifiez `ON *` et que vous **n'avez pas** de base courante, vous allez affecter les droits au niveau du serveur !

Les droits `EXECUTION` , `FILE` , `PROCESS` , `RELOAD` , `REPLICATION CLIENT` , `REPLICATION SLAVE` , `SHOW DATABASES` , `SHUTDOWN` et `SUPER` sont des droits d'administration, qui ne peuvent être donnés que globalement (avec la syntaxe `ON *.*`).

Les autres droits peuvent être donnés globalement ou à des niveaux plus spécifiques.

Les seuls droits `priv_type` que vous pouvez donner au niveau d'une table sont `SELECT` , `INSERT` , `UPDATE` , `DELETE` , `CREATE` , `DROP` , `GRANT OPTION` , `INDEX` et `ALTER` .

Les seuls droits `priv_type` que vous pouvez donner au niveau d'une colonne (avec la clause `column_list`) sont `SELECT` , `INSERT` et `UPDATE` .

`GRANT ALL` assigne des droits que vous possédez au niveau où vous le possédez. Par exemple, si vous utilisez `GRANT ALL ON db_name.*` , qui est un droit de niveau de base de données, aucun des droits globaux, comme `FILE` ne sera donné.

MySQL vous permet de donner des droits au niveau d'une base de données, même si la base de données n'existe pas, pour vous aider à préparer l'utilisation de la base de données. Actuellement, MySQL ne vous permet pas de créer des droits pour une table si la table n'existe pas.

MySQL ne supprime pas les droits lorsqu'un utilisateur efface une table ou une base.

Notez bien : les caractères joker `'_'` et `'%'` sont autorisés lors de la spécification de noms dans la commande `GRANT` . Cela signifie que si vous voulez utiliser par exemple le caractère littéral `'_'` comme nom de base, vous devez le spécifier sous la forme `'\_'` dans la commande `GRANT` , pour éviter à l'utilisateur d'accéder à d'autres bases, dont le nom pourrait correspondre au masque d'expression régulière ainsi créé. Utilisez plutôt `` GRANT ... ON `foo\_bar`.* TO ... `` .

Afin de permettre l'identification des utilisateurs depuis des hôtes arbitraires, MySQL supporte la spécification du nom d'utilisateur `nom_utilisateur` sous la forme `user@host` . Si vous voulez spécifier un nom d'utilisateur `user` qui contient des caractères spéciaux tels que `'-'` , ou une chaîne d'hôte `host` qui contient des caractères joker (comme `'%'`), vous pouvez placer le nom de l'utilisateur ou de l'hôte entre guillemets (par exemple, `'test-utilisateur'@'test-nomdhote'`).

Vous pouvez spécifier des caractères jokers dans le nom d'hôte. Par exemple, `user@'%.loc.gov'` fait correspondre l'utilisateur `user` de n'importe quel hôte du domaine `loc.gov` , et `user@'144.155.166.%'` fait correspondre l'utilisateur `user` à n'importe quelle adresse de la classe C `144.155.166` .

La forme simple de `user` est synonyme de `user@"%"` .

MySQL ne supporte pas de caractères joker dans les noms d'utilisateur. Les utilisateurs anonymes sont définis par l'insertion de ligne avec `User=''` dans la table `mysql.user` , ou en créant un utilisateur avec un nom vide, grâce à la commande `GRANT` .

```
mysql> GRANT ALL ON test.* TO ''&#64;'localhost' ...
```

Attention : si vous autorisez des utilisateurs anonymes à se connecter à votre serveur, vous devriez aussi donner ces droits à tous les utilisateurs locaux `user@localhost` car sinon, la ligne dans la table `mysql.user` sera utilisée lorsque l'utilisateur se connectera au serveur MySQL depuis la machine locale ! (Ce compte est créé durant l'installation de MySQL.)Vous pouvez vérifier si cela s'applique à vous en exécutant la requête suivante :

```
mysql> SELECT Host,User FROM mysql.user WHERE User='';
```

Si vous voulez effacer les utilisateurs anonymes d'un serveur, utilisez ces commandes :

```
mysql> DELETE FROM mysql.user WHERE Host='localhost' AND User='';
mysql> FLUSH PRIVILEGES;
```

Actuellement, la commande `GRANT` supporte uniquement les noms d'hôte, colonne, table et bases de données d'au plus 60 caractères. Un nom d'utilisateur peut être d'au plus 16 caractères.

Les droits pour les tables et colonnes sont combinés par OU logique, avec les quatre niveaux de droits. Par exemple, si la table `mysql.user` spécifie qu'un utilisateur a un droit global de `SELECT` , ce droit ne pourra pas être annulé au niveau base, table ou colonne.

Les droits d'une colonne sont calculés comme ceci :

```
droit global
OR (droit de base de données ET droit d'hôte)
OR droit de table
OR droit de colonne
```

Dans la plupart des cas, vous donnez des droits à un utilisateur en utilisant un seul des niveaux de droits ci−dessus, ce qui fait que la vie n'est pas aussi compliquée. Le détails de la procédure de vérification des droits et disponible dans Sécurité générale et accès à MySQL .

Si vous donnez des droits à une paire utilisateur/hôte qui n'existe pas dans la table `mysql.user` , une ligne sera créée et restera disponible jusqu'à son effacement avec la commande `DELETE` . En d'autre termes, `GRANT` crée une ligne dans la table `user` , mais `REVOKE` ne la supprime pas. Vous devez le faire explicitement avec la commande `DELETE` .

Avec MySQL version 3.22.12 ou plus récent, si un nouvel utilisateur est créé, ou si vous avez les droits de `GRANT` globaux, le mot de passe sera configuré avec le mot de passe spécifié avec la clause `IDENTIFIED BY` , si elle est fournie. Si l'utilisateur a déjà un mot de passe, il sera remplacé par ce nouveau.

Attention : si vous créez un nouvel utilisateur, mais ne spécifiez pas de clause `IDENTIFIED BY` , l'utilisateur n'aura pas de mot de passe. Ce n'est pas sécuritaire.

Les mots de passe peuvent aussi être modifiés avec la commande `SET PASSWORD` . Syntaxe de `SET PASSWORD` .

Si vous ne voulez pas transmettre le mot de passe en texte clair, vous pouvez immédiatement utiliser l'option `PASSWORD` suivi du mot de passe déjà chiffré avec la fonction `PASSWORD()` ou l'API C `make_scrambled_password(char *to, const char *password)` .

Si vous donnez les droits de base, une ligne sera ajoutée dans la table `mysql.db` . Lorsque les droits sur cette base seront supprimés avec la commande `REVOKE` , cette ligne disparaîtra.

Si un utilisateur n'a pas de droit sur une table, elle ne sera pas affichée lorsqu'il demandera la liste des tables avec la commande `SHOW TABLES` . Si un utilisateur n'a pas de droit dans une base, le nom de la base ne sera pas affiché par `SHOW DATABASES` à moins que l'utilisateur n'ai un droit de `SHOW DATABASES` .

La clause `WITH GRANT OPTION` donne à l'utilisateur le droit de donner les droits qu'il possède à d'autres utilisateurs. La plus grande prudence est recommandée pour cette commande, car il permettra à terme à deux utilisateurs de combiner les droits dont ils disposent.

Vous ne pouvez pas donner un droit que vous ne possédez pas. le droit de `GRANT OPTION` ne vous donne le droit que de donner les droits que vous possédez.

Sachez que si vous donnez à quelqu'un le droit de `GRANT OPTION` , tous les droits que possède cet utilisateur seront distribuables. Supposez que vous donnez à un utilisateur le droit d' `INSERT` dans une base de données. Si vous donnez le droit de `SELECT` sur une base, et spécifiez l'option `WITH GRANT OPTION` , l'utilisateur peut distribuer non seulement son droit de `SELECT` , mais aussi son droit de `INSERT` . Si vous donnez ensuite le droit de `UPDATE` , il pourra alors distribuer `INSERT` , `SELECT` et `UPDATE` .

Il est recommandé de ne pas donner de droits de `ALTER` à un utilisateur normal. Si vous le faîtes, l'utilisateur pourra essayer de contourner le système de droits en renommant des tables.

`MAX_QUERIES_PER_HOUR #` , `MAX_UPDATES_PER_HOUR #` et `MAX_CONNECTIONS_PER_HOUR #` sont nouveaux en MySQL 4.0.2. Ces deux options limitent le nombre de requêtes et de modifications qu'un utilisateur peut réclamer dans une heure. Si `#` vaut 0 (valeur par défaut), alors cela signifie qu'il n'y a pas de limitations pour cet utilisateur. Limiter la consommation de ressources . Note: pour spécifier l'une de ces options pour un utilisateur existant sans ajouter d'autres privilèges additionnels, utilisez `GRANT USAGE ... WITH MAX_...` .

MySQL peut vérifier les attributs X509 en plus des éléments d'identifications habituels, comme le nom d'utilisateur et le mot de passe. Pour spécifier des options SSL pour un compte MySQL, utilisez la clause `REQUIRE` de la commande `GRANT` . Pour des informations générales sur SSL et MySQL, voyez Utilisation des connexions sécurisées .

Il y a différentes possibilités pour limiter le type de connexions d'un compte :

- Si un compte ne doit pas utiliser SSL ou X509, les connexions sont autorisées si le mot de passe et le nom d'utilisateur sont valides. Cependant, les connexions non−chiffrées peuvent aussi être utilisées par le client, si le client dispose des bons certificats et clés.
- L'option `REQUIRE SSL` limite le serveur aux connexions chiffrées avec SSL. Notez que cette option peut être omise s'il y a des lignes d'identifications qui autorisent les connexions non chiffrées.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'root'&#64;'localhost'
    -> IDENTIFIED BY 'goodsecret' REQUIRE SSL;
```

- `REQUIRE X509` signifie que le client doit avoir un certificat valide, mais que le certificat exact, l'émetteur et le sujet n'ont pas d'importance. La seule obligation est qu'il faut pouvoir vérifier la signature auprès d'une des autorités de certification.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'root'&#64;'localhost'
    -> IDENTIFIED BY 'goodsecret' REQUIRE X509;
```

- `REQUIRE ISSUER 'issuer'` impose aux connexions l'utilisation d'un certificat X509 émis par l'autorité de certification `'issuer'` . Si le client présente un certificat d'une autre autorité, le serveur rejette la connexion. L'utilisation des certificats X509 implique toujours un chiffrement, ce qui fait que l'option `SSL` est inutile.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'root'&#64;'localhost'
    -> IDENTIFIED BY 'goodsecret'
    -> REQUIRE ISSUER '/C=FI/ST=Some-State/L=Helsinki/
       O=MySQL Finland AB/CN=Tonu Samuel/Email=tonu&#64;example.com';
```

  Notez que la valeur de `ISSUER` doit être saisie comme une seule chaîne.
- `REQUIRE SUBJECT 'subject'` impose à la connexion à la présentation d'un certificat X509 avec le sujet `'subject'` . Si le client présente un certificat qui est valide, mais avec un autre sujet, le serveur rejette la connexion.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'root'&#64;'localhost'
    -> IDENTIFIED BY 'goodsecret'
    -> REQUIRE SUBJECT '/C=EE/ST=Some-State/L=Tallinn/
       O=MySQL demo client certificate/
       CN=Tonu Samuel/Email=tonu&#64;example.com';
```

  Notez que la valeur de `SUBJECT` doit être saisie comme une seule chaîne.
- `REQUIRE CIPHER 'cipher'` est nécessaire pour s'assurer que des tailles de clé et chiffrements suffisantes sont utilisées. SSL peut être faible si de vieux algorithmes avec des clés courtes sont utilisées. En utilisant cette option, vous pouvez spécifier un chiffrement spécifique.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'root'&#64;'localhost'
    -> IDENTIFIED BY 'goodsecret'
    -> REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';
```

Les clauses `SUBJECT` , `ISSUER` et `CIPHER` peuvent être combinées avec la clause `REQUIRE` comme ceci :

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'root'&#64;'localhost'
    -> IDENTIFIED BY 'goodsecret'
    -> REQUIRE SUBJECT '/C=EE/ST=Some-State/L=Tallinn/
       O=MySQL demo client certificate/
       CN=Tonu Samuel/Email=tonu&#64;example.com'
    -> AND ISSUER '/C=FI/ST=Some-State/L=Helsinki/
       O=MySQL Finland AB/CN=Tonu Samuel/Email=tonu&#64;example.com'
    -> AND CIPHER 'EDH-RSA-DES-CBC3-SHA';
```

Notez que les valeurs de `SUBJECT` et `ISSUER` doivent être saisies comme une seule chaîne.

Depuis MySQL 4.0.4, le mot clé `AND` est optionnel entre les clauses de `REQUIRE` .

L'ordre des options est libre, mais les options doivent être spécifiées une seule fois.

Notez que si vous utilisez des droits de niveau table ou colonne même pour un utilisateur, le serveur vérifiera alors ces droits pour tous les utilisateurs, et cela ralentira MySQL un peu.

Lorsque `mysqld` démarre, tous les droits sont stockés en mémoire. Les droits de bases, tables et colonnes prennent aussitôt effet, et les droits des utilisateurs prendront effet dès leur prochaine configuration. Les modifications sur les tables de droits que vous effectuez avec les commandes

`GRANT` et `REVOKE` sont prises en compte immédiatement par le serveur. Si vous modifiez manuellement les tables (avec `INSERT` , `UPDATE` , etc...), vous devez exécuter la commande `FLUSH PRIVILEGES` , ou la commande en ligne `mysqladmin flush-privileges` pour indiquer au serveur qu'il doit recharger les droits. Quand est−ce que les droits prennent effets .

Les différences notables entre l'ANSI SQL et MySQL pour la commande `GRANT` sont :

- Les droits MySQL sont donnés pour une combinaison nom d'utilisateur + nom d'hôte, et non pas pour un nom d'hôte seulement.
- L'ANSI SQL n'a pas de droits globaux ou de niveau base de données, et l'ANSI SQL ne supporte pas tous les types de droits que MySQL supporte. MySQL ne supporte pas le droit ANSI SQL de `TRIGGER` ou `UNDER` .
- MySQL ne supporte pas les droits standard SQL `TRIGGER` et `UNDER` .
- Les droits ANSI SQL sont structurés de manière hiérarchique. Si vous supprimez un utilisateur, tous les droits donnés à cet utilisateur seront supprimés. Avec MySQL, les droits ne sont pas automatiquement supprimés, et vous devez les supprimer manuellement, si besoin.
- Avec MySQL, si vous avez le droit de `INSERT` sur uniquement quelques colonnes de la table, vous pourrez exécuter des insertions. Les colonnes pour lesquelles vous n'avez pas de droit prendront alors leur valeur par défaut. L'ANSI SQL vous impose d'avoir les droits d' `INSERT` sur toutes les colonnes.
- Lorsque vous détruisez une table avec ANSI SQL, tous les droits liés à la table sont supprimés. Si vous supprimez un droit en ANSI SQL, tous les droits qui étaient basés sur ce droit sont supprimés. Avec MySQL, les droits peuvent être abandonnés explicitement avec la commande `REVOKE` , ou en manipulant les tables de droits de MySQL.

14.5.1.3 Syntaxe de SET PASSWORD

```
 SET PASSWORD = PASSWORD('some password')
SET PASSWORD FOR user = PASSWORD('some password')
```

La commande `SET PASSWORD` assigne un mot de passe à un compte utilisateur existant.

La première syntaxe modifie le mot de passe de l'utilisateur courant. Tout client qui s'est connecté avec un compte non−anonyme peut changer le mot de passe pour ce compte.

La seconde syntaxe modifie le mot de passe pour un compte tiers, sur le serveur. Seuls les clients qui ont accès aux bases `mysql` peuvent faire cela. La valeur de `user` doit être donnée au format `user_name@host_name` , où `user_name` et `host_name` sont tels que listés dans les colonnes `User` et `Host` de la table `mysql.user` . Par exemple, si vous avez une ligne avec les champs `User` et `Host` qui valent `'bob'` et `'%.loc.gov'` , vous pouvez écrire la commande suivante :

```
 mysql> SET PASSWORD FOR 'bob'&#64;'%.loc.gov' = PASSWORD('newpass');
```

C'est l'équivalent de la commande suivante :

```
 mysql> UPDATE mysql.user SET Password=PASSWORD('newpass')
    -> WHERE User='bob' AND Host='%.loc.gov';
mysql> FLUSH PRIVILEGES;
```

14.5.2 Commandes d'entretien des tables

14.5.2.1 Syntaxe de ANALYZE TABLE

```
 ANALYZE TABLE nom_de_table[,nom_de_table...]
```

Cette commande analyse et stocke la clé de distribution de la table. Durant l'analyse, la table est verrouillée en lecture. Cette commande fonctionne avec les tables `MyISAM` et `BDB` .C'est l'équivalent de la commande en ligne `myisamchk -a` .

MySQL utilise les clés de distribution pour décider dans quel ordre les tables doivent être rassemblées lors des jointures qui ne s'effectuent pas sur une constante.

La commande retourne une table avec les colonnes suivantes :

Colonne	Valeur
Table	Nom de la table
Op	`` analyze '' (toujours)
Msg_type	Un des status , error , info ou warning .
Msg_text	Le message.

Vous pouvez vérifier la clé de distribution stockée avec la commande `SHOW INDEX` . Syntaxe de `SHOW DATABASES` .

Si la table n'a pas changé depuis la dernière commande `ANALYZE TABLE` , elle ne sera pas analysée à nouveau.

14.5.2.2 Syntaxe de BACKUP TABLE

```
BACKUP TABLE nom_de_table[,nom_de_table...] TO '/chemin/vers/le/dossier/de/sauvegardes'
```

Note : cette commande est obsolète. Nous travaillons à une solution de remplacement améliorée, qui proposera des sauvegardes à chaud. Durant ce temps, le script `mysqlhotcopy` peut être utilisé.

Cette commande copie le nombre minimal de fichiers de table dont en a besoin pour la restaurer vers le dossier de sauvegardes après avoir rafraîchit les changements dans le disque. Cela ne fonctionne actuellement que pour les tables au format `MyISAM` . Pour les tables `MyISAM` , elle ne copie que les fichiers `.frm` (définition) et `.MYD` (données), le fichier d'index `.MYD` pouvant, lui, être reconstruit à partir des deux autres.

Avant d'utiliser cette commande, merci de lire Sauvegardes de base de données .

Pendant la sauvegarde, un verrou de lecture est posé sur chaque table, une par une, lors de leur copie. Si vous voulez sauvegarder une image instantanée de plusieurs table, vous devez d'abord exécuter un `LOCK TABLES` obtenant un verrou de lecture pour chaque table concernée.

La commande retourne une table avec les colonnes suivantes :

Colonne	Valeur
Table	Nom de la table
Op	Toujours `` backup ''
Msg_type	status , error , info ou encore warning .
Msg_text	Le message.

Notez que `BACKUP TABLE` n'est disponible en MySQL que depuis la version 3.23.25.

14.5.2.3 Syntaxe de CHECK TABLE

```
CHECK TABLE tbl_name[,tbl_name...] [option [option...]]

option = QUICK | FAST | MEDIUM | EXTENDED | CHANGED
```

`CHECK TABLE` ne fonctionne qu'avec les tables `MyISAM` et `InnoDB` . Avec les tables `MyISAM` , c'est l'équivalent de la commande `myisamchk -m table_name` sur la table.

Par défaut, l'option `MEDIUM` est utilisée.

Cette commande vérifie l'intégrité des tables. Pour les tables `MyISAM` , des statistiques importantes sont mises à jour. La commande retourne les informations suivantes sur la table dans les colonnes suivantes :

Colonne	**Valeur**
Table	Nom de la table.
Op	Toujours ``check''.
Msg_type	Un des statut status , error , info ou warning .
Msg_text	Le message.

Notez que vous pouvez obtenir de nombreuses lignes d'informations pour chaque table. La dernière ligne sera du type `Msg_type status` et doit être normalement `OK` . Si vous n'obtenez pas de statut `OK` ou `Not checked` , il vous faudra exécuter une réparation de la table. Utilisation de `myisamchk` pour maintenir les tables et recouvrir les données . `Not checked` signifie que la table a indiqué qu'il n'y a pas de vérification à faire.

Les différents types de vérifications sont les suivants :

Type	**Signification**
QUICK	N'analyse pas les lignes pour vérifier les liens erronés.
FAST	Ne vérifie que les tables qui n'ont pas été correctement fermées.
CHANGED	Ne vérifie que les tables qui ont changées depuis la dernière vérification, ou bien qui n'ont pas été correctement fermées.
MEDIUM	Analyse les lignes pour s'assurer que les liens effacés sont corrects. Cette option calcule aussi la somme de contrôle des lignes, et la vérifie avec la somme de contrôle des clés.
EXTENDED	Fait une vérification complète des liens pour chaque ligne. Cela vérifie que la table est totalement cohérente, mais cela peut prendre beaucoup de temps.

Pour les tables à format de dynamique de type `MyISAM` , une vérification de table sera toujours démarrée avec une option de niveau `MEDIUM` . Pour les tables à format de ligne statique, nous évitons les niveaux de `QUICK` et `FAST` car les lignes sont rarement corrompues.

Vous pouvez combiner les options de vérification comme ceci :

```
CHECK TABLE test_table FAST QUICK;
```

L'exemple ci−dessus va simplement faire une vérification de la table, pour s'assurer qu'elle a été correctement fermée. **Note** : dans certains cas, `CHECK TABLE` va modifier la table! Cela arrive si la table a été marquée comme " `corrupted` " et " `not closed properly` " mais `CHECK TABLE` n'a trouvé aucun problème dans la table. Dans ce cas, `CHECK TABLE` va marquer la table comme correcte.

Si une table est corrompue, il est probable que les problèmes sont dans les fichiers d'index et non pas dans les données. Tous les types de vérifications présentés ci−dessus vérifient les index soigneusement, et ils devraient trouver la plupart des erreurs.

Si vous voulez simplement vérifier une table que vous supposez correcte, vous pouvez n'utiliser aucune option, ou l'option `QUICK` . Cette dernière peut aussi être utilisée si vous êtes pressé, et que vous pouvez prendre le risque minime que `QUICK` ne trouve pas d'erreur dans votre fichier. Dans la plupart des cas, MySQL doit trouver toutes les erreurs de données, pour un usage normal. Si cela arrive, alors la table est marquée comme 'corrupted', auquel cas, la table ne pourra pas être utilisée tant qu'elle n'a pas été réparée).

`FAST` et `CHANGED` sont surtout destinées à être utilisées depuis un script : par exemple, il peut être exécuté depuis une tâche `cron` , si vous voulez vérifier la table de temps en temps. Dans la plupart des cas, l'option `FAST` doit être préférée à `CHANGED` : le seul cas ou vous pourriez préférez `CHANGED` est lorsque vous soupçonnez avoir trouvé un bogue dans les tables `MyISAM` .

`EXTENDED` ne doit être utilisé qu'après une vérification normale, et que vous obtenez toujours des erreurs étranges lorsque MySQL essaie de modifier une ligne ou trouve une ligne avec clé (ce qui est très rare, si une vérification a réussie).

Certains problèmes rapportés par la commande `CHECK TABLE` , ne peuvent être corrigés automatiquement :

- `Found row where the auto_increment column has the value 0` .

  Cela signifie que vous avez dans votre table une ligne qui contient la valeur 0 alors qu'elle est de type `AUTO_INCREMENT` . (Il est possible de créer une ligne où la colonne `AUTO_INCREMENT` vaut 0 en spécifiant explicitement la valeur 0 dans la colonne avec la commande `UPDATE` .

  Ce n'est pas une erreur en soit, mais cela peut poser des problèmes si vous décidez de sauver cette table dans un fichier texte, et de la restaurer, ou encore d'appliquer la commande `ALTER TABLE` sur la table. Dans ce cas, la colonne `AUTO_INCREMENT` va changer automatiquement de valeur, en suivant les règles des colonnes de type `AUTO_INCREMENT` , qui vont causer un problème de clé doublon.Pour se débarasser de cette alerte, vous devez utiliser une commande `UPDATE` sur la table, pour mettre une valeur différente de 0 dans cette colonne.

14.5.2.4 Syntaxe de CHECKSUM TABLE

```
CHECKSUM TABLE tbl_name[,tbl_name ...] [ QUICK | EXTENDED ]
```

Calcule la somme de contrôle de la table.

Si `QUICK` est spécifié, la somme de contrôle instantanée est retournée, ou `NULL` si la table ne supporte pas les sommes de contrôle instantanées. Syntaxe de `CREATE TABLE` .

En mode `EXTENDED` , toute la table est lue, ligne par ligne, et la somme de contrôle est calculée. Cela peut être très lent pour les tables de grande taille.

Par défaut, sans `QUICK` ni `EXTENDED` , MySQL retourne la somme de contrôle si la table le supporte, et sinon, scanne la table.

Cette commande a été ajoutée en MySQL 4.1.1.

14.5.2.5 Syntaxe de OPTIMIZE TABLE

```
OPTIMIZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name [, tbl_name] ...
```

`OPTIMIZE TABLE` doit être utilisée si une grande partie de la base a été effacée, ou si vous avez fait de nombreuses modifications dans une table à format de ligne dynamique (des tables qui ont des colonnes de type `VARCHAR` , `BLOB` et `TEXT`). Les lignes effacées sont conservées dans une liste, et les prochaines opérations d' `INSERT` réutilisent les vieilles positions de lignes. Vous pouvez vous servir de la commande `OPTIMIZE TABLE` pour récupérer l'espace utilisé et défragmenter le fichier de données.

Dans la plupart des installations, vous n'avez pas à utiliser `OPTIMIZE TABLE` . Même si vous faites beaucoup de mises à jour sur des colonnes à taille dynamique, il n'est pas évident que vous ayez à passer cette commande plus d'une fois par semaine ou par mois, et uniquement sur quelques tables.

Pour le moment, `OPTIMIZE TABLE` fonctionne uniquement avec les tables de type `MyISAM` et `BDB` . Pour les tables `BDB` , `OPTIMIZE TABLE` est actuellement l'équivalent de `ANALYZE TABLE` . Syntaxe de `ANALYZE TABLE` .

Vous pouvez vous arranger pour que OPTIMIZE TABLE fonctionne sur d'autres types de tables, en démarrant mysqld avec --skip-new ou --safe-mode , mais dans ce cas, OPTIMIZE TABLE est simplement l'équivalent de ALTER TABLE .

OPTIMIZE TABLE fonctionne comme ceci :

- Si la table contient des lignes effacées ou des lignes fragmentées, la table est compactée.
- Si les pages d'index ne sont pas triées, OPTIMIZE TABLE les trie.
- Si les statistiques ne sont pas à jour (et que la table n'a pas pu effectuer de réparation en triant l'index), elles sont mises à jour.

Notez que la table est verrouillée durant la commande OPTIMIZE TABLE .

Avant MySQL 4.1.1, OPTIMIZE n'était pas reportée dans le log binaire. Depuis MySQL 4.1.1 elles le sont, à moins que l'attribut optionnel NO_WRITE_TO_BINLOG ou son alias LOCAL ne soit utilisé.

14.5.2.6 Syntaxe de REPAIR TABLE

```
REPAIR TABLE tbl_name[,tbl_name...] [QUICK] [EXTENDED] [USE_FRM]
```

REPAIR TABLE répare une table possiblement corrompue. Par défaut, elle a le même effet que myisamchk --recover tbl_name sur une table. REPAIR TABLE fonctionne unqiuement avec les tables MyISAM .Normalement, vous n'avez pas à exécuter cette commande, mais si une catastrophe vous frappe, vous êtes presque assurés de retrouver vos données dans les tables MyISAM, avec la commande REPAIR TABLE . Si vos tables sont souvent corrompues, vous devrie toutefois rechercher la cause de ce problème! Que faire si MySQL crashe constamment . Problèmes avec les tables MyISAM .

REPAIR TABLE répare autant que possible les tables corrompues. La commande retourne la table suivante :

Colonne	Valeur
Table	Nom de la table
Op	Toujours repair
Msg_type	Un des statut status , error , info ou warning .
Msg_text	Le message

La commande REPAIR TABLE pourrait afficher plusieurs messages pour chaque table. La dernière ligne doit être du format Msg_type status et doit être normalement OK . Si vous n'obtenez pas OK , vous devez essayer de réparer votre table avec la commande myisamchk -o , car REPAIR TABLE de supporte pas encore toutes les options de myisamchk . Dans un futur proche, nous allons rendre cette commande encore plus souple.

Si l'option QUICK est fournie, alors MySQL va essayer de ne réparer que le fichier d'index. Ce type de réparation est le même que myisamchk --recover --quick .

Si vous utilisez l'option EXTENDED , alors MySQL va essayer de créer l'index ligne par ligne, au lieu de créer un index à la fois, par tri. C'est une méthode qui peut s'avérer plus efficace que de trier sur des clés de taille fixe, si vous avez des clés CHAR longues qui se compressent bien. Ce type de réparation est l'équivalent de myisamchk --safe-recover .

Depuis MySQL 4.0.2, il existe le mode USE_FRM pour REPAIR . Utilisez−le si le fichier .MYI manque, ou si son entête est corrompu. Avec ce mode, MySQL va recréer le fichier .MYI , en utilisant les informations du fichier .frm . Ce type de réparation ne peut pas être fait avec myisamchk .

Attention : si le serveur s'arrête durant l'opération `REPAIR TABLE` , il est important d'exécuter à nouveau la commande `REPAIR TABLE` après le redémarrage (il est bon de faire une sauvegarde de toutes manières). Dans le pire scénario, vous pourriez vous retrouver avec un nouvel index sans relation avec les données, et la prochaine opération risque d'écraser le fichier de données. C'est peu probable, mais possible.

Avant MySQL 4.1.1, les commandes `REPAIR TABLE` n'étaient pas écrites dans le log binaire. Depuis MySQL 4.1.1, elles sont écrites dans le log binaire à moins que la clause `NO_WRITE_TO_BINLOG` ne soit utilisée (aussi connue sous le nom de `LOCAL`).

14.5.2.7 Syntaxe de RESTORE TABLE

```
RESTORE TABLE nom_de_table[,nom_de_table...] FROM '/chemin/vers/le/dossier/de/sauvegardes'
```

Restaure la ou les tables à partir d'une sauvegarde effectuée avec `BACKUP TABLE` . Les tables existantes ne seront pas écrasées et dans ce cas là, vous obtiendrez une erreur. La restauration prendra plus de temps que la sauvegarde à cause de la reconstruction du fichier d'index. Plus vous avez de clefs, plus la restauration sera longue. Tout comme `BACKUP TABLE` , `RESTORE TABLE` fonctionne seulement avec les tables `MyISAM` .

La sauvegarde de chaque table est constituée du fichier de format `.frm` et du fichier de données `.MYD` . L'opération de restauration restaure ces fichiers, puis les utilise pour reconstruire le fichier d'index `.MYI` . La restauration prend plus de temps que la sauvegarde, car il faut reconstituer l'index. Plus la table a d'index, plus cela prendra de temps.

Cette commande retourne un tableau avec les colonnes suivantes :

Colonne	**Valeur**
Table	Nom de la table
Op	Toujours `` restore ''
Msg_type	status , error , info ou encore warning .
Msg_text	Le message.

14.5.3 Syntaxe de SHOW

`SET` vous permet de modifier des variables et options.

`SHOW` a de très nombreuses formes, pour donner des informations sur les bases, tables, colonnes ou le serveur. Cette section les décrit.

```
 SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [LIKE 'pattern']
SHOW CREATE DATABASE db_name
SHOW CREATE TABLE tbl_name
SHOW DATABASES [LIKE 'pattern']
SHOW [STORAGE] ENGINES
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW GRANTS FOR user
SHOW INDEX FROM tbl_name [FROM db_name]
SHOW INNODB STATUS
SHOW [BDB] LOGS
SHOW PRIVILEGES
SHOW [FULL] PROCESSLIST
SHOW STATUS [LIKE 'pattern']
SHOW TABLE STATUS [FROM db_name] [LIKE 'pattern']
SHOW [OPEN] TABLES [FROM db_name] [LIKE 'pattern']
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern']
SHOW WARNINGS [LIMIT [offset,] row_count]
```

Si la syntaxe d'une commande `SHOW` inclut la clause `LIKE 'pattern'` , `'pattern'` est une expression régulière qui peut contenir les jokers `'%'` et `'_'` . Ce expression est utile pour restreindre la commande à une partie des valeurs normalement retournées.Notez qu'il y a d'autres formes pour ces commandes, décrites à d'autres endroits du manuel :

- La commande `SET PASSWORD` pour assigner un mot de passe à un compte est présentée dans Syntaxe de `SET PASSWORD` .
- La commande `SHOW` a des formes pour afficher des informations sur la réplication, pour le maître et l'esclave :

  ```
   SHOW BINLOG EVENTS
  SHOW MASTER LOGS
  SHOW MASTER STATUS
  SHOW SLAVE HOSTS
  SHOW SLAVE STATUS
  ```

  Ces formes de `SHOW` sont décrites dans Commandes de réplication .

14.5.3.1 Syntaxe de SET

```
SET variable_assignment [, variable_assignment] ...

variable_assignment:
     user_var_name = expr
   | [GLOBAL | SESSION] system_var_name = expr
   | &#64;&#64;[global. | session.]system_var_name = expr
```

`SET` permet de configurer plusieurs options qui affectent le comportement de votre serveur ou de votre client.

En MySQL 4.0.3, nous avons ajouté les options `GLOBAL` et `SESSION` et permis la modification des variables systèmes les plus importantes dynamiquement, durant l'exécution du serveur. Le système de variables que vous pouvez utiliser est décrit dans Variables dynamiques système .

Dans les anciennes versions de MySQL, nous avions autorisé l'utilisation de la syntaxe `SET OPTION` , mais elle est maintenant abandonnée. Omettez simplement le mot `OPTION` .

Les exemples suivants montrent les différentes syntaxes qu'on peut utiliser pour configurer des variables.

Une variable utilisateur s'écrit sous la forme `@var_name` et peut être configurée comme ceci :

```
SET &#64;var_name = expr;
```

Plus d'informations sur les variables utilisateurs sont données dans Variables définies par l'utilisateur .Les variables système peuvent être identifiées dans une commande `SET` sous la forme `var_name` . Le nom peut être optionnellement précédé par `GLOBAL` ou `@@global.` pour indiquer que cette variable est globale, ou par `SESSION` , `@@session.` , ou `@@` pour indiquer que cette variable est une variable de session. `LOCAL` et `@@local.` son synonymes de `SESSION` et `@@session.` . Si aucune option n'est présente, `SET` spécifie une variable de session.

La syntaxe `@@var_name` pour les variables système est supportée pour rendre la syntaxe MySQL compatible avec les autres bases.

Si vous configurez plusieurs variables sur une seule ligne de commande, le dernier mode `GLOBAL | SESSION` utilisé est pris en compte.

```
 SET sort_buffer_size=10000;
SET &#64;&#64;local.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION sort_buffer_size=1000000;
SET &#64;&#64;sort_buffer_size=1000000;
SET &#64;&#64;global.sort_buffer_size=1000000, &#64;&#64;local.sort_buffer_size=1000000;
```

Si vous utilisez `SESSION` (par défaut) l'option que vous configurez garde son effet jusqu'à ce que la session courante se termine, ou que vous modifiez à nouveau cette option. Si vous utilisez `GLOBAL` , qui requière le privilège `SUPER` , l'option est gardée en mémoire et utilisée pour les nouvelles connexion jusqu'au redémarrage du serveur. Si vous voulez qu'un changement reste permanent, vous devez l'effectuer dans l'un des fichiers d'options de MySQL. Utiliser les fichiers d'options .Pour éviter un mauvais usage, MySQL donnera une erreur si vous utilisez `SET GLOBAL` avec une variable qui ne peut être inutilisée que par `SET SESSION` ou si vous n'utilisez pas `SET GLOBAL` avec une variable globale.

Si vous voulez configurer une variable `SESSION` à une valeur `GLOBAL` ou une valeur `GLOBAL` à la valeur par défaut de MySQL, vous pouvez la configurer à `DEFAULT` .

```
SET max_join_size=DEFAULT;
SET &#64;&#64;session.max_join_size=&#64;&#64;global.max_join_size;
```

Vous pouvez obtenir une liste de la plupart des variables avec `SHOW VARIABLES` . Syntaxe de `SHOW VARIABLES` . Vous pouvez obtenir la valeur d'une variable spécifique avec la syntaxe `@@[global.|local.]nom_variable` :

```
SHOW VARIABLES LIKE 'max_join_size';
SHOW GLOBAL VARIABLES LIKE 'max_join_size';
```

Vous pouvez aussi obtenir une valeur spécifique d'une variable en utilisant la syntaxe `@@[global.|local.]var_name` avec `SELECT` :

```
SELECT &#64;&#64;max_join_size, &#64;&#64;global.max_join_size;
```

Lorsque vous lisez la valeur d'une variable avec la syntaxe `SELECT @@var_name` (c'est à dire, sans spécifier `global.` , `session.` ou `local.`), MySQL retourne la valeur de `SESSION` si elle existe, et la valeur `GLOBAL` sinon.

Vous trouverez ici une description des variables qui utilisent une syntaxe non−standard de `SET` . Les définitions des autres variables peuvent être trouvées dans la section des Variables sytème du serveur , avec les options de démarrage ou dans la description de `SHOW VARIABLES` .

Même si ces variables ne sont pas affichées par `SHOW VARIABLES` , vous pouvez obtenir leur valeur avec la commande `SELECT` (à l'exception de `CHARACTER SET`). Par exemple :

```
mysql> SELECT &#64;&#64;AUTOCOMMIT;
+--------------+
| &#64;&#64;autocommit |
+--------------+
|            1 |
+--------------+
```

AUTOCOMMIT= {0 | 1}
: Si définie à `1` tous les changements dans une table se feront en une seule fois. Pour démarrer une transaction multi−commandes, vous devez utiliser la commande `BEGIN` . Syntaxe des `BEGIN/COMMIT/ROLLBACK` . Si définie à `0` vous devez utiliser `COMMIT` / `ROLLBACK` pour accepter/annuler cette transaction. Notez que quand vous passez du mode non `AUTOCOMMIT` vers le mode `AUTOCOMMIT` , MySQL fera un `COMMIT` automatique sur toutes les transactions en cours.

BIG_TABLES = {0 | 1}
: Si définie à `1` , toutes les tables temporaires sont stockées sur le disque plutôt qu'en mémoire. Cela sera un peu plus lent, mais vous n'obtiendrez jamais l'erreur `The table nom_de_table is full` pour les grands `SELECT` qui requièrent une table temporaire. La valeur par défaut pour une nouvelle connexion est `0` (qui est d'utiliser la mémoire pour les tables temporaires). Cette option se nommait avant `SQL_BIG_TABLES` .

CHARACTER SET {charset_name | DEFAULT}
: Cela change le jeu de caractère dans toutes les chaînes du et vers le client avec le jeu donné. Jusqu'à maintenant, la seule option pour `nom_jeu_de_caractères` est `cp1251_koi8` , mais vous pouvez facilement ajouter d'autres possibilités en éditant le fichier `sql/convert.cc` dans la distribution des sources MySQL. Le jeu de caractères par défaut peut être restauré en utilisant la valeur `DEFAULT` de `nom_jeu_de_caractères DEFAULT` .

  Notez que la syntaxe pour configurer l'option `CHARACTER SET` diffère de la syntaxe pour configurer les autres options.

FOREIGN_KEY_CHECKS = {0 | 1}

Si cette option vaut `1` (par défaut), les contraintes de clé étrangères des tables `InnoDB` sont vérifiées. Si cette option vaut `0` , elles sont ignorées. Désactiver les clés étrangères peut être pratique pour recharger des tables `InnoDB` dans un ordre différent que celui qu'impose les relations de contraintes. Cette variable a été ajoutée en MySQL 3.23.52. Contraintes `FOREIGN KEY` .

IDENTITY = valeur

Cette variable est un synonyme de la variable `LAST_INSERT_ID` . Elle existe pour des raisons de compatibilité avec les autres bases. Depuis MySQL 3.23.25, vous pouvez lire sa valeur avec `SELECT @@IDENTITY` . Depuis MySQL 4.0.3, vous pouvez aussi modifier cette valeur avec `SET IDENTITY` .

INSERT_ID = valeur

Configure la valeur à utiliser par l'appel suivant à la commande `INSERT` ou `ALTER TABLE` lors de l'insertion d'une valeur `AUTO_INCREMENT` . Cela est souvent utilisé par le log des modifications.

LAST_INSERT_ID = valeur

Configure la valeur qui doit être retournée par `LAST_INSERT_ID()` . C'est enregistré dans le log de mises à jour quand vous utilisez `LAST_INSERT_ID()` dans une commande qui met à jour une table.

NAMES {'charset_name' | DEFAULT}

`SET NAMES` spécifie les valeurs des trois variables systèmes de session `character_set_client` , `character_set_connection` et `character_set_results` avec le jeu de caractères donné.La valeur par défaut de ces variables peut être rappelée avec `DEFAULT` .

Notez que la syntaxe de `SET NAMES` diffère en cela des autres options. Cette commande est disponible depuis MySQL 4.1.0.

SQL_AUTO_IS_NULL = {0 | 1}

Si définie à `1` (par défaut) alors on peut trouver la dernière ligne insérée dans une table avec une colonne `AUTO_INCREMENT` avec la construction suivante :

```
WHERE auto_increment_column IS NULL
```

Ceci est utilisé par des programmes ODBC tel que Access. `SQL_AUTO_IS_NULL` a été ajouté en MySQL 3.23.52.

SQL_BIG_SELECTS = {0 | 1}

Si configuré à `0` , MySQL interrompra les requêtes `SELECT` qui prendront probablement trop de temps. C'est utile lorsqu'une clause `WHERE` déconseillée a été utilisée. Une grosse requête est définie comme étant un `SELECT` qui devra probablement étudier plus de `max_join_size` lignes. La valeur par défaut d'une nouvelle connexion est `1` (qui permet toutes les requêtes `SELECT`).

SQL_BUFFER_RESULT = {0 | 1}

`SQL_BUFFER_RESULT` forcera les résultats des requêtes `SELECT` à être placés dans une table temporaire. Cela aidera MySQL à libérer les verrous sur table plus tôt et améliorera les cas où le jeu de résultats de la requête prend trop de temps à être envoyée au client.

SQL_LOG_BIN = {0 | 1}

Si cette option vaut `0` , aucun log n'est fait dans le log binaire du client, si le client a les droits de `SUPER` .

SQL_LOG_OFF = {0 | 1}

Si cette option vaut `1` , aucun log n'est fait dans le log standard du client, si le client a les droits de `SUPER` .

SQL_LOG_UPDATE = {0 | 1}

Si définie à `0` , aucune trace des requêtes ne sera gardée dans le log des mises à jour pour le client, si le client à le privilège `SUPER` . Cette variable est abandonnée depuis la version 5.0.0 et est remplacée par `SQL_LOG_BIN` (Changements de la version5.0.0 (22 Dec 2003: Alpha)).

SQL_QUOTE_SHOW_CREATE = {0 | 1}

Si vous le configurez à `1` , `SHOW CREATE TABLE` protégera les noms de tables et de colonnes. Ceci est activé par défaut, pour que la réplication des tables avec des noms à risques fonctionne. Syntaxe de `SHOW CREATE TABLE` .

SQL_SAFE_UPDATES = {0 | 1}
: Si définit à `1` , MySQL annulera si un `UPDATE` ou un `DELETE` est exécuté alors qu'il n'utilise pas de clef ou de `LIMIT` dans la clause `WHERE` . Cela permet de bloquer les requêtes erronées crées à la main.

SQL_SELECT_LIMIT = valeur | DEFAULT
: Le nombre maximal des enregistrements que doivent retourner les requêtes `SELECT` . Si un `SELECT` possède une clause `LIMIT` , celle−ci est utilisée. La valeur par défaut pour une nouvelle connexion est ``illimitée.'' Si vous avez changé la limite, la valeur par défaut peut être retrouvée en utilisant la valeur `DEFAULT` avec `SQL_SELECT_LIMIT` .

SQL_WARNINGS = {0 | 1}
: Cette variable contrôle le fait que les insertion mono−ligne `INSERT` produisent une chaîne d'information si une alerte survient. La valeur par défaut est 0. Donnez la valeur de 1 pour avoir un message d'information. Cette variable a été ajoutée en MySQL 3.22.11.

TIMESTAMP = valeur_timestamp | DEFAULT
: Configure le temps pour ce client. C'est utilisé pour obtenir le timestamp d'origine si vous utilisez le log de mises à jour pour restaurer des lignes. `valeur_timestamp` doit être un timestamp Unix, et non un timestamp MySQL.

UNIQUE_CHECKS = {0 | 1}
: Si cette option vaut `1` (par défaut), les tests d'unicité sur les index secondaires des tables `InnoDB` sont effectués. Si l'option vaut `0` , aucun test d'unicité n'est fait. Cette variable a été ajouté en MySQL 3.23.52. Contraintes `FOREIGN KEY` .

14.5.3.2 Commande SHOW CHARACTER SET

La commande `SHOW CHARACTER SET` montre tous les jeux de caractères disponibles. Il faut une clause facultative `LIKE` pour limiter les jeux de caractères à afficher.

Par exemple :

```
mysql> SHOW CHARACTER SET LIKE 'latin%';
+---------+-----------------------------+-------------------+--------+
| Charset | Description                 | Default collation | Maxlen |
+---------+-----------------------------+-------------------+--------+
| latin1  | ISO 8859-1 West European    | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          | latin7_general_ci |      1 |
+---------+-----------------------------+-------------------+--------+
4 rows in set (0.00 sec)
```

Remarques sur la liste précédente :

- La colonne `Maxlen` affiche le nombre maximum d'octets utilisés pour stocker un caractère.

14.5.3.3 Syntaxe de SHOW COLLATION

```
SHOW COLLATION [LIKE 'pattern']
```

Le résultat de `SHOW COLLATION` inclut tous les jeux de caract´res disponibles. Vous pouvez utiliser optionnellement la clause `LIKE` pour limiter le nombre de réponses.

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------+---------+
| Collation         | Charset | Id | Default | Compiled | Sortlen |
```

```
+-------------------+---------+----+---------+----------+---------+
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       0 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       0 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
| latin1_spanish_ci | latin1  | 94 |         |          |       0 |
+-------------------+---------+----+---------+----------+---------+
7 rows in set (0.00 sec)
```

La colonne `Default` indique si une collation est la collation par défaut pour son jeu de caractères. `Compiled` indique si le jeu de caractères est compilé dans le serveur ou non. `Sortlen` est en relation avec la quantité de mémoire nécessaire pour trier des chaînes exprimées dans le jeu de caractéres.

14.5.3.4 Syntaxe de SHOW COLUMNS

```
SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [LIKE 'pattern']
```

`SHOW COLUMNS` liste les colonnes de la table. Si les types de colonnes sont différents de ceux que vous avez utilisé avec la commande `CREATE TABLE` , c'est que MySQL a modifié silencieusement le type lors de la création. Les conditions de cette modification sont décrites dans Changement de type de colonne automatique .Le mot clé `FULL` peut être utilisé depuis MySQL 3.23.32. Il fait afficher les droits dont vous disposez pour chaque colonne. Depuis MySQL 4.1, `FULL` affiche aussi les commentaires par colonne.

Vous pouvez utiliser `db_name.tbl_name` comme syntaxe alternative à `tbl_name FROM db_name` . Ces deux commandes sont équivalentes :

```
mysql> SHOW COLUMNS FROM mytable FROM mydb;
mysql> SHOW COLUMNS FROM mydb.mytable;
```

`SHOW FIELDS` est un synonyme de `SHOW COLUMNS` . Vous pouvez aussi lister les colonnes d'une table avec la commande `mysqlshow db_name tbl_name` .

La commande `DESCRIBE` fournit une information similaire à `SHOW COLUMNS` . Syntaxe de `DESCRIBE` (Obtenir des informations sur les colonnes) .

14.5.3.5 Syntaxe de SHOW CREATE DATABASE

```
SHOW CREATE DATABASE db_name
```

La requête suivante montre une commande `CREATE DATABASE` qui va créer une base de donnée. Commande ajoutée en MySQL 4.1.

```
mysql> SHOW CREATE DATABASE test\G
*************************** 1. row ***************************
       Database: test
Create Database: CREATE DATABASE `test`
                 /*!40100 DEFAULT CHARACTER SET latin1 */
```

14.5.3.6 Syntaxe de SHOW CREATE TABLE

```
SHOW CREATE TABLE tbl_name
```

Affiche la commande `CREATE TABLE` nécessaire pour créer une table donnée.

```
mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE t (
  id int(11) default NULL auto_increment,
```

```
  s char(60) default NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM
```

`SHOW CREATE TABLE` va protéger le nom de la table et des colonnes selon l'option `SQL_QUOTE_SHOW_CREATE` . Syntaxe de `SET` .

14.5.3.7 Obtenir des informations sur les bases, tables, colonnes et index

```
SHOW DATABASES [LIKE 'pattern']
```

`SHOW DATABASES` liste les bases de données disponible sur le serveur MySQL. Vous pouvez aussi obtenir cette liste avec l'utilitaire `mysqlshow` . Depuis MySQL 4.0.2, vous ne verrez que les bases pour lesquelles vous avez des droits, à moins que vous n'ayez le droit de `SHOW DATABASES` .

Si le serveur a été lancé avec l'option `--skip-show-database` , vous ne pouvez pas utiliser cette commande à moins que vous n'ayez le droit de `SHOW DATABASES` .

14.5.3.8 Syntaxe SHOW ENGINES

```
SHOW [STORAGE] ENGINES
```

`SHOW ENGINES` affiche les informations sur les moteurs de stockage du serveur. C'est particulièrement utile pour connaître les moteurs supportés par votre serveur, ou le moteur par défaut. Cette commande a été ajoutée en MySQL 4.1.2. `SHOW TABLE TYPES` est un synonyme, mais est abandonnée.

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
   Type: MyISAM
Support: DEFAULT
Comment: Default type from 3.23 with great performance
*************************** 2. row ***************************
   Type: HEAP
Support: YES
Comment: Hash based, stored in memory, useful for temporary tables
*************************** 3. row ***************************
   Type: MEMORY
Support: YES
Comment: Alias for HEAP
*************************** 4. row ***************************
   Type: MERGE
Support: YES
Comment: Collection of identical MyISAM tables
*************************** 5. row ***************************
   Type: MRG_MYISAM
Support: YES
Comment: Alias for MERGE
*************************** 6. row ***************************
   Type: ISAM
Support: NO
Comment: Obsolete table type; Is replaced by MyISAM
*************************** 7. row ***************************
   Type: MRG_ISAM
Support: NO
Comment: Obsolete table type; Is replaced by MRG_MYISAM
*************************** 8. row ***************************
   Type: InnoDB
Support: YES
Comment: Supports transactions, row-level locking and foreign keys
*************************** 9. row ***************************
   Type: INNOBASE
Support: YES
Comment: Alias for INNODB
*************************** 10. row ***************************
   Type: BDB
Support: YES
Comment: Supports transactions and page-level locking
*************************** 11. row ***************************
   Type: BERKELEYDB
Support: YES
Comment: Alias for BDB
```

La valeur `Support` indique que le moteur est supporté, et si le moteur est le moteur par défaut. Par exemple, si le serveur est lancé avec l'option `--default-table-type=InnoDB` alors la valeur de la colonne `Support` de la ligne `InnoDB` contiendra `DEFAULT` .

14.5.3.9 Syntaxe de SHOW ERRORS

```
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW COUNT(*) ERRORS
```

Cette commande est similaire à `SHOW WARNINGS` , hormis le fait qu'au lieu d'afficher les erreurs, alertes et notes, elle n'affiche que les erreurs. `SHOW ERRORS` est disponible depuis MySQL 4.1.0.La clause `LIMIT` a la même syntaxe que celle de la commande `SELECT` . Syntaxe de `SELECT` .

La commande `SHOW COUNT(*) ERRORS` affiche le nombre d'erreurs. Vous pouvez aussi connaître ce nombre en lisant la variable `error_count` :

```
SHOW COUNT(*) ERRORS;
SELECT &#64;&#64;error_count;
```

Pour plus d'informations, voyez Syntaxe de `SHOW WARNINGS` .

14.5.3.10 SHOW GRANTS

```
SHOW GRANTS FOR user
```

`SHOW GRANTS FOR user` affiche la commande nécessaire pour donner les même droits qu'un utilisateur existant.

```
mysql> SHOW GRANTS FOR 'root'&#64;'localhost';
+---------------------------------------------------------------------+
| Grants for root&#64;localhost                                       |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'&#64;'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

Depuis MySQL 4.1.2, pour lister les droits de la session courante, vous pouvez connaître le nom d'utilisateur de la session avec ces commandes :

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

Avant MySQL 4.1.2, vous pouviez connaître le nom d'identification de l'utilisation avec la fonction `CURRENT_USER()` (nouveau en MySQL 4.0.6). Puis, utilisez cette valeur avec la commande `SHOW GRANTS` . Fonctions d'informations .

`SHOW GRANTS` est disponible depuis MySQL 3.23.4.

14.5.3.11 Syntaxe de SHOW INDEX

```
SHOW INDEX FROM tbl_name [FROM db_name]
```

`SHOW INDEX` retourne les informations sur les index de la table, dans un format proche de celui de `SQLStatistics` en ODBC. `SHOW INDEX` retourne les champs suivants :

Table
: Le nom de la table.

Non_unique
: 0 si l'index ne peut pas contenir de doublons, et 1 s'il le peut.

Key_name
: Le nom de l'index.

Seq_in_index
: Le numéro de la colonne dans l'index, en commençant à 1.

Column_name
: Le nom de la colonne.

Collation
: Comment la colonne est triée dans l'index. Avec MySQL, les valeurs peuvent être `'A'` (Ascendant) ou `NULL` (non trié).

Cardinality
: Le nombre de valeurs uniques dans l'index. C'est une valeur qui est mise à jour avec la commande `ANALYZE TABLE` ou `myisamchk -a` . `Cardinality` est compté en se basant sur des statistiques entières : il n'est pas toujours exacte pour les petites tables.

Sub_part
: Le nombre de caractères indexé si la colonne n'est que partiellement indexée. `NULL` si la colonne entière est indexée.

Packed
: Indique comment la clé est compactée. `NULL` si elle ne l'est pas.

Null
: Contient `YES` si la colonne contient `NULL` , `''` sinon.

Index_type
: La méthode d'indexation utilisée (`BTREE` , `FULLTEXT` , `HASH` , `RTREE`).

Comment
: Différentes remarques. Avant MySQL 4.0.2 lorsque la colonne `Index_type` a été ajoutée, `Comment` indiquait si un index était `FULLTEXT` .

Les colonnes `Packed` et `Comment` ont été ajoutée en MySQL 3.23.0. Les colonnes `Null` et `Index_type` ont été ajoutées en MySQL 4.0.2.Vous pouvez utiliser la syntaxe `db_name.tbl_name` comme alternative à `tbl_name FROM db_name` . Ces deux commandes sont équivalentes :

```
 mysql> SHOW INDEX FROM mytable FROM mydb;
mysql> SHOW INDEX FROM mydb.mytable;
```

`SHOW KEYS` est un synonyme `SHOW INDEX` . Vous pouvez aussi lister les index d'une table avec la commande en ligne `mysqlshow -k db_name tbl_name` .

14.5.3.12 Syntaxe de SHOW INNODB STATUS

```
SHOW INNODB STATUS
```

Cette commente donne des informations exhaustives sur le moteur de stockage `InnoDB` .

14.5.3.13 Syntaxe de SHOW LOGS

```
SHOW [BDB] LOGS
```

La commande `SHOW LOGS` affiche les informations d'état de vos fichiers de logs. Actuellement, elle n'affiche que les informations pour les fichiers de log des tables Berkeley DB.

- `File` affiche le chemin complet jusqu'au fichier de log.
- `Type` affiche le type de fichier de log (`BDB` pour les tables de types Berkeley DB)
- `Status` affiche le status du fichier de log (`FREE` si le fichier peut être supprimé, ou `IN USE` si le fichier est utilisé par une transaction en cours)

14.5.3.14 SHOW PRIVILEGES

```
SHOW PRIVILEGES
```

Cette commande est implémentée en MySQL 4.1.0. `SHOW PRIVILEGES` affiche la liste des droits que le serveur MySQL supporte.

```
mysql> show privileges;
+------------+--------------------------+-----------------------------------------------------+
| Privilege  | Context                  | Comment                                             |
+------------+--------------------------+-----------------------------------------------------+
| Select     | Tables                   | Lire des lignes d'une table                         |
| Insert     | Tables                   | Insérer des lignes dans une table                   |
| Update     | Tables                   | Modifier les lignes existantes                      |
| Delete     | Tables                   | Effacer des lignes existantes                       |
| Index      | Tables                   | Créer ou effacer des indexs                         |
| Alter      | Tables                   | Modifier la structure d'une table                   |
| Create     | Databases,Tables,Indexes | Créer une nouvelle base ou table                    |
| Drop       | Databases,Tables         | Effacer une base ou table                           |
| Grant      | Databases,Tables         | Donner à d'autres les droits courants               |
| References | Databases,Tables         | Avoir des références sur les tables                 |
| Reload     | Server Admin             | Rafraîchir les droits, tables et logs               |
| Shutdown   | Server Admin             | Eteindre le serveurver                              |
| Process    | Server Admin             | Voir la version texte des requêtes en cours         |
| File       | File access on server    | Lire et écrire des fichiers sur le serveur          |
+------------+--------------------------+-----------------------------------------------------+
14 rows in set (0.00 sec)
```

14.5.3.15 Syntaxe de SHOW PROCESSLIST

```
SHOW [FULL] PROCESSLIST
```

`SHOW [FULL] PROCESSLIST` affiche la liste de processus qui sont en cours d'exécution. Vous pouvez aussi obtenir ces informations avec la commande en ligne `mysqladmin processlist` . Si vous avez les droits de `SUPER` , vous pourrez aussi voir les autres threads. Sinon, vous ne pourrez voir que les votre. Syntaxe de `KILL` . Si vous n'utilisez pas l'option `FULL` , seuls les 100 premiers caractères de chaque requête seront affichés.

Cette commande est très pratique si vous obtenez trop d'erreurs '`too many connections`' et que vous voulez savoir ce qui se passe. MySQL réserve une connexion supplémentaire pour un client ayant les droits de `SUPER` , de façon à ce qu'il y ait toujours la possibilité de se connecter et de vérifier le système (en supposant que vous ne donnez pas ce droit à tous vos utilisateurs).

Certains états sont souvent disponible dans le résultat de `mysqladmin processlist`

`Checking table`
: Le thread fait une vérification (automatique) de la table.

`Closing tables`
: Le thread est en train d'écrire les données modifiées sur le disque, et il va fermer les tables. Cela doit être une opération très rapide. Si ce n'est pas le cas, vous devriez vérifier si vous n'avez pas un disque plein, ou que le disque est sous haute charge.

`Connect Out`
: Connexion d'un esclave sur le maître.

`Copying to tmp table on disk`
: Le résultat temporaire était plus grand que `tmp_table_size` et le thread passe d'une table en mémoire à une table sur disque.

`Creating tmp table`
: Le thread est en train de créer une table temporaire pour contenir le résultat d'une requête.

`deleting from main table`
: Lors de l'exécution de la première partie d'une requête d'effacement multi−table, et que MySQL n'a commencé à effacer que dans la première table.

`deleting from reference tables`
: Lors de l'exécution de la deuxième partie d'une requête d'effacement multi−table, et que MySQL a commencé à effacer dans les autres tables.

`Flushing tables`

Le thread exécute la commande `FLUSH TABLES` et il attend que tous les threads ferme leur tables.

`Killed`
: Quelqu'un a envoyé une commande `KILL` et le thread s'annuler la prochaine fois qu'il vérifie l'option de kill. Cette option est vérifiée dans chaque boucle majeure de MySQL, mais dans certains cas, il peut lui prendre un court instant avant de s'arrêter. Si le thread est verrouillé par un autre thread, l'arrêt va prendre effet aussitôt que l'autre thread lève son verrou.

`Sending data`
: Le thread traite des lignes pour une commande `SELECT` et il envoie les données au client.

`Sorting for group`
: Le thread est en train de faire un tri pour satisfaire une clause `GROUP BY` .

`Sorting for order`
: Le thread est en train de faire un tri pour satisfaire une clause `ORDER BY` .

`Opening tables`
: Cela signifie simplement que le thread essaie d'ouvrir une table. Ce doit être une opération très rapide, à moins que quelque chose ne retarde l'ouverture. Par exemple, une commande `ALTER TABLE` ou `LOCK TABLE` peut empêcher l'ouverture de table, jusqu'à l'achèvement de la commande.

`Removing duplicates`
: La requête utilisait `SELECT DISTINCT` de telle manière que MySQL ne pouvait pas optimiser les lignes distinctes au début du traitement. A cause de cela, MySQL doit effectuer une opération de plus pour supprimer toutes les lignes en doubles, avant d'envoyer les lignes au client.

`Reopen table`
: Le thread a reçu un verrou pour une table, mais a noté après l'avoir reçu que la structure de la table a changé. Il a libéré le verrou, fermé la table, et maintenant il essaie de la rouvrir.

`Repair by sorting`
: Le thread répare la table en utilisant la méthode de tri pour créer l'index.

`Repair with keycache`
: Le thread répare la table en utilisant la méthode de création des clés à partir du cache de clé. C'est bien plus lent que `la réparation par tri` .

`Searching rows for update`
: Le thread effectue une première phase pour trouver toutes les lignes qui satisfont les critères avant de les modifier. Cela doit être fait si `UPDATE` modifie l'index qui sera utilisé pour trouver les lignes.

`Sleeping`
: Le thread attend que le client envoie une nouvelle commande.

`System lock`
: Le thread attend le verrou externe pour la table. Si vous n'utilisez pas de serveurs MySQL multiples qui exploitent les mêmes tables, vous pouvez désactiver les verrous systèmes avec l'option `--skip-external-locking` .

`Upgrading lock`
: Le gestionnaire de `INSERT DELAYED` essaie d'obtenir un verrou pour insérer des lignes.

`Updating`
: Le thread recherche des lignes pour les modifier.

`User Lock`
: Le thread attend un `GET_LOCK()` .

`Waiting for tables`
: Le thread a reçu l'annonce que la structure de table a été modifiée, et il doit réouvrir la table pour obtenir une nouvelle structure. Pour être capable de réouvrir la table, il doit attendre que les autres threads aient fermé la table en question.

  Cette annonce survient lorsqu'un autre autre thread a été utilisé avec la commande `FLUSH TABLES` ou une des commandes suivantes, appliquées à la table en question : `FLUSH TABLES table_name` , `ALTER TABLE` , `RENAME TABLE` , `REPAIR TABLE` , `ANALYZE TABLE` ou `OPTIMIZE TABLE` .

`waiting for handler insert`
: Le gestionnaire de `INSERT DELAYED` a traité toutes insertions, et en attend de nouvelles.

La plupart des états sont des opérations très rapides. Si le thread s'attarde dans un de ces états pour plusieurs secondes, il doit y avoir un problème qui mérite d'être étudié.Il existe encore d'autres états qui ne sont pas mentionné ci−dessus, mais la majorité sont utilisés pour trouver des bogues dans `mysqld` .

14.5.3.16 Syntaxe de SHOW STATUS

```
SHOW STATUS [LIKE 'pattern']
```

`SHOW STATUS` affiche des informations sur le statut du serveur. Cette information est aussi accessible avec la commande la commande `mysqladmin extended-status` .Un résultat partiel est présenté ci−dessous. La liste complète des variables dépend de votre serveur. Leur signification individuelle est présentée dans la section Variables de statut du serveur .

```
mysql> SHOW STATUS;
+--------------------------+------------+
| Variable_name            | Value      |
+--------------------------+------------+
| Aborted_clients          | 0          |
| Aborted_connects         | 0          |
| Bytes_received           | 155372598  |
| Bytes_sent               | 1176560426 |
| Connections              | 30023      |
| Created_tmp_disk_tables  | 0          |
| Created_tmp_tables       | 8340       |
| Created_tmp_files        | 60         |
...
| Open_tables              | 1          |
| Open_files               | 2          |
| Open_streams             | 0          |
| Opened_tables            | 44600      |
| Questions                | 2026873    |
...
| Table_locks_immediate    | 1920382    |
| Table_locks_waited       | 0          |
| Threads_cached           | 0          |
| Threads_created          | 30022      |
| Threads_connected        | 1          |
| Threads_running          | 1          |
| Uptime                   | 80380      |
+--------------------------+------------+
```

Avec la clause `LIKE` , la commande peut limiter l'affichage des variables à celles qui vérifient un masque :

```
mysql> SHOW STATUS LIKE 'Key%';
+--------------------+----------+
| Variable_name      | Value    |
+--------------------+----------+
| Key_blocks_used    | 14955    |
| Key_read_requests  | 96854827 |
| Key_reads          | 162040   |
| Key_write_requests | 7589728  |
| Key_writes         | 3813196  |
+--------------------+----------+
```

14.5.3.17 SHOW TABLE STATUS

```
SHOW TABLE STATUS [FROM db_name] [LIKE wild]
```

`SHOW TABLE STATUS` (nouveau en version 3.23) fonctionne comme `SHOW STATUS` , mais fournit des informations sur les tables. Vous pouvez aussi obtenir ces informations en utilisant la commande en ligne `mysqlshow --status db_name` . Les données suivantes sont retournées :

`Name`
: Nom de la table.

`Type`
: Type de table. Types de tables MySQL .

`Row_format`
: Le format de stockage de ligne (`Fixed` , `Dynamic` ou `Compressed`).

`Rows`
: Nombre de lignes.

`Avg_row_length`
: Taille moyenne d'une ligne.

`Data_length`
: Taille du fichier de données.

`Max_data_length`
: Taille maximale du fichier de données. Pour les formats de lignes fixe, c'est le nombre maximal de lignes dans la table. Pour les formats de lignes dynamique, c'est le nombre total d'octets qui peuvent être stockés dans la table, avec le pointeur de données utilisé.

`Index_length`
: Taille du fichier d'index.

`Data_free`
: Nombre d'octets alloués mais non utilisés.

`Auto_increment`
: Prochaine valeur d'auto_increment.

`Create_time`
: Date de création de la table.

`Update_time`
: Date de dernière modification de la table.

`Check_time`
: Date de dernier entretien de la table.

`Collation`
: Le jeu de caractères et la collation de la table (nouveau en 4.1.1)

`Checksum`
: La somme de contrôle en direct (si elle existe). (nouveau en 4.1.1)

`Create_options`
: Options supplémentaires utilisées avec `CREATE TABLE` .

`Comment`
: Le commentaire utilisé lors de la création de la table (ou des informations sur pourquoi MySQL n'a pu accéder aux informations de la table).

Les tables `InnoDB` indiqueront l'espace disque libre dans le commentaire de table.

14.5.3.18 Syntaxe de SHOW TABLES

```
SHOW [OPEN] TABLES [FROM db_name] [LIKE 'pattern']
```

`SHOW TABLES` liste les tables permanentes (non `TEMPORARY`) dans une base de données. Vous pouvez obtenir cette liste avec la commande en ligne `mysqlshow db_name` . **Note :** Si vous n'avez pas les droits sur une table, la table n'apparaîtra pas dans le résultat de `SHOW TABLES` et `mysqlshow db_name` .

`SHOW OPEN TABLES` liste les tables qui sont actuellement ouvertes dans le cache de table. Comment MySQL ouvre et ferme les tables . Le champ `Comment` du résultat indique le nombre d'ouverture de la table en cache (`cached`) et le nombre d'utilisation (`in_use`). `OPEN` est disponible depuis MySQL 3.23.33.

14.5.3.19 Syntaxe de SHOW VARIABLES

```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern']
```

`SHOW VARIABLES` affiche les valeurs des variables systèmes de MySQL. Vous pouvez aussi obtenir ces informations avec la commande `mysqladmin variables` . Les options `GLOBAL` et `SESSION` ont été ajoutées en MySQL 4.0.3. Avec `GLOBAL` , vous obtiendrez les valeurs qui seront utilisées pour les nouvelles connexions au serveur MySQL. Avec `SESSION` , vous recevez les valeurs effectives pour la connexion en cours. Si vous ne précisez ni l'une ni l'autre, la valeur par défaut est `SESSION` . `LOCAL` est un synonyme de `SESSION` .

Si les valeurs par défaut ne vous conviennent pas, vous pouvez modifier la plupart de ces variables, en ligne de commande, lorsque `mysqld` est lancé. Voir Options de ligne de commande `mysqld` et Syntaxe de `SET` .

Voici un extrait du résultat de la commande. La liste complète des variables et de leur valeur peut être différente pour votre serveur. La signification de chaque variable est présentée dans Variables sytème du serveur . Des informations sur comment optimiser ces valeurs sont disponibles dans la section Choix des paramètres du serveur .

```
mysql> SHOW VARIABLES;
+---------------------------------+------------------------------+
| Variable_name                   | Value                        |
+---------------------------------+------------------------------|
| back_log                        | 50                           |
| basedir                         | /usr/local/mysql             |
| bdb_cache_size                  | 8388572                      |
| bdb_log_buffer_size             | 32768                        |
| bdb_home                        | /usr/local/mysql             |
...
| max_connections                 | 100                          |
| max_connect_errors              | 10                           |
| max_delayed_threads             | 20                           |
| max_error_count                 | 64                           |
| max_heap_table_size             | 16777216                     |
| max_join_size                   | 4294967295                   |
| max_relay_log_size              | 0                            |
| max_sort_length                 | 1024                         |
...
| timezone                        | EEST                         |
| tmp_table_size                  | 33554432                     |
| tmpdir                          | /tmp/:/mnt/hd2/tmp/          |
| version                         | 4.0.4-beta                   |
| wait_timeout                    | 28800                        |
+---------------------------------+------------------------------+
```

Avec la clause `LIKE` , la commande n'affichera que les variables qui vérifie le masque fourni :

```
mysql> SHOW VARIABLES LIKE 'have%';
+--------------------+----------+
| Variable_name      | Value    |
+--------------------+----------+
| have_bdb           | YES      |
| have_innodb        | YES      |
| have_isam          | YES      |
| have_raid          | NO       |
| have_symlink       | DISABLED |
| have_openssl       | YES      |
| have_query_cache   | YES      |
+--------------------+----------+
```

14.5.3.20 SHOW WARNINGS | ERRORS

```
SHOW WARNINGS [LIMIT [offset,] row_count]
SHOW COUNT(*) WARNINGS
```

Cette commande affiche les erreurs, alertes et notes qui ont été générées par la dernière commande. Les erreurs et alertes sont remises à zéro pour chaque nouvelle commande qui utilisent une table. Cette commande a été implémentée depuis MySQL 4.1.0. Une commande connexe, `SHOW ERRORS` , affiche uniquement les erreurs. Syntaxe de `SHOW ERRORS` .

La liste de messages est remise à zéro au début de chaque commande qui utilise la table.

La commande `SHOW COUNT(*) WARNINGS` affiche le nombre total d'erreurs, d'alertes et de notes. Vous pouvez aussi lire ce nombre avec la variable `warning_count` :

```
SHOW COUNT(*) WARNINGS;
SELECT &#64;&#64;warning_count;
```

La valeur de `warning_count` peut être plus grande que le nombre de messages affichés par `SHOW WARNINGS` si la variable système `max_error_count` est configurée assez bas pour que tous les messages ne soient pas stockés. Un exemple plus loin dans cette section montre ce qui arrive.La clause `LIMIT` a la même syntaxe que la commande `SELECT` . Syntaxe de `SELECT` .

Le serveur MySQL retourne le nombre total d'alertes et d'erreurs que vous avez obtenu lors de la dernière commande. Ils sont disponibles avec la fonction `mysql_warning_count()` . `mysql_warning_count()` .

Jusqu'à `max_error_count` messages peuvent être stockés (variable globale et spécifique aux threads).

Vous pouvez lire le nombre d'erreurs dans `@error_count` et le nombre d'alertes dans `@warning_count` .

`SHOW WARNINGS` affiche aussi toutes les erreurs, alertes et notes de la dernière commande, alors que `SHOW ERRORS` ne montre que les erreurs.

```
mysql> DROP TABLE IF EXISTS no_such_table;
mysql> SHOW WARNINGS;

+-------+------+-------------------------------+
| Level | Code | Message                       |
+-------+------+-------------------------------+
| Note  | 1051 | Unknown table 'no_such_table' |
+-------+------+-------------------------------+
```

Notez que depuis MySQL 4.1.0, nous avons ajouté un nouveau système d'alertes, et peu de commandes MySQL génère des alertes. 4.1.1 supporte toutes sortes d'alertes pour `LOAD DATA INFILE` et les commandes DML telles que `INSERT` , `UPDATE` et `ALTER` .

Par exemple, voici une situation simple qui produit des alertes de conversions pour une commande d'insertion :

```
mysql> create table t1(a tinyint NOT NULL, b char(4));
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t1 values(10,'mysql'),(NULL,'test'),(300,'open source');
Query OK, 3 rows affected, 4 warnings (0.15 sec)
Records: 3  Duplicates: 0  Warnings: 4

mysql> show warnings;
+---------+------+---------------------------------------------------------------+
| Level   | Code | Message                                                       |
+---------+------+---------------------------------------------------------------+
| Warning | 1263 | Data truncated for column 'b' at row 1                        |
| Warning | 1261 | Data truncated, NULL supplied to NOT NULL column 'a' at row 2 |
| Warning | 1262 | Data truncated, out of range for column 'a' at row 3          |
| Warning | 1263 | Data truncated for column 'b' at row 3                        |
+---------+------+---------------------------------------------------------------+
4 rows in set (0.00 sec)
```

Le nombre maximal d'alertes peut être spécifié en utilisant la variable de serveur `'max_error_count'` , `SET max_error_count=[count]` ; Par défaut, c'est 64. Pour désactiver les alertes, donnez simplement la valeur de 0 à la variable. Si `max_error_count` vaut 0, alors le nombre d'alertes représente toujours le nombre d'alertes qui ont eu lieu, mais aucun message d'erreur n'est accessible.

Par exemple, observez la commande `ALTER` suivante, pour l'exemple ci−dessus, qui retourne uniquement une alerte, même si le nombre total d'alertes est de 3 lorsque `'max_error_count'` =1.

```
mysql> show variables like 'max_error_count';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_error_count | 64    |
+-----------------+-------+
1 row in set (0.00 sec)

mysql> set max_error_count=1;
Query OK, 0 rows affected (0.00 sec)

mysql> alter table t1 modify b char;
Query OK, 3 rows affected, 3 warnings (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 3

mysql> show warnings;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1263 | Data truncated for column 'b' at row 1 |
+---------+------+----------------------------------------+
1 row in set (0.00 sec)

mysql>
```

14.5.4 Autres commandes d'administration

14.5.4.1 Syntaxe de CACHE INDEX

```
CACHE INDEX
 table_index_list [, table_index_list] ...
 IN key_cache_name

table_index_list:
 tbl_name [[INDEX] (index_name[, index_name] ...)]
```

La commande `CACHE INDEX` assigne un index de table à un cache de clé spécifique. Cette commande est uniquement disponible pour les tables `MyISAM` .

La commande suivante assigne les index des tables `t1` , `t2` et `t3` au cache de clé appelé `hot_cache` :

```
 mysql> CACHE INDEX t1, t2, t3 IN hot_cache;
+---------+--------------------+----------+----------+
| Table   | Op                 | Msg_type | Msg_text |
+---------+--------------------+----------+----------+
| test.t1 | assign_to_keycache | status   | OK       |
| test.t2 | assign_to_keycache | status   | OK       |
| test.t3 | assign_to_keycache | status   | OK       |
+---------+--------------------+----------+----------+
```

La syntaxe de `CACHE INDEX` vous permet de spécifier des index particuliers à un cache. Cependant, l'implémentation courante assigne tous les index de la table au cache, et il n'y a donc pas d'intérêt à spécifier autre chose que le nom de la table.

Le cache de clé utilisé dans une commande `CACHE INDEX` peut être créé en lui donnant une taille avec une commande de configuration, ou la configuration du serveur. Par exemple :

```
mysql> SET GLOBAL keycache1.key_buffer_size=128*1024;
```

Les paramètres du cache de clé sont accessibles depuis une variable système structurée. Variables système structurées .

Un cache de clé doit exister avant que vous ne l'utilisiez :

```
 mysql> CACHE INDEX t1 in non_existent_cache;
ERROR 1283 (HY000): Unknown key cache 'non_existent_cache'
```

Par défaut, les index de table sont assignée au cache de clé par défaut, créé au moment du démarrage du serveur. Lorsqu'un cache de clé est détruit, tous les index qui lui étaient assigné sont transmis au cache par défaut.

Les assignations d'index affectent le serveur globalement : si un client assigne un index à un cache donné, ce cache sera utilisé pour tous les requêtes, quelque soit le client qui émet la requête.

`CACHE INDEX` a été ajouté en MySQL 4.1.1.

14.5.4.2 Syntaxe de FLUSH

```
FLUSH flush_option [,flush_option] ...
```

Vous devez utiliser la commande `FLUSH` si vous voulez effacer certains caches internes de MySQL. Pour exécuter `FLUSH` , vous devez avoir le droit `RELOAD` . `flush_option` peut être l'une des suivantes :

`HOSTS`

Vide le cache des hôtes. Vous devez vider ce cache si certaines des adresses IP de vos clients changent, ou si vous obtenez des erreurs du type `Host ... is blocked` . Lorsque plus de `max_connect_errors` erreurs successives surviennent pour un hôte, lors des connexions au serveur MySQL, MySQL suppose qu'il y a un problème, et interdit l'accès à l'hôte. Erreur `Host '...' is blocked` . Vous pouvez démarrer `mysqld` avec `-O max_connect_errors=999999999` pour éviter ce message.

`DES_KEY_FILE`
: Recharge les clés DES depuis le fichier de stockage spécifié par `--des-key-file` lors du démarrage du serveur.

`LOGS`
: Ferme et réouvre tous les fichiers de log. Si vous avez spécifié un fichier de log de mise à jour, ou un fichier de log binaire sans extension, le numéro d'extension du fichier de log sera incrémenté d'une unité. Si vous avez utilisé une extension dans le nom du fichier, MySQL va fermer et réouvrir le même fichier. Le log des mises à jour . Ceci est la même chose que d'envoyer le signal `SIGHUP` au serveur `mysqld` .

`PRIVILEGES`
: Recharge les privilèges des tables de droits dans la base `mysql` .

`QUERY CACHE`
: Défragmente le cache des requêtes pour mieux en utiliser la mémoire. Cette commande n'effacera aucune requête du cache, à la différence de `RESET QUERY CACHE` .

`TABLES`
: Ferme toutes les tables ouvertes, et force les tables utilisées à se refermer. Cela vide aussi le cache de requêtes.

`[TABLE | TABLES] nom_de_table [,nom_de_table...]`
: Vide du cache uniquement les tables nommées.

`TABLES WITH READ LOCK`
: Ferme toutes les tables ouvertes, et verrouille en lecture toute les tables et bases, jusqu'à ce que vous exécutiez une commande `UNLOCK TABLES` . C'est très pratique pour générer des sauvegardes, si vous avez un système de fichiers comme Veritas, qui peut prendre des photos du système.

`STATUS`
: Remet la plupart des variables de statut à zéro. A n'utiliser que pour corriger une requête. Comment rapporter des bogues ou problèmes .

`USER_RESOURCES`
: Remet toutes les ressources à zéro. Cela va autoriser de nouveau les utilisateurs qui ont été bloqués. Limiter la consommation de ressources .

Vous pouvez aussi accéder à toutes les commandes décrites plus haut en les donnant en arguments à `mysqladmin` (exemple : `flush-hosts` , `flush-logs` , `reload` , ou encore `flush-tables`).Reportez−vous aussi à la commande `RESET` avec la réplication. Syntaxe de `RESET` .

14.5.4.3 Syntaxe de KILL

```
KILL [CONNECTION | QUERY] thread_id
```

Chaque connexion à `mysqld` utilise un thread unique. Vous pouvez voir les threads en cours d'exécution en utilisant la commande `SHOW PROCESSLIST` et en terminer un avec la commande `KILL thread_id` .

Depuis MySQL 5.0.0, `KILL` autorise les options `CONNECTION` et `QUERY` :

- `KILL CONNECTION` est similaire à `KILL` sans option : elle termine la connexion associée avec le thread `thread_id` .
- `KILL QUERY` termine la requête que la connexion est actuellement en train de traiter, mais laisse la connexion ouverte.

Si vous avez le droit `PROCESS` , vous pouvez voir tous les threads. Si vous avez le droit `SUPER` , vous pouvez terminer tout les threads. Sinon, vous ne pouvez terminer que vos propres threads.

Vous pouvez aussi utiliser les commandes `mysqladmin processlist` et `mysqladmin kill` pour examiner et terminer les threads.

Note : vous ne pouvez actuellement pas utiliser `KILL` avec la bibliothèque du serveur embarqué, car celui−ci utilise les threads de l'application hôte, il ne crée pas ses propres threads.

Quand vous exécutez un `KILL` , un thread spécifique est crée pour ce thread.

Dans la plupart des cas, la terminaison du thread pourra prendre un certain temps vu que le thread de terminaison est invoqué à intervalles spécifiques.

- Pour les boucles de `SELECT` , `ORDER BY` et `GROUP BY` , le thread de terminaison est vérifié après avoir lu un enregistrement. S'il est activé, la requête est abandonnée.
- Lors d'un `ALTER TABLE` le thread de terminaison est vérifié avant la lecture de chacune des colonnes de la table d'origine. S'il est activé, la commande est abandonnée et la table temporaire effacée.
- Lors d'un `UPDATE` ou d'un `DELETE` , le thread de terminaison est vérifié après chaque lecture de bloc et chaque mise à jour ou suppression de ligne. S'il est activé, la requête est abandonnée. Notez que si vous utilisez les transactions, les modifications ne seront pas perdues !
- `GET_LOCK()` stoppera avec `NULL` .
- Un thread `INSERT DELAYED` videra rapidement toutes les lignes en mémoire et se terminera.
- Si le thread est dans le gestionnaire des verrous de tables (état : `Locked`), le verrou sur la table sera vite enlevé.
- Si le thread est en attente de libération d'espace disque lors d'un appel à `write` , l'opération est avortée avec un message d'erreur indiquant que le disque est plein.

14.5.4.4 Syntaxe de LOAD INDEX INTO CACHE

```
LOAD INDEX INTO CACHE
 table_index_list [, table_index_list] ...

table_index_list:
 tbl_name
   [[INDEX] (index_name[, index_name] ...)]
   [IGNORE LEAVES]
```

La commande `LOAD INDEX INTO CACHE` précharge un index dans un cache de clé, qui est explicitement nommé dans dans la commande `CACHE INDEX` , ou dans le cache par défaut. `LOAD INDEX INTO CACHE` ne sert que pour les tables `MyISAM` .

L'option `IGNORE LEAVES` fait que les blocs terminaux de l'index ne sont pas lus.

La commande suivante précharge les noeuds des tables `t1` et `t2` :

```
mysql> LOAD INDEX INTO CACHE t1, t2 IGNORE LEAVES;
+---------+--------------+----------+----------+
| Table   | Op           | Msg_type | Msg_text |
+---------+--------------+----------+----------+
| test.t1 | preload_keys | status   | OK       |
| test.t2 | preload_keys | status   | OK       |
+---------+--------------+----------+----------+
```

Cette commande charge tous les index de `t1` . Elle ne charge que les index non−terminaux de `t2` .La syntaxe de `LOAD INDEX INTO CACHE` vous permet de spécifier seulement des index particuliers à charger dans la table. Cependant, l'implémentation courante charge tous les index : il n' y a pas de raison pour utiliser autre chose que le nom de la table.

`LOAD INDEX INTO CACHE` a été ajouté en MySQL 4.1.1.

14.5.4.5 Syntaxe de la commande RESET

```
RESET reset_option [, reset_option] ...
```

La commande `RESET` sert à remettre à zéro des données. C'est aussi une version plus puissante de la commande `FLUSH` . Syntaxe de `FLUSH` .

Pour exécuter la commande `RESET` , vous devez avoir les droits `RELOAD` .

`MASTER`
: Efface tous les logs binaires listés dans le fichier d'index, et l'index binlog est vidé. Dans les version antérieures à la version 3.23.26, cette commande s'appelait `FLUSH MASTER` (Master) Commandes SQL pour contrôler les maîtres .

`QUERY CACHE`
: Supprime tous les résultats de requêtes du cache de requête.

`SLAVE`
: Annule la position de réplication de l'esclave dans les historiques du maître. Dans les version antérieures à la version 3.23.26, cette commande s'appelait `FLUSH SLAVE` (Slave) Commandes SQL pour contrôler les esclaves .

14.6 Commandes de réplication

Cette section décrit les commandes liées à la réplication. Un groupe de commande peut être utilisé pour contrôler le serveur. L'autre groupe sert avec les esclaves.

14.6.1 Requêtes SQL pour contrôler les maîtres de réplication

La réplication est contrôlable via l'interface SQL. Cette section présente les commandes qui contrôlent les maîtres de réplication. La section Commandes SQL pour contrôler les esclaves présente les commandes pour gérer les esclaves.

14.6.1.1 PURGE MASTER LOGS

```
 PURGE &#123;MASTER | BINARY&#125; LOGS TO 'log_name'
PURGE &#123;MASTER | BINARY&#125; LOGS BEFORE 'date'
```

Efface tous les logs binaires listés dans l'index de logs, qui sont antérieurs à la date ou au log indiqué. Les logs sont aussi supprimés de cette liste : le log donné en paramètre devient alors le premier de la liste.Exemple :

```
 PURGE MASTER LOGS TO 'mysql-bin.010';
PURGE MASTER LOGS BEFORE '2003-04-02 22:46:26';
```

La variante `BEFORE` est disponible en MySQL 4.1; sont argument de date peut être au format `'YYYY-MM-DD hh:mm:ss'` . `MASTER` et `BINARY` sont synonymes, mais `BINARY` ne peut être utilisé que depuis MySQL 4.1.1.Si vous avez un esclave actif qui est actuellement en train de lire l'un des logs que vous voulez effacer, la commande ne fera rien, et échouera avec une erreur. Cependant, si l'esclave est inactif, et que vous effacez un des logs qu'il utilisait, l'esclave sera incapable de reprendre la réplication. Cette commande peut être utilisée sans problème durant la réplication : vous n'avez pas besoin d'arrêter les esclaves.

Pour purger les logs, suivez cette procédure :

- Sur chaque esclave, utilisez la commande `SHOW SLAVE STATUS` pour vérifier quel log est lu.

- Faite une liste des logs sur le maître avec `SHOW MASTER LOGS` .
- Déterminez le plus ancien log parmi ceux utilisés par les esclaves. C'est votre limite. Si vous les esclaves sont à jour, alors ce sera le dernier log de la liste.
- Faites une sauvegarde de tous les logs que vous allez effacer. Cette étape est optionnelle, mais c'est une bonne idée.
- Purgez tous les logs jusqu'à celui qui précède votre limite.

14.6.1.2 RESET MASTER

```
RESET MASTER
```

Efface tous les fichiers de logs binaires dans le fichier d'index, et vide le fichier d'index des logs.Cette commande s'appelait `FLUSH MASTER` avant MySQL 3.23.26.

14.6.1.3 SET SQL_LOG_BIN

```
SET SQL_LOG_BIN = &#123;0|1&#125;
```

Inactive ou active le log binaire de la connexion courante (`SQL_LOG_BIN` est une variable de session), si le client se connecte avec un compte qui a les droits de `SUPER` . La commande est ignorée si le client n'a pas de droits.

14.6.1.4 SHOW BINLOG EVENTS

```
SHOW BINLOG EVENTS [ IN 'log_name' ] [ FROM pos ] [ LIMIT [offset,] row_count ]
```

Affiche les événements du log binaire. Si vous ne spécifiez pas `'log_name'` , le premier log binaire sera affiché.

La clause `LIMIT` a la même syntaxe que celle de la commande `SELECT` . Syntaxe de `SELECT` .

Cette commande est disponible en MySQL 4.0

14.6.1.5 SHOW MASTER LOGS

```
SHOW MASTER LOGS
```

Liste les logs binaires disponibles sur le maître. Vous devriez utiliser cette commande avant `PURGE MASTER LOGS` pour savoir jusqu'où vous pouvez aller.

14.6.1.6 SHOW MASTER STATUS

```
SHOW MASTER STATUS
```

Affiche les informations d'état du log binaire du maître.

14.6.1.7 SHOW SLAVE HOSTS

```
SHOW SLAVE HOSTS
```

Affiche la liste des esclaves actuellement enregistrée sur le maître. Notez que les esclaves qui ne sont pas lancé avec l'option `--report-host=nom_d_esclave` ne seront pas visible dans cette liste.

14.6.2 Commandes SQL de contrôle des esclaves de réplication

La réplication est contrôlable via l'interface SQL. Cette section présente les commandes qui contrôlent les esclaves de réplication. La section Commandes SQL pour contrôler les maîtres présente les commandes pour gérer les maîtres.

14.6.2.1 CHANGE MASTER TO

```
CHANGE MASTER TO master_def [, master_def] ...

master_def =
      MASTER_HOST = 'host_name'
    | MASTER_USER = 'user_name'
    | MASTER_PASSWORD = 'password'
    | MASTER_PORT = port_num
    | MASTER_CONNECT_RETRY = count
    | MASTER_LOG_FILE = 'master_log_name'
    | MASTER_LOG_POS = master_log_pos
    | RELAY_LOG_FILE = 'relay_log_name'
    | RELAY_LOG_POS = relay_log_pos
    | MASTER_SSL = &#123;0|1&#125;
    | MASTER_SSL_CA = 'ca_file_name'
    | MASTER_SSL_CAPATH = 'ca_directory_name'
    | MASTER_SSL_CERT = 'cert_file_name'
    | MASTER_SSL_KEY = 'key_file_name'
    | MASTER_SSL_CIPHER = 'cipher_list'
```

Modifie les paramètres que l'esclave utilise pour se connecter et pour communiquer avec le serveur maître. Les valeurs possibles pour `master_def` sont présentées ci−dessus.Les options de log de relais, `RELAY_LOG_FILE` et `RELAY_LOG_POS` , sont disponibles depuis MySQL 4.0.

Les options SSL, `MASTER_SSL` , `MASTER_SSL_CA` , `MASTER_SSL_CAPATH` , `MASTER_SSL_CERT` , `MASTER_SSL_KEY` et `MASTER_SSL_CIPHER` , sont disponibles depuis MySQL 4.1.1. Vous pouvez changer ces options même sur les esclaves qui sont compilé sans le support SSL. Elles seront sauvées dans le fichier `master.info` mais ignorées jusqu'à ce que le support SSL soit activé.

Par exemple :

```
 mysql> CHANGE MASTER TO
    ->     MASTER_HOST='master2.mycompany.com',
    ->     MASTER_USER='replication',
    ->     MASTER_PASSWORD='bigs3cret',
    ->     MASTER_PORT=3306,
    ->     MASTER_LOG_FILE='master2-bin.001',
    ->     MASTER_LOG_POS=4,
    ->     MASTER_CONNECT_RETRY=10;
mysql> CHANGE MASTER TO
    ->     RELAY_LOG_FILE='slave-relay-bin.006',
    ->     RELAY_LOG_POS=4025;
```

`MASTER_USER` , `MASTER_PASSWORD` , `MASTER_SSL` , `MASTER_SSL_CA` , `MASTER_SSL_CAPATH` , `MASTER_SSL_CERT` , `MASTER_SSL_KEY` , et `MASTER_SSL_CIPHER` sont des informations qui permettent à l'esclave de se connecter au maître. Si vous omettez certains paramètres, les paramètres omis conserveront leur ancienne valeur. Par exemple, si le mot de passe sur le maître a changé, il suffit de faire :

```
 mysql> STOP SLAVE; -- if replication was running
mysql> CHANGE MASTER TO MASTER_PASSWORD='new3cret';
mysql> START SLAVE; -- if you want to restart replication
```

pour indiquer à l'esclave le nouveau mot de passe : il n'y a pas besoin de spécifier les informations qui n'ont pas changé, comme l'hôte, le port, l'utilisateur, etc... `MASTER_HOST` , `MASTER_PORT` sont le nom d'hôte ou l'adresse IP du maître, et son port TCP. Notez que si `MASTER_HOST` est égal à `localhost` , alors, comme généralement avec MySQL, le port sera ignoré si les sockets Unix sont utilisables.

Si vous spécifiez `MASTER_HOST` ou `MASTER_PORT` , l'esclave supposera que le serveur maître est différent du précédent, même si vous spécifier les mêmes valeurs d'hôte et de port que précédemment. Dans ce cas, les anciennes valeurs et position de l'historique binaire ne sont plus valides. Ainsi, si vous ne spécifiez pas `MASTER_LOG_FILE` et `MASTER_LOG_POS` dans la commande, `MASTER_LOG_FILE=''` et `MASTER_LOG_POS=4` sont ajoutés silencieusement.

`MASTER_LOG_FILE` et `MASTER_LOG_POS` sont les coordonnées auxquelles le thread d'I/O doit commencer à lire chez le maître, lorsque le thread redémarrera. Si vous spécifiez l'un d'entre eux, vous ne pourrez pas spécifier `RELAY_LOG_FILE` ou `RELAY_LOG_POS` . Si `MASTER_LOG_FILE` , ni `MASTER_LOG_POS` n'ont été spécifiés, alors les dernières coordonnées du **thread esclave** d'avant la commande `CHANGE MASTER` seront utilisées. Cela assure que la réplication ne connaît pas de discontinuité, même si le thread esclave était en retard sur le thread d'I/O, alors que vous ne voulez changer que le mot de passe. Ce comportement sécuritaire a été introduit à partir de MySQL versions 4.0.17 et 4.1.1. Avant ces versions, les coordonnées utilisées celles du thread d'I/O, avant que la commande `CHANGE MASTER` soit émise, ce qui conduisait à des pertes d'événements au niveau du maître, et donc, la corruption de la réplication.

`CHANGE MASTER` **efface tous les logs de relais** (et en démarre de nouveaux), à moins que vous ne spécifiez l'option `RELAY_LOG_FILE` ou `RELAY_LOG_POS` (dans ce cas, les logs de relais seront conservés; depuis MySQL 4.1.1 la variable globale `RELAY_LOG_PURGE` sera automatiquement mise à 0). `CHANGE MASTER TO` modifie `master.info` et `relay-log.info` .

`CHANGE MASTER` sert à configurer un esclave lorsque vous avez une sauvegarde du maître, son log et la position qui correspond à la sauvegarde du maître. Vous pouvez utiliser la commande `CHANGE MASTER TO MASTER_LOG_FILE='log_name_on_master', MASTER_LOG_POS=log_offset_on_master` sur l'esclave après la restauration de la sauvegarde. Le premier exemple ci−dessus (`CHANGE MASTER TO MASTER_HOST='master2.mycompany.com' etc`) modifie les coordonnées du maître et de son log binaire. Cela est utile lorsque vous voulez que l'esclave réplique le maître. Le second exemple, moins fréquent, sert lorsque l'esclave a des logs de relais que vous voulez utiliser à nouveau. Pour cela, le maître n'a pas besoin d'être rejoint : il suffit d'utiliser la commande `CHANGE MASTER TO` et de lancer le thread SQL `START SLAVE SQL_THREAD` . Vous pouvez même utiliser cela dans une configuration de réplication, sur un serveur indépendant, pour assurer la restauration après crash. Supposez que votre serveur soit planté, et que vous avez restauré la sauvegarde. Vous voulez que le serveur exécute à nouveau ses propres logs (non pas des logs de relais, mais ses logs binaires), qui sont par exemple, stockés sous le nom `myhost-bin.*` . Tout d'abord, faite une copie des fichiers de log dans un entrepôt, au cas où une erreur de manipulation surviendrait, et que le serveur vide ses logs. Si vous utilisez MySQL 4.1.1 ou plus récent, utilisez la commande suivante pour plus de sécurité : `SET GLOBAL RELAY_LOG_PURGE=0` .

Puis, lancez le serveur sans `log-bin` , et avec un nouvel identifiant (différent du précédent), avec l'option `relay-log=myhost-bin` (pour faire croire au serveur que ses propres logs sont des logs de relais), et `skip-slave-start` . Puis, envoyez cette commande :

```
 mysql> CHANGE MASTER TO
    ->     RELAY_LOG_FILE='myhost-bin.153',
    ->     RELAY_LOG_POS=410,
    ->     MASTER_HOST='some_dummy_string';
mysql> START SLAVE SQL_THREAD;
```

Le serveur va alors lire et exécuter ses propres logs, et rattraper les données jusqu'au crash.

Une fois la restauration finie, faites `STOP SLAVE` , éteignez le serveur, supprimez `master.info` et `relay-log.info` , puis relancez le serveur avec ses options originales.

Pour le moment, spécifier `MASTER_HOST` (même avec une valeur insignifiante) est obligatoire pour que le serveur pense qu'il est un esclave. Donner au serveur un nouvel identifiant, différent du précédent, est aussi obligatoire, car sinon, le serveur va voir des événements avec son identifiant, et il va conclure que c'est une réplication circulaire, et il va les ignorer. Dans le futur, nous envisageons de nous débarasser de ces petites contraintes.

14.6.2.2 LOAD DATA FROM MASTER

```
LOAD DATA FROM MASTER
```

Fait une sauvegarde du maître et la copie vers l'esclave. Met à jour les valeurs de `MASTER_LOG_FILE` et `MASTER_LOG_POS` pour que la réplication reprennent à la bonne position. Respecte les interdictions de réplications de tables et de bases spécifiées par les options `replicate-*` .

L'utilisation de cette commande est sujette aux conditions suivantes :

- Fonctionne avec les tables `MyISAM` .
- Pose un verrou global en lecture sur le maître durant la sauvegarde, qui empêche les modifications sur le maître durant la phase de chargement.

Dans le futur, il est prévu de faire que cette commande fonctionne avec les tables `InnoDB` , et qu'elle se passe du verrou global en utilisant des fonctionnalités de sauvegarde à chaud non−bloquantes.

Si vous chargez de très grosses tables, pensez à augmenter les valeurs des options `net_read_timeout` et `net_write_timeout` sur vos maître et esclave. Variables sytème du serveur .

Notez que `LOAD DATA FROM MASTER` **ne copie pas** les tables de droits de la base `mysql` . C'est fait pour simplifier l'utilisation de droits et utilisateurs différents sur le maître et les esclaves.

Cette commande requiert les droits de `RELOAD` et `SUPER` sur le maître, et le droit de `SELECT` sur toutes les tables du maître qui seront lues. Toutes les tables du maître sur lesquelles l'utilisateur n'a pas de droits de `SELECT` seront ignorées par `LOAD DATA FROM MASTER` ; ceci est dû au fait que le maître va masquer ces tables à l'utilisateur : `LOAD DATA FROM MASTER` utilise `SHOW DATABASES` pour connaître les tables à charger, mais `SHOW DATABASES` ne retourne que les bases pour lesquelles l'utilisateur a des droits. Voyez Syntaxe de `SHOW DATABASES` . Sur l'esclave, l'utilisateur qui envoie la commande `LOAD DATA FROM MASTER` doit avoir les droits de création et d'effacement des tables et bases impliquées.

14.6.2.3 Syntaxe de LOAD TABLE tbl_name FROM MASTER

```
LOAD TABLE tbl_name FROM MASTER
```

Télécharge une copie d'une table depuis le maître vers l'esclave. Cette commande est implémentée pour déboguer la commande `LOAD DATA FROM MASTER` . Elle requiert un compte pour se connecter au maître, avec les droits de `RELOAD` et `SUPER` , ainsi que les droits de `SELECT` sur la table a charger. Coté esclave, l'utilisateur qui émet la commande doit avoir les droits de `LOAD TABLE FROM MASTER` pour créer et effacer les tables. Lisez les informations sur les problèmes réseau dans `LOAD DATA FROM MASTER` ; elles s'appliquent aussi ici. Notez aussi que les limitations de `LOAD DATA FROM MASTER` s'appliquent aussi (par exemple, `LOAD TABLE FROM MASTER` ne fonctionne que sur les tables de type `MyISAM`).

14.6.2.4 MASTER_POS_WAIT()

```
SELECT MASTER_POS_WAIT('master_log_file', master_log_pos)
```

C'est une fonction et non pas une commande. Elle sert à s'assurer que l'esclave a atteint (lu et exécuté) les événements du log binaire du maître jusqu'à une certaine position. Voyez la section See Fonctions diverses pour une description complète.

14.6.2.5 RESET SLAVE

```
RESET SLAVE
```

Force l'esclave a oublier toute les positions de réplications dans les logs du maître. Cette commande permet de faire un démarrage propre : elle efface les fichiers `master.info` et `relay-log.info` , et les logs de relais, puis créer un nouveau log de relais. **Note :** tous les logs de relais sont effacés, même si il n'ont pas été totalement exécutés par le thread SQL. (C'est un état qui est probable si l'esclave de réplication est fortement chargé, ou si vous avez lancé une commande `STOP SLAVE` .) Les informations de connexions stockées dans le fichier `master.info` reprennent immédiatement les valeurs spécifiées dans les options de démarrage, si elles étaient précisées. Ces informations incluent notamment le nom de l'hôte maître, le port, l'utilisateur et le mot de passe. Si le thread esclave était au milieu d'une réplication temporaire lorsqu'il a été arrêté, et que `RESET SLAVE` a été émise, ces tables temporaires sont aussi effacées.

Cette commande s'appelait `FLUSH SLAVE` avant MySQL 3.23.26.

14.6.2.6 SET GLOBAL SQL_SLAVE_SKIP_COUNTER

```
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = n
```

Ignore les `n` prochains événements du maître. C'est une commande pratique pour rattraper les arrêts de réplications causés par une commande.Cette commande n'est valide que lorsque le thread esclave ne fonctionne pas. Sinon, elle produit une erreur.

Avant MySQL 4.0, omettez le mot clé `GLOBAL` dans la commande.

14.6.2.7 SHOW SLAVE STATUS

```
SHOW SLAVE STATUS
```

Affiche des informations sur les paramètres essentiels des threads esclaves. Si vous utilisez cette commande avec le client `mysql` , vous pouvez utiliser le terminateur de commande `\G` plutôt que le point−virgule pour avoir un format plus lisible :

```
 mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
       Slave_IO_State: Waiting for master to send event
          Master_Host: localhost
          Master_User: root
          Master_Port: 3306
        Connect_Retry: 3
      Master_Log_File: gbichot-bin.005
  Read_Master_Log_Pos: 79
       Relay_Log_File: gbichot-relay-bin.005
        Relay_Log_Pos: 548
Relay_Master_Log_File: gbichot-bin.005
     Slave_IO_Running: Yes
    Slave_SQL_Running: Yes
      Replicate_Do_DB:
  Replicate_Ignore_DB:
           Last_Errno: 0
           Last_Error:
         Skip_Counter: 0
  Exec_Master_Log_Pos: 79
      Relay_Log_Space: 552
      Until_Condition: None
```

```
       Until_Log_File:
        Until_Log_Pos: 0
   Master_SSL_Allowed: No
   Master_SSL_CA_File:
   Master_SSL_CA_Path:
      Master_SSL_Cert:
    Master_SSL_Cipher:
       Master_SSL_Key:
Seconds_Behind_Master: 8
```

Suivant votre version de MySQL, vous pourriez ne pas voir tous les champs qui sont dans cet exemple. Notamment, il y a plusieurs champs qui ne sont disponibles qu'avec MySQL 4.1.1.

Les champs affichés par `SHOW SLAVE STATUS` ont les définitions suivantes :

`Slave_IO_State`
: Une copie de la colonne `State` de la commande `SHOW PROCESSLIST` pour le thread d'I/O. Elle va vous indiquer si le thread essaye de se connecter au maître, attend des événements, se reconnecte, etc. Les différents états possibles sont listés dans la section Détails d'implémentation de la réplication . Etudier cette colonne est nécessaire, par exemple, car le thread peut fonctionner mais ne pas réussir à se connecter au maître : seule cette colonne vous indiquera ce type de problèmes. D'un autre coté, l'état du thread SQL n'est pas indiqué, car les problèmes sont bien plus simples avec lui : soit il fonctionne, et il n'y a pas de problème; soit il ne fonctionne pas, et vous trouverez les messages d'erreur dans la colonne `Last_Error` , décrite plus bas.

  Ce champ a été ajouté en MySQL 4.1.1.

`Master_Host`
: L'hôte maître courant.

`Master_User`
: Le nom de l'utilisateur utilisé pour se connecter au maître.

`Master_Port`
: Le port courant sur le maître.

`Connect_Retry`
: La valeur courante de l'option `master-connect-retry` .

`Master_Log_File`
: Le nom du fichier de log binaire que le thread d'I/O utilise sur le maître.

`Read_Master_Log_Pos`
: La position que le thread d'I/O a atteint dans le fichier de log binaire du maître.

`Relay_Log_File`
: Le nom du fichier de log de relais dans lequel le thread SQL est actuellement en train de lire et de travailler.

`Relay_Log_Pos`
: La position à laquelle le thread SQL est en train de travailler.

`Relay_Master_Log_File`
: Le nom du fichier de log binaire du maître qui contient le dernier événement exécuté par le thread SQL.

`Slave_IO_Running`
: Indique si le thread d'I/O est lancé ou pas.

`Slave_SQL_Running`
: Indique si le thread SQL est lancé ou pas.

`Replicate_Do_DB, Replicate_Ignore_DB`
: La liste des bases de données qui ont été spécifiée dans l'option `--replicate-do-db` et `--replicate-ignore-db` , éventuellement.

`Replicate_Do_Table, Replicate_Ignore_Table, Replicate_Wild_Do_Table, Replicate_Wild_Ignore_Table`
: La liste des tables qui ont été spécifiées respectivement dans les options `--replicate-do-table` , `--replicate-ignore-table` , `--replicate-wild-do-table` et `--replicate-wild-ignore_table` , éventuellement.

  Ces champs ont été ajoutés en MySQL 4.1.1.

`Last_Errno, Last_Error`
: `Last_Errno` est le numéro d'erreur de la plus récent requête exécutée. La valeur de 0 signifie ``pas d'erreur''. `Last_Error` est le message d'erreur de la plus récent requête exécutée. Par exemple :

```
 Last_Errno: 1051
Last_Error: error 'Unknown table 'z'' on query 'drop table z'
```

Le message indique que la table `z` existait sur le maître et a été effacée, mais qu'elle n'existe pas sur l'esclave, et que `DROP TABLE` a échoué sur l'esclave. Cela peut arriver si vous avez oublié de copier une table dans l'esclave avant de lancer la réplication.Une chaîne vide signifie ``pas d'erreur''. Si `Last_Error` n'était pas vide, alors le même message apparaîtra dans le log d'erreur de l'esclave.

`Skip_Counter`
: La dernière valeur utilisée par `SQL_SLAVE_SKIP_COUNTER` .

`Exec_Master_Log_Pos`
: La position dans les logs binaires du maître (`Relay_Master_Log_File`) pour le dernier événement exécuté par le thread SQL. ((`Relay_Master_Log_File` , `Exec_Master_Log_Pos`) dans le log binaire du maître correspond à (`Relay_Log_File` , `Relay_Log_Pos`) dans le log de relais.

`Relay_Log_Space`
: La taille combinée de tous les logs de relais.

`Until_Condition, Until_Log_File, Until_Log_Pos`
: Les valeurs spécifiées dans la clause `UNTIL` de la commande `START SLAVE` .

`Until_Condition` peut prendre ces valeurs :

- `None` (aucune) si aucune clause `UNTIL` n'a été spécifiée
- `Master` (maître), si l'esclave lit depuis une position donnée, dans le log binaire du maître
- `Relay` (relais) si l'esclave lit dans une position donnée dans le log de relais.

`Until_Log_File` et `Until_Log_Pos` indique le nom du fichier de log et la position qui définissent le point où le thread SQL va s'arrêter d'exécuter.

Ces champs ont été ajoutés en MySQL 4.1.1.

`Master_SSL_Allowed, Master_SSL_CA_File, Master_SSL_CA_Path, Master_SSL_Cert, Master_SSL_Cipher, Master_SSL_Key`
: Ces champs indiquent les paramètres SSL utilisés par l'esclave pour se connecter au maître, s'ils sont fournis. `Master_SSL_Allowed` prend ses valeurs :

- `Yes` (oui) si la connexion SSL au maître est autorisée
- `No` (non) si la connexion SSL au maître est interdite
- `Ignored` (ignoré) si la connexion SSL au maître est autorisée par l'esclave mais que le support de SSL n'est pas là.

Les valeurs des autres champs correspondent aux valeurs des options `--master-ca` , `--master-capath` , `--master-cert` , `--master-cipher` et `--master-key` .

Ces champs ont été ajoutés en MySQL 4.1.1.

`Seconds_Behind_Master`
: Le nombre de secondes qui se sont écoulées depuis le timestamp du dernier événement maître exécuté par le thread SQL. Ce sera `NULL` si aucun événement n'a été exécuté, ou après une commande `CHANGE MASTER` et `RESET SLAVE` . Cette colonne sert à mesurer le retard de l'esclave sur le maître. Cela fonctionne même si le maître et l'esclave on des horloges réglées différemment.

Ce champ a été ajouté en MySQL 4.1.1.

14.6.2.8 START SLAVE

```
 START SLAVE [thread_name [, thread_name] ... ]
START SLAVE [SQL_THREAD] UNTIL
    MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos
START SLAVE [SQL_THREAD] UNTIL
    RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos

 thread_name = IO_THREAD | SQL_THREAD
```

`START SLAVE` , appelé sans option, démarre les deux threads esclaves. Le thread I/O lire les requêtes du maître et les stocke dans le log de relais. Le thread SQL lire le log de relais, et exécute les requêtes. Notez que si `START SLAVE` réussi à lancer le thread esclave, elle se terminera sans erreur.

Mais même dans ce cas, il se peut que le thread esclave se lance, puis s'arrête (car il n'a pas pu se connecter au maître, ou lire le log binaire ou tout autre problème). `START SLAVE` ne vous préviendra pas de cet événement. Vous devez vérifier le log d'erreur de l'esclave pour voir si des messages ont été générés, ou encore vérifier que tout fonctionne avec la commande `SHOW SLAVE STATUS` .Depuis `MySQL` 4.0.2, vous pouvez ajouter les options `IO_THREAD` ou `SQL_THREAD` à la commande, pour nommer les threads que vous lancez.

Depuis `MySQL` 4.1.1, une clause `UNTIL` peut être ajoutée pour spécifier que l'esclave doit commencer à un certain point dans le log binaire, ou dans le log de relais. Lorsque le thread SQL atteint ce point, il s'arrête. Si l'option `SQL_THREAD` est spécifiée dans la commande, seule le thread SQL est lancé. Sinon, les deux threads sont lancés. Si le thread SQL est déjà lancé, la clause `UNTIL` est ignorée, et une alerte est émise.

Avec la clause `UNTIL` , vous devez spécifier à la fois un fichier de log et une position. Ne confondez pas les options du maître et celles du log de relais.

Toute condition `UNTIL` est annulée par une commande `STOP SLAVE` , ou une commande `START SLAVE` qui n'inclut pas de condition `UNTIL` , ou encore un redémarrage serveur.

La clause `UNTIL` peut être utile pour déboguer la réplication, ou pour vous assurer que la réplication s'effectue jusqu'à un certain point. Par exemple, si une commande imprudente `DROP TABLE` a été exécutée sur le maître, vous pouvez utiliser la clause `UNTIL` pour dire à l'esclave de s'exécuter jusqu'à ce moment, puis de s'arrêter. Pour trouver cet événement, utilisez l'utilitaire `mysqlbinlog` sur le log du maître, ou sur le log de relais, ou encore utilisez la commande `SHOW BINLOG EVENTS` .

Si vous utilisez la clause `UNTIL` pour faire des réplications par portions, il est recommandé de lancer l'esclave avec l'option `--skip-slave-start` pour éviter que le thread SQL ne se lance lorsque l'esclave se lance. Il est probablement idéale d'utiliser cette option dans un fichier d'options plutôt qu'en ligne de commande, pour qu'un redémarrage intempestif ne l'oublie pas.

La commande `SHOW SLAVE STATUS` affiche un champ qui indique la valeur courante de la clause `UNTIL` .

14.6.2.9 STOP SLAVE

```
STOP SLAVE [thread_name [, thread_name] ... ]

thread_name = IO_THREAD | SQL_THREAD
```

Arrête l'esclave. Tout comme `START SLAVE` , cette commande peut être utilisée avec les options `IO_THREAD` et `SQL_THREAD` pour identifier le thread par son nom.

Cette commande s'appelait `SLAVE STOP` avant `MySQL` 4.0.5. Actuellement, `SLAVE STOP` est toujours disponible pour assurer la compatibilité ascendante, mais c'est une commande abandonnée.

15 Types de tables MySQL

MySQL supporte plusieurs moteurs de stockage, qui gère différents types de tables. Les moteurs de tables MySQL peuvent être transactionnels ou non−transactionnels.

- Le moteur de tables originale était `ISAM` , qui gérait des tables non−transactionnelles. Ce moteur a été remplacé par le moteur `MyISAM` et ne doit plus être utilisé. Il est abandonné et ne doit plus être utilisé. depuis MySQL 4.1, et il sera supprimé en MySQL 5.0.
- En MySQL 3.23.0, les moteurs `MyISAM` et `HEAP` ont été introduits. `MyISAM` est une version améliorée de `ISAM` . Le moteur `HEAP` propose des tables stockées en mémoire. Le moteur `MERGE` a été ajouté en MySQL 3.23.25. Il permet le regroupement de tables `MyISAM` identique sous la forme d'une seule table. Tous les trois moteurs sont non−transactionnels, et sont tous inclus par défaut. Notez que le moteur `HEAP` est maintenant nommé `MEMORY` .
- Les moteurs `InnoDB` et `BDB` gèrent des tables transactionnelles, et ont été introduits en MySQL 3.23. Les deux font partie de la distribution source de MySQL 3.23.34a. `BDB` est inclus dans les distributions MySQL−Max pour les systèmes d'exploitation qui le supporte. `InnoDB` est aussi inclus dans les distributions binaires MySQL−Max de MySQL 3.23. Depuis MySQL 4.0, `InnoDB` est inclus par défaut dans toutes les distributions binaires. Dans les distributions source, vous pouvez l'activer ou pas en configurant la compilation.
- `NDBCluster` est le moteur de stockage de MySQL MySQL Cluster qui implémente des tables réparties sur plusieurs serveurs. Il est disponible avec les distributions source depuis MySQL 4.1.2.

Ce chapitre décrit les différents moteurs de tables MySQL, hormis `InnoDB` , qui est présenté dans le chapitre Le moteur de tables `InnoDB` et `NDBCluster` qui est présenté dans le chapitre Introduction à MySQL Cluster .

Lorsque vous créez une table, vous pouvez indiquer à MySQL le type de table avec la clause `ENGINE` ou `TYPE` lors de la commande de `CREATE TABLE` :

```
 CREATE TABLE t (i INT) ENGINE = INNODB;
CREATE TABLE t (i INT) TYPE = MEMORY;
```

`ENGINE` est le terme recommandé, mais il ne peut pas être utilisé avant MySQL 4.0.18. `TYPE` est disponible depuis MySQL 3.23.0, la première version de MySQL qui dispose de plusieurs moteurs de tables.

Si vous omettez l'option `ENGINE` ou `TYPE` , le type de table par défaut sera utilisé. C'est généralement `MyISAM` . Cela peut être changé en modifiant la variable système `table_type` .

Pour convertir une table d'un type à l'autre, utilisez la commande `ALTER TABLE` , pour indiquer le nouveau type :

```
 ALTER TABLE t ENGINE = MYISAM;
ALTER TABLE t TYPE = BDB;
```

See Syntaxe de `CREATE TABLE` and Syntaxe de `ALTER TABLE` . Si vous essayez d'utiliser un moteur de stockage qui n'est pas compilé ou qui est désactivé, MySQL créera une table de type `MyISAM` . Ce comportement est pratique pour copier des tables entre serveurs MySQL qui supportent différents moteurs. Par exemple, dans une architecture de réplication, votre serveur maître supporte des tables transactionnelles, mais l'esclave n'utilise que des tables non−transactionnelles, pour améliorer la vitesse.

Cette substitution automatique par une table de type `MyISAM` pour un type de moteur indisponible peut être gênant pour un nouvel utilisateur MySQL. En MySQL 4.1 et plus récent, une alerte est générée lorsque le type de la table est modifié.

MySQL crée toujours un fichier `.frm` pour stocker le type de la table et les informations de définition. Les données et les index de la table peuvent être stockés ailleurs, en fonction du type de tables. Le serveur crée le fichier `.frm` par dessus le moteur de stockage. Les moteurs peuvent créer des fichiers supplémentaires, en fonction de leurs besoins.

Les avantages des tables transactionnelles (TST) sont :

- Plus sûr. Même si MySQL crashe ou que vous avez un problème matériel, vous pouvez récupérer vos données, soit par un recouvrement automatique, soit à partir d'une sauvegarde combinée avec le log des transactions.
- Vous pouvez combiner plusieurs commandes et les accepter toutes d'un seul coup avec la commande `COMMIT` .
- Vous pouvez utiliser `ROLLBACK` pour ignorer vos modifications (si vous n'êtes pas en mode auto−commit).
- Si une mise à jour échoue, tout vos changements seront annulés. (Avec les tables NTST tous les changements opérés sont permanents)
- Gère mieux les accès concurrents si la table reçoit simultanément plusieurs lectures.

Notez que pour utiliser le moteur `InnoDB` en MySQL 3.23, vous devez configurer au moins l'option de démarrage `innodb_data_file_path` . En 4.0 et plus récent, `InnoDB` utilise les valeurs par défaut de la configuration, si vous ne les spécifiez pas. Configuration `InnoDB` .

Avantages des tables non−transactionnelles (NTST) :

- Plus rapides puisqu'il n'y a pas de traitement des transactions.
- Utilisent moins d'espace disque puisqu'il n'y a pas de traitement des transactions.
- Utilisent moins de mémoires pour les mises à jour.

Vous pouvez combiner les tables TST et NTST dans la même requête pour obtenir le meilleur des deux types. Cependant, dans une transaction sans auto−validation, les modifications à une table non−transactionnelles seront toujours immédiatement enregistrés, et ne pourront pas être annulé.

15.1 Le moteur de tables MyISAM

`MyISAM` est le type par défaut de table en MySQL version 3.23. Il est basé sur `ISAM` et ajoute de nombreuses extensions pratiques.

Chaque table `MyISAM` est stockée en trois fichiers. Les fichiers portent le nom de la table, et ont une extension qui spécifie le type de fichier. Le fichier `.frm` stocke la définition de la table. L'index est stocké dans un fichier avec l'extension `.MYI` (`MYIndex`), et les données sont stockées dans un fichier avec l'extension `.MYD` (`MYData`).

Pour spécifier explicitement que vous souhaitez une table `MyISAM` , indiquez le avec l'option `ENGINE` ou `TYPE` lors de la création de la table :

```
 CREATE TABLE t (i INT) ENGINE = MYISAM;
CREATE TABLE t (i INT) TYPE = MYISAM;
```

Normalement, les options `ENGINE` et `TYPE` sont inutiles : `MyISAM` est le type par défaut de table en MySQL, à moins d'avoir été spécifié autrement.Vous pouvez vérifier ou réparer une table `MyISAM` avec l'utilitaire `myisamchk` . Utilisation de `myisamchk` pour résoudre un crash . Vous pouvez aussi compresser les tables `MyISAM` avec l'utilitaire `myisampack` pour réduire leur taille sur le disque. `myisampack` , le générateur de tables MySQL compressées .

Voici les nouveautés des tables `MyISAM` :

- Toutes les clés numériques sont stockées avec l'octet de poids fort en premier, pour améliorer la compression. Cela rend les données indépendantes du système d'exploitation et de la machine. La seule règle pour assurer la portabilité binaire des fichiers est que la machine doit utiliser des entiers signés pour le complément à 2 (c'est le cas de toutes les machines ces 20 dernières années), et un format de nombre à virgule flottante compatible IEEE (c'est aussi le format dominant). Le seul point où la portabilité n'est pas assurée est les machine portables, qui ont des processeurs originaux.

  Il n'y a pas de coût spécifique à stocker les données dans ce format. Les octets dans la table sont généralement non−alignés, et cela ne prend pas longtemps d'aligner des octets. De plus, le code qui lit les valeurs des colonnes n'est pas critique par rapport au reste du code.
- Support des grands fichiers (63 bits) sur les systèmes de fichiers et les systèmes d'exploitation qui supportent les grands fichiers.
- Les lignes de taille dynamique sont bien moins fragmentées lors de l'utilisation d'insertion et d'effacement. Cela se fait en combinant automatiquement les blocs adjacent libres, et en étendant la taille des blocs avec le suivant s'il est vide.
- Le nombre maximal d'index par table est de 64 (32 avant MySQL 4.1.2). Cela peut être changé en recompilant. Le nombre de colonnes maximal par index est 16.
- La taille maximale d'une clé est de 1000 octets (500 avant MySQL 4.1.2). Cela peut être changé en recompilant MySQL. Dans le cas où la clé est plus longue que 250 octets, une taille de bloc de clé plus large est utilisée, en remplacement des 1024 octets par défaut.
- Les colonnes `BLOB` et `TEXT` peuvent être indexées.
- Les valeurs `NULL` sont autorisées dans une colonne indexée. Elles prennent 0 à 1 octets par clé.
- Les valeurs numériques sont stockées avec l'octet de poids fort en premier, pour permettre une meilleure compression.
- Les fichiers d'index sont généralement plus petits en `MyISAM` qu'en `ISAM` . Cela signifie que `MyISAM` va utiliser moins de ressources systèmes que `ISAM` , mais il prendra plus de processeur lors de l'insertion de données dans un index compressé.
- Lorsque les lignes sont insérées dans un ordre trié (comme lorsque vous utilisez une colonne de type `AUTO_INCREMENT`), l'arbre des clé sera scindé, pour que noeud principal ne contienne qu'une clé. Cela va améliorer l'utilisation d'espace dans l'arbre des clés.
- La gestion interne des colonnes `AUTO_INCREMENT` . `MyISAM` va automatiquement modifier cette valeur lors d'une insertion ou d'une modification. La valeur courante d' `AUTO_INCREMENT` peut être modifiée avec `myisamchk` . Cela va rendre les colonnes `AUTO_INCREMENT` plus rapide (au moins 10%) et les anciens nombres ne seront pas réutilisés, comme avec les vieilles tables `ISAM` . Notez que lorsque une clé `AUTO_INCREMENT` est définie à la fin d'une clé multiple, l'ancien comportement est toujours présent.
- Vous pouvez insérer de nouvelles lignes dans une table qui n'a aucun bloc vide dans le fichier de données, en même temps que d'autres threads lisent le fichier de données (insertion simultanée). Un bloc vide peut provenir d'une modification de ligne à format dynamique (les données sont maintenant plus petites). Lorsque tous les blocs vide sont à nouveau utilisés, les insertions suivantes peuvent être simultanées.
- Vous pouvez placer les fichiers de données et d'index dans différents dossiers pour obtenir plus de vitesse avec les options de table `DATA DIRECTORY` et `INDEX DIRECTORY` , dans la commande `CREATE TABLE` . Syntaxe de `CREATE TABLE` .
- Depuis MySQL version 4.1, chaque colonne de caractères peut avoir un jeu de caractères distinct.
- Il y a un indicateur dans le fichier `MyISAM` qui indique si la table a été correctement fermée. Si `mysqld` est lancé avec l'option `--myisam-recover` , les tables `MyISAM` vont automatiquement être vérifiées et réparées, si elles n'ont pas été correctement refermées.
- `myisamchk` va marquer les tables comme vérifiéz s'il est exécuté avec l'option `--update-state` . `myisamchk --fast` va uniquement vérifier les tables qui n'ont pas cette marque.

- `myisamchk -a` stocke les statistiques pour les parties de clés (et non plus pour les clés complètes, comme avec `ISAM`).
- `myisampack` peut compresser des colonnes `BLOB` et `VARCHAR` . `pack_isam` ne le peut pas.

`MyISAM` supporte aussi les fonctionnalités suivantes, dont MySQL pourra profiter sous peu :

- Support du vrai type `VARCHAR` ; une colonne `VARCHAR` commence avec une taille, stockée sur 2 octets.
- Les tables ayant des colonnes `VARCHAR` peuvent avoir un format de lignes fixe ou dynamique.
- `VARCHAR` et `CHAR` peuvent prendre jusqu'à 64 ko.
- Un index de hashage peut être utilisé avec `UNIQUE` . Cela vous permettra d'avoir un index `UNIQUE` sur toute combinaison de colonnes de la table. Vous ne pourrez pas utiliser un index `UNIQUE` pour une recherche.

15.1.1 Options de démarrage MyISAM

Les options suivantes de `mysqld` permettent de modifier le comportement des tables `MyISAM` :

`--myisam-recover=mode`
: Active le mode de restauration automatique des tables `MyISAM` corrompues.

`--delay-key-write=ALL`
: N'écrit pas les buffers de clés entre deux écritures dans une table `MyISAM` .

  Note : Si vous faîtes cela, vous ne devez pas utiliser les tables `MyISAM` avec d'autres programmes (comme depuis un autre serveur MySQL ou avec `myisamchk`) lorsque la table est utilisée. Sinon, vous allez obtenir une corruption d'index.

  Utiliser `--external-locking` n'aidera pas les tables qui utilisent `--delay-key-write` .

Options de ligne de commande `mysqld` .Les variables systèmes suivantes affectent le comportement des tables `MyISAM` :

`bulk_insert_buffer_size`
: La taille du cache d'index lors des insertions de masse. **Note :** c'est une limite par **par thread** !

`myisam_max_extra_sort_file_size`
: Utilisée pour aider MySQL à décider quand utiliser le cache de clé lent mais sûr. **Note :** ce paramètre était donné en megaoctets avant MySQL 4.0.3, et en octets depuis 4.0.3.

`myisam_max_sort_file_size`
: N'utilise pas la méthode de tri rapide pour créer un index, si un fichier temporaire dépasserait cette taille. **Note :** ce paramètre était donné en megaoctets avant MySQL 4.0.3, et en octets depuis 4.0.3.

`myisam_sort_buffer_size`
: La taille du buffer lors de la restauration de table.

Variables sytème du serveur .La restauration automatique est activée si vous lancez `mysqld` avec l'option `--myisam-recover` . Dans ce cas, lorsque le serveur ouvre la table `MyISAM` , il vérifie si la table a été marquée comme crashée ou si le compteur de tables ouvertes n'est pas zéro ou si le serveur utilise `--skip-external-locking` . Si une des conditions précédente est vraie, il arrive ceci :

- La table est analysée pour rechercher des erreurs.
- Si le serveur trouve une erreur, il essaie de faire une réparation rapide (avec le tri, sans recréer de données).
- Si la réparation échoue à cause d'une erreur dans le fichier de données (par exemple, une erreur de clé), le serveur essaie à nouveau, en re−créant le fichier de données.

- Si la réparation échoue encore, le serveur essaie encore avec une ancienne méthode réparation (écrire les lignes les unes après les autres, sans tri). Cette méthode devrait être capable de réparer tout les types d'erreurs, et elle occupe peu de place sur le disque.

Si la restauration n'est toujours pas capable de retrouver toutes les lignes, et que vous n'avez pas spécifié l'option `FORCE` dans la valeur de l'option `--myisam-recover` , la réparation automatique s'annule, avec le message d'erreur suivant :

```
Error: Couldn't repair table: test.g00pages
```

Si vous spécifiez la valeur `FORCE` , une alerte comme celle−ci sera écrite dans les logs :

```
Warning: Found 344 of 354 rows when repairing ./test/g00pages
```

Notez que si la valeur de restauration automatique inclut `BACKUP` , le processus de restauration créera des fichiers avec des noms de la forme `tbl_name-datetime.BAK` . Vous devriez avoir une tâche régulière avec `cron` pour supprimer automatiquement ces fichiers dans les bases de données pour nettoyer le volume.

15.1.2 Espace requis pour les clefs

MySQL supporte plusieurs types d'index, mais le type normal est `ISAM` ou `MyISAM` . Ils utilisent un index `B-tree` , et vous pouvez avoir une approximation de la taille du fichier d'index en faisant la somme de `(longueur_clef+4)/0.67` pour toutes les clefs. (Cela est le pire des cas où les clefs sont insérées dans l'ordre et qu'aucune n'est compressée.

Les index de chaînes de caractères sont compressés par rapport aux espaces. Si la première partie de l'index est une chaîne, son préfixe sera aussi compressé. La compression des espaces rend le fichier d'index plus petit que ce que nous avions calculé précédemment si la colonne chaîne possède beaucoup d'espaces invisibles en début et fin de chaîne ou est une colonne `VARCHAR` qui n'est pas toujours pleinement utilisée. La compression des préfixes est utilisée sur les clefs qui commencent par un chaîne de caractères. La compression des préfixes s'il y a plusieurs chaînes avec des préfixes identiques.

Dans les tables `MyISAM` , vous pouvez aussi compresser les nombres en spécifiant `PACK_KEYS=1` lors de la création de la table. Cela vous aidera lorsque vous aurez plusieurs clefs de types entier qui auront un préfixe identique et que les nombres seront classé par ordre décroissant des grands octets.

15.1.3 Formats de table MyISAM

`MyISAM` supporte 3 différent types de tables. Deux des trois sont choisis automatiquement selon le type de colonne que vous utilisez. Le troisième, tables compressées, ne peut être crée qu'avec l'outil `myisampack` .

Quand vous créez une table avec `CREATE` ou en modifiez la structure avec `ALTER` vous pouvez, pour les tables n'ayant pas de champs `BLOB` forcer le type de table en `DYNAMIC` ou `FIXED` avec l'option `ROW_FORMAT=#` des tables. Bientôt, vous pourrez compresser/décompresser les tables en spécifiant `ROW_FORMAT=compressed | default` à `ALTER TABLE` . Syntaxe de `CREATE TABLE` .

15.1.3.1 Caractéristiques des tables statiques (taille fixée)

Ceci est le format par défaut. Il est utilisé lorsque la table ne contient pas de colonnes de type `VARCHAR` , `BLOB` , ou `TEXT` .

Ce format est le plus simple et le plus sûr. C'est aussi le format sur disque le plus rapide. La vitesse vient de la facilité avec laquelle les données peuvent être trouvées sur le disque. La recherche de quelque chose avec un index et un format statique est très simple. Multipliez juste le nombre de lignes par la longueur des lignes.

De même, lors du scannage d'une table, il est très facile de lire un nombre constant d'enregistrements avec chaque lecture du disque.

La sécurité est mise en évidence si votre ordinateur crashe lors de l'écriture dans un fichier de taille fixée MyISAM, dans ce cas, `myisamchk` peur facilement trouver où commence et finit chaque ligne. Il peut donc retrouver tous les enregistrements à part celui dont l'écriture a été interrompue. Notez qu'avec MySQL tous les index peuvent toujours être reconstruits :

- Toutes les colonnes `CHAR` , `NUMERIC` , et `DECIMAL` sont complétées par des espaces jusqu'à atteindre la longueur totale de la colonne.
- Très rapide.
- Facile à mettre en cache.
- Facile à reconstruire après un crash, car les enregistrements sont localisés dans des positions fixées.
- N'a pas à être réorganisé (avec `myisamchk`) sauf si un grand nombre de lignes est effacé et que vous voulez retourner l'espace libéré au système d'exploitation.
- Requière usuellement plus d'espace disque que les tables dynamiques.

15.1.3.2 Caractéristiques des tables à format de ligne dynamiques

Ce format est utilisé avec les tables qui contiennent des colonnes de type `VARCHAR` , `BLOB` ou `TEXT` , ou si la table a été créée avec l'option `ROW_FORMAT=dynamic` .

Ce format est un peu plus complexe, car chaque ligne doit avoir un entête pour indiquer sa longueur. Une ligne peut aussi être répartie sur plusieurs blocs, lorsqu'elle est agrandie lors d'une modification.

Vous pouvez utiliser la commande SQL `OPTIMIZE table` ou Shell `myisamchk` pour défragmenter une table. Si vous avez des données statiques que vous modifiez souvent dans la même table, avec des colonnes `VARCHAR` ou `BLOB` , il peut être une bonne idée de placer des colonnes dans une autre table, pour éviter la fragmentation :

- Toutes les colonnes de type chaîne sont dynamiques (hormis celle qui sont de taille inférieure à 4).
- Chaque ligne est précédée d'un octet qui indique quelles sont les lignes vides (`''` , bit à 1) et celle qui le ne sont pas (bit à 0). Une colonne vide n'est pas la même choses qu'une colonne qui contient `NULL` . Si une colonne a une taille de zéro après avoir supprimé les espaces finaux, ou un nombre a une valeur de zéro, il est marqué dans cet octet, et la colonne sera ignorée sur le disque. Les chaînes non vides sont sauvées avec un octet de plus pour y stocker la taille.
- Ce format prend généralement moins de place que des tables à format fixe.
- Chaque ligne consomme autant d'espace que nécessaire. Si une ligne devient trop grande, elle sera coupée en blocs et écrites dans le fichier de données. Cela engendre la fragmentation du fichier de données. Par exemple, si vous modifiez une ligne avec des informations qui excèdent la capacité courante de la ligne, la ligne sera fragmentée. Dans ce cas, vous pouvez avoir à exécuter la commande `myisamchk -r` de temps en temps pour améliorer les performances. Utilisez `myisamchk -ei tbl_name` pour obtenir des statistiques.
- Ce format de table n'est pas toujours facile à reconstituer après un crash, car une ligne peut

être fragmentée en de nombreux blocs, et un fragment peut manquer.
- La taille d'une ligne de format variable se calcule avec :

```
 3
+ (nombre de colonnes + 7) / 8
+ (nombre de colonnes de tailles chars)
+ taille compactée des colonnes numériques
+ taille des chaînes
+ (nombre de colonne de valeur NULL + 7) / 8
```

  Il y a un aussi un supplément de 6 octets pour chaque lien. Une ligne de format dynamique utilise un lien à chaque fois qu'une modification cause un agrandissement de la ligne. Chaque nouveau bloc lié fait au moins 20 octets, pour que le prochain agrandissement utilise aussi ce bloc. Si ce n'est pas le cas, un nouveau bloc sera lié, avec un autre coût de 6 octets. Vous pouvez vérifier le nombre de liens dans une table avec la commande `myisamchk -ed` . Tous les liens sont supprimés avec la commande `myisamchk -r` .

15.1.3.3 Caractéristiques des tables compressées

C'est un type en lecture seule qui est généré avec l'outil optionnel `myisampack` .

Toutes les distributions MySQL depuis la version 3.23.19 incluent `myisampack` par défaut (C'est le moment où MySQL a été mis sous GPL). Pour les versions plus anciennes `myisampack` n'était inclus qu'avec les licences ou contrats, mais le serveur peut toujours lire les tables compressées `myisampack` . Les tables compressées peuvent être décompressées avec `myisamchk` . Pour le moteur de stockage `ISAM` , les tables compressées peuvent être compressées avec `pack_isam` et décompressées avec `isamchk` .

Les tables compressées ont les avantages suivants :

- Les tables compressées prennent très peu d'espace disque. Cela réduit l'espace requis, ce qui est fort utile lors de l'utilisation de petits disques (comme les CD−ROM).
- Chaque ligne est compressée séparemment (optimisation des accès). L'entête d'un enregistrement est fixé (1−3 octets) selon le plus grand enregistrement dans la table. Chaque colonne est compressée différemment. Quelques un des types de compressions sont :
  - Compression des espaces en suffixe.
  - Compression des espaces en préfixe.
  - Les nombres avec la valeur `0` sont stockés en utilisant 1 octet.
  - Si les valeurs dans une colonne de type entier ont un petit intervalle, la colonne est stockée en utilisant le type le plus petit possible. Par exemple, une colonne `BIGINT` (8 octets) peut être stocké en tant que colonne `TINYINT` (1 octet) si toutes les valeurs sont entre `0` et `255` .
  - Si une colonne n'a qu'un petit éventail de valeurs, son type est changé en `ENUM` .
  - Une colonne peut utiliser une combinaison des compressions précédentes.
- Peut gérer les enregistrements de tailles fixes ou variables.

15.1.4 Problèmes avec les tables MyISAM

Le format de fichier que MySQL utilise pour stocker les données a été testé à l'extrême, mais il y a toujours des circonstances qui peuvent corrompre les tables d'une base de données.

15.1.4.1 Tables MyISAM corrompues

Même si le format des tables MyISAM est relativement sûr (tous les changements sont écrits avant que la requête SQL ne retourne quoi que ce soit), vous pouvez quand même vous trouver face à des tables corrompues si l'une des choses suivantes arrive :

- Le processus `mysqld` est tué au milieu d'une écriture.
- Arrêt inattendu de la machine (par exemple, coupure de courant).
- Un problème matériel.
- Vous utilisez un programme externe (comme myisamchk) sur une table active.
- Un bogue logiciel dans le code de MySQL ou de MyISAM.

Les symptômes typiques d'une table corrompue sont :

- Vous obtenez l'erreur
  ```
  Incorrect key file for table: '...'. Try to repair it
  ```
  pendant la sélection de données à partir de cette table.
- Les requêtes ne trouvent pas de lignes dans la table ou retournent des données incomplètes.

Vous pouvez réparer une table corrompue avec `REPAIR TABLE` . Vous pouvez aussi réparer une table, lorsque `mysqld` ne fonctionne pas, avec la commande `myisamchk` . Lorsque `mysqld` est arrêté, vous pouvez vérifier une table avec la commande `myisamchk` . Voyez la section Syntaxe de `CHECK TABLE` , Syntaxe de `REPAIR TABLE` et Syntaxe d'invocation de `myisamchk` .

Si vos tables sont souvent corrompues, vous devez essayez de trouver d'où vient le problème ! Dans ce cas, la chose la plus importante à savoir est, si la table est corrompue, si le serveur `mysqld` s'est interrompu. (cela peut être facilement vérifié en regardant s'il y a une entrée récente `restarted mysqld` dans le fichier d'erreurs de mysqld). Si ce n'est pas le cas, vous devez essayer d'effectuer une série de tests. Voyez Que faire si MySQL crashe constamment et Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .

15.1.4.2 Des clients utilisent la table, ou bien elle n'a pas été fermée correctement

Chaque fichier `MyISAM .MYI` possède un compteur dans l'entête qui peut être utilisé pour savoir si une table a été fermée Proprement.

Si vous obtenez l'avertissement suivant de la part de `CHECK TABLE` ou `myisamchk` :

```
# clients is using or hasn't closed the table properly
```

cela signifie que le compteur n'est plus synchrone. Cela ne signifie Pas que la table est corrompue, mais que vous devez au moins effectuer une vérification sur la table pour vous assurer de son bon fonctionnement.Le compteur fonctionne de la façon suivante :

- La première fois qu'une table est mise à jour dans MySQL, un compteur dans l'entête du fichier est incrémenté.
- Le compteur ne change pas pour les mises à jours suivantes.
- Lors de la fermeture de la dernière instance d'une table (à cause d'un `FLUSH` ou qu'il n'y a plus de place dans le cache de la table) le compteur est décrémenté si la table n'a pas été mise à jour.
- Lorsque vous réparez la table ou vérifiez quel est en bon état, le compteur est remis à zéro.
- Pour éviter les problèmes d'interactions avec d'autres processus qui peuvent vérifier la table, le compteur n'est pas décrémenté à la fermeture si sa valeur était zéro.

En d'autres termes, les seuls moyens d'obtenir ce genre d'erreur sont :

- Les tables `MyISAM` sont copiés sans `LOCK` et `FLUSH TABLES` .

- MySQL a planté entre une mise à jour et la fermeture finale. (Notez que la table peut encore être bonne, vu que MySQL écrit toujours pour tout entre deux requêtes.)
- quelqu'un a exécuté `myisamchk --recover` ou `myisamchk --update-state` sur une table qui était utilisée par `mysqld` .
- Plusieurs serveurs `mysqld` utilisent la table et l'un d'eux a exécuté dessus un `REPAIR` ou un `CHECK` pendant qu'elle était utilisée par un autre serveur. Dans ce cas là, l'utilisation de `CHECK` n'est pas très grave (même si vous obtiendrez des avertissements sur les autres serveurs), mais `REPAIR` doit être évitée vu qu'elle remplace actuellement le fichier de données part un nouveau, ce qui n'est pas signalé aux autres serveurs.En général, c'est une mauvaise idée que de partager un dossier de données avec plusieurs serveurs. Voyez la section Utiliser plusieurs serveurs sur la même machine pour plus de détails.

15.2 Tables assemblées MERGE

Les tables `MERGE` ont été ajoutée en MySQL version 3.23.25. Ce type de table est aussi connu sous le nom de `MRG_MyISAM` . Le code raisonnablement stable.

Une table `MERGE` est un groupe de tables `MyISAM` identiques qui sont utilisées comme une seule. ``Identique'' signifie que toutes les tables ont la même structure de colonnes et d'index. Vous ne pouvez pas regrouper des tables qui ont des index dans un ordre différent. Toutefois, une ou plusieurs tables peuvent être compressées avec `myisampack` . `myisampack` , le générateur de tables MySQL compressées .

Lorsque vous créez une table `MERGE` , MySQL crée deux fichiers sur le disque. Les fichiers ont pour nom celui de la table, et ont un extension qui indique le type de fichiers. Le fichier `.frm` stocke la définition de la table, et le fichier `.MRG` contient les noms des tables qui doivent être utilisées. Originalement, toutes les tables utilisées dans la même table `MERGE` devaient être dans la même base que la table `MERGE` . Cette restriction a été levée en MySQL 4.1.1.

Pour le moment, vous avez simplement besoin des droits de `SELECT` , `UPDATE` et `DELETE` sur les tables que vous avez rassemblé dans la table `MERGE` .

L'exemple suivant vous montre comme utiliser les tables `MERGE` :

```
 mysql> CREATE TABLE t1 (
    ->    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->    message CHAR(20));
mysql> CREATE TABLE t2 (
    ->    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->    message CHAR(20));
mysql> INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');
mysql> INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');
mysql> CREATE TABLE total (
    ->    a INT NOT NULL AUTO_INCREMENT,
    ->    message CHAR(20), INDEX(a))
    ->    TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

Notez que la colonne `a` est indexée dans la table `MERGE` , mais elle n'est pas déclarée comme `PRIMARY KEY` comme elle peut l'être dans les tables `MyISAM` sous−jacente. C'est nécessaire car une table `MERGE` ne peut pas assurer l'unicité de valeurs à travers les tables.Après la création de la table `MERGE` , vous pouvez faire des commandes comme :

```
 mysql> SELECT * FROM total;
+---+---------+
| a | message |
+---+---------+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+---+---------+
```

Pour redéfinir une table `MERGE` avec un autre groupe de tables `MyISAM` , vous pouvez faire ceci : Notez que vous pouvez aussi manipuler le fichier `.MRG` directement, à l'extérieur du serveur MySQL :

```
 shell> cd /mysql-data-directory/current-database
shell> ls -1 t1 t2 > total.MRG
shell> mysqladmin flush-tables
```

- Effacez la table avec la commande `DROP` , puis recréez la.
- Utilisez `ALTER TABLE tbl_name UNION=(...)` pour redéfinir les tables regroupées.
- Modifiez le fichier `.MRG` et utilisez la commande `FLUSH TABLE` sur la table `MERGE` et toutes les tables sous−jacentes, pour forcer le gestionnaire à relire la définition.

Les tables `MERGE` peuvent vous aider dans les situations suivantes :

- Gérer facilement un jeu de tables d'historique. Par exemple, vous pourriez placer les données de chaque mois dans un fichier séparé, en compresser certains avec `myisampack` puis créer une table `MERGE` pour les utiliser.
- Vous donner plus de vitesse. Vous pouvez répartir les grandes tables en lecture seule dans différentes parties du disque. Une table `MERGE` bâtie de cette façon peut être plus rapide qu'une grosse table (vous pouvez aussi et bien sûr, utiliser un système RAID pour arriver aux mêmes avantages).
- Effectuer des recherches plus efficaces. Si vous savez exactement ce que vous recherchez, vous pouvez faire des recherches dans une seule des tables individuelles pour les recherches, et utiliser la table `MERGE` pour les autres opérations. Vous pouvez même avoir de nombreuses tables `MERGE` actives, qui partagent les même fichiers.
- Des réparations plus efficaces. Il est plus facile de réparer les fichiers individuels qui sont rassemblés dans une table `MERGE` que de réparer une grande table.
- Fusion instantanée de plusieurs tables en une seule. Une table `MERGE` utilise les index des tables individuelles. Il n'y a pas besoin de gérer un seul index. Cela rend les tables `MERGE` **très** rapides à faire ou défaire. Notez que vous devez spécifier les définitions de clés lorsque vous créez la table `MERGE` !
- Si vous avez un jeu de table que vous rassemblez dans une grande à la demande ou pour un traitement batch, vous devriez utiliser une table `MERGE` . C'est bien plus rapide, et cela va vous faire économiser de l'espace disque.
- Contourner les limitations de taille du système d'exploitation.
- Vous pouvez créer un alias ou un synonyme pour une table, en utilisant simplement `MERGE` sur une seule. Il n'y a pas de coûts particulier en performance (hormis quelques appels de fonctions indirects, et des `memcpy()` avant chaque lecture).

Les inconvénients des tables de type `MERGE` sont :

- Vous devez utiliser des tables `MyISAM` identiques pour faire une table `MERGE` .
- `MERGE` utilise plus de pointeurs de fichiers. Si vous utilisez une table `MERGE` qui couvre 10 tables et que 10 utilisateurs l'utilisent, vous consommez 10*10 + 10 pointeurs de fichiers (10 fichiers de données, et 10 utilisateurs avec 10 fichiers d'index).
- Les lectures de clés sont plus lentes. Lorsque vous faîtes une lecture sur une clé, le gestionnaire `MERGE` doit faire une lecture dans tous les fichiers d'index des tables sous−jacentes, pour vérifier lequel est le plus proche de la valeur recherchée. Si vous faîtes une lecture du type "lit le suivant", le gestionnaire de table assemblée doit rechercher dans tous les buffers de clés pour la trouver. Uniquement lorsqu'un buffer clé est complet, doit il lire le prochain bloc. Cela rend l'accès aux clés `MERGE` bien plus lent que les recherches `eq_ref` , mais pas aussi lent que les recherches de type `ref` . Voyez la section Syntaxe de `EXPLAIN` (Obtenir des informations `SELECT`) pour plus d'informations sur `eq_ref` et `ref` .

15.2.1 Problèmes avec les tables MERGE

Voici une liste des problèmes connus avec les tables de type `MERGE` :

- Si vous utilisez la commande `ALTER TABLE` pour modifier une table de type `MERGE` en un autre type de table, la liste des tables sous−jacentes sera perdue. Au lieu de cela, les lignes des tables seront copiées dans la nouvelle table, puis on lui assignera le nouveau type.
- Avant MySQL 4.1.1, toutes les tables sous−jacentes et la table `MERGE` devaient être dans la même base de données.
- `REPLACE` ne fonctionne pas.
- Vous ne pouvez pas utiliser `DROP TABLE` , `ALTER TABLE` , `DELETE FROM` dans clause `WHERE` , `REPAIR TABLE` , `TRUNCATE TABLE` , `OPTIMIZE TABLE` , ou `ANALYZE TABLE` sur l'une des tables qui est dans une table `MERGE` ``ouverte''. Si vous faîtes cela, la table `MERGE` va utiliser la table originale, et vous obtiendrez des résultats étranges. Le plus simple est d'utiliser la commande `FLUSH TABLES` pour s'assurer qu'aucune table `MERGE` ne reste ``ouverte.''
- Une table `MERGE` peut pas pas supporter de contrainte de type `UNIQUE` sur toute la table. Lorsque vous faites une insertion, les données vont dans la première ou la dernière table (suivant la méthode d'insertion `INSERT_METHOD=xxx`) et cette table `MyISAM` s'assure que les données sont uniques, mais rien n'est fait pour vérifier l'unicité auprès des autres tables `MyISAM` tables.
- Avant MySQL 3.23.49, `DELETE FROM merge_table` utilisé sans clause `WHERE` va uniquement détruire la table assemblée, mais ne va pas toucher les tables sous−jacentes. En fait, le fichier `.MRG` est effacé, mais pas les tables.
- `RENAME TABLE` utilisé sur une table de type `MERGE` peut corrompre la table. Cela sera corrigé en MySQL 4.1.x.
- La création d'une table de type `MERGE` ne vérifie pas si les tables sous−jacentes sont compatibles. Si vous utilisez une table `MERGE` de cette façon, vous devriez rencontrer des problèmes très étranges.
- L'ordre des index dans la table `MERGE` et ses tables sous−jacentes doit être le même. Si vous utilisez la commande `ALTER TABLE` pour ajouter un index de type `UNIQUE` à une table qui est utilisée dans une table assemblée `MERGE` , puis que vous utilisez `ALTER TABLE` pour ajouter un index normal dans la table `MERGE` , l'ordre des clé sera différent suivant les tables, si jamais il y avait une vieille clé non unique. Ceci est dû au fait que `ALTER TABLE` place les clés `UNIQUE` avant les clés normales, pour être capable de détecter les doublons le plus tôt possible.
- `DROP TABLE` sur une table qui est utilisé par une table `MERGE` ne fonctionne pas sous Windows car le gestionnaire de `MERGE` garde les connexions vers les tables cachées sous la couche MySQL. Comme Windows ne vous permet pas d'effacer une table qui est ouverte, vous devez d'abord fermer toute les tables `MERGE` (avec la commande `FLUSH TABLES`) ou effacer la table `MERGE` avant de pouvoir effacer la table désirée. Nous allons corriger lorsque nous introduirons les vues. `VIEW` s.

15.3 Tables HEAP

Le moteur `MEMORY` storage crée des tables dont le contenu est stocké en mémoire. Avant MySQL 4.1, les tables `MEMORY` étaient appelées des tables `HEAP` . Depuis 4.1, `HEAP` est un synonyme de `MEMORY` , et `MEMORY` est le terme recommandé.

Chaque table `MEMORY` est associée à un fichier sur le dique. Le fichier a le nom de la table, et pour extension `.frm` pour indiquer la définition de la table.

Pour spécifier explicitement que vous voulez une table `MEMORY` , indiquez l'option `ENGINE` ou `TYPE` :

```
 CREATE TABLE t (i INT) ENGINE = MEMORY;
CREATE TABLE t (i INT) TYPE = HEAP;
```

Les tables `HEAP` utilisent un index de hachage, et sont stockées en mémoire. Elles sont très rapides, mais si MySQL crashe, vous perdrez toutes vos données. La table continuera d'exister car leur définition est stockée sur le serveur, dans le fichier `.frm` mais le contenu sera perdu au redémarrage du serveur. Les tables `HEAP` sont très pratiques pour être des tables temporaires.

Voici un exemple qui montre comment créer, utiliser et détruire une table `MEMORY` :

```
mysql> CREATE TABLE test TYPE=MEMORY
    ->     SELECT ip,SUM(downloads) AS down
    ->     FROM log_table GROUP BY ip;
mysql> SELECT COUNT(ip),AVG(down) FROM test;
mysql> DROP TABLE test;
```

Les tables `MEMORY` ont les caractéristiques suivantes :

- Les données pour les tables `HEAP` sont alloués par petits blocs. Les tables sont 100% dynamiques (en insertion). Aucune zone de débordement ou d'espace de clé supplémentaire n'est nécessaire. Les lignes effacées sont placées dans une liste, prêtes à être réutilisées.
- Les tables `MEMORY` peuvent avoir jusqu'à 32 index par table, 16 colonnes par index, et un maximum de 500 pour la tailles des clés.
- Avant MySQL 4.1, le moteur `MEMORY` n'implémentait que des index hash. Depuis MySQL 4.1, Les index hash sont le type par défaut, mais vous pouvez spécifier explicitement que l'index `MEMORY` doit être de type `HASH` ou `BTREE` en ajoutant la clause `USING` :

```
CREATE TABLE lookup
    (id INT, INDEX USING HASH (id))
    ENGINE = MEMORY;
CREATE TABLE lookup
    (id INT, INDEX USING BTREE (id))
    ENGINE = MEMORY;
```

  Les caractéristiques générales des hash et `B-tree` sont décrites dans la section Comment MySQL utilise les index .
- Vous pouvez avoir des clés non−uniques dans une table `MEMORY` . (C'est une fonctionnalité rare pour les index hash).
- Si vous avez un index hash sur une table `HEAP` avec un haut degré de duplication (de nombreux valeurs d'index contiennent la même valeur), les modifications dans cette table peuvent affecter les valeurs des clés et toutes les suppressions seront plus lentes. Le facteur de ralentissement est proportionnel au degré de duplication (ou inversement proportionnel à la cardinalité). Depuis la version 4.1, MySQL supporte les index `BTREE` les tables `HEAP` , que vous pouvez utiliser pour éviter le problème.
- Les tables `HEAP` utilisent un format de ligne fixe.
- `HEAP` ne supporte pas les colonnes de type `BLOB` / `TEXT` .
- `HEAP` ne supporte pas les colonnes de type `AUTO_INCREMENT` .
- Avant MySQL 4.0.2, `HEAP` ne supportait les index sur les valeurs `NULL` .
- Les tables `HEAP` sont partagées entre tous les clients (comme une autre table).
- La caractéristique des tables `MEMORY` qui fait que les tables sont stockées en mémoire est partagée avec les tables internes que le serveur crée à la volée lors du traitement des requêtes. Cependant, les tables internes ont aussi la capacité d'être converties en tables disques automatiquement, si elles deviennent trop grandes. La taille limite est déterminée par la valeur de `tmp_table_size` .

  Les tables `MEMORY` ne peuvent pas être converties en tables disques. Pour vous assurer que vous ne faîtes rien de dangeureux pour le serveur, vous pouvez utiliser la variable système `max_heap_table_size` pour imposer une taille maximale aux tables `MEMORY` . Pour des tables individuelles, vous pouvez utiliser l'option de table `MAX_ROWS` avec la commande `CREATE TABLE` .
- Vous avez besoin de suffisamment de mémoire pour accepter toutes les tables `HEAP` que vous allez utiliser simultanément.
- Pour libérer de la mémoire, vous devez exécuter la commande `DELETE FROM heap_table` , `TRUNCATE heap_table` ou `DROP TABLE heap_table` .

- Si vous voulez remplir les tables `MEMORY` au lancement du serveur MySQL, vous pouvez utiliser l'option `--init-file` . Par exemple, vous pouvez mettre les commandes telles que `INSERT INTO ... SELECT` et `LOAD DATA INFILE` pour lire des données dans une source de données persistante. Options de ligne de commande `mysqld` .
- Si vous utilisez la réplication, les tables `MEMORY` du maître se vident à l'extinction. Rependant, un esclave peut ne pas s'apercevoir que ces tables ont été vidées, et il risque de retourner des données invalides si vous l'utilisez. Depuis MySQL 4.0.18, lorsqu'une table `MEMORY` est utilisée sur le maître, il émet une commande `DELETE FROM` automatiquement, pour synchroniser l'esclave et le maître. Notez que mêmea avec cette stratégie, l'esclave aura des données obsolètes entre le moment où le maître s'éteind et celui où il est redémarré. Mais si vous utilisez l'option `--init-file` pour remplir la table `MEMORY` au lancement du serveur, elle s'assurera que cette intervalle est bien null.
- La mémoire nécessaire pour les tables `HEAP` sont :

```
 SUM_OVER_ALL_KEYS(max_length_of_key + sizeof(char*) * 2)
+ ALIGN(length_of_row+1, sizeof(char*))
```

  `ALIGN()` représente un facteur d'arrondi, car la taille de la ligne doit faire exactement un multiple de la taille du pointeur de `char` . `sizeof(char*)` vaut 4 sur les machines 32 bits et 8 sur une machine 64 bits.

15.4 Tables BDB ou BerkeleyDB

Sleepycat Software fournit à MySQL le moteur de stockage transactionnel Berkeley DB. Ce moteur est généralement appelée `BDB` . Le support des tables `BDB` est inclus par la distribution des sources de MySQL à partir de la version 3.23.34 et est activé dans le binaire MySQL−Max.

En utilisant les tables `DBD` , vos tables ont plus de chances de survivre aux crashs, et vous avez accès à `COMMIT` et `ROLLBACK` avec les transactions. La distribution des sources de MySQL fournit une distribution corrigée de `BDB` pour lui permettre de fonctionner d'une façon plus souple avec MySQL. Vous pouvez utiliser une version non−patchée de `BDB` avec MySQL.

Nous travaillons chez MySQL AB en coopération étroite avec Sleepycat pour garantir une bonne qualité d'interface MySQL/ `BDB` . Même si Berkeley DB est très surveillée et fiable, l'interface MySQL est considérée de qualité Gamma. Nous y travaillons et l'optimisons.

Lorsqu'ils utilisent les tables `BDB` , nous aiderons nos utilisateurs à trouver les problèmes et à créer des batteries de tests reproductibles pour tout problème ayant trait aux tables `BDB` . De tels tests seront aussi envoyés à Sleepycat qui nous aidera aussi à trouver et résoudre les problèmes. Vu qu'il s'agit d'une collaboration à deux niveaux, les problèmes concernant les tables `BDB` prendront un peu plus de temps à être résolus en comparaison avec les autres gestionnaires de tables. Toutefois, étant donné que le code de BerkeleyDB a lui−même été utilisé par plusieurs autres applications à part MySQL, nous n'envisageons pas de rencontrer de gros problèmes avec. Support offert par MySQL AB .

Pour des informations générales sur Berkeley DB, visitez le site de Sleepycat Web, http://www.sleepycat.com/ .

15.4.1 Systèmes d'exploitation supportés par BDB

Actuellement, nous savons que le gestionnaire `BDB` fonctionne avec les systèmes d'exploitation suivants :

- Linux 2.x Intel

- Sun Solaris (SPARC et x86)
- FreeBSD 4.x/5.x (x86, sparc64)
- IBM AIX 4.3.x
- SCO OpenServer
- SCO UnixWare 7.1.x

Il ne fonctionne pas sur les systèmes d'exploitations suivants :

- Linux 2.x Alpha
- Linux 2.x AMD64
- Linux 2.x IA64
- Linux 2.x s390
- Max OS X

Note : La liste ci−dessus n'est pas complète; nous la mettrons à jour au fur et à mesure que nous recevrons des informations à ce propos.

Si, après avoir compilé MySQL avec le support des tables `BDB` tables, obtenez l'erreur suivante dans le fichier de logs quand vous démarrez `mysqld` :

```
 bdb: architecture lacks fast mutexes: applications cannot be threaded
Can't init dtabases
```

Cela signifie que les tables `BDB` ne sont pas supportées par votre architecture. Dans ce cas, vous devez recompiler MySQL sans le support des tables `BDB` .

15.4.2 Installation de BDB

Si vous avez téléchargé une version binaire de MySQL qui inclus le support de BerkeleyDB, vous n'avez qu'à suivre les instructions classiques (MySQL−Max inclut le support `BDB`).

Si vous compilez MySQL depuis les sources, vous pouvez activer le support `BDB` en ajoutant au script `configure` l'option `--with-berkeley-db` en plus des autres options que vous avez. Téléchargez la distribution de MySQL 3.23.34 ou plus récent, allez dans la racine des sources, et tapez :

```
shell> ./configure --with-berkeley-db [other-options]
```

Pour plus d'informations, voyez Installer MySQL sous les autres systèmes Unix , `mysqld-max` , la version étendue du serveur `mysqld` et Installer MySQL à partir des sources .

15.4.3 Options de démarrage BDB

Les options suivantes de `mysqld` peuvent être utilisées pour modifier le comportement des tables `BDB` :

`--bdb-home=répertoire`
: Répertoire de base des tables `BDB` . Cela doit être le même répertoire que vous avez utilisés pour `--datadir` .

`--bdb-lock-detect=#`
: Détection des verrouillages Berkeley. (`DEFAULT` , `OLDEST` , `RANDOM` , ou `YOUNGEST`).

`--bdb-logdir=répertoire`
: Répertoire des fichiers de log de Berkeley DB.

`--bdb-no-recover`
: Ne pas démarrer Berkeley DB en mode de restauration.

`--bdb-no-sync`
: Ne pas vider les tampons synchroniquement.

`--bdb-shared-data`
: Démarrer Berkeley DB en mode multi−processus (Ne pas utiliser `DB_PRIVATE` lors de l'initialisation de Berkeley DB)

`--bdb-tmpdir=répertoire`
: Répertoire des fichiers temporaires de Berkeley DB.

`--skip-bdb`
: Désactive l'utilisation des tables `BDB` .

Options de ligne de commande `mysqld` .Les variables systèmes suivante affectent le comportement des tables `BDB` :

`bdb_max_lock`
: Le nombre maximal de verrous actifs sur une table `BDB` .

Variables sytème du serveur .

Si vous utilisez `--skip-bdb` , MySQL n'initialisera pas la bibliothèque Berkeley DB et cela économisera beaucoup de mémoire. Bien sûr, vous ne pouvez pas utiliser les table `BDB` si vous utilisez cette option. Si vous essayez de créer une table `BDB` , MySQL créera une table `MyISAM` à la place.

Normalement, vous devez démarrer `mysqld` sans `--bdb-no-recover` si vous avez l'intention d'utiliser des tables `BDB` . Cela peut cependant vous poser des problèmes si vous essayez de démarrer `mysqld` alors que des fichiers de log `BDB` sont corrompus. Lancer et résoudre des problèmes avec MySQL .

Vous pouvez spécifier le nombre maximal de verrous avec `bdb_max_lock` (10000 par défaut) que vous pouvez activer sur une table `BDB` . Vous devez l'augmenter si vous obtenez des erreurs du type :

```
 bdb: Lock table is out of available locks
Got error 12 from ...
```

lorsque vous avez fait de longues transactions ou quand `mysqld` doit examiner beaucoup de lignes pour calculer la requête.

Vous pouvez aussi changer les options `binlog_cache_size` et `max_binlog_cache_size` si vous utilisez de grandes transactions multi−lignes. Le log binaire des mises à jour .

15.4.4 Caractéristiques des tables BDB

Chaque table `BDB` est stocké sur le disque en deux fichiers. Les fichiers portent le nom de la table, et ont des extensions qui indiquent le type de fichier. Un fichier `.frm` stocke la définition de la table, et le fichier `.db` contient les données et les index.

Pour spécifier explicitement que vous voulez une table `BDB` , indiquez l'option de création de table `ENGINE` ou `TYPE` :

```
 CREATE TABLE t (i INT) ENGINE = BDB;
CREATE TABLE t (i INT) TYPE = BDB;
```

`BerkeleyDB` est un synonyme de `BDB` pour les options `ENGINE` et `TYPE` .

Le moteur de tables `BDB` fournit un modèle transactionnel. La façon dont vous utilisez ces tables dépend du mode de validation :

- Si vous utilisez le mot d'auto−validation (ce qui est le mode par défaut), les modifications dans les tables `BDB` sont validées immédiatement, et ne peuvent pas être annulées.
- Si vous utilisez le mode de validation manuel, les modifications ne seront rendues

permanentes que si vous envoyez la commande `COMMIT` . AU lieu de valider, vous pouvez aussi annuler avec la commande `ROLLBACK` pour détruire les modifications.Vous pouvez démarrer une transaction avec la commande `BEGIN WORK` pour suspendre le mode d'auto−validation, ou avec `SET AUTOCOMMIT=0` pour le désactiver explicitement.

Syntaxe des `BEGIN/COMMIT/ROLLBACK` .

Le moteur de tables `BDB` a les caractéristiques suivantes :

- Les tables `BDB` peuvent avoir jusqu'à 31 index par table, 16 colonnes par index, et un taille maximale de 1024 octets par index (500 octets avant MySQL 4.0).
- MySQL requiert une clé `PRIMARY KEY` dans chaque table `BDB` pour être capable de faire référence aux lignes précédemment lues. Si vous n'en créez pas, MySQL va gérer une telle clé de manière cachée. La clé cachée a une taille de 5 octets, et est incrémentée à chaque nouvelle insertion.
- La clé `PRIMARY KEY` sera plus rapide que n'importe quelle autre clé, car la `PRIMARY KEY` est stockée avec les données. Comme les autres clés sont stockées sous la forme données + `PRIMARY KEY` , il est important de garder une clé `PRIMARY KEY` aussi courte que possible pour économiser de l'espace disque, et améliorer la vitesse.

  Ce comportement est similaire à celui d' `InnoDB` , où des clés primaires courtes économisent de l'espace pour la clé primaire et pour les index secondaire aussi.
- Si toutes les colonnes auxquelles vous accédez dans une table `BDB` font partie du même index dans la clé primaire, alors MySQL peut exécuter la requête sans avoir à lire la ligne elle−même. Dans une table `MyISAM` , ce qui précède n'est valable que si les colonnes font partie du même index.
- Scanner séquentiellement est plus lent qu'avec `MyISAM` car les tables `BDB` stockent les données dans un fichier `B-tree` et non pas dans un fichier séparé.
- Les clés ne sont pas compressées avec les clés précédentes, comme pour les tables `ISAM` et `MyISAM` . En d'autres termes, les informations de clés prennent un peu plus d'espace pour les tables `BDB` , comparativement aux tables `MyISAM` qui n'utilisent pas l'option `PACK_KEYS=0` .
- Il y a souvent des trous dans les tables `BDB` pour vous permettre d'insérer de nouvelles lignes au milieu de l'arbre de données. Cela rend les tables `BDB` un peu plus grandes que les tables `MyISAM` .
- `SELECT COUNT(*) FROM table_name` est très lent, car les tables `BDB` ne maintiennent pas un compte de leur lignes dans la table.
- L'optimiseur a besoin de connaître une approximation du nombre de lignes dans la table. MySQL résout ce problème en comptant les insertions et en conservant ce compte dans un segment séparé pour chaque table `BDB` . Si vous ne faites pas souvent de `DELETE` ou `ROLLBACK` , ce nombre sera plutôt précis pour l'optimiseur MySQL, mais comme MySQL ne stocke ce nombre qu'à la fermeture de la table, il peut être incorrecte si MySQL s'interrompt inopinément. Cela ne doit pas être fatal si ce nombre n'est pas à 100% correct. Vous pouvez forcer la mise à jour de ce nombre avec la commande `ANALYZE TABLE` ou `OPTIMIZE TABLE` . Syntaxe de `ANALYZE TABLE` . Syntaxe de `OPTIMIZE TABLE` .
- Le verrouillage interne des tables `BDB` est fait au niveau page.
- `LOCK TABLES` fonctionne avec les tables `BDB` sur les autres tables. Si vous n'utilisez pas le verrou `LOCK TABLE` , MySQL va poser un verrou interne multiple sur la table, pour s'assurer que la table est bien verrouillée, si un autre thread tente de poser un verrou.
- Pour permettre les annulations de transaction, `BDB` gère un fichier de log. Pour maximiser les performances, vous devriez placer ces fichiers sur un autre disque que celui de votre base, en utilisant l'option `--bdb-logdir` .
- MySQL fait un point de contrôle à chaque fois qu'un nouveau fichier de log `BDB` est démarré, et supprime les fichiers de logs anciens qui ne sont pas utiles. Si vous exécutez la commande `FLUSH LOGS` , vous placerez un nouveau point de contrôle pour les tables Berkeley

DB.Pour la restauration après crash, vous devez utiliser les sauvegardes et le log binaire de MySQL. Sauvegardes de base de données .

Attention : si vous effacez les anciens fichiers de log qui sont en cours d'utilisation, `BDB` ne sera pas capable de faire la restauration et vous risquez de perdre des données.

- L'application doit toujours être prête à gérer des cas où une modification sur une table `BDB` peut être annulée, ou une lecture abandonnée pour cause de blocage de verrous.
- Si vous atteignez la capacité maximale du disque avec la table `BDB` , vous allez obtenir une erreur (probablement l'erreur 28), et la transaction va s'annuler. C'est un comportement différent des tables `MyISAM` et `ISAM` qui vont attendre que `mysqld` ait trouvé de l'espace disque avant de continuer.

15.4.5 Ce que nous devons corriger dans BDB dans un futur proche :

- Il est très lent d'ouvrir de nombreuses tables `BDB` en même temps. Si vous utilisez des tables `BDB` , il ne faut pas avoir un cache de table trop grand (par exemple, > 256) et vous devriez utiliser l'option `--no-auto-rehash` avec le client `mysql` . Nous envisageons de corriger cela en partie en version 4.0.
- `SHOW TABLE STATUS` ne fourni pas encore beaucoup d'informations pour les tables `BDB` tables.
- Optimiser les performances.
- Ne pas utiliser les verrous de pages lorsque l'on scanne les tables.

15.4.6 Restrictions avec les tables BDB

Voilà les restrictions que vous pouvez rencontrer en travaillant avec les tables `BDB` :

- Les tables `BDB` enregistrent dans le fichier `.db` le chemin vers le fichier tel qu'il était lorsqu'il a été crée. Cela fait que les tables `BDB` ne peuvent être changées de répertoire !
- Lors de la sauvegarde de tables `BDB` , vous devez utiliser `mysqldump` ou effectuer des sauvegardes de tous les fichiers de table (les fichiers `.frm` et `.db`) et les fichiers de log `BDB` . Les fichiers de log de `BDB` sont les fichiers dans le répertoire de base des données nommés `log.XXXXXXXXXX` (dix chiffres);
- Si une colonne accepte les valeurs `NULL` , avec un index unique, seule une seule valeur `NULL` est autorisée. Cela diffère du comportement des autres moteurs.

15.4.7 Erreurs pouvant survenir lors de l'utilisation des tables BDB

- Si vous obtenez l'erreur suivante dans le fichier `hostname.err log` lors du démarrage de `mysqld` :
  ```
  bdb:  Ignoring log file: .../log.XXXXXXXXXX: unsupported log version #
  ```
  cela signifie que la nouvelle version de `BDB` ne supporte pas l'ancien format de log. Dans ce cas, vous devez effacer tous les logs `BDB` du dossier des données (les fichiers dont le nom est au format `log.XXXXXXXXXX`) et redémarrer `mysqld` . Nous vous recommandons aussi d'exécuter un `mysqldump --opt` de vos vieilles tables `BDB` , de les effacer, puis de restaurer les copies.
- Si vous n'êtes pas en mode auto−commit et que vous effacez une table qu'un autre thread utilise, vous obtiendrez le message d'erreur suivant dans le fichier d'erreurs de MySQL :
  ```
   001119 23:43:56  bdb:  Missing log fileid entry
  001119 23:43:56  bdb:  txn_abort: Log undo failed for LSN:
                         1 3644744: Invalid
  ```
  Ce n'est pas une erreur très grave mais nous ne vous recommandons pas d'effacer vos tables si vous n'êtes pas en mode auto−commit, tant que ce problème n'est pas résolu (la solution n'est pas triviale).

15.5 Tables ISAM

Le moteur de table originale de MySQL est le moteur `ISAM` . Il a été le seul moteur disponible jusqu'en MySQL 3.23, lorsque la version améliorée `MyISAM` a été inventée. `ISAM` est maintenant obsolète. Depuis MySQL 4.1, il est inclus dans le source, mais pas activé dans les versions binaires. Il disparaitra en MySQL 5.0. Les versions embarquées de MySQL ne supportent pas les tables `ISAM` par défaut.

Etant donné l'obsolescence de `ISAM` , et comme `MyISAM` est une version améliorée de `ISAM` , il est recommandé de convertir les tables `ISAM` en `MySAM` dès que possible. Pour convertir une table `ISAM` en `MyISAM` , utilisez la commande `ALTER TABLE` :

```
mysql> ALTER TABLE tbl_name TYPE = MYISAM;
```

Pour plus d'informations sur `MyISAM` , voyez Tables `MyISAM` .Chaque table `ISAM` est stockée dans trois fichiers. Les fichiers portent le nom de la table, et ont une extension qui indique leur type. Un fichier `.frm` stocke la définition de table. Le fichier de données a pour suffixe `.ISD` . Le fichier d'index a l'extension `.ISM` .

`ISAM` utilise les index `B-tree` .

Vous pouvez réparer ou vérifier une table `ISAM` avec l'utilitaire `isamchk` . Utilisation de `myisamchk` pour résoudre un crash .

`ISAM` possède les fonctionnalités/propriétés suivantes :

- Clefs compressées et de tailles fixes
- Enregistrements de taille fixée ou dynamique
- 16 clefs avec 16 parties de clefs/clefs
- Taille maximale de la clef 256 (défaut)
- Les données sont enregistrées au format machine; c'est rapide, mais c'est dépendant de la machine/système d'exploitation.

La plupart des choses vraies pour les tables `MyISAM` le sont pour les tables `ISAM` . La différence majeure comparées aux tables `MyISAM` sont :

- Les tables `ISAM` ne sont pas binairement portables entre les plate−formes/systèmes d'exploitations.
- Ne peut pas gérer les tables de taille supérieure à 4Go.
- Ne supporte que la compression des préfixes sur les chaînes de caractères.
- Limites de clefs plus basses.
- Les tables à taille de ligne dynamique sont plus fragmentées.
- Ne supporte pas les tables `MERGE` .
- Les tables sont vérifiées et réparées avec `isamchk` plutôt que `myisamchk` .
- Les tables sont compressées avec `pack_isam` plutôt que `myisampack` .
- Impossible d'utiliser les commandes `BACKUP TABLE` et `RESTORE TABLE` .
- Impossible d'utiliser les commandes d'entretien `CHECK TABLE` , `REPAIR TABLE` , `OPTIMIZE TABLE` et `ANALYZE TABLE` .
- Pas de support pour les index en texte plein ou spatiaux.
- Pas de support pour les jeux de caractères multiples.
- Les index ne peuvent pas être assignés à des caches de clés spécifiques.

16 Tables InnoDB

16.1 Présentation des tables InnoDB

`InnoDB` fournit à MySQL un gestionnaire de table transactionnelle (compatible `ACID`), avec validation (commits), annulations (rollback) et capacités de restauration après crash. `InnoDB` utilise un verrouillage de lignes, et fournit des lectures cohérentes comme Oracle, sans verrous. Ces fonctionnalités accroissent les possibilités d'utilisation simultanées des tables, et les performances. Il n'y a pas de problème de queue de verrous avec `InnoDB` , car les verrous de lignes utilisent très peu de place. Les tables `InnoDB` sont les premières tables MySQL qui supportent les contraintes de clés étrangères (`FOREIGN KEY`).

`InnoDB` a été conçu pour maximiser les performances lors du traitement de grandes quantités de données. Son efficacité processeur n'est égalée par aucun autre moteur de base de données.

Techniquement, `InnoDB` est un gestionnaire de table placé sous MySQL. `InnoDB` dispose de son propre buffer pour mettre en cache les données et les index en mémoire centrale. `InnoDB` stocke les tables et index dans un espace de table, qui peut être réparti dans plusieurs fichiers. Ceci diffère des tables comme, par exemple, MyISAM où chaque table est stockée dans un fichier différent. Les tables `InnoDB` peuvent prendre n'importe quelle taille, même sur les systèmes d'exploitation dont la limite est de 2 Go par fichier.

`InnoDB` est inclus dans les distributions binaires par défaut depuis MySQL 4.0. Pour des informations sur le support `InnoDB` en MySQL 3.23, voyez la section `InnoDB` en MySQL 3.23 .

`InnoDB` est utilisé en production dans plusieurs sites où de grandes capacités de stockages et des performances accrues sont nécessaires. Le fameux site web Slashdot.org utilise `InnoDB` . Mytrix, Inc. stocke plus de 1 To de données dans une base `InnoDB` , et un autre site gère une moyenne de 800 insertions/modifications par secondes avec `InnoDB` .

`InnoDB` est sous licence GNU GPL License Version 2 (de Juin 1991). Si vous distribuez MySQL et `InnoDB` , et que votre application ne satisfait pas les restrictions de la licence GPL, vous devez acheter une licence commerciale **MySQL Pro** sur https://order.mysql.com/?sub=pg.

16.2 Informations de contact InnoDB

Informations de contact de Innobase Oy, producteur de `InnoDB` . Site web : http://www.innodb.com/ . Courrier électronique : Heikki.Tuuri@innodb.com

```
 phone: 358-9-6969 3250 (bureau) 358-40-5617367 (portable)
Innobase Oy Inc.
World Trade Center Helsinki
Aleksanterinkatu 17
P.O.Box 800
00101 Helsinki
Finland
```

16.3 InnoDB avec MySQL version 3.23

Depuis MySQL version 4.0, `InnoDB` est activé par défaut. Les informations suivantes ne s'appliquent qu'à la série 3.23.

Les tables `InnoDB` sont inclues dans la distribution source de MySQL depuis la version 3.23.34a et sont activées dans les exécutables `MySQL-Max` de la série 3.23. Pour Windows, les exécutables `MySQL-Max` sont inclus dans la distribution standard.

Si vous avez téléchargé la version binaire de MySQL qui inclut le support d' `InnoDB` , suivez simplement les instructions du manuel MySQL pour installer la version binaire de MySQL. Si vous avez déjà MySQL−3.23 d'installé, le plus simple pour installer `MySQL-Max` est de remplacer l'exécutable serveur `mysqld` avec l'exécutable correspondant de la distribution `MySQL-Max` . MySQL et `MySQL-Max` ne diffèrent qu'au niveau de ce programme. Voyez Installer MySQL sous les autres systèmes Unix et `mysqld-max` , la version étendue du serveur `mysqld` .

Pour compiler MySQL avec le support `InnoDB` , téléchargez MySQL MySQL−3.23.34a ou plus récent sur le site de http://www.mysql.com/ et configurez MySQL avec l'option `--with-innodb` . Voyez le manuel MySQL pour les instructions d'installation de MySQL. Installer MySQL à partir des sources .

Pour utiliser les tables `InnoDB` dans `MySQL-Max` version 3.23 vous devez spécifier des paramètres dans la section `[mysqld]` du fichier de configuration `my.cnf` , ou, sous Windows, dans `my.ini` . Si vous ne configurez pas `InnoDB` dans le fichier d'options, `InnoDB` ne démarrera pas. Depuis MySQL 4.0, `InnoDB` utilise des valeurs par défaut pour ces paramètres si vous n'en spécifiez pas. Cependant, il est recommandé d'utiliser des valeurs appropriées pour votre système afin d'obtenir performances valables. Le paramétrage est détaillé dans la section Configuration `InnoDB` .)

Au minimum, en version 3.23, vous devez spécifier avec `innodb_data_file_path` où seront les fichiers de données, et les tailles de fichiers. Si vous ne mentionnez pas `innodb_data_home_dir` dans `my.cnf` , le comportement par défaut est de créer ces fichiers dans le dossier de données `datadir` de MySQL. Si vous spécifiez `innodb_data_home_dir` comme une chaîne vide, vous pouvez donner des chemins absolus à vos fichiers de données dans `innodb_data_file_path` .

Le minimum est de modifier la section `[mysqld]` à la ligne

```
innodb_data_file_path=ibdata:30M
```

Mais pour obtenir de bonnes performances, il est recommandé de spécifier des options supplémentaires. Options de démarrage `InnoDB` .

16.4 Configuration InnoDB

Pour utiliser les tables `InnoDB` en MySQL versions 3.23, voyez `InnoDB` en MySQL 3.23 .

En MySQL 4.0, vous n'avez rien à faire pour obtenir le support des tables `InnoDB` . Si vous ne souhaitez pas utiliser les tables `InnoDB` , vous pouvez ajouter l'option `skip-innodb` dans votre fichier d'options MySQL.

Les deux ressources disques importantes gérées par `InnoDB` sont sa table de données et son fichier de log.

Si vous ne spécifiez aucune options de configuration `InnoDB` , MySQL 4.0 et plus récent créera un fichier de données auto−croissant appelé `ibdata1` et deux fichiers de log de 5 Mo appelés `ib_logfile0` et `ib_logfile1` dans le dossier de données MySQL. En MySQL 4.0.0 et 4.0.1, le fichier de données est de 64Mo et pas auto−croissant. En MySQL 3.23, `InnoDB` ne démarrera pas si vous ne fournissez pas d'options de configuration.

Note : pour obtenir les meilleures performances, vous devez explicitement configurer les paramètres InnoDB dans les exemples ci−dessous. Naturellement, il vous faudra adapter les configurations à votre architecture.

Pour configurer le fichier de données `InnoDB` , utilisez l'option `innodb_data_file_path` dans la section `[mysqld]` du fichier `my.cnf` . Sous Windows, vous pouvez utiliser `my.ini` à la place. La valeur de `innodb_data_file_path` doit être une liste d'un ou plusieurs fichiers. Si vous indiquez plusieurs fichiers, séparez les noms par des caractères points−virgule (`';'`) :

```
innodb_data_file_path=datafile_spec1[;datafile_spec2]...
```

Par exemple, une configuration qui crée explicitement un espace de table, avec les mêmes caractéristiques que la configuration par défaut de MySQL 4.0 est :

```
[mysqld]
innodb_data_file_path=ibdata1:10M:autoextend
```

Cette configuraiton crée un fichier de données de 10 Mo `ibdata1` , auto−croissant. Il n'y a pas de dossier de sauvegarde d'indiqué : par défaut, c'est le dossier de données de MySQL.Les tailles sont spécifiées avec les suffixes `M` et `G` pour indiquer des Mo et des Go.

Une table contenant 50Mo de données, appelée `ibdata1` et un fichier 50Mo auto−croissant, appelé `ibdata2` dans le dossier de données est configuré comme ceci :

```
[mysqld]
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

La syntaxe complète de la spécification de fichier de données inclut le nom du fichier, sa taille, et différents attributs :

```
file_name:file_size[:autoextend[:max:max_file_size]]
```

L'attribut `autoextend` et ceux qui le suivent peuvent être utilisés uniquement pour le dernier fichier de données de la ligne `innodb_data_file_path` . `autoextend` est disponible depuis MySQL 3.23.50 et 4.0.2.Si vous spécifiez le dernier fichier avec l'option `autoextend` , `InnoDB` va augmenter la taille du dernier fichier de données jusqu'à ce qu'il n'y ait plus de place dans l'espace de table. Les incréments se feront par bloc de 8 Mo.

Si le disque est plein, vous aurez à ajouter un autre fichier sur un autre disque. Les informations pour reconfigurer une table existante sont données dans la section Ajout et suppression des fichiers de logs et de données `InnoDB` .

`InnoDB` ne connait pas la taille maximale des fichiers sur votre système : il faut donc être prudent lorsque la taille des fichiers ne peut dépasser 2Go. Pour spécifier la taille maximale des fichiers auto−croissant, utilisez l'attribut `max` . La ligne de configuration suivante permet au fichier `ibdata1` de croitre jusqu'à 500Mo :

```
[mysqld]
innodb_data_file_path=ibdata1:10M:autoextend:max:500M
```

`InnoDB` crée les fichiers de données dans le dossier de données de MySQL. Pour spécifier explicitement un dossier, utilisez l'option `innodb_data_home_dir` . Par exemple, pour créer deux fichiers appelés `ibdata1` et `ibdata2` mais pour les placer dans le dossier `/ibdata` , configurez `InnoDB` comme ceci :

```
[mysqld]
innodb_data_home_dir = /ibdata
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

Note : `InnoDB` ne crée pas les dossiers : assurez vous que `/ibdata` existe avant de lancer le serveur. C'est aussi vrai pour les fichiers de log. Utilisez la commande Unix et DOS `mkdir` pour créer les dossiers nécessaires.

`InnoDB` forme le chemin de chaque fichier en concaténant textuellement la valeur de `innodb_data_home_dir` devant le nom du fichier, en ajoutant un slash si nécessaire. Si l'option `innodb_data_home_dir` n'est pas mentionnée dans `my.cnf` , la valeur par défaut est le dossier ``point'' `./` , c'est à dire le dossier de données de MySQL.

Si vous spécifier l'option `innodb_data_home_dir` sous forme de chaîne vide, vous pouvez spécifier des noms de chemins absolus dans la valeur de `innodb_data_file_path` . L'exemple ci−desosus est équivalent au précédent :

```
 [mysqld]
innodb_data_home_dir =
innodb_data_file_path=/ibdata/ibdata1:50M;/ibdata/ibdata2:50M:autoextend
```

Exemple de fichier `my.cnf` simple Supposons que vous avez un serveur avec 128 Mo de RAM et un disque dur. Voici un exemple de configuration de fichier `my.cnf` ou `my.ini` pour `InnoDB` . Nous supposons que vous exécutez MySQL−Max−3.23.50 ou plus récent, ou MySQL−4.0.2 ou plus récent, qui utilisent l'attribut `autoextend` .Cet exemple devrait convenir à une majorité d'utilisateurs, Unix et Windows, qui ne souhaitent pas répartir leur fichiers de données `InnoDB` et leurs logs sur plusieurs disques. Cette configuration crée un fichier de données auto−croissant, appelé `ibdata1` et deux fichiers de log `InnoDB ib_logfile0` et `ib_logfile1` dans le dossier de données MySQL. De plus, le petit fichier d'archive `InnoDB ib_arch_log_0000000000` sera placé dans `datadir` .

```
 [mysqld]
# Vous pouvez placer d'autres options MYSQL ici
# ...
# Le fichier de données doit contenir vos données et index.
# Assurez vous que vous avez l'espace disque nécessaire.
innodb_data_file_path = ibdata1:10M:autoextend
#
# Utilisez un buffer de taille 50 à 80 % de votre mémoire serveur
set-variable = innodb_buffer_pool_size=70M
set-variable = innodb_additional_mem_pool_size=10M
#
# Utiisez un fichier de log de taille 25 % du buffer mémoire
set-variable = innodb_log_file_size=20M
set-variable = innodb_log_buffer_size=8M
#
innodb_flush_log_at_trx_commit=1
```

Vérifiez que le serveur MySQL a les droits de créer ces fichiers dans le `datadir` . Plus généralement, le serveur doit avoir les droits d'accès dans les dossiers où il va créer des fichiers, de données ou de log.Notez que le fichier de données doit être inférieure à 2Go sur certains systèmes d'exploitation. La taille combinée des fichiers de log doit être inférieure à 4Go. La taille combinée des fichiers de données doit être inférieure à 10Go.

Lorsque vous créez pour la première fois une base de données `InnoDB` , il est mieux de lancer le serveur depuis la commande en ligne. `InnoDB` va afficher des informations sur la création de la base, et vous verrez commence ça se passe. Voyez la section plus bas, pour une illustration. Par exemple, sous Windows, vous pouvez démarrer `mysqld-max.exe` , qui est stocké dans `C:\mysql\bin` , vous pouvez le démarrer comme ceci :

```
 C:\> C:\mysql\bin\mysqld-max --console
```

Si vous n'envoyez par de données sur l'écran, vérifiez le fichier de log pour savoir ce que `InnoDB` a indiqué durant le lancement.

Voyez Créer un Tablespace `InnoDB` pour un exemple des informations affichées par `InnoDB` .

Où mettre le fichier d'options sous Windows? Les règles sous Windows sont les suivantes :

- Un seul des deux fichiers `my.cnf` ou `my.ini` doit être créé.
- Le fichier `my.cnf` doit être placé dans le dossier racine du disque `C:` .
- Le fichier `my.ini` doit être placé dans le dossier `WINDIR` , e.g, `C:\WINDOWS` ou `C:\WINNT` . Vous pouvez utiliser la commande `SET` de MS−DOS pour afficher la valeur de `WINDIR` :
  ```
   C:\> SET WINDIR
  windir=C:\WINNT
  ```
- Si votre PC utilise un gestionnaire de démarrage où le `C:` n'est pas votre disque de démarrage, alors votre seule option est d'utilise le fichier `my.ini` .

Oz placer les fichiers d'options sous Unix? Sous Unix, `mysqld` lit les options dans les fichiers suivants, s'ils existent, et dans cet ordre :

- `/etc/my.cnf` Options globales.
- `COMPILATION_DATADIR/my.cnf` Options spécifiques au serveur.
- `defaults-extra-file` Le fichier spécifié avec `--defaults-extra-file=...` .
- `~/.my.cnf` Options spécifiques à l'utilisateur.

`COMPILATION_DATADIR` est le dossier de données de MySQL qui a été spécifié lors de l'utilisation du script `./configure` , avant la compilation de `mysqld` . (typiquement, `/usr/local/mysql/data` pour une installation binaire, ou `/usr/local/var` pour une installation source).

Si vous voulez vous assurer que `mysqld` lit les options uniquement depuis un fichier spécifique, vous pouvez utiliser l'option `--defaults-option` comme première option de ligne de commande, au démarrage du serveur :

```
mysqld --defaults-file=your_path_to_my_cnf
```

Exemple de fichier `my.cnf` complexe : supposons que vous avez un serveur Linux avec 2 Go de RAM et trois disques de 60 Go (situés dans les dossiers `/` , `/dr2` et `/dr3` . Voici ci−dessous un exemple de configuration possible pour `my.cnf` , de `InnoDB` .

```
 [mysqld]
# Vous pouvez placer d'autres options MYSQL ici
# ...
innodb_data_home_dir =
#
# Le fichier de données doivent contenir vos données et index.
innodb_data_file_path = /ibdata/ibdata1:2000M;/dr2/ibdata/ibdata2:2000M:autoextend
#
# Utilisez un buffer de taille 50 à 80 % de votre mémoire serveur
# mais assurez vous sous Linux que l'utilisation totale est inférieure à 2Go
set-variable = innodb_buffer_pool_size=1G
set-variable = innodb_additional_mem_pool_size=20M
innodb_log_group_home_dir = /dr3/iblogs
#
# innodb_log_arch_dir doit être le même que  innodb_log_group_home_dir
# (starting from 4.0.6, you can omit it)
innodb_log_arch_dir = /dr3/iblogs
set-variable = innodb_log_files_in_group=2
#
# Utilisez un fichier de log de taille 15 % du buffer mémoire
set-variable = innodb_log_file_size=250M
set-variable = innodb_log_buffer_size=8M
#
innodb_flush_log_at_trx_commit=1
set-variable = innodb_lock_wait_timeout=50
#
# Décommentez les prochaines lignes, si vous voulez les utiliser
#innodb_flush_method=fdatasync
#set-variable = innodb_thread_concurrency=5
```

Notez que nous avons placé deux fichier de données sur des disques différents. `InnoDB` va remplir l'espace de tables jusqu'au maximum. Dans certains cas, les performances seront améliorées si les données ne sont pas toutes placées sur le même disque physique. Placer les fichiers de log dans des disques séparés est souvent une bonne chose. Vous pouvez aussi utiliser des partitions de disques brutes (`raw devices`) comme fichier de données. Utiliser les `Raw Devices` pour l'espace de tables . **Attention** : en Linux x86, vous devez être très prudent, et ne pas utiliser trop de mémoire. `glibc` va autoriser les processus à dépasser la pile de thread, et votre système va crasher. Cela représente un risque réel si la valeur de 2 Go :

```
 innodb_buffer_pool_size
+ key_buffer_size
+ max_connections*(sort_buffer_size+read_buffer_size+binlog_cache_size)
+ max_connections*2MB
```

Chaque thread va utiliser une pile (souvent 2Mo, mais les exécutables MySQL uniquement 256 ko) et dans le pire des scénarios, `sort_buffer + read_buffer_size` de mémoire supplémentaire.Depuis MySQL 4.1, vous pouvez utiliser 64Go de mémoire physique sur Windows 32 bits. Voyez la description de `innodb_buffer_pool_awe_mem_mb` dans Options de démarrage `InnoDB` .

Comment optimiser d'autres paramètres du serveur `mysqld` **?** Les valeurs qui conviennent à la majorité des utilisateurs sont :

```
 [mysqld]
skip-external-locking
set-variable = max_connections=200
set-variable = read_buffer_size=1M
set-variable = sort_buffer_size=1M
# key_buffer vaut de 5 à 50%
# de la RAM disponible, suivant l'utilisation des
# tables MyISAM, mais gardez
# key_buffer + InnoDB en deça de < 80% de votre RAM
set-variable = key_buffer_size=...
```

16.5 Options de démarrage InnoDB

Cette section décrit les options d' `InnoDB` . Depuis MySQL 4.0, toutes les options peuvent être spécifiées sous la forme `--opt_name=value` en ligne de commande, ou dans les fichiers d'options. Avant MySQL 4.0, les options numériques devaient être spécifiée avec `--set-variable=opt_name=value` ou la syntaxe `-O opt_name=value` .

`innodb_additional_mem_pool_size`
: La taille du buffer mémoire d' `InnoDB` , pour ses dictionnaires d'informations, et ses structures internes de données. Une valeur pratique est 2Mo, mais plus vous aurez de tables dans votre application, plus vous devrez augmenter cette valeur. Si `InnoDB` est à court de mémoire, il va allouer de la mémoire auprès du système, et écrire des messages dans le fichier de logs MySQL.

`innodb_buffer_pool_awe_mem_mb`
: Taille du pool de buffer, en Mo, s'il est placé dans la mémoire AWE de Windows 32 bits. Disponible depuis MySQL version 4.1.0, et uniquement utile sur Windows 32 bits. Si votre système Windows 32 bits supporte plus de 4 Go de mémoire, dite aussi Address Windowing Extensions, vous pouvez allouer le pool de buffer `InnoDB` dans la mémoire physique AWE en utilisant ce paramètre. La valeur maximal est de 64000. Si ce paramètre est spécifié, alors innodb_buffer_pool_size est la fenêtre dans l'espace d'adresse 32 bits de mysqld où `InnoDB` place la mémoire AWE. Une bonne valeur pour innodb_buffer_pool_size est alors 500M.

`innodb_buffer_pool_size`
: La taille de buffer mémoire que `InnoDB` utiliser pour mettre en cache les données et les index de tables. Plus cette valeur est grand, et moins vous ferez d'accès disques. Sur un serveur dédiés, vous pouvez monter cette valeur jusqu'à 80% de la mémoire physique de la machine. Ne lui donnez pas une valeur trop grande, car cela peut engendrer l'utilisation de mémoire sur le disque par votre serveur.

`innodb_data_file_path`
: Chemin individuel vers les fichiers de données, et leur taill. Le chemin complet de chaque fichier de données est créé en concaténant `innodb_data_home_dir` avec les chemins spécifiés ici. La taille du fichier est spécifiée en méga−octets, ce qui explique la présence du 'M' après les spécifications ci−dessus. Depuis la version 3.23.44, vous pouvez donner au fichier une taille supérieure à 4 Go sur les systèmes d'exploitation qui acceptent les gros fichiers. Sur certains systèmes, la taille doit être inférieure à 2 Go. Si vous ne spécifiez pas `innodb_data_file_path` , le comportement par défaut depuis la version 4.0 est de créer un fichier auto−croissant de 10 Mo, appelé `ibdata1` . Depuis la version 3.23.44, vous pouvez donner une taille de fichier de plus de 4Go sur les systèmes d'exploitation qui supportent les grands fichiers. Vous pouvez aussi utiliser les partition raw. Utiliser les `Raw Devices` pour l'espace de tables .

`innodb_data_home_dir`
: La partie commune du chemin de tous les fichiers de données `InnoDB` . Si vous ne mentionnez pas cette option, la valeur par défaut sera celle du dossier de données MySQL. Vous pouvez aussi spécifier une chaîne vide, et dans ce cas, les chemins spécifiés dans `innodb_data_file_path` seront des chemins absolus.

innodb_fast_shutdown
: Par défaut, `InnoDB` fait une purge complète et vide le buffer d'insertion avant une extinction. Ces opérations peuvent prendre beaucoup de temps. Si vous donnez à ce paramètre la valeur de 1, `InnoDB` ignore ces opérations d'extinction. Cette option est valable depuis MySQL 3.23.44 et 4.0.1. Sa valeur par défaut est 1 depuis la version 3.23.50.

`innodb_file_io_threads`
: Nombre de pointeurs de fichier de `InnoDB` . Normalement, cette valeur doit être de 4, mais sur des disques Windows, les accès peuvent être améliorés en augmentant cette valeur.

`innodb_file_per_table`
: Cette option fait que `InnoDB` va stocker chaque table dans un fichier `.ibd` indépendant. Voyez la section sur les espaces de tables multiples. Espaces de tables multiples : mettre un fichier par table . Cette option a été ajoutée en MySQL 4.1.1.

`innodb_flush_log_at_trx_commit`
: Normalement, cette option vaut 1, ce qui signifie que lors de la validation de la transaction, les logs sont écrits sur le disque, et les modifications faites par la transaction deviennent permanentes, et survivront un crash de base. Si vous souhaitez réduire la sécurité de vos données, et que vous exécutez de petites transactions, vous pouvez donner une valeur de 0 à cette option, pour réduire les accès disques.

`innodb_flush_method`
: (Disponible depuis 3.23.40 et plus récent) La valeur par défaut pour cette option est `fdatasync` . Une autre option est `O_DSYNC` .

innodb_force_recovery
: Attention : cette option ne doit être définie que dans les cas où vous voulez exporter les données d'une base corrompue, dans une situation d'urgence. Les valeurs possibles de cette option vont de 1 à 6. La signification des valeurs est décrite dans Forcer la restauration . Par mesure de sécurité, `InnoDB` empêche les modifications de données si la valeur de cette option est supérieure à 0. Cette option est disponible depuis MySQL 3.23.44.

`innodb_lock_wait_timeout`
: Le délai d'expiration des transactions `InnoDB` , en cas de blocage de verrou, avant d'annuler. `InnoDB` détecte automatiquement les blocages de verrous et annule alors les transactions. Si vous utilisez la commande `LOCK TABLES` , ou un autre gestionnaire de table transactionnelles que `InnoDB` dans la même transaction, un blocage de verrou peut survenir, et `InnoDB` ne pourra pas le détecter. Ce délai est donc pratique pour résoudre ces situations.

`innodb_log_arch_dir`
: Le dossier où les logs complétés doivent être archivés, si nous utilisons l'archivage de logs. La valeur de ce paramètre doit être actuellement la même que la valeur de `innodb_log_group_home_dir` .

`innodb_log_archive`
: Cette valeur doit être actuellement de 0. Au moment de la restauration de données à partir d'une sauvegarde, à l'aide des log binaires de MySQL, il n'y a actuellement pas besoin d'archiver les fichiers de log `InnoDB` .

`innodb_log_buffer_size`
: La taille du buffer que `InnoDB` utilise pour écrire les log dans les fichiers de logs, sur le disque. Les valeurs utiles vont de 1 Mo à 8 Mo. Un grand buffer de log permet aux grandes transactions de s'exécuter sans avoir à écrire de données dans le fichier de log jusqu'à la validation. Par conséquent, si vous avez de grandes transactions, augmenter cette taille va réduire les accès disques.

`innodb_log_file_size`
: Taille de chaque fichier de log dans un groupe de log, exprimé en méga−octets. Les valeurs pratiques vont de 1Mo à une fraction de la taille du buffer de log (1 / le nombre de logs, en fait). Plus la taille est grande, moins de points de contrôles seront utilisés, réduisant les accès disques. La taille combinée des logs doit être inférieure à 4 Go sur les systèmes 32 bits.

`innodb_log_files_in_group`

Nombre de fichiers de logs dans le groupe de log. `InnoDB` écrit dans ces fichiers de manière circulaire. Une valeur de 2 est recommandée. C'est la valeur par défaut.

`innodb_log_group_home_dir`

Le dossier pour les fichiers de logs. Il doit avoir la même valeur que `innodb_log_arch_dir` . Si vous ne spécifiez pas de paramètre de log `InnoDB` , la configuration par défaut va créer deux fichiers de logs de 5 Mo, appelés `ib_logfile0` et `ib_logfile1` dans le dossier de données MySQL.

innodb_max_dirty_pages_pct

Cette entier va de 0 à 100. Par défaut, il vaut 90. Le thread principal de `InnoDB` essaie de transmettre les pages au pool de buffer, pour qu'un pourcentage maximal de `innodb_max_dirty_pages_pct` soit encore en attente de flush. Cette option est disponible depuis 4.0.13 et 4.1.1. Si vous avez le droit de `SUPER` , ce pourcentage peut être changée durant l'exécution du serveur :

```
SET GLOBAL innodb_max_dirty_pages_pct = value;
```

`innodb_mirrored_log_groups`

Nombre de copies identiques de groupe de log que nous conservons. Actuellement, cette valeur doit être au minimum de 1.

`innodb_open_files`

Ce n'est utile que si vous utilisez les espaces de tables multiples. Cette option spécifie que le nombre maximal de fichier `.ibd` que `InnoDB` peut garder ouvert simultanément. La valeur minimum est de 10. La valeur maximum est de 300. Disponible depuis MySQL version 4.1.1.

Les pointeurs de fichiers utilisés par `.ibd` sont réservés pour `InnoDB` . Ils sont indépendants de ceux spécifiés par `--open-files-limit` , et n'affectent pas les opérations de cache.

innodb_thread_concurrency

`InnoDB` essaie de garder le nombre de thread système concurents inférieur à la limite de ce paramètre. La valeur par défaut est 8. Si vous avez des problèmes de performances, et que `SHOW INNODB STATUS` revèle que des threads attendent des sémaphores, essayez de diminuer ou augmenter ce paramètre. Si vous avez un serveur avec de nombreux processeurs et disques, vous pouvez essayer d'augmenter la valeur, pour utiliser au mieux les ressources disponibles. Une vlaeur recommandée est la somme du nombre de processeurs et de disques que vous avez. Une valeur de 500 ou supérieur, supprime la vérification de concurence. Cette option a été ajoutée depuis MySQL 3.23.44 et 4.0.1.

16.6 Créer des bases InnoDB

Supposons que vous avez installé MySQL et que vous avez édité le fichier d'options de façon à ce qu'il contiennent les paramètres de configuration nécessaires de `InnoDB` . Avant de démarrer MySQL, vous devez vérifier que les dossiers que vous avez spécifié pour les fichiers de données `InnoDB` et les fichiers de logs existent, et que vous avez des accès suffisants dans ces dossiers. `InnoDB` ne peut pas créer de dossiers, uniquement des fichiers. Vérifiez aussi que vous avez d'espace disque pour les données et les logs.

Lorsque vous démarrez MySQL, `InnoDB` va commencer à créer vos fichiers de données et vos fichiers de log. InnDB va afficher ceci :

```
 ~/mysqlm/sql > mysqld
InnoDB: The first specified datafile /home/heikki/data/ibdata1
did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file /home/heikki/data/ibdata1 size to 134217728
InnoDB: Database physically writes the file full: wait...
InnoDB: datafile /home/heikki/data/ibdata2 did not exist:
new to be created
InnoDB: Setting file /home/heikki/data/ibdata2 size to 262144000
InnoDB: Database physically writes the file full: wait...
```

```
InnoDB: Log file /home/heikki/data/logs/ib_logfile0 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile0 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile1 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile1 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile2 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile2 size to 5242880
InnoDB: Started
mysqld: ready for connections
```

Une nouvelle base de données `InnoDB` a été créée. Vous pouvez vous connecter au serveur MySQL avec votre client MySQL habituel, comme `mysql` . Lorsque vous arrêtez le serveur MySQL avec `mysqladmin shutdown` , `InnoDB` va afficher :

```
 010321 18:33:34  mysqld: Normal shutdown
010321 18:33:34  mysqld: Shutdown Complete
InnoDB: Starting shutdown...
InnoDB: Shutdown completed
```

Vous pouvez observer vos fichiers de données et de logs, et vous apercevrez les fichiers créés. Le dossier de log va aussi contenir un petit fichier appelé `ib_arch_log_0000000000` . Ce fichier est le résultat de la création de base, à partir duquel `InnoDB` a désactivé l'archivage des logs. Lorsque MySQL va être redémarré, l'affichage sera :

```
 ~/mysqlm/sql > mysqld
InnoDB: Started
mysqld: ready for connections
```

16.6.1 Si quelque chose se passe mal à la création de la base de données

Si `InnoDB` renvoie une erreur de système d'exploitation lors d'une opération sur fichier, habituellement le problème est l'un des suivants :

- Vous n'avez pas crée les dossiers de données ou de logs `InnoDB` .
- `mysqld` n'a pas le droit de créer des fichiers dans ces dossiers.
- `mysqld` ne lit pas le bon fichier `my.cnf` ou `my.ini` , et donc ne voit pas les options que vous spécifiez.
- Le disque ou l'espace disque alloué est plein.
- Vous avez crée un sous−dossier dont le nom est le même que celui d'un fichier de données que vous avez spécifié.
- Il y a une erreur de syntaxe dans `innodb_data_home_dir` ou `innodb_data_file_path` .

Si quelque chose se passe mal lors de la création d'une base de données `InnoDB` , vous devez effacer tous les fichiers créés par `InnoDB` . Cela inclut tous les fichiers de données, tous les journaux, les archives. Dans le cas où vous avez déjà crées des tables `InnoDB` , effacez aussi les fichiers `.frm` (et tous les fichiers `.ibd` si vous utilisez les espaces de tables multiples) concernés dans le dossier de données de MySQL. Vous pourrez alors essayer une nouvelle création de base de données `InnoDB` .

16.7 Créer des tables InnoDB

Supposons que vous avez démarré le client MySQL avec la commande `mysql test` . Pour créer une table au format `InnoDB` vous devez spécifier le type `TYPE = InnoDB` lors de la création de table, dans la commande SQL :

```
 CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A)) TYPE = InnoDB;
```

Cette commande SQL va créer une table et un index sur la colonne `A` dans la base `InnoDB` constituée par les fichiers de données que vous avez spécifié dans le fichier de configuration `my.cnf` . De plus, MySQL va créer un fichier `CUSTOMER.frm` dans le dossier de données de MySQL `test` . En interne, `InnoDB` va ajouter une entrée dans son propre dictionnaire de données une entrée pour la table `'test/CUSTOMER'` . De cette façon, vous pouvez créer plusieurs table avec le même nom de `CUSTOMER` , mais dans d'autres bases MySQL, et les noms de seront pas en conflit avec `InnoDB` .Vous pouvez demander la quantité d'espace disponible dans l'espace de tables `InnoDB` avec la commande de statut de MySQL pour toutes les tables de type `TYPE = InnoDB` . La quantité d'espace disponible apparaît dans la section de commentaire de la commande `SHOW` . Par exemple :

```
SHOW TABLE STATUS FROM test LIKE 'CUSTOMER'
```

Notez que les statistiques que `SHOW` vous donne sur les tables `InnoDB` ne sont que des approximations : elles sont utilisées pour les optimisations SQL par MySQL. Les tailles réservées d'index et de table, exprimées en octets, sont précises.

16.7.1 Comment utiliser les transactions de InnoDB avec différentes API

Par défaut, chaque client qui se connecte à MySQL commence avec le mode d'auto−validation activé, ce qui valide automatiquement toutes les requêtes que vous soumettez. Pour utiliser des requêtes mutli−commandes, vous pouvez désactiver l'auto−validation avec la commande `SET AUTOCOMMIT = 0` et utiliser les commandes `COMMIT` et `ROLLBACK` pour valider ou annuler vos transactions. Si vous voulez laisser le mode d'auto−validation tranquille, vous pouvez placer vos commandes entre `START TRANSACTION` et `COMMIT` ou `ROLLBACK` . Avant MySQL 4.0.11, vous deviez utiliser la commande `BEGIN` au lieu de `START TRANSACTION` . L'exemle suivant montre deux transactions. La première est valideé, et la seconde est annulée.

```
 shell> mysql test
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.50-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A))
    -> TYPE=InnoDB;
Query OK, 0 rows affected (0.00 sec)
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO CUSTOMER VALUES (10, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO CUSTOMER VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM CUSTOMER;
+------+--------+
| A    | B      |
+------+--------+
|   10 | Heikki |
+------+--------+
1 row in set (0.00 sec)
mysql>
```

Avec des interfaces comme PHP, Perl `DBI/DBD` , `JDBC` , ODBC ou l'interface C, vous pouvez envoyer des commandes de contrôle de transaction comme `COMMIT` au serveur sous la forme de chaîne, tout comme une autre commande `SELECT` et `INSERT` . Certaines interfaces proposent des fonctions spécifiques pour les validations et annulations de transactions.

16.7.2 Convertir des tables MyISAM vers InnoDB

Important : vous ne devez pas convertir les tables de la base `mysql` , telles que `user` ou `host`) en type `InnoDB` . Ces tables doivent conserver le moteur `MyISAM` .

Si vous voulez que toutes les tables non−système soient crées directement en MySQL, depuis MySQL 3.23.43, ajoutez la ligne `default-table-type=innodb` dans la section `[mysqld]` de votre fichier `my.cnf` ou `my.ini` .

`InnoDB` n'a pas d'optimisation particulière pour séparer la création d'index. Par conséquent, cela ne sert à rien d'exporter et de réimporter les données de la table pour créer les index après. Le moyen le plus rapide pour mettre la table au format `InnoDB` est d'insérer directement les lignes dans une table `InnoDB` , c'est à dire, utiliser `ALTER TABLE ... TYPE=INNODB` , ou créer un table `InnoDB` vide, avec la même définition et de faire l'insertion de toutes les lignes avec `INSERT INTO ... SELECT * FROM ...` .

Si vous avez une contrainte `UNIQUE` sur des clés secondaires, depuis MySQL 3.23.52, vous pouvez accélérer l'importation dans les tables en désactivant la vérification de cette contrainte durant l'insertion : `SET UNIQUE_CHECKS=0;` Pour les grosses tables, cela économise beaucoup d'accès disques car `InnoDB` peut alors utiliser un buffer d'insertion pour écrire les lignes par bloc.

Pour avoir un meilleur contrôle sur le processus d'insertion, il est mieux de faire les insertions des grosses tables par blocs :

```
INSERT INTO newtable SELECT * FROM oldtable
  WHERE yourkey > something AND yourkey <= somethingelse;
```

Une fois que toutes les données ont été insérées dans la table, vous pouvez la renommer.Durant la conversion des grosses tables, vous pouvez donner à `InnoDB` un gros buffer pour réduire les accès disques. Ne le portez pas au−delà de 80% de votre mémoire physique. Donnez aussi de gros fichiers de logs, et un buffer de log important.

Assurez vous que vous n'allez pas manquer d'espace : les tables `InnoDB` prennent beaucoup plus d'espace que les tables `MyISAM` . Si la commande `ALTER TABLE` manque d'espace, elle va lancer une annulation, et cela peut prendre des heures sur un disque plein. Durant les insertions, `InnoDB` utiliser un buffer d'insertion pour rassembler les lignes d'index secondaires avec les index, en groupe. Cela économise beaucoup d'accès disques. Durant l'annulation, ce mécanisme n'est pas utilisé, et elle peut prendre jusqu'à 30 fois plus de temps.

Dans le cas d'un annulation immensément longue, si vous n'avez pas de données critiques dans votre base, il est mieux de tuer le processus, d'effacer tous les fichiers, et de recommencer, plutôt que d'attendre la fin de l'annulation. Pour la procédure complète, voyez Forcer la restauration .

16.7.3 Comment les colonnes AUTO_INCREMENT fonctionnent avec InnoDB

Si vous spécifiez une colonne `AUTO_INCREMENT` dans une table, la table `InnoDB` va ajouter dans le dictionnaire de données un compteur spécial appelé le compteur auto−incrément, qui est utilisé pour assigner les nouvelles valeurs de la colonne. Le compteur est stocké uniquement en mémoire, et non pas sur le disque.

`InnoDB` utilise l'algorithme suivant pour initialiser le compteur auto−incrément pour la table `T` qui contient la colonne de type `AUTO_INCREMENT` appelée `ai_col` : après le démarrage du serveur, lorsqu'un utilisateur fait une insertion dans la table `T` , `InnoDB` exécute la commande suivante :

```
SELECT MAX(ai_col) FROM T FOR UPDATE;
```

La valeur lue par la commande est incrémenté d'une unité, et assignée à la colonne auto−incrément et au compteur de table. Si la table est vide, la valeur de `1` est assignée. Si le compteur n'est pas initialisé et que l'utilisateur appelle la commande `SHOW TABLE STATUS` qui affiche les informations de la table `T` , le compteur est initialisé mais pas incrémenté, et stocké pour être utilisé ultérieurement. Notez que dant cette initialisation, nous posons un verrou exclusif en lecture sur la table, et le verrou durera jusqu'à la fin de la transaction. `InnoDB` follows the same procedure for initializing the auto−increment counter for a freshly created table.

Note that if the user specifies `NULL` or `0` for the `AUTO_INCREMENT` column in an `INSERT` , `InnoDB` treats the row as if the value had not been specified and generates a new value for it.

After the auto−increment counter has been initialized, if a user inserts a row that explicitly specifies the column value, and the value is bigger than the current counter value, the counter is set to the specified column value. If the user does not explicitly specify a value, `InnoDB` increments the counter by one and assigns the new value to the column.

When accessing the auto−increment counter, `InnoDB` uses a special table level `AUTO-INC` lock that it keeps to the end of the current SQL statement, not to the end of the transaction. The special lock release strategy was introduced to improve concurrency for inserts into a table containing an `AUTO_INCREMENT` column. Two transactions cannot have the `AUTO-INC` lock on the same table simultaneously.

Note that you may see gaps in the sequence of values assigned to the `AUTO_INCREMENT` column if you roll back transactions that have gotten numbers from the counter.

The behavior of the auto−increment mechanism is not defined if a user assigns a negative value to the column or if the value becomes bigger than the maximum integer that can be stored in the specified integer type.

16.7.4 Contraintes de clés étrangères FOREIGN KEY

Depuis la version 3.23.43b, InnoDB respecte les contraintes de clé étrangères.La syntaxe des définitions de contraintes de clés étrangères de InnoDB est la suivante :

```
[CONSTRAINT symbol] FOREIGN KEY [id] (index_col_name, ...)
   REFERENCES tbl_name (index_col_name, ...)
   [ON DELETE &#123;CASCADE | SET NULL | NO ACTION | RESTRICT&#125;]
   [ON UPDATE &#123;CASCADE | SET NULL | NO ACTION | RESTRICT&#125;]
```

Les deux tables doivent être de type `InnoDB` , dans la table, il doit y avoir un INDEX où les clés étrangères sont listées comme **première** colonne, dans le même ordre, et dans la table référencée, il doit y avoir un INDEX où les colonnes référencées sont listées comme **premières** colonnes, dans le même ordre. Les préfixes d'index ne sont pas supportés pour les clés de contrainte. `InnoDB` ne crée pas automatiquement les index nécessaires pour les clés étrangères : vous devez ls créer vous−même. Les index sont nécessaires pour accélérer les vérifications de contrainte, et éviter un scan de table.

Les colonnes correspondantes de la contrainte dans la table et la table de référence doivent avoir le même type, pour éviter les conversions lors des comparaisons. La **taille et la présente du signe pour les entiers doit être les mêmes** . La taille des chaînes doivent être les mêmes. Si vous spécifiez une action `SET NULL` , assurez vous que vous **n'avez pas déclaré les colonnes de la table fille** `NOT NULL` .

Si MySQL vous retourne une numéro d'erreur 1005 lors de la comande `CREATE TABLE` , et un message d'erreur de numéro 150, alors la création de la table a échoué à cause de la contrainte de clé étrangère, qui n'a pas été correctement formulée. Similairement, si une commande `ALTER TABLE` échoue et indique une erreur 150, c'est que la définition de la clé étrangère est incorrectement formulée dans la table modifiée. Depuis la version 4.0.13, vous pouvez utiliser la commande `SHOW INNODB STATUS` pour avoir une explication détaillée de la dernière erreur de clé étrangère `InnoDB` sur le serveur.

Depuis la version 3.23.50, `InnoDB` ne vérifie pas la clé étrangère pour les clés étrangères ou les clés référencées qui contienent des valeurs `NULL` .

Une entorse aux standards : si dans la table parente, il y a plusieurs lignes qui ont la même valeur de clé référencée, alors `InnoDB` effectue les vérifications de clé étrangères comme si les autres parents avec la même valeur de clé n'existaient pas. Par exemple, si vous avez défini une contrainte de type `RESTRICT` et qu'il y a une ligne fille avec plusieurs lignes parente, `InnoDB` n'acceptera pas l'effacement d'aucun des parents.

Depuis la version 3.23.50, vous pouvez aussi associer la clause `ON DELETE CASCADE` ou `ON DELETE SET NULL` avec la contrainte de clé étrangère. Les options correspondante `ON UPDATE` sont disponibles depuis la version 4.0.8. Si `ON DELETE CASCADE` est spécifiée, et qu'une ligne de la table parente est effacée, alors `InnoDB` va automatiquement effacer toute les lignes qui sont dans la table fille et dont les valeurs de clé étrangère sont celles référencées dans la ligne parente. Si `ON DELETE SET NULL` est spécifiée, les lignes filles sont automatiquement modifiée pour que la colonne de la clé étrangère prenne la valeur de `NULL` .

Une entorse aux standards : si `ON UPDATE CASCADE` ou `ON UPDATE SET NULL` cascade récursivement jusqu'à **la même table** , elle agira comme pour un `RESTRICT` . Cela est fait pour éviter les boucles infinies des modifications en cascade. Une clause `ON DELETE SET NULL` auto−référente, d'un autre coté, fonctionne depuis la version 4.0.13. La clause `ON DELETE CASCADE` auto−référente à toujours fonctionné.Un exemle :

```
 CREATE TABLE parent(id INT NOT NULL,
                   PRIMARY KEY (id)
) TYPE=INNODB;
CREATE TABLE child(id INT, parent_id INT,
                   INDEX par_ind (parent_id),
                   FOREIGN KEY (parent_id) REFERENCES parent(id)
                     ON DELETE CASCADE
) TYPE=INNODB;
```

Voici un exemple plus complexe où la table `product_order` a des clés étrangères sur deux tables. La première clé est un index à deux colonnes, dans la table `product` . Les autres clés sont mono−colonnes, dans la table `customer` :

```
 CREATE TABLE product (category INT NOT NULL, id INT NOT NULL,
                      price DECIMAL,
                      PRIMARY KEY(category, id)) TYPE=INNODB;
CREATE TABLE customer (id INT NOT NULL,
                      PRIMARY KEY (id)) TYPE=INNODB;
CREATE TABLE product_order (no INT NOT NULL AUTO_INCREMENT,
                      product_category INT NOT NULL,
                      product_id INT NOT NULL,
                      customer_id INT NOT NULL,
                      PRIMARY KEY(no),
                      INDEX (product_category, product_id),
                      FOREIGN KEY (product_category, product_id)
                        REFERENCES product(category, id)
                        ON UPDATE CASCADE ON DELETE RESTRICT,
                      INDEX (customer_id),
                      FOREIGN KEY (customer_id)
                        REFERENCES customer(id)) TYPE=INNODB;
```

Depuis la version 3.23.50, `InnoDB` vous permet d'ajouter une nouvelle clé à une table, grâce à la syntaxe

```
 ALTER TABLE yourtablename
    ADD [CONSTRAINT symbol] FOREIGN KEY [id] (index_col_name, ...)
    REFERENCES tbl_name (index_col_name, ...)
    [ON DELETE &#123;CASCADE | SET NULL | NO ACTION | RESTRICT&#125;]
    [ON UPDATE &#123;CASCADE | SET NULL | NO ACTION | RESTRICT&#125;]
```

N'oubliez pas de commencer par créer les index nécessaires en premier! . Vous pouvez aussi ajouter des clés étrangères reflexives, en utilisant la commande `ALTER TABLE` . Depuis la version 4.0.13, `InnoDB` supporte la syntaxe `ALTER TABLE` pour supprimer une clé étrangère :

```
 ALTER TABLE yourtablename
    DROP FOREIGN KEY fk_symbol
```

Si la clause `FOREIGN KEY` inclut un nom de contraite `CONSTRAINT` lors de la création, vous pouvez utiliser ce nom pour effacer la clé. Les contraintes peuvent porter un nom depuis MySQL 4.0.18. SInon, la valeur `fk_symbol` est généré en interne par `InnoDB` lorsque la clé étrangère est créée. Pour savoir quel symbole utiliser pour effacer une clé étrangère, utilisez la commande `SHOW CREATE TABLE` . Par exemple :

```
 mysql> SHOW CREATE TABLE ibtest11c\G
```

```
*************************** 1. row ***************************
       Table: ibtest11c
Create Table: CREATE TABLE `ibtest11c` (
  `A` int(11) NOT NULL auto_increment,
  `D` int(11) NOT NULL default '0',
  `B` varchar(200) NOT NULL default '',
  `C` varchar(175) default NULL,
  PRIMARY KEY  (`A`,`D`,`B`),
  KEY `B` (`B`,`C`),
  KEY `C` (`C`),
  CONSTRAINT `0_38775` FOREIGN KEY (`A`, `D`)
REFERENCES `ibtest11a` (`A`, `D`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `0_38776` FOREIGN KEY (`B`, `C`)
REFERENCES `ibtest11a` (`B`, `C`)
ON DELETE CASCADE ON UPDATE CASCADE
) TYPE=InnoDB CHARSET=latin1
1 row in set (0.01 sec)

 mysql> ALTER TABLE ibtest11c DROP FOREIGN KEY 0_38775;
```

Depuis MySQL version 3.23.50, l'analyseur `InnoDB` autorise l'utilisation des guillemets obliques autour des noms de tables et colonnes dans une clause `FOREIGN KEY ... REFERENCES ...` . Depuis MySQL 4.0.5, l'analyseur `InnoDB` prend aussi en compte la variable système `lower_case_table_names` .

Dans `InnoDB` en versions < 3.23.50, `ALTER TABLE` et `CREATE INDEX` ne doivent pas être utilisé avec des tables qui ont des contraintes de clés étrangères, ou qui sont référencées dans des clés étrangères : une commande `ALTER TABLE` supprime toutes les clés étrangères qui sont définies pour cette table. Vous ne devriez pas utiliser `ALTER TABLE` sur la table référencée, mais utiliser `DROP TABLE` puis `CREATE TABLE` pour modifier le schéma. Lorsque MySQL exécute la commande `ALTER TABLE` , il risque d'utiliser en interne la commande `RENAME TABLE` , et cela va poser des problèmes pour les clés étrangères qui reposent sur cette table. Une commande `CREATE INDEX` est traitée par MySQL comme une commande `ALTER TABLE` , et ces restrictions s'appliquent aussi.

Lorsqu'il vérifie les clés étrangères, `InnoDB` pose des verrous de lignes partagées sur les lignes des tables qu'il utilise. `InnoDB` vérifie immédiatement les contraintes de clés étrangères : la vérification n'attend pas la validation de la transaction.

Si vous voulez ignorer les contraintes de clés étrangères durant, par exemple, une opération de `LOAD DATA` , vous pouvez utiliser la commande `SET FOREIGN_KEY_CHECKS=0` .

`InnoDB` vous permet d'effacer n'importe quelle table, même si cela va casser les contraintes de clés étranègres qui référence cette table. Lorsque vous supprimez une table, la contrainte de clé étrangère qui y était attachée est aussi supprimée.

Si vous recréez une table qui a été supprimée, sa définition doit se conformer aux contraintes des clés étrangères qui la référencent. Elle doit avoir les bons types et noms de colonnes, et doit avoir les bonnes clés, comme indiqué ci−dessus. Si ces contraintes ne sont pas vérifiées, MySQL vous gratifiera d'une erreur 1005, et vous enverra lire le message numéro 150.

Depuis la version 3.23.50 `InnoDB` retourne la définition de clé étrangère lorsque vous utilisez la commande

```
 SHOW CREATE TABLE tbl_name;
```

De plus, `mysqldump` produit aussi les définitions correctes de tables, sans oublier les clés étrangères.Vous pouvez aussi lister les clés étrangères d'une table `T` avec

```
 SHOW TABLE STATUS FROM db_name LIKE 'tbl_name'
```

Les contraintes de clés étragnères sont listées dans les commentaires de la table.Lors des vérifications des contraintes, `InnoDB` pose des verrous de lignes sur les lignes parents ou enfants qu'il utilise. `InnoDB` vérifie immédiatement les contraintes de clés : la vérification n'est pas reportée jusqu'à la validation de la transaction.

Pour simplifier l'importation de données dans des tables qui ont des contraintes, `mysqldump` ajoute automatiquement la commande qui met la variable `FOREIGN_KEY_CHECKS` à 0, depuis MySQL version 4.1.1.

Cela évite des problèmes spécifiques avec les tables qui doivent être chargées dans un ordre particulier. Pour les versions antérieures, vous pouvez désactiver manuellement la variable depuis `mysql` lors du chargement du fichier comme ceci :

```
 mysql> SET FOREIGN_KEY_CHECKS = 0;
mysql> SOURCE dump_file_name;
mysql> SET FOREIGN_KEY_CHECKS = 1;
```

Cela vous permet de faire l'importation des données des tables dans n'importe quel ordre. Cela accélère aussi l'opération d'importation. `FOREIGN_KEY_CHECKS` est disponible depuis MySQL 3.23.52 et 4.0.3.

Mettre `FOREIGN_KEY_CHECKS` à 0 peut aussi être utile pour les opérations de `LOAD DATA` .

`InnoDB` permet l'effacement de n'importe quelle table, même si cela casse les contraintes de clés étrangères. Lorsque vous effacez une table, les contraintes définies sur cette table sont aussi effacées.

Si vous recréez une table qui a été effacée, elle doit avoir une définition qui est compatible avec les clés étrangères qui l'utilise. Elle doit avoir les bonnes colonnes et les index. Si cela n'est pas vrai, MySQL retourne une erreur 1005, et fait référence à un message d'erreur numéro 150.

16.7.5 InnoDB et la réplication MySQL

La réplication MySQL fonctionne pour les tables `InnoDB` comme pour les tables `MyISAM` . Il est aussi possible d'utiliser la réplication pour que les tables de l'esclave ne soient pas les mêmes que les tables du maître. Par exemple, vous pouvez répliquer les modifications d'une table `InnoDB` sur le maître dans une table `MyISAM` sur l'esclave.

Pour configurer un nouvel esclave sur le maître, vous devez faire une copie de l'espace de table `InnoDB` , des fichiers de log, ainsi que les fichiers `.frm` des tables `InnoDB` , et les placer sur l'esclave. Pour une procédure à suivre pour réaliser cela, voyez Transporter une base de données `InnoDB` vers une autre machine .

Si vous pouvez arrêter le maître ou l'esclave, faîtes une sauvegarde à l'arrêt de l'espace de table `InnoDB` et des fichiers de logs, puis utilisez les pour redémarrer l'esclave. Pour faire un nouvel esclave sans arrêter le serveur, utilisez le logiciel commercial `InnoDB Hot Backup` tool .

Il y a des limitations mineures à la réplication `InnoDB` :

- `LOAD TABLE FROM MASTER` ne fonctionne pas pour les tables `InnoDB` . Il y a des palliatifs : 1) exportez la table du maître, et envoyez la sur l'esclave, ou, 2) utilisez `ALTER TABLE tbl_name TYPE=MyISAM` sur le maître avant de configurer la réplication avec `LOAD TABLE tbl_name FROM MASTER` , et ensuite, `ALTER TABLE` pour remettre les tables en mode `InnoDB` après cela.
- Avant MySQL 4.0.6, `SLAVE STOP` ne respectait la limite de transaction. Une transaction incomplète était annulée, et la prochaine commande `SLAVE START` n'éxécutait que le reste de la transaction, ce qui conduisait à un échec.
- Avant MySQL 4.0.6, un crash de l'esclave au milieu de d'une transaction multi−commande causait le même problème que `SLAVE STOP` .
- Avant MySQL 4.0.11, la réplication de la commande `SET FOREIGN_KEY_CHECKS=0` ne fonctionnait pas correctement.

La plupart de ces limitations peuvent être levées en utilisant un serveur récent, pour lequel les limitations n'existent pas.

Les transactions qui échouent sur le serveur n'affectent pas la réplication. La réplication MySQL est basée sur le log binaire où MySQL écrit les requêtes SQL qui modifient des données. Un escalve lit le log binaire du maître, et éxécute les mêmes commandes SQLO. Cependant, les commandes d'une transaction ne sont pas écrite avant la fin de la transaction, où toutes les commandes sont écrites d'un coup. Si une transaction échoue, par exemple, à cause d'une clé étrangère, ou si la transaction est annulée, aucune requête ne sera écrite dans le log binaire, et la transaction ne sera pas du tout éxécutée sur le serveur.

16.7.6 Espaces de tables multiples : chaque table InnoDB a son fichier @filename{.ibd}

Depuis MySQL 4.1.1, vous pouvez stocker chaque table `InnoDB` et ses index dans son propre fichier. Cette fonctionnalité est appelée ``espaces de tables multiples'', car chaque table dispose de son propre espace de table.

Note importante : si vous passez en version `InnoDB` 4.1.1 ou plus récent, il devient très difficile de retourner en versions 4.0 ou 4.1.0! Ceci est dû au fait que les versions antérieures de `InnoDB` ne sont pas compatibles avec les espaces de tables multiples.

Si vous devez revenir à une vieille version 4.0, vous devez faire des exports des tables, et recréer tout votre espace de tables `InnoDB` . Si vous n'avez pas créé de nouvelles tables sous `InnoDB` >= 4.1.1, et que vous devez revenir rapidement en arrière, vous pouvez passer directement en versions 4.0.18, ou plus récent. Avant de faire un retour en arrière direct en versions 4.0, vous devez terminer toutes les connexions, et laisser `mysqld` vider les buffers d'insertion, jusqu'à ce que `SHOW INNODB STATUS` indique que le thread principal soit dans un état de `waiting for server activity` . Alors, vous pouvez éteindre le serveur `mysqld` et démarrer votre version 4.0.18 ou plus récent. Un retour en arrière direct n'est pas recommandé, car il n'a pas été totalement testé.

Depuis MySQL version 4.1.1, vous pouvez stocker chaque table `InnoDB` et ses index dans son propre fichier. Cette fonctionnalité est appelé espaces de tables multiples, car chaque table a son propre espace de table.

Vous pouvez activer cette fonctionnalité en ajoutant une ligne dans le groupe `[mysqld]` du fichier `my.cnf` :

```
 [mysqld]
innodb_file_per_table
```

Après redémarrage du serveur, `InnoDB` va stocker chaque table dans son propre fichier `tablename.ibd` du dossier de données, où les tables sont stockées. C'est la même méthode que `MyISAM` , mais si MySQL divise les tables en un fichier de données et un fichier d'index, `tablename.MYD` et `tablename.MYI` , `InnoDB` place les données et les index dans le même fichier `.ibd` . Le fichier `tbl_name.frm` est toujours créé, comme d'habitude.

Si vous supprimez la ligne `innodb_file_per_table` du fichier `my.cnf` , alors `InnoDB` créera les tables dans le fichier de données.

`innodb_file_per_table` affecte seulement la création de tables. Si vous démarrez le serveur avec cette option, les nouvelles tables sont créées avec le fichier `.ibd` , mais vous pouvez toujours accéder aux tables qui existent dans l'espace de table partagées `ibdata` . Si vous supprimez cette option, les nouvelles tables seront crées dans l'espace de tables, mais vous pouvez toujours aux tables qui ont été créées indépendamment.

`InnoDB` a toujours besoin du système d'espace de tables, les fichiers `.ibd` ne sont pas suffisants. Le système d'espaces de table est constitué des fichiers classiques `ibdata` . `InnoDB` y place son dictionnaire de données interne, et ses historiques d'annulation.

Vous ne pouvez pas déplacer les fichiers `.ibd` librement , comme vous pouvez le faire avec les tables `MyISAM` . Ceci est dû au fait que les définitions de tables sont stockées dans le système d'espace de tables `InnoDB` , et aussi, parce que `InnoDB` doit préserver la cohérence des identifiants de transactions et les numéros de séquence des logs.

Vous pouvez déplacer le fichier `.ibd` et la table associée d'une base à l'autre (dans la même installation MySQL/ `InnoDB`) avec la classique commande `RENAME` :

```
RENAME TABLE old_db_name.tbl_name TO new_db_name.tbl_name;
```

Si vous avez une sauvegarde ``propre'' du fichier `.ibd` , vous pouvez restaurer l'installation MySQL comme ceci :

- Utilisez cette commande `ALTER TABLE` :
  ```
  ALTER TABLE tbl_name DISCARD TABLESPACE;
  ```
  Attention : cette commande efface le fichier `.ibd` .
- Placez le fichier de sauvegarde `.ibd` dans le bon dossier de base de données.
- Utilisez cette commande `ALTER TABLE` :
  ```
  ALTER TABLE tbl_name IMPORT TABLESPACE;
  ```

Dans ce contexte, un fichier `.ibd` ``propre'' signifie :

- Il n'y a pas de modifications non validées par des transactions dans le fichier `.ibd` file.
- Il n'y a pas de buffer d'insertion non intégrés dans le fichier `.ibd` .
- La purge a supprimé toutes les lignes marquées pour l'effacement dans le fichier `.ibd` .
- `mysqld` a envoyé toute les pages modifiées au fichier `.ibd` depuis le buffer de fichier.

Vous pouvez faire une sauvegarde propre du fichier `.ibd` en suivant la méthode :

- Cessez toute activité sur le serveur `mysqld` et validez toutes les transactions.
- Attendez que `SHOW INNODB STATUS\G` indique qu'il n'y a plus de transaction active dans la base, et que le thread principal de `InnoDB` est dans l'état `Waiting for server activity` . Vous pouvez alors faire une copie du fichier `.ibd` .

Une autre méthode (non libre) de faire une sauvegarde propre du fichier `.ibd` est de :

- Utilisez `InnoDB Hot Backup` pour faire une sauvegarde de votre installation `InnoDB` .
- Lancez un second serveur `mysqld` sur la sauvegarde, et laissez le nettoyer les fichiers `.ibd` de la sauvegarde.

La liste de tâche inclut la possibilité de déplacer les fichiers `.ibd` vers une autre installation MySQL/ `InnoDB` . Cela impose la remise à zéro des numéros de transactions et des séquences de fichiers de log du fichier `.ibd` .

16.8 Ajouter et retirer des données et des logs InnoDB

Cette section décrit ce que vous pouvez faire lors que votre espace de tables `InnoDB` n'a plus d'espace, ou que vous voulez changer la taille des fichiers de logs.

Depuis la version 3.23.50 et 4.0.2, le moyen le plus facile pour augmenter la taille de l'espace de tables `InnoDB` est de le configurer immédiatement comme auto−croissant. Spécifiez l'attribut `autoextend` pour le dernier fichier de données dans la définition de l'espace de table. `InnoDB` va augmenter la taille du fichier automatiquement de 8 Mo dès qu'il n'a plus d'espace dans le fichier.

Alternativement, vous pouvez augmenter la taille de votre espace de tables en ajoutant un autre fichier de données. Pour cela, vous devez éteindre le serveur MySQL, éditer le fichier `my.cnf` pour ajouter votre nouveau fichier de données à la fin de la ligne `innodb_data_file_path` , et relancer le serveur MySQL.

Si votre dernier fichier a déjà été configuré avec le mot−clé `autoextend` , la procédure d'édition du fichier `my.cnf` doit prendre en compte la taille que le fichier occupe actuellement. Vous devez lire la taille du fichier, l'arrondir au megaoctet inférieur le plus proche (1024 * 1024 octets), et spécifier la taille arrondie dans la ligne d'options `innodb_data_file_path` . Puis, vous pouvez ajouter un autre fichier de données. N'oubliez pas que seul le dernier fichier de données de la ligne `innodb_data_file_path` peut être spécifié comme auto−croissant.

Par exemple, supposons que l'espace de tables dispose juste d'un fichier de données auto−croissant, appelé `ibdata1` :

```
 innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:10M:autoextend
```

Supposons qu'au cours du temps, le fichier a atteind la taille de 988 Mo. Ci−dessous, la ligne de configuration va ajouter un autre fichier auto−croissant après celui qui existe.

```
 innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend
```

Lorsque vous ajoutez un nouveau fichier de données, assurez vous qu'il n'existe pas, pour que `InnoDB` puisse le créer et l'initialiser, lorsque vous redémarrerez le serveur.

Actuellement, vous ne pouvez pas supprimer de fichiers de données. Pour réduire la taille de votre espace de table, utilisez cette procédure :

- Utilisez `mysqldump` pour exportez toutes vos tables `InnoDB` .
- Stoppez le serveur.
- Supprimez tous les fichiers de tables existants.
- Configurez un nouvel espace de table.
- Relancez le serveur.
- Importez les fichiers de vos tables.

Si vous voulez changer le nombre ou la taille de vos fichiers de log `InnoDB` , vous devez éteindre le serveur MYSQL et vous assurer qu'il s'est arrêté sans erreur. Puis, copiez les anciens fichiers de log dans une archive, car si vous rencontrez un problème ultérieurement, vous en aurez besoin pour restaurer votre base. Effacer les anciens fichiers de log du dossier de logs, éditez le fichier `my.cnf` , et redémarrez le serveur MySQL. `InnoDB` vous indiquera au démarrage qu'il va créer de nouveaux fichiers de log.

16.9 Sauver et restaurer une base InnoDB

La clé d'une gestion prudente de votre serveur est la sauvegarde régulière des données.

`InnoDB Hot Backup` est un utilitaire de sauvegarde en ligne, qui vous permet de faire des sauvegardes pendant que `InnoDB` fonctionne. `InnoDB Hot Backup` ne vous impose pas l'arrêt de votre serveur, et il ne pose pas de verrous qui vont perturber le fonctionnement normal de votre serveur. `InnoDB Hot Backup` est un utilitaire propriétaire, qui n'est pas inclut dans la distribution MySQL. Voyez le site web de `InnoDB Hot Backup` : http://www.innodb.com/manual.php , pour plus d'informations.

Si vous pouvez arrêter votre serveur MySQL, alors faites une sauvegarde binaire de votre base comme ceci :

- Arrêtez le serveur MySQL et assurez vous qu'il s'est bien arrêté sans erreur.
- Copiez tous les fichiers de données dans votre entrepôt.
- Copiez tous les fichiers de log `InnoDB` dans votre entrepôt.
- Copiez vos fichiers de configuration `my.cnf` dans l'entrepôt.
- Copiez tous les fichiers `.frm` de vos tables `InnoDB` dans votre entrepôt.

En plus de prendre des sauvegardes binaires comme décrit ci−dessus, vous devriez aussi prendre des exports de vos tables avec `mysqldump` . La raison à cela est que le fichier binaire peut être corrompu sans que vous vous en rendiez compte. Les tables exportées sont stockées sous forme de fichier texte, lisible à l'oeil, et bien plus simple à sauver que les fichiers binaires. Repérer la corruption d'une table dans les fichiers exportés est bien plus facile, et comme le format est bien plus simple, il y a moins de chances que les données soient corrompues.

En fait, c'est une bonne idée que de faire une exportation des tables au moment où vous sauvez les fichiers binaires des bases. Vous devez arrêter tous les clients qui utilisent le serveur. Puis, vous pouvez faire la sauvegarde binaire et l'export : vous aurez ainsi une archive cohérente en deux formats.

Pour restaurer une base `InnoDB` à partir d'une sauvegarde binaire, vous devez utiliser le serveur MySQL avec les logs général et d'archive activés. Par log général, nous entendons le log de MySQL, et non pas le log spécifique d' `InnoDB` .

```
mysqlbinlog yourhostname-bin.123 | mysql
```

Pour restaurer les données après un crash MySQL, la seule chose à faire est de relancer le serveur. `InnoDB` va automatiquement vérifier les historiques, et reprendre toutes les opérations qui ont eu lieu jusqu'à présent. `InnoDB` va automatiquement annuler les transactions qui n'ont pas été achevées. Durant la restauration, `InnoDB` va afficher des séquences semblables à celle−ci :

```
 ~/mysqlm/sql > mysqld
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

Si votre base ou vos disques se corrompent, vous devez faire la restauration à partir de sauvegardes. En cas de corruption, vous devriez trouver une sauvegarde qui n'est pas corrompue. A partir de la sauvegarde, faites une restauration à partir des fichiers de logs généraux, en suivant les instructions du manuel.

16.9.1 Forcer la restauration

S'il survient une corruption de base de données, vous souhaiterez exporter vos tables de la base avec `SELECT INTO OUTFILE` , et généralement, la plupart des données seront intactes et correctes. Mais la correction peut faire que `SELECT * FROM table` ou une opération en tâche de fond d' `InnoDB` crashe, ou même, que le processus de récupération de `InnoDB` crashe. Depuis `InnoDB` version 3.23.44, il y a une option du fichier d'options qui vous permet de forcer `InnoDB` a démarrer, et vous permet d'éviter le lancement des opérations en tâche de fond, pour que vous puissiez exporter vos tables. Par exemple, vous pouvez configurer :

```
[mysqld]
innodb_force_recovery = 4
```

Avant MySQL 4.0, utilisez cette syntaxe :

```
[mysqld]
set-variable = innodb_force_recovery=4
```

Les autres possibilités pour `innodb_force_recovery` sont listées ci−dessous. La base ne doit pas être utilisée lorsque vous utilisez ces options. Comme mesure de sécurité, `InnoDB` empêchera un utilisateur de faire des commandes `INSERT`, `UPDATE` et `DELETE` lorsque cette option est supérieure à 0.Depuis la version 3.23.53 et 4.0.4, vous êtes autorisés à utiliser les commandes `DROP` et `CREATE` sur une table, même si la restauration forcée est active. Si vous savez qu'une table particulière vous pose des problèmes, vous pouvez l'effacer. Vous pouvez aussi utiliser cette commande pour stopper une annulation sauvage, qui seraient causée par des importations de masse ou `ALTER TABLE`. Vous pouvez tuer le processu `mysqld` et utiliser l'option `innodb_force_recovery=3` pour relancer votre base sans l'annulation. Alors, la suppression de la table avec `DROP` vous aidera.

Un nombre plus grand signifie que toutes les précautions des nombres inférieurs sont inclus. Si vous êtes capables d'exporter vos tables avec une option au maximum de 4, alors vous êtes sûr que très peu de données sont perdues. L'option 6 est plus dramatique, car les pages de la base de données sont laissées dans un état obsolète, et cela va introduire encore plus de corruption dans les arbres `B-tree` et les structures de la base.

- 1 (SRV_FORCE_IGNORE_CORRUPT) laisse le serveur fonctionner même s'il détecte une page corrompue. Essaie d'éviter les index et pages avec `SELECT * FROM table`, ce qui aide à l'export.
- 2 (SRV_FORCE_NO_BACKGROUND) empêche le thread principal de fonctionner. Si un crash survenait durant la purge, cela l'évitera.
- 3 (SRV_FORCE_NO_TRX_UNDO) ne lance pas les annulations de transactions après la restauration.
- 4 (SRV_FORCE_NO_IBUF_MERGE) empêche les opérations de vidange du buffer d'insertion. Si ce sont eux qui causent le crash, il vaut mieux les ignorer. Ne recalcule pas les statisttiques de tables.
- 5 (SRV_FORCE_NO_UNDO_LOG_SCAN) ne regarde pas les logs d'annulations lors du lancement de la base. `InnoDB` va traiter les transactions, même incomplètes, comme validées.
- 6 (SRV_FORCE_NO_LOG_REDO) ne lance pas le rattrapage des opérations avec le log, après la restauration.

16.9.2 Points de contrôle

`InnoDB` utilise un mécanisme de contrôle appelé points de contrôle flou. `InnoDB` va écrire des blocs de données modifiées depuis un buffer vers le disque par petits paquets : il n'y a pas besoin de tout écrire en une seule fois, ce qui en général, conduit à l'arrêt du traitement des autres instructions pour quelques instants.

Durant une restauration de base, `InnoDB` recherche un point de contrôle écrit dans les fichiers de log. Il sait que toutes les modifications de la base placées avant ce point de contrôle sont aussi présentes sur le disque de la base. Puis, `InnoDB` analyse le fichier de log, et applique les modifications qui ont eu lieu depuis le point de contrôle.

`InnoDB` écrit dans les fichiers de log en mode circulaire. Toutes les modifications validées qui font que le buffer de MySQL est différent de la version sur le disque doivent être disponibles dans les fichiers de log, au cas où `InnoDB` aurait besoin pour une restauration. Cela signifie que lorsque `InnoDB` commence à réutiliser le fichier d'historique, il doit commencer par s'assurer que le disque a reçu les modifications qui sont dans le buffer. En d'autres termes, `InnoDB` doit placer un point de contrôle, et souvent, cela se traduit par l'écriture de données du buffer sur le disque.

Ceci explique pourquoi utiliser de grands fichiers de log peut éviter des accès disques pour les points de contrôle. Il est recommandé d'utiliser une taille de fichier d'historique aussi grande que le buffer, voire même plus grande. L'inconvénient des grands fichiers est que la restauration dure alors plus longtemps, puisqu'il y a plus de modifications à appliquer.

16.10 Transférer une base de données InnoDB vers une autre machine

Sur Windows, `InnoDB` stocke les noms de bases et de tables en interne, et toujours en minuscules. Pour déplacer des bases au format binaire, entre Unix et Windows, ou le contraire, vous devez donner des noms en minuscules à toutes vos bases et tables. Un moyen simple de faire cela sous Unix est d'ajouter la ligne suivante dans la section `[mysqld]` de votre fichier d'options `my.cnf` avant de démarrer la création de tables.

```
set-variable=lower_case_table_names=1
```

Sous Windows, cette option vaut 1 par défaut.

Les fichiers de données et de logs de `InnoDB` sont compatibles en mode binaire sur toutes les plates−formes si le format des nombre à virgule flottante est le même. Vous pouvez déplacer une base de données `InnoDB` en copiant tous les fichiers concernés, que nous avons déjà listés dans la section Sauvegarde et recouvrement des bases de données `InnoDB` . Si les formats des nombres à virgules flottantes sont différents mais que vous n'avez pas utilisé les types de données `FLOAT` ou `DOUBLE` dans vos tables alors la procédure est là même : copiez juste les fichiers concernés. Si les formats sont différents et que vous utilisez de tels types de données, vous devez utiliser `mysqldump` et `mysqlimport` pour transférer les tables.

Un bon moyen d'avoir de bonnes performances est de couper le mode auto−commit quand vous importez des données dans votre base de données, en supposant que votre espace de tables possède assez d'espace pour la grande partie d'annulations (`rollback`) que la grande transaction importée génère. Ne faites le commit qu'après avoir importé une table entière, ou un segment de table.

16.11 Modèle de transactions et verrouillage InnoDB

Le modèle transactionnel d' `InnoDB` a pour but de combiner les avantages des bases de données multi−version aux verrouillages traditionnels en deux phases. `InnoDB` fait un verrouillage de ligne, et exécute les requêtes par défaut avec des lectures cohérentes non bloquante, de la même façon qu'Oracle. Les verrous `InnoDB` sont stockés de manière efficace, pour que l'escalade de transaction ne soit pas nécessaire : typiquement, plusieurs utilisateurs sont autorisés à verrouiller toutes les lignes dans une base, ou un sous ensemble aléatoire de ligne, sans que `InnoDB` ne soit à court de mémoire.

16.11.1 InnoDB et AUTOCOMMIT

Avec `InnoDB` , toutes les opérations sont placées dans une transaction. Si le mode d'auto−validation est activé, chaque commande SQL est une transaction à part entière. MySQL démarre toujours une nouvelle transaction lorsque le mode d'auto−validation est activé.

Si l'auto−validation est désactivée avec `SET AUTOCOMMIT = 0` , alors nous pouvons considérer qu'une transaction est toujours commencée. Une commande SQL `COMMIT` ou `ROLLBACK` termine la transaction courante et en commence une autre. Ces deux commandes vont libérer tous les verrous `InnoDB` qui étaient posés durant la transaction. Un `COMMIT` signifie que les modifications durant la transaction seront enregistrés, et rendus visibles aux autres. Un `ROLLBACK` , d'un autre coté, annule toutes les modifications.

Si la connexion a activé l'auto−validation, l'utilsiateur peut faire une transaction multi−commandes en commençant la transaction avec la commande `START TRANSACTION` ou `BEGIN` et en la terminant avec `COMMIT` ou `ROLLBACK` .

16.11.2 InnoDB et SET ... TRANSACTION ISOLATION LEVEL ...

En terme de niveau d'isolation des transactions SQL−92, le comportement par défaut de `InnoDB` est `REPEATABLE READ` . Depuis la version 4.0.5, `InnoDB` offre 4 niveaux différents d'isolation de transactions, tels que décrit dans la norme SQL−92. Vous pouvez configurer le niveau d'isolation par défaut dans le groupe `[mysqld]` du fichier `my.cnf` :

```
transaction-isolation = &#123;READ-UNCOMMITTED | READ-COMMITTED
                        | REPEATABLE-READ | SERIALIZABLE&#125;
```

Un utilisateur peut changer le niveau d'isolation d'une session ou des nouvelles connexion avec la commande `SET TRANSACTION` . La syntaxe est la suivante :

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
                      &#123;READ UNCOMMITTED | READ COMMITTED
                       | REPEATABLE READ | SERIALIZABLE&#125;
```

Notez qu'il n'y a pas de tiret dans les noms de niveaux de la syntaxe SQL.Le comportement par défaut est de configurer le niveau d'isolation de la prochaine transaction non démarrée. Si vous utilisez le mot clé `GLOBAL` , la commande configure le niveau d'isolation globalement, pour toutes les nouvelles connexions, mais pas pour les connexions existantes. Vous devez avoir les droits de `SUPER` pour faire cela. En utilisant le mot clé `SESSION` , vous allez configurer le niveau d'isolation des prochaines transactions de la session courante. Tous les clients sont autorisés à changer le niveau d'isolation de la session, même au milieu d'une transaction, ainsi que le niveau d'isolation de la prochaine transaction. Dans les versions 3.23.50 et plus anciennes, `SET TRANSACTION` n'avait pas d'effet sur les tables `InnoDB` . Dans les versions < 4.0.5, seules `REPEATABLE READ` et `SERIALIZABLE` étaient disponibles.

Vous pouvez connaître les niveaux d'isolation de transaction de session et global avec ces commandes :

```
 SELECT &#64;&#64;global.tx_isolation;
SELECT &#64;&#64;tx_isolation;
```

Pour le verrouillage de ligne, `InnoDB` utilise le système dit de verrouillage de la prochaine clé. Cela signifie qu'en plus des lignes d'index, `InnoDB` peut aussi verrouiller le ``trou'' après une ligne d'index pour éviter les insertions des autres utilisateurs. Un verrou de prochaine clé représente un verrou qui bloque la ligne d'index et le trou qui le précède. Un verrou de trou représente un verrou qui ne bloque que le trou avant la ligne.

Voici une description détaillée de chaque niveau d'isolation de `InnoDB` :

- `READ UNCOMMITTED` This is also called ``dirty read'' : Les commandes `SELECT` non−verrouillantes sont effectuées sans rechercher de versions plus récente de la ligne. Elles sont donc non−cohérentes, sous ce niveau d'isolation. SInon, ce niveau fonctionne comme le niveau `READ COMMITTED` .
- `READ COMMITTED` Proche du niveau d'isolation d'Oracle. Toutes les commandes `SELECT ... FOR UPDATE` et `SELECT ... LOCK IN SHARE MODE` ne verrouille que les lignes d'index, et non pas les trous entre elles, ce qui permet l'insertion de nouvelles lignes, adjacentes aux lignes verrouillées. `UPDATE` et `DELETE` qui utilisent un seul index avec une condition de recherche unique ne verrouille que la ligne d'index trouvée, par le trou qui la précède. Mais, pour un intervalle de lignes traitées par `UPDATE` et `DELETE` , `InnoDB` doit méler des verrous de prochaine clé ou des verrous de trous et bloquer les insertions des autres utilisateurs dans les espaces couvert par l'intervalle. Ceci est nécessaire car les ``lignes fantômes'' doivent être bloquées pour la réplication MySQL et la restauration. La lecture cohérente se comporte comme avec Oracle : chaque lecture cohérente, même dans une transaction, lit et modifie son propre contexte. Lecture cohérente .
- `REPEATABLE READ` C'est le niveau d'isolation par défaut d' `InnoDB` . Les commandes `SELECT ... FOR UPDATE` , `SELECT ... LOCK IN SHARE MODE` , `UPDATE` et `DELETE` qui utilisent un index unique dans une condition de recherche, ne verrouille que la ligne d'index trouvée, et non pas le trou précédent. SInon, ces opérations utilisent le verrouillage de prochaine clé, en verrouillant l'intervalle utilisé, et bloque les insertions des utilisateurs. Avec les lectures cohérentes il y a une importante différence avec le niveau d'isolation précédent : dans ce mode, toutes les lectures cohérentes d'une même transaction liront les mêmes données, établies par la première lecture. Cette convention signifie que si vous émettez plusieurs commandes `SELECT` dans la même transaction, ces `SELECT` seront cohérents les uns avec les autres. Lecture cohérente .
- `SERIALIZABLE` Ce niveau est identique au précédent, mais toutes les lectures `SELECT` sont implicitement converties en `SELECT ... LOCK IN SHARE MODE` .

16.11.3 Lecture cohérente non−bloquante

Une lecture cohérente signifie que `InnoDB` utilise son système de multi−versionage pour présenter à une requête, une photo de la base à un moment donné. La requête va alors voir les différentses modifications apportées par les transactions qui ont eu lieu avant cette date, et masquera les transactions ont eu lieu depuis, ou n'ont pas été archivées. L'exception à cette règle est que la requête verra les modification apportées la requête qui a émis cette commande.

Si vous utilisez le niveau d'isolation `REPEATABLE READ` , alors les lectures cohérentes dans une même transaction liront le même bilan. Vous pouvez obtenir un bilan plus récent pour vos requêtes en archivant la requête courante, et en démarrant une autre.

Les lectures cohérentes sont le mode par défaut de traitement des commandes `SELECT` par `InnoDB` avec les niveaux d'isolation `READ COMMITTED` et `REPEATABLE READ` . Une lecture cohérente ne pose aucun verrou sur les tables auxquelles elle accède, et par conséquent, les autres utilisateurs peuvent librement modifier ces tables en même temps qu'une lecture cohérente est exécutée.

16.11.4 Verrous de lecture SELECT ... FOR UPDATE et SELECT ... LOCK IN SHARE MODE

Une lecture cohérente n'est pas toujours pratique, dans certaines circonstances. Supposons que vous voulez ajouter une ligne dans votre table `CHILD` , et vous assurer que l'enfant a déjà un parent dans la table `PARENT` .

Supposez que vous utilisiez une lecture cohérente, pour lire la table `PARENT` , et que vous découvrez le parent de l'enfant dans cette table. Pouvez vous ajouter tranquillement la ligne fille dans la table `CHILD` ? Non, car il peut arriver que durant ce temps, un autre utilisateur a effacé la ligne parente dans la table `PARENT` , et vous n'en êtes pas conscient.La solution est d'exécuter la commande `SELECT` en mode verrouillage, avec `LOCK IN SHARE MODE` .

```
SELECT * FROM PARENT WHERE NAME = 'Jones' LOCK IN SHARE MODE;
```

Effectuer une lecture en mode partagé signifie que vous allons lire les dernières données disponibles, et que nous allons poser un verrou sur les lignes que nous lisons. Si les dernières données appartiennent à une transaction non validée d'un autre utilisateur, nous allons attendre que ce soit fait. Un verrou partagé évite que les autres utilisateurs ne modifient ou n'effacent la ligne que nous lisons. Après que nous ayons obtenu le nom du parent, nous pouvons tranquillement ajouter le nom du fils dans la table `CHILD` , et valider notre transaction. Cet exemple montre comment implenter l'intégrité référentielle dans votre application.Ajoutons un autre exemple : nous avons un champs compteur dans la table `CHILD_CODES` que nous utilisons pour assigner un identifiant unique à chaque enfant que nous ajoutons dans la table `CHILD` . Evidemment, en utilisant une lecture cohérente ou une lecture partagée pour lire la valeur courante du compteur n'est pas une bonne idée, car deux utilisateurs de la base peuvent simultanément lire la même valeur de compteur, et nous allons obtenir une erreur de clé doublon lorsque nous ajouterons le second des deux fils.

Ici, `LOCK IN SHARE MODE` n'est pas une bonne solutionm, car si deux utilisateurs lisent le compteur en même temps, au moins l'un des deux sera bloqué lorsqu'il tentera de modifier le compteur.

Dans ce cas, il y a deux bonnes méthodes pour implémenter la lecture et l'incrémentation du compteur : (1) modifiez le compteur d'une unité, et lisez le après cela ou (2) lisez le compteur d'abord, avec un verrou en mode `FOR UPDATE` , puis incrémentez le :

```
SELECT COUNTER_FIELD FROM CHILD_CODES FOR UPDATE;
UPDATE CHILD_CODES SET COUNTER_FIELD = COUNTER_FIELD + 1;
```

Une commande `SELECT ... FOR UPDATE` va lire les dernières données disponibles pour chaque ligne, et pose un verrou dessus en même tant qu'il lit. De cette façon, il pose le même verrou que la commande `UPDATE` .Notez que la commande ci−dessus est simplement un exemple de fonctionnement de `SELECT ... FOR UPDATE` . En MySQL, la tâche spécifique de création d'un identifiant unique peut être réalisée avec un seul accès à la table :

```
UPDATE child_codes
      SET counter_field = LAST_INSERT_ID(counter_field + 1);
SELECT LAST_INSERT_ID();
```

La comande `SELECT` ne fait que lire l'identifiant spécifique à la connexion. Il ne fait aucun accès à la table.

16.11.5 Verrou de clé suivante : éviter le problème des lignes fantômes

Avec le verrouillage de ligne, `InnoDB` utilise un algorithme appelé le verrouillage de la clé suivante. `InnoDB` fait un verrouillage de telle sorte que lorsqu'il fait une recherche ou un scan d'index, il pose des verrous partagés ou exclusifs sur les lignes d'index qu'il rencontre. Par conséquent, les verrous de lignes sont plus exactement appelés des verrous d'index.

Les verrous que `InnoDB` posent affectent aussi l'espace qui le sépare de la ligne suivante. Si un utilisateur a un verrou partagé ou exclusif sur une ligne L dans un index, alors un autre utilisateur ne peut faire d'insertion immédiatement avant la ligne L, dans l'ordre de l'index. Ce verrouillage est fait pour éviter le problème de la ligne fantôme. Supposons que je veuille lire et verrouiller tous les enfants ayant un identifiant supérieur à 100 dans la table `CHILD` , puis modifier certains champs des lignes ainsi identifiées :

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

Supposons qu'il y ait un index sur la table CHILD , sur la colonne ID . Notre requête va scanner l'index à partir de la première ligne où ID est plus grand que 100. Maintenant, si le verrou posé sur les lignes d'index n'empêche pas l'utilisation des intervalles entre les lignes d'index, un nouvel enfant peut être inséré dans la table durant la lecture. Et maintenant, si ma transaction exécute la commande :

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

je vais trouver un nouvel enfant dans le résultat de ma requête. Ceci va à l'encontre du principe d'isolation des transactions : une transaction doit être capable de s'exécuter sans que les lectures soient affectées durant ce temps. Si vous considérons un intervalle de données, alors la nouvelle ligne 'fantôme' va casser le principe d'isolation.Lorsque InnoDB scanne un index, il va aussi verrouille l'espace après la dernière ligne de l'index. C'est ce qui est arrivé dans l'exemple ci−dessus : le verrou posé par InnoDB va éviter qu'une insertion n'intervienne dans la table où ID serait plus grand que 100.

Vous pouvez utiliser le verrouillage de la clé suivant pour implémenter des vérifications d'unicité dans votre application : si vous lisez des données en mode partagé, et que vous ne voyez pas de duplicata de la ligne que vous allez insérer, alors vous pouvez l'insérer sans problème, en sachant que le verrou de clé suivante va vous garantir que durant ce temps, personne ne pourra insérer de ligne, qui déboucherait sur un duplicata de la votre. Le verrou de clé suivante permet de verrouiller aussi la non−existence de ligne dans votre table.

16.11.6 Un exemple de lecture cohérente avec InnoDB

Supposons que vous exécutez MySQL avec le niveau d'isolation des transactions de REPEATABLE READ . Lorsque vous demandez une lecture cohérente avec une commande SELECT ordinaire, InnoDB va donner à votre transaction une date jalon, en fonction de laquelle votre requête va voir la base. Ainsi, si une transaction B efface une ligne après l'assignation de ce jalon, vous ne verrez pas cette ligne. De même pour une insertion ou une modification.

Vous pouvez faire avancer votre date jalon en validant votre transaction et en exécutant un autre SELECT .

Cela s'appelle le contrôle simultané multi−versions.

```
                   User A                User B

               SET AUTOCOMMIT=0;      SET AUTOCOMMIT=0;
time
|             SELECT * FROM t;
|             empty set
|                                    INSERT INTO t VALUES (1, 2);
|
v             SELECT * FROM t;
              empty set
                                     COMMIT;

               SELECT * FROM t;
              empty set;

               COMMIT;

               SELECT * FROM t;
              ---------------------
              |    1    |    2    |
              ---------------------
```

De cette façon, A voir la ligne insérée par B uniquement lorsque B a validé son insertion, et que A a validé sa propre transaction, de façon à ce que la date jalon soit plus récente que la validation de B.Si vous voulez avoir une vision aussi ``fraîche'' que possible de votre base, vous devez utiliser le verrou en lecture :

```
SELECT * FROM t LOCK IN SHARE MODE;
```

16.11.7 Les verrous posés par différentes requêtes SQL avec InnoDB

Un verrou de lecture, une commande `UPDATE` ou `DELETE` pose généralement des verrous sur toutes les lignes qui sont analysée durant l'opération. La présence d'une clause `WHERE` n'a pas d'importance. `InnoDB` ne se souviens pas de la condition `WHERE` exacte, mais sait quel intervalles d'index ont été scannés. Les verrous sont du type 'prochaine clé', et cela empêche aussi les insertions dans l'``espace'' immédiatement après la ligne.

Si le verrou est exclusif, alors `InnoDB` essaie de lire les groupes d'index et de poser un verrou dessus.

Si vous n'avez d'index valable pour votre requête, et que MySQL scanne toute la table, chaque ligne sera verrouillée, et bloquera ainsi toutes les insertions. Il est important de bien configurer ses index, poru que les requêtes ne fasse pas de scan de table inutiles.

- `SELECT ... FROM ...` : ceci est une lecture cohérente, qui lit un bilan de la base, et ne pose aucun verrou.
- `SELECT ... FROM ... LOCK IN SHARE MODE` : pose un verrou partagé sur la prochaine clé sur tous les index que la lecture rencontre.
- `SELECT ... FROM ... FOR UPDATE` : pose un verrou exclusif sur la prochaine clé sur tous les index que la lecture rencontre.
- `INSERT INTO ... VALUES (...)` : pose un verrou exclusif sur la ligne insérée. Notez que ce verrou n'est pas un verrou de clé, et il n'empêche pas les autres utilisateurs d'insérer des lignes. Si une erreur de clé double apparaît, un verrou sera posé partagé sera posé sur la ligne doublon.
- Durant l'initialisation d'une colonne `AUTO_INCREMENT` dans une table, `InnoDB` pose un verrou exclusif à la fin de l'index associé à la colonne `AUTO_INCREMENT` . Lors de l'accession au compteur d'incrémentation, `InnoDB` utilise un verrou spécifique, en mode `AUTO-INC` où le verrou ne dure que jusqu'à la fin de la requête SQL courante, au lieu de la fin de la transaction. `InnoDB` et `AUTOCOMMIT` .

  Avant MySQL 3.23.50, `SHOW TABLE STATUS` posait aussi un verrou exclusif sur les tables ayant une colonne `AUTO_INCREMENT` . Cela signifie que la commande `SHOW TABLE STATUS` pouvait aussi causer un blocage de verrou, ce qui surprenait beaucoup les utilisateurs. Depuis MySQL 3.23.50, `InnoDB` lit la valeur d'une table dont la colonne `AUTO_INCREMENT` a été initialisée, sans poser de verrou.
- `INSERT INTO T SELECT ... FROM S WHERE ...` : pose un verrou exclusif sur chaque ligne inséré dans `T` . Effectue la recherche sur `S` sous la forme d'une lecture cohérente, mais pose un verrou partagé sur l'index de prochaine clé de `S` si MySQL a activé le log. `InnoDB` doit poser un verrou dans cette dernière situation, car en cas d'exécution des instructions dans une phase de restauration, toutes les requêtes doivent être exécutées dans le même ordre.
- `CREATE TABLE ... SELECT ...` effectue une commande `SELECT` sous la forme d'une lecture cohérente, ou avec des verrous partagés, comme précédemment.
- `REPLACE` est similaire à une insertion, si il n'y a pas de collision sur la clé unique. Sinon, une verrou exclusif sur l'index de prochaine clé est posé sur la ligne qui sera modifiée.
- `UPDATE ... SET ... WHERE ...` : pose un verrou exclusif sur l'index de prochaine clé, à chaque ligne que la recherche trouve.
- `DELETE FROM ... WHERE ...` : pose un verrou exclusif sur l'index de prochaine clé à chaque ligne que la recherche trouve.
- Si la contrainte de `FOREIGN KEY` est définie sur une table, toute insertion, modification ou effacement qui requiert la vérification de la contrainte va poser un verrou de ligne sur la ligne dont il doit vérifier la contrainte. De plus, dans certains cas où la contrainte échoue, `InnoDB` pose ces verrous.
- `LOCK TABLES ...` : pose un verrou de table. L'implémentation de la couche MySQL pose ce verrou. La détection automatique des blocages de `InnoDB` ne peut détecter les blocages lorsque de tels verrous sont posés. Voyez la section suivante. Détection des blocages et annulation .De plus, comme MySQL ne connaît pas le verrouillage de lignes, il est possible que vous posiez un verrou sur une table où un autre utilisateur a déjà posé un verrou. Mais

cela ne pose pas de problème quant à l'intégrité de la requête. Restrictions sur les tables `InnoDB` .

16.11.8 Quand est−ce que MySQL valide ou annule implicitement une transaction?

MySQL ouvre les connexions des clients en mode d'auto−validation, par défaut. Lorsque l'auto−validation est activée, MySQL fait une validation après chaque commande SQL, si la commande n'a pas retourné d'erreur.

Si vous n'avez pas de mode d'auto−validation, et que vous fermez une connexion sans valider explicitement vos transactions, alors MySQL annule votre transaction.

Si une erreur est retournée par une commande SQL, le comportement de la transaction dépend de l'erreur. Restrictions sur les tables `InnoDB` .

Les commandes SQL suivantes causent une validation implicite de la transaction courante :

- `ALTER TABLE` , `BEGIN` , `CREATE INDEX` , `DROP DATABASE` , `DROP INDEX` , `DROP TABLE` , `LOAD MASTER DATA` , `LOCK TABLES` , `RENAME TABLE` , `SET AUTOCOMMIT=1` , `START TRANSACTION` , `TRUNCATE` , `UNLOCK TABLES` .
- `CREATE TABLE` (elle valide uniquement avant MySQL 4.0.13 et si le log binaire MySQL est utilisé).
- La commande `CREATE TABLE` de `InnoDB` est traitée comme une seule transaction. Cela signifie qu'une commande `ROLLBACK` ne va pas annuler la commande `CREATE TABLE` qui a été faite dans la transaction.

16.11.9 Détection des blocages et annulation

`InnoDB` détecte automatiquement les blocages de transactions et annule une ou plusieurs transactions pour l'éviter. Depuis la version 4.0.5, `InnoDB` va essayer d'annuler les petites transactions. La taille de la transaction est déterminée par le nombre de lignes qu'elle a inséré, modifié ou effacé. Avant la version 4.0.5, `InnoDB` annulait toujours la transaction qui avait posé le dernier verrou avant le blocage, c'est à dire, un cycle dans le graphe des transactions.

`InnoDB` ne peut pas détecter les blocages causés par la commande MySQL `LOCK TABLES` , ou si un verrou est posé par un autre gestionnaire de table que `InnoDB` . Vous devez résoudre ces situations avec l'option `innodb_lock_wait_timeout` du fichier de configuration.

Lorsque `InnoDB` effectue une annulation de transaction, tous les verrous de cette transaction sont libérés. Cependant, si une commande SQL est annulée pour cause d'erreur, certains verrous de la transaction peuvent être conservés. Deci est dû au fait que `InnoDB` enregistre les verrous dans un format qui ne permet pas de savoir qui l'a posé.

16.11.10 Comment gérer les blocages de verrous?

Les blocages de verrous sont un problème classique des bases de données transactionnelles, mais ils ne sont pas dangereux, à moins qu'ils ne se répètent si souvent que vous ne puissiez pas exécuter tranquillement certaines transactions. Normalement, vous devriez écrire vos applications de manière à ce qu'elles soient prêtes à tenter à nouveau une transaction si la transaction est annulée pour cause de blocage.

`InnoDB` utilise un verrouillage de lignes automatique. Vous pouvez obtenir des blocages sur une ligne, même si votre transactions ne fait que modifier ou insérer une seule ligne. Cela est dû au fait que les opérations ne sont pas réellement 'atomiques' : elles posent automatiquement des verrous (éventuellement plusieurs) sur les lignes d'index de l'enregistrement concerné.

Vous pouvez gérer ces blocages et réduire leur nombre avec les trucs suivants :

- Utilisez la commande `SHOW INNODB STATUS` avec MySQL version supérieure à 3.23.52 et 4.0.3, pour déterminer la cause du dernier blocage. Cela peut vous aider à optimiser votre application pour les éviter.
- Soyez toujours prêts à tenter une nouvelle fois une transaction si elle échoue à cause d'un blocage. Les verrous ne sont pas dangeureux. Essayez juste une autre fois.
- Validez souvent vos transactions. Les petites transactions sont moins sujettes aux blocages.
- Si vous utilisez des verrous en lectures avec `SELECT ... FOR UPDATE` ou `... LOCK IN SHARE MODE` , essayez d'utiliser un niveau d'isolation plus bas comme `READ COMMITTED` .
- Accédez à vos tables et lignes dans un ordre fixé. Les transactions vont alors former des queues, et non pas des blocages.
- Ajoutez de bons index à vos tables. Vos requêtes devront scanner moins souvent les tables, et poseront donc moins de verrous. Utilisez `EXPLAIN SELECT` pour déterminer si MySQL choisit les bons index pour vos requêtes.
- Limitez votre utilisation des verrous. Si vous pouvez vous permettre de faire retourner à une commande `SELECT` des données un peu anciennes, n'ajoutez pas la clause `FOR UPDATE` ou `LOCK IN SHARE MODE` . Utiliser le niveau d'isolation `READ COMMITTED` est bon ici, car chaque lecture cohérente dans la même transaction lira avec des données aussi fraîches que possible à chaque fois.
- En dernier recours, vous pouvez forcer les verrous avec la commande :

  ```
   LOCK TABLES t1 WRITE, t2 READ, ...;
  [faire quelquechose avec les tables t1 et t2];
   UNLOCK TABLES;
  ```

  Les verrous de niveau de table forcent les transactions à se mettre en ligne, et les blocages sont évités. Notez que `LOCK TABLES` démarre implicitement une transaction, tout comme `BEGIN` , et `UNLOCK TABLES` termine une transaction avec un `COMMIT` .
- Une dernière solution est de créer un sémaphore auxiliaire sous la forme d'une table avec une seule ligne. Chaque transaction modifie cette table avant d'accéder aux autres tables. Dans ce cas, toutes les transactions se font en ordre séquentiel. Notez que dans cette configuration, même l'algorithme `InnoDB` de détection des blocages fonctionne, car le sémaphore est un verrou de ligne. Avec les verrous de niveau de table de MySQL, nous devons nous résoudre à une méthode de délai d'expiration pour résoudre un verrou.

16.12 Conseils pour l'amélioration des performances InnoDB

- Si l'outil Unix `top` ou si le `Task Manager` de Windows montre que l'utilisation du CPU, lors de la charge de travail, est inférieure à 70%, votre calcul est probablement limité par les disques. Vous faites peut être trop d'écriture de transaction, ou le tampon de traitement ("buffer pool") est peut être trop petit. Augmenter la taille du tampon peut aider, mais il ne faut pas qu'il dépasse 80% de la mémoire physique.
- Regrouper les modifications dans une seule transaction. `InnoDB` doit écrire les données de log sur le disque à chaque validation de transaction, si la transaction a fait des modifications dans la base. Comme la vitesse de rotation d'un disque est typiquement de 167 revolutions/seconde au mieux, cela réduit le nombre de validations à 167 par seconde, si le disque ne veut pas induire le système d'exploitation en erreur.
- Si vous pouvez accepter la perte des toutes dernières transactions validées, vous pouvez modifier dans le fichier `my.cnf` le paramètre `innodb_flush_log_at_trx_commit` à 0. `InnoDB` essaie d'écrire sur

le disque au moins une fois par seconde, mais cette écriture n'est plus garantie.

- Utilisez de gros fichiers de log, même aussi grand que le pool de buffer. Lorsque `InnoDB` a écrit les fichiers de log, il doit écrire les contenus modifiés du pool de buffer sur le disque, avec un jalon. Les fichiers de logs de petites tailles imposeront des écritures inutiles. L'inconvénient d'un gros fichier de log est que le temps de restauration sera plus long.
- Le buffer de logs doit être assez grand, au moins 8 Mo.
- Utilisez le type de colonne `VARCHAR` au lieu de `CHAR` si vous stockez des chaînes de taille variable, ou si une colonne peut contenir plusieurs valeurs `NULL` . Une colonne `CHAR(N)` prend toujours `N` octets pour stocker les données, même si la chaîne est plus petite, ou que sa valeur est `NULL` . Des tables plus petites rentrent plus facilement dans les buffers et réduisent les accès disques.
- (Valable depuis MySQL version 3.23.39 et plus récent) Dans certaines versions de MySQL et Unix, l'écriture des fichiers sur les disques avec `fdatasync` et d'autres commandes similaires sont étonnament lentes. La méthode par défaut de `InnoDB` est la fonction `fdatasync` . Si vous n'êtes pas satisfaits avec les performances en écriture de la base, vous pouvez essayer d'utiliser l'option `innodb_flush_method` dans le fichier `my.cnf` avec la valeur `O_DSYNC` , même si `O_DSYNC` semble être la méthode la plus lente sur la plupart des systèmes.
- Lors de l'importation de données avec `InnoDB` , assurez vous que MySQL n'a pas `autocommit=1` . Sinon, chaque insertion implique une écriture sur le disque. Ajoutez cette commande dans votre fichier SQL d'import :

```
SET AUTOCOMMIT=0;
/* commandes d'importation SQL ... */
COMMIT;
```

  Si vous utilisez l'option `--opt` de l'utilitaire `mysqldump` , vous allez obtenir des fichiers d'export qui sont rapides à importer dans une table `InnoDB` , même sans utiliser l'astuce de la transaction ci−dessus : `SET AUTOCOMMIT=0; ... COMMIT;` .
- Attention aux annulations lors des insertions de masse : `InnoDB` utilise une buffer d'insertion pour éviter les accès disques, mais la fonction d'annulation correspondante n'utilise pas ce mécanisme. Une annulation qui utilise beaucoup d'accès disque est environ 30 fois plus lente que l'insertion équivalent. Interrompre la base ne va pas aider, car l'annulation reprendra lors du redémarrage de la base. La seule solution pour ce débarasser de cette annulation lénifiante est d'augmenter la taille du pool de buffer, pour que l'annulation soit limitée par le processeur et non plus par le disque. Ou alors, effacez toute la base `InnoDB` . Forcer la restauration .
- Attention aux opérations qui ont de gros impacts sur le disque : utilisez `DROP TABLE` ou `TRUNCATE` depuis MySQL 4.0 et, pour vider une table, et non pas `DELETE FROM votre_table` .
- Utilisez les `INSERT` multiples pour réduire les communications entre le client et le serveur, si vous devez insérer plusieurs lignes :

```
INSERT INTO yourtable VALUES (1, 2), (5, 5);
```

  Ce conseil est valable pour tous les types des tables, pas seulement `InnoDB` .
- Si vous avez une contrainte `UNIQUE` sur les clés secondaires, depuis MySQL version 3.23.52 et 4.0.3, vous pouvez accélérer les imports de tables en désactivant temporairement les vérifications d'unicité durant l'importation :

```
SET UNIQUE_CHECKS=0;
```

  Pour les grandes tables, cela économise de nombreux accès disques, car les tables `InnoDB` peuvent utiliser le buffer d'insertion pour écrire des index secondaires par paquets.
- Si vous avez des contraîntes `FOREIGN KEY` dans vos tables, depuis MySQL 3.23.52 et 4.0.3, vous pouvez accélérer les imports de tables en désactivant temporairement les vérifications d'unicité durant l'importation :

```
SET FOREIGN_KEY_CHECKS=0;
```

  Pour les grandes tables, cela économise beaucoup d'accès disques.
- Si vous avez des requêtes récurrentes dans vos tables, qui ne sont pas modifiées souvent, vous pouvez utiliser le cache de requêtes depuis MySQL 4.0 :

```
[mysqld]
query_cache_type = ON
query_cache_size = 10M
```

En MySQL 4.0, le cache temporaire fonctionne uniquement si l'auto−commit est activé. Cette restriction a été levée depuis MySQL 4.1.1.

16.12.1 Le moniteur InnoDB

Depuis la version 3.23.42, `InnoDB` inclut le moniteur `InnoDB` (`InnoDB Monitor`) qui affiche des informations sur l'état interne des tables `InnoDB` . Depuis les versions 3.23.52 et 4.0.3 vous pouvez aussi utiliser la commande SQL `SHOW INNODB STATUS` pour lire les informations de `InnoDB Monitor` depuis le client SQL. Les données lue sont importantes pour les réglages de performances. Si vous utilisez le client `mysql` interactif, le résultat est plus lisible si vous remplacez le point virgule classique par `\G` :

```
SHOW INNODB STATUS\G
```

Une autre méthode pour utiliser le `InnoDB Monitors` est de le laisser écrit continuellement des données dans la sortie de `mysqld` : (note : le client MySQL n'affichera rien). Lorsqu'il est activé, `InnoDB` peut afficher des données toutes les 15 secondes. Si vous utilisez `mysqld` comme démon, alors ces données sont dirigées vers le fichier de log `.err` dans le dossier `datadir` . Les données lue sont importantes pour les réglages de performances. Sous Windows, vous devez lancer `mysqld-max` depuis une fenêtre MS−DOS, avec l'option `--console` , si vous voulez voir le résultat affiché dans la fenête.

Il existe aussi `innodb_lock_monitor` qui affiche les mêmes informations que `innodb_monitor` , mais qui indique aussi les verrous posés par les transactions.

- Les verrous de tables et de ligne de chaque transaction,
- les attentes de verrous par les transactions,
- Les attentes de sémaphores pour les threads,
- les requêtes en attente d'accès aux fichiers,
- les statistiques sur les buffers, et
- les statistiques sur les purges et buffers d'insertion sur le thread principal `InnoDB` .

Vous pouvez démarrer le moniteur `InnoDB` avec la commande SQL suivante :

```
CREATE TABLE innodb_monitor(a int) type = innodb;
```

et le stopper avec

```
DROP TABLE innodb_monitor;
```

La syntaxe `CREATE TABLE` est simplement un moyen de passer une commande à une table `InnoDB` , via l'analyseur MySQL : la table créée n'est pas importante pour le moniteur `InnoDB` . Si vous interrompez le serveur lorsque le moniteur fonctionne, et que vous voulez redémarrer le moniteur, vous devez alors effacer la table avant de pouvoir exécuter la même commande `CREATE TABLE` pour démarrer le moniteur. Cette syntaxe est susceptible de changer dans le futur. Un exemple de résultat du moniteur `InnoDB` :

```
 mysql> SHOW INNODB STATUS\G
*************************** 1. row ***************************
Status:
=====================================
030709 13:00:59 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 18 seconds
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 413452, signal count 378357
--Thread 32782 has waited at btr0sea.c line 1477 for 0.00 seconds the semaphore:
X-lock on RW-latch at 41a28668 created in file btr0sea.c line 135
a writer (thread id 32782) has reserved it in mode wait exclusive
number of readers 1, waiters flag 1
Last time read locked in file btr0sea.c line 731
Last time write locked in file btr0sea.c line 1347
Mutex spin waits 0, rounds 0, OS waits 0
```

```
RW-shared spins 108462, OS waits 37964; RW-excl spins 681824, OS waits 375485
------------------------
LATEST FOREIGN KEY ERROR
------------------------
030709 13:00:59 Transaction:
TRANSACTION 0 290328284, ACTIVE 0 sec, process no 3195, OS thread id 34831 inser
ting
15 lock struct(s), heap size 2496, undo log entries 9
MySQL thread id 25, query id 4668733 localhost heikki update
insert into ibtest11a (D, B, C) values (5, 'khDk' ,'khDk')
Foreign key constraint fails for table test/ibtest11a:
,
  CONSTRAINT `0_219242` FOREIGN KEY (`A`, `D`) REFERENCES `ibtest11b` (`A`, `D`)
 ON DELETE CASCADE ON UPDATE CASCADE
Trying to add in child table, in index PRIMARY tuple:
 0: len 4; hex 80000101; asc ....;; 1: len 4; hex 80000005; asc ....;; 2: len 4;
 hex 6b68446b; asc khDk;; 3: len 6; hex 0000114e0edc; asc ...N..;; 4: len 7; hex
 00000000c3e0a7; asc .......;; 5: len 4; hex 6b68446b; asc khDk;;
But in parent table test/ibtest11b, in index PRIMARY,
the closest match we can find is record:
RECORD: info bits 0 0: len 4; hex 8000015b; asc ...[;; 1: len 4; hex 80000005; a
sc ....;; 2: len 3; hex 6b6864; asc khd;; 3: len 6; hex 0000111ef3eb; asc ......
;; 4: len 7; hex 800001001e0084; asc .......;; 5: len 3; hex 6b6864; asc khd;;
------------------------
LATEST DETECTED DEADLOCK
------------------------
030709 12:59:58
*** (1) TRANSACTION:
TRANSACTION 0 290252780, ACTIVE 1 sec, process no 3185, OS thread id 30733 inser
ting
LOCK WAIT 3 lock struct(s), heap size 320, undo log entries 146
MySQL thread id 21, query id 4553379 localhost heikki update
INSERT INTO alex1 VALUES(86, 86, 794,'aA35818','bb','c79166','d4766t','e187358f'
,'g84586','h794',date_format('2001-04-03 12:54:22','%Y-%m-%d %H:%i'),7
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 0 page no 48310 n bits 568 table test/alex1 index symbole
trx id 0 290252780 lock mode S waiting
Record lock, heap no 324 RECORD: info bits 0 0: len 7; hex 61613335383138; asc a
a35818;; 1:
*** (2) TRANSACTION:
TRANSACTION 0 290251546, ACTIVE 2 sec, process no 3190, OS thread id 32782 inser
ting
130 lock struct(s), heap size 11584, undo log entries 437
MySQL thread id 23, query id 4554396 localhost heikki update
REPLACE INTO alex1 VALUES(NULL, 32, NULL,'aa3572','','c3572','d6012t','', NULL,'
h396', NULL, NULL, 7.31,7.31,7.31,200)
*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 0 page no 48310 n bits 568 table test/alex1 index symbole
trx id 0 290251546 lock_mode X locks rec but not gap
Record lock, heap no 324 RECORD: info bits 0 0: len 7; hex 61613335383138; asc a
a35818;; 1:
*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 0 page no 48310 n bits 568 table test/alex1 index symbole
trx id 0 290251546 lock_mode X locks gap before rec insert intention waiting
Record lock, heap no 82 RECORD: info bits 0 0: len 7; hex 61613335373230; asc aa
35720;; 1:
*** WE ROLL BACK TRANSACTION (1)
------------
TRANSACTIONS
------------
Trx id counter 0 290328385
Purge done for trx's n:o < 0 290315608 undo n:o < 0 17
Total number of lock structs in row lock hash table 70
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0 0, not started, process no 3491, OS thread id 42002
MySQL thread id 32, query id 4668737 localhost heikki
show innodb status
---TRANSACTION 0 290328384, ACTIVE 0 sec, process no 3205, OS thread id 38929 in
serting
1 lock struct(s), heap size 320
MySQL thread id 29, query id 4668736 localhost heikki update
insert into speedc values (1519229,1, 'hgjhjgghggjgjgjgjgjggjgjgjgjgjgggjgjgjlhh
gghggggghhjhghgggggghjhghghghghghhhhghghghjhhjghjghjkghjghjghjghjfhjfh
---TRANSACTION 0 290328383, ACTIVE 0 sec, process no 3180, OS thread id 28684 co
mmitting
1 lock struct(s), heap size 320, undo log entries 1
MySQL thread id 19, query id 4668734 localhost heikki update
insert into speedcm values (1603393,1, 'hgjhjgghggjgjgjgjgjggjgjgjgjgjgggjgjgjlh
hgghggggghhjhghgggggghjhghghghghghhhhghghghjhhjghjghjkghjghjghjghjfhjf
---TRANSACTION 0 290328327, ACTIVE 0 sec, process no 3200, OS thread id 36880 st
arting index read
LOCK WAIT 2 lock struct(s), heap size 320
MySQL thread id 27, query id 4668644 localhost heikki Searching rows for update
update ibtest11a set B = 'kHdkkkk' where A = 89572
------- TRX HAS BEEN WAITING 0 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 0 page no 65556 n bits 232 table test/ibtest11a index PRIM
ARY trx id 0 290328327 lock_mode X waiting
Record lock, heap no 1 RECORD: info bits 0 0: len 9; hex 73757072656d756d00; asc
 supremum.;;
------------------
---TRANSACTION 0 290328284, ACTIVE 0 sec, process no 3195, OS thread id 34831 ro
llback of SQL statement
```

```
ROLLING BACK 14 lock struct(s), heap size 2496, undo log entries 9
MySQL thread id 25, query id 4668733 localhost heikki update
insert into ibtest11a (D, B, C) values (5, 'khDk' ,'khDk')
---TRANSACTION 0 290327208, ACTIVE 1 sec, process no 3190, OS thread id 32782
58 lock struct(s), heap size 5504, undo log entries 159
MySQL thread id 23, query id 4668732 localhost heikki update
REPLACE INTO alex1 VALUES(86, 46, 538,'aa95666','bb','c95666','d9486t','e200498f
','g86814','h538',date_format('2001-04-03 12:54:22','%Y-%m-%d %H:%i'),
---TRANSACTION 0 290323325, ACTIVE 3 sec, process no 3185, OS thread id 30733 in
serting
4 lock struct(s), heap size 1024, undo log entries 165
MySQL thread id 21, query id 4668735 localhost heikki update
INSERT INTO alex1 VALUES(NULL, 49, NULL,'aa42837','','c56319','d1719t','', NULL,
'h321', NULL, NULL, 7.31,7.31,7.31,200)
--------
FILE I/O
--------
I/O thread 0 state: waiting for i/o request (insert buffer thread)
I/O thread 1 state: waiting for i/o request (log thread)
I/O thread 2 state: waiting for i/o request (read thread)
I/O thread 3 state: waiting for i/o request (write thread)
Pending normal aio reads: 0, aio writes: 0,
 ibuf aio reads: 0, log i/o's: 0, sync i/o's: 0
Pending flushes (fsync) log: 0; buffer pool: 0
151671 OS file reads, 94747 OS file writes, 8750 OS fsyncs
25.44 reads/s, 18494 avg bytes/read, 17.55 writes/s, 2.33 fsyncs/s
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf for space 0: size 1, free list len 19, seg size 21,
85004 inserts, 85004 merged recs, 26669 merges
Hash table size 207619, used cells 14461, node heap has 16 buffer(s)
1877.67 hash searches/s, 5121.10 non-hash searches/s
---
LOG
---
Log sequence number 18 1212842764
Log flushed up to   18 1212665295
Last checkpoint at  18 1135877290
0 pending log writes, 0 pending chkp writes
4341 log i/o's done, 1.22 log i/o's/second
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 84966343; in additional pool allocated 1402624
Buffer pool size   3200
Free buffers       110
Database pages     3074
Modified db pages  2674
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages read 171380, created 51968, written 194688
28.72 reads/s, 20.72 creates/s, 47.55 writes/s
Buffer pool hit rate 999 / 1000
--------------
ROW OPERATIONS
--------------
0 queries inside InnoDB, 0 queries in queue
Main thread process no. 3004, id 7176, state: purging
Number of rows inserted 3738558, updated 127415, deleted 33707, read 755779
1586.13 inserts/s, 50.89 updates/s, 28.44 deletes/s, 107.88 reads/s
----------------------------
END OF INNODB MONITOR OUTPUT
============================
1 row in set (0.05 sec)
```

Quelques notes sur le résultat :

- Si la section `TRANSACTIONS` rapporte des attentes de verrous, alors votre application a des problèmes de rétention de verrous. Le rapport peut vous aider à trouver les blocages de verrous, et les transactions bloquantes.
- La section `SEMAPHORES` rapporte les threads en attente d'un sémaphore, et les statistiques d'attentes pour un mutex ou un verrou en lecture/écriture. Les longues attentes peuvent être dues à des requêtes paralèlles, ou des problèmes avec la programmation des threads. Donner à la variable `innodb_thread_concurrency` une valeur plus petite que le `8` par défaut peut réduire ce type de situations.
- La section `BUFFER POOL AND MEMORY` vous donne des statistiques sur les pages qui ont été lues et écrites. Vous pouvez calculer depuis ces nombres combien d'accès disques vos requêtes réalisent actuellement.
- La section `ROW OPERATIONS` montre ce que le thread principal fait.

16.13 Implémentation du multi−versionnage

Comme InnoDB est une base de donnes multi−versionnée, elle doit conserver des informations des vieilles versions des lignes dans l'espace de tables. Cette information est stockée dans la structure de données que nous appelons un segment d'annulation, d'après la structure similaire d'Oracle.

En interne, InnoDB ajoute deux champs à chaque ligne stockée dans la base. Un champ de 6 octets note l'identifiant de la dernière transaction qui a inséré ou modifier la ligne. De plus, un effacement est traité en interne comme une modification, où un bit spécial dans la ligne sert à marquer la ligne comme effacée. Chaque ligne contient aussi une colonne de 7 octets appelé un pointeur d'annulation. Ce pointeur fait référence à une ligne dans un fichier d'annulation qui contient les informations nécessaires pour reconstruire le contenu de la ligne qui a été modifiée.

InnoDB utilise les informations dans le segment d'annulation pour effectuer les opérations d'annulation nécessaires dans une annulation de transaction. Il utilise aussi ces informations pour reconstruire les anciennes versions d'une ligne lors d'une lecture cohérente.

Le log d'annulation dans le segment d'annulation est divisé entre les insertions et les modifications. Le log d'insertion n'est nécessaire que lors des annulations de transactions, et peut être vidé dès que la requête est validée. Le log de modification est utilisé lors des lectures cohérentes, et les informations y sont supprimées une fois que toutes les transactions qui font une lecture cohérente sont terminées.

Vous devez vous rappeler que valider vos transactions régulièrement, même ces transactions qui ne font que des lectures cohérentes. Sinon, InnoDB ne peut pas vider le log de modification, et le segment d'annulation va grossir énormément.

La taille physique d'une ligne du log d'annulation dans le segment d'annulation est typiquement plus petite que la ligne correspondante insérée ou modifiée. Vous pouvez utiliser ces informations pour calculer l'espace nécessaire à vos segments d'annulation.

Dans nos schémas de multi−versionnage, une ligne n'est pas physiquement supprimée de la table immédiatement lorsque vous l'effacez avec une requête SQL. Uniquement lorsque InnoDB va supprimer les lignes du log de modification, il vaut aussi supprimer physiquement la ligne, et les index. Cette opération d'effacement est appelée une purge, et elle est plutôt rapide, et aussi rapide que la requête SQL elle−même.

16.14 Structures de tables et d'index

MySQL enregistre la structure de table dans le fichier .frm , du dossier de données. Mais les tables InnoDB ont aussi leur propre entrée dans les tables internes InnoDB . Lorsque MySQL efface une table ou une base, il efface le ou les fichiers .frm et aussi les lignes correspondantes dans les tables d'administration InnoDB . C'est la raison qui fait que vous ne pouvez pas déplacer les tables InnoDB entre les bases simplement en déplaçant le fichier .frm , et pourquoi DROP DATABASE ne fonctionnait pas sous InnoDB en MySQL versions <= 3.23.43.

Chaque table InnoDB a un index spécial appelé un index en grappe, où les données des lignes sont enregistrées. Si vous définissez une clé primaire pour votre table PRIMARY KEY , alors l'index de la clé primaire de la table sera un index en grappe.

Si vous ne définissez pas de clé primaire pour votre table, InnoDB va générer un index en grappe, ou les lignes sont ordonnées dans l'ordre des identifiants que InnoDB assigne aux lignes de la table.

L'identifiant de ligne vaut 6 octets, et s'accroît au fur et à mesure que les lignes sont ajoutées. Les lignes sont alors ordonnées dans leur ordre d'insertion.

Accéder à une ligne via l'index en grappe est rapide, car la ligne de données sera dans la même page que l'index. Dans de nombreuses bases, les données sont traditionnellement stockées dans un autre endroit. Si la table est grand, l'index en grappe économise de nombreux accès disques, comparativement aux solutions traditionnelles.

Les lignes des index qui ne sont pas en grappe (ce sont les index secondaires) dans `InnoDB` , contiennent la valeur de la clé primaire de la ligne. `InnoDB` utilise cette clé primaire pour rechercher la valeur de la ligne dans l'index en grappe. Notez que si la clé primaire est longue, les index secondaires utiliseront plus de place.

`InnoDB` compare les chaînes `CHAR` et `VARCHAR` de différentes longues en complétant la chaîne la plus courte avec des espaces.

16.14.1 Structure physique d'un index

Tous les index de `InnoDB` sont des index `B-tree` où les lignes d'index sont stockées dans un noeud terminal de l'arbre. La taille par défaut d'une page d'index est de 16ko. Lorsque de nouvelles lignes sont insérées, `InnoDB` essaie de laisser 1 / 16 de la page de libre pour les prochaines insertions et modifications dans les lignes d'index

Si les lignes d'index sont insérées dans un ordre séquentiel (croissant ou décroissant), les pages d'index résultantes seront environs pleines à 15/16. Si les lignes sont insérées dans un ordre aléatoire, les pages seront pleines de 1/2 à 15/16. Si le taux de remplissage d'une page d'index tombe à 1/2, `InnoDB` va essayer de contracter l'arbre d'index pour libérer la page.

16.14.2 Bufferisation des insertions

Une situation courante dans les applications de base de données apparaît lorsque la clé primaire est un identifiant unique, et que les nouvelles lignes sont insérées dans un ordre ascendant. Par conséquent, les insertions dans l'index en grappe ne nécessitent pas de lectures aléatoires dans le disque.

D'un autre coté, les index secondaires sont généralement non−uniques, et les insertions surviennent dans un ordre aléatoire. Cela causerait de nombreux accès disques aléatoire, si `InnoDB` de disposait pas d'un mécanisme spécial.

Si une ligne doit être insérée dans un index secondaire non unique, `InnoDB` vérifie si la page d'index fait partie du buffer. Dans ce cas, `InnoDB` va faire directement l'insertion dans une structure de buffer destinée à l'insertion. Le buffer d'insertion est conservé petit, pour qu'il reste dans le buffer général, et les insertions sont faites très vite.

Le buffer d'insertion est périodiquement fusionné avec l'arbre d'index secondaires dans la base. Souvent, nous fusionnons plusieurs insertions dans la même page de l'arbre d'index, et donc, nous économisons des accès disques. Il a été mesuré que les insertions sont jusqu'à 15 fois plus rapides de cette façon.

16.14.3 Index hash adaptatifs

Si une base de données est suffisamment petite pour tenir en mémoire, alors le plus rapide pour faire des requêtes est d'utiliser les index hash. `InnoDB` a un mécanisme automatique pour surveiller les recherches utilisant les index d'une table, et si `InnoDB` remarque que la requête pourrait profiter d'un index hash, un tel index est automatiquement constitué.

Mais notez que les index hash sont toujours bâtis à partir d'un index `B-tree` existant. `InnoDB` peut bâtir un index hash sur un préfixe de taille arbitraire de clé `B-tree` , suivant le modèle de recherche que `InnoDB` remarque dans l'index `B-tree` . Un index hash peut être partiel : il n'est pas obligatoire que tout l'index `B-tree` soit mis en cache dans le pool. `InnoDB` va bâtir des index hash à la demande pour les tables dont les index sont souvent sollicités.

En un sens, grâce au mécanisme d'index hash adaptatif, `InnoDB` s'adapte tout seul à la mémoire interne, et se rapproche des architectures de bases en mémoire vive.

16.14.4 Structure physique d'une ligne

- Chaque ligne d'index de `InnoDB` contient un entête de 6 octets. L'entête est utilisé pour lier des lignes consécutives ensembles, et aussi pour le verrouillage de ligne.
- Les lignes dans un index en grappe contient des champs pour toutes les colonnes définies par l'utilisateur. De plus, il y a un champ de 6 octets pour l'identification de transaction, et un champs de 7 octets pour le pointeur d'annulation.
- Si l'utilisateur n'a pas défini de clé primaire pour la table, chaque ligne de l'index en grappe contient aussi une colonne supplémentaire de 6 octets, qui sert d'identification.
- Chaque ligne d'index secondaire contient aussi les champs définis pour la clé de l'index en grappe.
- Une ligne contient aussi un pointeur pour chaque champs de la ligne. Si la taille totale des champs représentent moins de 128 octets, alors le pointeur fait 1 octets, sinon 2.
- En interne, `InnoDB` stocke les colonnes de taille fixe comme `CHAR(10)` dans un format à taille fixe. `InnoDB` supprime les espaces terminaux des colonnes `VARCHAR` . Notez que MySQL peut convertir en interne les colonnes `CHAR` en `VARCHAR` . Changement de type de colonne automatique .
- Une valeur SQL `NULL` réserve 0 octets si elle est stockée dans une colonne à taille variable. Dans une colonne à taille fixe, elle utilise toute la largeur de la colonne. La raison de la réservation de toute la colonne pour les valeurs `NULL` est que lors de la mise à jour de la colonne depuis la valeur `NULL` vers une valeur non−nulle, il n'y aura pas de fragementation sur le disque.

16.15 Gestion de l'espace fichiers et des entrées/sorties disque

16.15.1 Accès disques

`InnoDB` utilise des accès disques asynchrones : `InnoDB` crée quelques threads pour s'occuper des opérations de lecture, notamment les lectures anticipées.

Voici deux heuristiques de lectures anticipées de `InnoDB` :

- Dans une lecture anticipée séquentielle, si `InnoDB` remarque que l'accès à une partie de l'espace de table est séquentiel, il fait plusieurs lectures à l'avance.

- Dans une lecture aléatoire, si `InnoDB` remarque qu'une partie de l'espace de tables semble être totalement lu dans le buffer, il va aussi faire une lecture à l'avance.

Depuis la version 3.23.40b, `InnoDB` utilise une nouvelle technique de flush de fichier, appelée doublewrite. Elle apporte de la sécurité lors de la restauration après crash du système d'exploitation, ou un problème électrique, et améliore les performances sous Unix, pour plusieurs distributions, en réduisant le besoin de synchronisation.

`Doublewrite` (Double écriture, en français) signifie que `InnoDB` , avant d'écrire les pages dans le fichier de données, les écrits dans une zone continue d'espace, appelée un buffer de double écriture. Une fois que cette écriture est faite, que le buffer de double écriture a été vidé, `InnoDB` écrit les pages à leur destination finale. Si le système d'exploitation crashe entre temps, `InnoDB` va trouver une bonne copie des données dans le buffer, lors de la restauration.

16.15.2 Utiliser les raw devices pour l'espace de tables

Depuis MySQL 3.23.41, vous pouvez utiliser des partitions raw disk pour stocker les fichiers d'espace de tables. En utilisant un raw disk, vous pouvez faire des E/S non bufferisée sous Windows et sans couche de contrôle sous Unix, ce qui améliore les performances.

Lorsque vous créer un nouveau fichier de données, vous devez mettre le mot `newraw` immédiatement après la taille du fichier de données dans `innodb_data_file_path` . La partition doit être aussi grande que la taille que vous spécifiez. Notez que 1Mo avec `InnoDB` vaut 1024 * 1024 octets, alors que 1Mo signifie généralement 1000000 d'octets pour un disque.

```
 [mysqld]
innodb_data_home_dir=
innodb_data_file_path=/dev/hdd1:3Gnewraw;/dev/hdd2:2Gnewraw
```

Lors de votre prochain redémararge du serveur, `InnoDB` remarque le mot `newraw` et initialise un nouvelle partition. Cependant, ne créez ou ne modifiez pas les tables `InnoDB` pour le moment. SInon, lors de votre prochain redémarrage serveur, `InnoDB` va reinitialiser la partition et vous aurez tout perdu. Depuis la version 3.23.44, par mesure de sécurité, `InnoDB` empêche les utilisateurs de modifier des données dans une partition `newraw` .

Une fois que `InnoDB` a inialisé la nouvelle partition, stoppez le serveur, remplacez `newraw` dans le fichier d'options par `raw` :

```
 [mysqld]
innodb_data_home_dir=
innodb_data_file_path=/dev/hdd1:5Graw;/dev/hdd2:2Graw
```

Puis relancez le serveur et `InnoDB` va permettre les modifications dans cet espace de table.

Sous Windows, depuis la version 4.1.1, vous pouvez allouer une partition de disque comme ceci :

```
 [mysqld]
innodb_data_home_dir=
innodb_data_file_path=//./D::10Gnewraw
```

Le `//./` correspond à la syntaxe Windows de `\\.\` , pour accéder aux disques physiques.

Lorsque vous utilisez une partition raw disque, assurez vous que vous avez les permissions qui permettent de lire et écrire avec le compte qui fait tourner le serveur.

16.15.3 Gestion de l'espace fichier

Les fichiers de données que vous définissez dans le fichier de configuration forment l'espace de données `InnoDB` . Les fichiers sont simplement concaténés pour former un espace de données, et il n'y a pas de parallélisme utilisé. Actuellement, vous ne pouvez pas spécifier où l'espace est alloué pour vos tables, hormis en utilisant la méthode suivante : pour chaque nouvel espace de données créé, `InnoDB` va allouer de l'espace en partant depuis la fin.

L'espace de tables est constitué de pages de taille 16 ko. Les pages sont groupées en ensembles de 64 pages consécutives. Les 'fichiers' dans un espace de tables sont appelés des segments en `InnoDB` . Le nom du segment d'annulation est un peu trompeur, car il contient en fait de nombreux segments dans l'espace de table.

Pour chaque index de `InnoDB` , nous allons créer deux segments : un pour les noeuds non terminaux du B tree, et un autre pour les noeuds terminaux. L'idée ici est d'améliorer la séquence des noeuds terminaux, qui contiennent les données.

Lorsqu'un segment croit dans un espace de table, `InnoDB` alloue les 32 premières pages spécifiquement pour un segment. Après cela, `InnoDB` commence à allouer des extensions au segment. `InnoDB` peut ajouter aux grands segments jusqu'à 4 extension en même temps, pour améliorer la séquence de données.

Certaines pages dans l'espace de table contiennent des cartes des autres pages, et donc, quelques extensions dans un espace de table `InnoDB` ne pourront pas être allouées en tant que segment, mais comme pages individuelles.

Lorsque vous exécutez une commande `SHOW TABLE STATUS FROM ... LIKE ...` pour demander quel est l'espace libre dans la table, `InnoDB` va rapporter les extensions qui sont encore totalement libre. `InnoDB` se réserve toujours quelques extensions pour la gestion interne. Ces extensions réservées ne sont pas inclues dans l'espace libre.

Lorsque vous effacez des donnes d'une table, `InnoDB` va contracter les index `B-tree` correspondants. Suivant les effacements qui vont libérer des pages individuelles ou des extensions, de l'espace sera rendu aux autres tables. Effacer une table ou en effacer toutes les lignes va rendre obligatoirement de l'espace aux autres tables, mais souvenez vous que les lignes effacées ne pourront être physiquement effacée qu'après une opération de purge, si aucune transaction n'en a plus besoin.

16.15.4 Défragmentation des tables

S'il y a plusieurs insertions et suppressions dans les index d'une table, les index peuvent devenir fragmentés. Par fragmentation, nous voulons dire que l'ordre physique de la page d'index n'est pas proche de l'ordre alphabétique des enregistrements dans les pages, ou qu'il y a plusieurs pages non−utilisées dans le bloc de 64 pages qui ont été allouées à l'index.

Une manière d'accélérer les index est d'extraire périodiquement les données de la table dans un fichier avec `mysqldump` , d'effacer la table puis de la recréer.

Une autre manière de défragmenter consiste à exécuter un `ALTER` sur le type de la table pour le changer en `MyISAM` puis de le repasser en `InnoDB` .

```
ALTER TABLE tbl_name TYPE=InnoDB
```

Cela fait que MySQL va reconstruire la table. Un autre moyen de réaliser la défragmentation est d'utiliser l'utilitaire `mysqldump` pour exporter la table vers un fichier texte, effacer la table, la recréer et importer le fichier de données.

Si les insertions dans un index sont toujours ascendantes et que les lignes supprimées le sont à la fin du fichier, alors l'algorithme de gestion de l'espace fichiers de `InnoDB` garantit qu'aucune fragmentation n'aura lieu dans l'index.

16.16 Gestion des erreurs InnoDB

La gestion des erreurs avec `InnoDB` n'est pas toujours la même que celle spécifiée par les standards ANSI SQL. En accord avec les standards ANSI, toute erreur survenant durant une commande SQL devrait provoquer l'annulation de celle−ci. `InnoDB` n'annule qu'une partie de la commande de temps en temps, ou la totalité de la transaction. La liste suivante spécifie la gestion des erreurs de `InnoDB` .

- Si vous dépassez l'espace d'un fichier dans l'espace des tables, vous obtiendrez l'erreur MySQL `'Table is full'` et `InnoDB` annulera la requête SQL.
- Un deadlock de transaction ou un dépassement de temps dans un attente de verrouillage fait annuler toute la transaction à `InnoDB` .
- Une erreur de clef dupliquée annule l'insertion de la ligne concernée, même dans une requête du genre `INSERT INTO ... SELECT ...` . Cela fera probablement en sorte que la commande SQL sera annulée si vous n'avez pas spécifié l'option `IGNORE` dans votre requête.
- Une erreur 'row too long' annule la commande SQL.
- Les autres erreurs sont la plupart du temps détectées au niveau de la couche MySQL du code et annulent la commande SQL correspondante.

16.16.1 Codes d'erreurs InnoDB

Voici une liste non−exhaustive des erreurs courantes et spécifiques à `InnoDB` que vous pouvez rencontrer, avec des détails pour les corriger.

1005 (`ER_CANT_CREATE_TABLE` *)*
: Impossible de créer la table. Si le message d'erreur fait référence à une erreur de code `errno` 150, la création de la table a échoué à cause d'une contrainte de clé étrangère, qui n'est pas correctement formée.

1016 (`ER_CANT_OPEN_FILE` *)*
: Impossible de trouver le fichier de table `InnoDB` dans les fichiers de données `InnoDB` alors que le fichier `.frm` s. Résolutions de problèmes avec le dictionaire de données `InnoDB` .

1114 (`ER_RECORD_FILE_FULL` *)*
: `InnoDB` n'a plus d'espace libre dans l'espace de table. Essayez de reconfigurer l'espace de table, et d'ajouter un nouveau fichier de données.

1205 (`ER_LOCK_WAIT_TIMEOUT` *)*
: Le délai d'expiration du verrou a été dépassé. La transaction a été annulée.

1213 (`ER_LOCK_DEADLOCK` *)*
: Blocage de transactions. Vous devriez relancer la transaction.

1216 (`ER_NO_REFERENCED_ROW` *)*
: Vous essayez d'ajouter une ligne, mais il n'y a pas de ligne parente, et une contrainte de clé étrangère échoue. Vous devez ajouter le parent en premier.

1217 (`ER_ROW_IS_REFERENCED` *)*
: Vous essayez d'effacer une ligne parent qui a des enfants, et une contrainte de clé étrangère échoue. Vous devez effacer la ligne fille en premier.

16.16.2 Codes d'erreur sytème

Sous Unix, pour connaître la signification d'un message d'erreur systè,e, utilisez le programme `perror` qui est livré avec la distribution MySQL.

La table suivante fournit une liste de certains code d'erreur système Linux. Pour une version complète, voyez Linux source code .

1 (`EPERM`)
: Operation not permitted

2 (`ENOENT`)
: No such file or directory

3 (`ESRCH`)
: No such process

4 (`EINTR`)
: Interrupted system call

5 (`EIO`)
: I/O error

6 (`ENXIO`)
: No such device or address

7 (`E2BIG`)
: Arg list too long

8 (`ENOEXEC`)
: Exec format error

9 (`EBADF`)
: Bad file number

10 (`ECHILD`)
: No child processes

11 (`EAGAIN`)
: Try again

12 (`ENOMEM`)
: Out of memory

13 (`EACCES`)
: Permission denied

14 (`EFAULT`)
: Bad address

15 (`ENOTBLK`)
: Block device required

16 (`EBUSY`)
: Device or resource busy

17 (`EEXIST`)
: File exists

18 (`EXDEV`)
: Cross−device link

19 (`ENODEV`)
: No such device

20 (`ENOTDIR`)
: Not a directory

21 (`EISDIR`)
: Is a directory

22 (`EINVAL`)
: Invalid argument

23 (`ENFILE`)
: File table overflow

24 (`EMFILE` *)*
: Too many open files

25 (`ENOTTY` *)*
: Inappropriate ioctl for device

26 (`ETXTBSY` *)*
: Text file busy

27 (`EFBIG` *)*
: File too large

28 (`ENOSPC` *)*
: No space left on device

29 (`ESPIPE` *)*
: Illegal seek

30 (`EROFS` *)*
: Read−only file system

31 (`EMLINK` *)*
: Too many links

La table suivante fournit une liste de certains code d'erreur système Windows. Pour une version complète, voyez Microsoft website .

1 (`ERROR_INVALID_FUNCTION` *)*
: Incorrect function.

2 (`ERROR_FILE_NOT_FOUND` *)*
: The system cannot find the file specified.

3 (`ERROR_PATH_NOT_FOUND` *)*
: The system cannot find the path specified.

4 (`ERROR_TOO_MANY_OPEN_FILES` *)*
: The system cannot open the file.

5 (`ERROR_ACCESS_DENIED` *)*
: Access is denied.

6 (`ERROR_INVALID_HANDLE` *)*
: The handle is invalid.

7 (`ERROR_ARENA_TRASHED` *)*
: The storage control blocks were destroyed.

8 (`ERROR_NOT_ENOUGH_MEMORY` *)*
: Not enough storage is available to process this command.

9 (`ERROR_INVALID_BLOCK` *)*
: The storage control block address is invalid.

10 (`ERROR_BAD_ENVIRONMENT` *)*
: The environment is incorrect.

11 (`ERROR_BAD_FORMAT` *)*
: An attempt was made to load a program with an incorrect format.

12 (`ERROR_INVALID_ACCESS` *)*
: The access code is invalid.

13 (`ERROR_INVALID_DATA` *)*
: The data is invalid.

14 (`ERROR_OUTOFMEMORY` *)*
: Not enough storage is available to complete this operation.

15 (`ERROR_INVALID_DRIVE` *)*
: The system cannot find the drive specified.

16 (`ERROR_CURRENT_DIRECTORY` *)*
: The directory cannot be removed.

17 (`ERROR_NOT_SAME_DEVICE` *)*
: The system cannot move the file to a different disk drive.

18 (`ERROR_NO_MORE_FILES` *)*
: There are no more files.

19 (`ERROR_WRITE_PROTECT` *)*
: The media is write protected.

20 (`ERROR_BAD_UNIT` *)*
: The system cannot find the device specified.

21 (`ERROR_NOT_READY` *)*
: The device is not ready.

22 (`ERROR_BAD_COMMAND` *)*
: The device does not recognize the command.

23 (`ERROR_CRC` *)*
: Data error (cyclic redundancy check).

24 (`ERROR_BAD_LENGTH` *)*
: The program issued a command but the command length is incorrect.

25 (`ERROR_SEEK` *)*
: The drive cannot locate a specific area or track on the disk.

26 (`ERROR_NOT_DOS_DISK` *)*
: The specified disk or diskette cannot be accessed.

27 (`ERROR_SECTOR_NOT_FOUND` *)*
: The drive cannot find the sector requested.

28 (`ERROR_OUT_OF_PAPER` *)*
: The printer is out of paper.

29 (`ERROR_WRITE_FAULT` *)*
: The system cannot write to the specified device.

30 (`ERROR_READ_FAULT` *)*
: The system cannot read from the specified device.

31 (`ERROR_GEN_FAILURE` *)*
: A device attached to the system is not functioning.

32 (`ERROR_SHARING_VIOLATION` *)*
: The process cannot access the file because it is being used by another process.

33 (`ERROR_LOCK_VIOLATION` *)*
: The process cannot access the file because another process has locked a portion of the file.

34 (`ERROR_WRONG_DISK` *)*
: The wrong diskette is in the drive. Insert %2 (Volume Serial Number: %3) into drive %1.

36 (`ERROR_SHARING_BUFFER_EXCEEDED` *)*
: Too many files opened for sharing.

38 (`ERROR_HANDLE_EOF` *)*
: Reached the end of the file.

39 (`ERROR_HANDLE_DISK_FULL` *)*
: The disk is full.

112 (`ERROR_DISK_FULL` *)*
: The disk is full.

123 (`ERROR_INVALID_NAME` *)*
: The filename, directory name, or volume label syntax is incorrect.

1450 (`ERROR_NO_SYSTEM_RESOURCES` *)*
: Insufficient system resources exist to complete the requested service.

16.17 Restrictions sur les tables InnoDB

- Une table ne peut pas contenir plus de 1000 colonnes.
- La taille maximale d'une clé est de 1024 octets.

- La taille maximale d'une ligne, hormis pour les colonnes de type `BLOB` et `TEXT` , et légèrement inférieure à la moitié d'une page de base, c'est à dire, que la taille maximale d'une ligne est d'environs 8000 octets. Les colonnes `LONGBLOB` et `LONGTEXT` doivent être un peut plus petite que 4Go, et la taille totale d'une ligne, incluant les colonnes `BLOB` et `TEXT` doivent être de 4 Go. `InnoDB` stocke les 512 premiers octets des valeurs `BLOB` et `TEXT` dans la ligne, et le reste dans une page séparée.
- Sur certains systèmes d'exploitation, le fichier de données est limité à 2 Go.
- La taille combinée des fichiers de log doit être inférieure à 4 Go.
- La taille minimale d'un espace de tables est de 10Mo. La taille maximale d'un espace de tables est de 4 milliards de pages de bases (64To). C'est aussi la taille maximal d'une table.
- Les tables `InnoDB` ne supportent pas les index `FULLTEXT` .
- Sous Windows, `InnoDB` stocke les noms de bases de données et les noms de tables en interne, et en minuscule. Pour passer une base au format binaire de Unix à Windows, ou le contraire, vous devez créer toutes vos bases et tables en minuscules.
- **Attention** : **NE convertissez PAS** les tables de droits du format `MyISAM` en `InnoDB` ! Cela n'est pas supporté. Si vous faîtes cela, MySQL ne va pas redémarrer jusqu'à ce que vous restauriez vos données avec une vieille sauvegarde, ou que vous regénériez ces tables avec le script `mysql_install_db` .
- `InnoDB` ne conserve pas de compte interne de ligne pour une table. Cela serait en fait compliqué, à cause du multi−versionnage. Pour traiter une commande `SELECT COUNT(*) FROM T` , `InnoDB` doit scanner l'index de la table, ce qui prendra du temps si la table n'est pas enregistrée dans le buffer. Pour accélérer le compte, vous devez créer un compteur de table vous−même, et votre application le modifiera à chaque ajout ou suppression. Si votre table ne change pas souvent, l'utilisation du cache sera une bonne solution. `SHOW TABLE STATUS` peut aussi être utilisé pour obtenir un décompte approximatif des lignes. Paramétrage optimale d' `InnoDB` .
- Pour une colonne `AUTO_INCREMENT` , vous devez toujours définir un index pour la table, et cet index doit contenir uniquement la colonne `AUTO_INCREMENT` . Dans les tables `MyISAM` , la colonne `AUTO_INCREMENT` peut faire partie d'un index multi−colonne.
- `InnoDB` ne supporte pas l'option de configuration initale des colonnes `AUTO_INCREMENT` dans les commandes `CREATE TABLE` ou `ALTER TABLE` . Pour configurer cette valeur avec une table `InnoDB` , insérez une ligne avec une valeur inférieure d'une unité à votre valeur de départ, ou bien insérez la première ligne en spécifiant la première valeur.
- Lorque vous redémarrez le serveur MySQL, `InnoDB` peut réutiliser une ancienne valeur de la séquence `AUTO_INCREMENT` (c'est à dire, une valeur qui a été assignée à une transaction annulée).
- Lorsque la colonne `AUTO_INCREMENT` n'a plus de valeurs, `InnoDB` passe le `BIGINT` à `-9223372036854775808` et les `BIGINT UNSIGNED` à `1` . Cependant, les valeurs `BIGINT` sont codées sur 64 bits, alors même si vous insérez 1 million lignes par seconde, cela vous prendra un million d'années avant d'atteindre la limite des `BIGINT` . Avec les autres types de colonnes, une erreur de clé doublon sera émise. C'est identique au fonctionnement des tables `MyISAM` , le comportement générale de MySQL, et ce n'est pas caractéristique d'un moteur spécifique.
- `DELETE FROM TABLE` ne regénère pas la table, mais au lieu de cela, il efface les lignes une à une, ce qui est bien plus lent. Dans les prochaines versions, MySQL va pouvoir utiliser la commande `TRUNCATE` qui est très rapide.
- `TRUNCATE tbl_name` est synonyme de `DELETE FROM tbl_name` pour `InnoDB` et ne remet pas à zéro le compteur de `AUTO_INCREMENT` .
- `SHOW TABLE STATUS` ne donne pas de statistiques exactes pour les tables `InnoDB` , hormis pour la taille physique réservée par la table. Le nombre de lignes n'est qu'une estimation utilisée pour les optimisations SQL.
- Si vous essayez de créer un index unique sur un préfixe d'une colonne, vous allez obtenir une erreur :

```
CREATE TABLE T (A CHAR(20), B INT, UNIQUE (A(5))) TYPE = InnoDB;
```

  Si vous créez un index non−unique sur un préfixe de colonne, `InnoDB` va créer un index pour toute la colonne.

Ces restrictions sont levées depuis les versions 4.0.14 et 4.1.1.

- `INSERT DELAYED` n'est pas supportés par les tables `InnoDB` .
- La commande MySQL `LOCK TABLES` ne reconnait pas le verrouillage de ligne `InnoDB` réalisé dans les commandes SQL achevées : cela signifie que vous pouvez poser un verrou sur une table même si il existe une transaction qui a été posée par un autre utilisateur. Par conséquent, votre opération doit attendre que les autres tables soient libres, et elle peut aussi entrer en conflit avec une autre requête. De plus, un blocage de verrous est possible mais il ne met pas en danger l'intégrité des transactions, car le verrou de ligne posé par `InnoDB` se charge toujours de l'intégrité. Enfin, un verrou de table évite aux autre transactions de poser un verrou de ligne (en conflit avec le mode de verrous) sur la table.
- Avant MySQL 3.23.52, la réplication fonctionnait toujours en mode d'auto−validation. Par conséquent, les lectures cohérentes de l'esclaves voyait aussi les transactions partiellement traitées, et la cohérence n'était pas assurée. Cette restriction a été levée en MySQL 3.23.52.
- La commande `LOAD TABLE FROM MASTER` de configuration de la réplication ne fonctionne pas pour les tables `InnoDB` . Un palliatif consiste à modifier la table en `MyISAM` sur le maître, faire la configuration, puis repasser la table en format `InnoDB` .
- La taille par défaut d'une page de base avec `InnoDB` est de 16Ko. En recompilant le code, vous pouvez donner une valeur allant de 8Ko à 64Ko. Vous mettre à jour les valeurs des constantes `UNIV_PAGE_SIZE` et `UNIV_PAGE_SIZE_SHIFT` dans le fichier `univ.i` .

16.18 Résolution de problèmes avec InnoDB

- Une règle générale est que lorsqu'une opération échoue ou que vous soupsonnez un bug, vous devez lire le fichier de log d'erreurs de MySQL, qui porte généralement le nom de `Array` , ou, sous Windows, `mysql.err` .
- Lorsque vous recherchez une solution, il est généralement bon de faire tourner le serveur depuis la ligne de commande, et non pas avec le script `mysqld_safe` ou comme service Windows. Vous verrez ce que `mysqld` affiche directement dans le terminal, et vous aurez une meilleure idée de ce qui se passe. Sous Windows, vous devez lancer le serveur avec `--console` pour voir le résultat dans la console.
- Utilisez le moniteur `InnoDB` pour avoir des informations sur un problème. Si le problème est lié aux performacnes, ou si votre serveur semble gelé, vous devriez utiliser `innodb_monitor` pour afficher les informations sur l'état interne de `InnoDB` . Si le problème est lié aux verrous, utilisez `innodb_lock_monitor` . Si le problème est lié à la création de table ou lié aux dictionnaires de données, utilisez `innodb_table_monitor` pour voir le contenu du dictionnaire de données internes d' `InnoDB` .
- Si vous suspectez qu'une table est corrompue, utilisez la commande `CHECK TABLE` sur cette table.

16.18.1 Solutions pour le dictionnaire de données InnoDB

Un problème spécifique avec les tables est que le serveur MySQL garde les données dans un dictionnaire de données `.frm` , qu'il stocke dans le dossier de base de données, alors que `InnoDB` stocke aussi les informations dans son propre dossier de données dans l'espace de tables. Si vous déplacez le fichier `.frm` , ou si vous utilisez `DROP DATABASE` en MySQL versions 3.23.44 et plus anciennes, ou si le serveur crashe au milieu d'une opération dans le dictionnaire de données, le fichier `.frm` peut être déconnecté du dictionnaire de données interne `InnoDB` .

Un symptôme de ce déphasage est que les commandes `CREATE TABLE` échouent. Si cela arrive, vous devez lire le fichier de log d'erreurs. Si le fichier de log dit que la table existe déjà dans le dictionnaire interne d' `InnoDB` , vous avez un fichier de table orphelin dans l'espace de table `InnoDB` , qui n'a plus de fichier `.frm` correspondant. Le message d'erreur ressemble à ceci :

```
 InnoDB: Error: table test/parent already exists in InnoDB internal
InnoDB: data dictionary. Have you deleted the .frm file
InnoDB: and not used DROP TABLE? Have you used DROP DATABASE
InnoDB: for InnoDB tables in MySQL version <= 3.23.43?
InnoDB: See the Restrictions section of the InnoDB manual.
InnoDB: You can drop the orphaned table inside InnoDB by
InnoDB: creating an InnoDB table with the same name in another
InnoDB: database and moving the .frm file to the current database.
InnoDB: Then MySQL thinks the table exists, and DROP TABLE will
InnoDB: succeed.
```

Vous pouvez effacer le fichier orphelin en suivant les instructions du message d'erreur.

Un autre symptôme de déphasage st que MySQL affiche MySQL une erreur qui dit qu'il ne peut pas ouvrir le fichier `.InnoDB` :

```
ERROR 1016: Can't open file: 'child2.InnoDB'. (errno: 1)
```

Dans les logs d'erreurs, vous trouverez le message suivant :

```
 InnoDB: Cannot find table test/child2 from the internal data dictionary
InnoDB: of InnoDB though the .frm file for the table exists. Maybe you
InnoDB: have deleted and recreated InnoDB data files but have forgotten
InnoDB: to delete the corresponding .frm files of InnoDB tables?
```

Cela signifie qu'il y a un fichier `.frm` orphelin sans table correspondante dans les données `InnoDB` . Vous pouvez effacer simplement le fichier orphelin `.frm` .Si MyQSL crashe au milieu d'une opération `ALTER TABLE` , vous pouvez vous retrouver avec un fichier temporaire de table, orphelin, dans le code `InnoDB` . Avec `innodb_table_monitor` vous pouvez voir une table dont le nom est `#sql...` , mais comme MySQL ne vous permet pas d'accéder à une table avec un tel nom vous ne pouvez ni l'effacer, ni la détruire. La solution est d'utiliser un mécanisme spécial, disponible depuis MySQL 3.23.48.

Lorsque vous avez une table orpheline `#sql_id` dans un espace de table, vous pouvez faire que `InnoDB` le renomme en `rsql_id_recover_innodb_tmp_table` avec la commande :

```
CREATE TABLE `rsql_id_recover_innodb_tmp_table`(...) TYPE=InnoDB;
```

Les guillemets obliques autour du nom de table sont nécessaires car une table temporaire contient un tiret `'-'` .

La définition de la table doit être similaire à celle de la table temporaire. Si vous ne connaissez pas la définition de la table temporire, vous pouvez utiliser une définition arbitraire dans la commande `CREATE TABLE` précédente, et ensuite, remplacer le fichier `rsql_id.frm` par le fichier `#sql_id.frm` de la table temporaire. Notez que pour copier ou pour renommer un fichier dans le shell, vous devez mettre le nom du fichier entre guillemets, car le nom du fichier contient `'#'` . Alors, vous pouvez exporter et effacer la table renommée.

17 Introduction à MySQL Cluster

Ce chapitre est en cours de rédaction. Les autres documents decrivants le cluster MySQL sont disponibles à http://www.mysql.com/cluster et http://dev.mysql.com/doc .

MySQL Cluster utiliser le nouveau moteur de table `NDBCluster` pour faire fonctionner plusieurs serveurs MySQL en cluster.

17.1 Présentation de MySQL Cluster

Un Cluster MySQL est un groupe de processus qui s'exécutent sur plusieurs serveurs MySQL, des noeuds `NDBCluster` , et des processus d'administration, ainsi que des processus d'accès spécialisés. Tous ces programmes fonctionnent ensemble pour former un Cluster MySQL. Lorsque les données sont stockées dans le moteur `NDBCluster` , les tables sont réparties sur les noeuds `NDBCluster` . Ces tables sont directement accessibles depuis tous les autres serveurs MySQL faisant partie du Cluster. Par conséquent, si une application met à jour une ligne, tous les autres serveurs le verront immédiatement.

Les données stockées dans le moteur de table de MySQL Cluster peut être dupliquées, et sont capables de gérer une indisponibilité d'un noeud sans autre impact que l'annulation des transactions qui utilisaient ces données. Cela ne devrait pas être un problème, cas les applications transactionnelles sont écrites pour gérer les échecs de transaction.

En plaçant MySQL Cluster en Open Source, MySQL rend les hautes performances, haute disponibilité et grands trafics accessibles à tous ceux qui en ont besoin.

17.2 Histoire de MySQL Cluster

17.3 Licence et support de MySQL Cluster

17.4 Concepts de base de MySQL Cluster

MySQL Cluster est constitué de trois types de programmes différents :

- Premièrement, un jeu de processus serveurs MySQL. Ce sont des serveurs MySQL traditionnelles, avec le nouveau moteur de table `NDBCluster` qui autorise l'accès aux tables en cluster.
- Le second type de processus est représenté par les noeuds de stockage de `NDBCluster` . Ces processus contiennent les données stockées dans MySQL Cluster. Les données de MySQL Cluster sont réparties entre les différents noeuds du cluster, et sont aussi doublées dans le cluster.
- Le troisième type de processus est les processus d'administration. Ces processus sont utilisées pour gérer la configuration du cluster.

Nous appelons ces processus de cluster les noeuds du cluster. Mettre en place la configuration du cluster implique la configuration de chaque noeud dans le cluster, et la configuration de chaque moyen de communication entre les noeuds du cluster. MySQL Cluster est actuellement configuré avec le pré−requis que les noeuds sont homogènes en terme de puissance processeur, espace mémoire et largeur de bande. De plus, pour activer un point de configuration, il a été décidé de placer toute la configuration du cluster dans un seul fichier de configuration.

Les processus d'administration gère le fichier de configuration du cluster, et les logs. Tous les noeuds dans le cluster contactent le serveur d'administration pour lire leur configuration : ils doivent donc savoir où le serveur d'administration réside en premier lieu. Lorsqu'un événement pertinent survient dans un moteur de stockage, il est transféré au serveur d'administration qui l'écrit dans le log.

De plus, il y a un nombre arbitraire de clients connectés au cluster. Ils sont de deux types. D'abord, les clients MySQL normaux, qui ne sont pas spécifiques à MySQL Cluster. MySQL Cluster est accessible à partir des applications MySQL écrites en PHP, Perl, C, C++, Java, Ruby, etc. Deuxièmement, les clients d'administration. Ces clients accèdent au serveur d'administration, et émettent des commandes pour lancer ou arrêter correctement des noeuds, lancer ou arrêter la trace serveur (pour les versions de débogage), pour afficher la configuration courante, voir l'état des noeuds du cluster, afficher les versions et noeuds, lancer les sauvegardes, etc.

17.5 Haute disponibilité avec MySQL Cluster

17.5.1 Gestion des indisponibilités de noeuds

17.5.2 Gestion du redémarrage d'un noeud

17.5.3 Modèle de restauration après crash

17.5.4 Gestion d'un redémarrage système

17.5.5 Comment MySQL Cluster évite les cloisenements réseau

17.5.6 Sauvegarde en ligne de MySQL Cluster

17.6 Configuration de MySQL Cluster

17.7 Gestion de processus avec MySQL Cluster

17.8 Administration de MySQL Cluster

17.9 Création de tables NDBCluster

17.10 Modèle de transaction de MySQL Cluster

17.11 Structures de lignes et d'index avec MySQL Cluster

17.11.1 Structure de ligne de taille fixe

17.11.2 Parties de lignes à taille dynamique

17.11.3 Gestion des BLOB avec MySQL Cluster

17.11.4 Structure d'un index

17.11.5 Structure d'un index ordonné

17.12 Méthodes d'accès aux tables avec MySQL Cluster

17.12.1 Accès aux clés primaires

17.12.2 Accès aux hash uniques

17.12.3 Scan de table complète

17.12.4 Scan d'intervalle

17.13 Optimisation des performances avec MySQL Cluster

17.14 Gestion des erreurs avec MySQL Cluster

17.15 Restrictions sur les tables NDBCluster

17.16 Solutions aux problèmes avec MySQL Cluster

17.17 Liste de tâches pour MySQL Cluster

18 Introduction à MaxDB

MaxDB est une base de donnée au niveau de l'entreprise. MaxDB est le nouveau nom d'un système de gestion de bases de données, anciennement appelé SAP DB.

18.1 Historique de MaxDB

L'histoire de SAP DB commence au début des années 1980 lorsqu'il était développé en tant que produit commercial (Adabas). La base de données a changé de nom plusieurs fois durant sa vie. Lorsque SAP AG, une société allemande de Walldorf, a pris en main le développement de ce système de base de données, il portait le nom SAP DB .

SAP a développé ce système de base de données pour servir de système de stockage pour toutes les application SAP lourdes, à savoir SAP R/3. SAP DB avait pour objectif de fournir une alternative aux systèmes de bases de données d'éditeurs tiers, comme Oracle, Microsoft SQL Server, ou DB2 d'IBM. En octobre 2000, SAP a publié SAP DB sous licence GNU GPL (Licence GNU), et en a fait ainsi un logiciel Open Source. En octobre 2003, plus de 2 000 clients l'utilisaient comme système de base de données séparé, en dehors de leur base de donnée principale, faisant partie de la solution APO/LiveCache . En mai 2003, un partenariat technologique a été conclu entre MySQL AB et SAP AG. Ce partenariat autorise MySQL AB à développer davantage SAP DB, à changer son nom et à vendre des licences commerciales de SAP DB aux clients qui ne veulent pas être limités par les restrictions découlant de la licence GNU GPL (Licence GNU). En août 2003, MySQL AB a rebaptisé SAP DB : MaxDB .

18.2 Licence et support MaxDB

MaxDB peut être utilisé sous les mêmes licences que celles qui sont disponibles pour les autres produits distribués par MySQL AB (Licences MySQL). Par conséquent MaxDB est disponible sous licence GNU General Public (Licence GNU), et sous licence commerciale (Support et licences MySQL).

MySQL fournira un support sur MaxDB à ceux qui ne sont pas clients de SAP.

La première version sous bannière MySQL AB, est MaxDB 7.5.00, qui sortira à la fin 2003.

18.3 Liens traitant de MaxDB

Le site officiel où trouver des informations sur MaxDB est http://www.mysql.com/maxdb . A terme, toutes les informations disponibles sur http://www.sapdb.org y seront placées.

18.4 Concepts de base de MaxDB

MaxDB agit en mode client/serveur. Il a été développé pour répondre aux exigences d'installations répondant à un grand nombre de transactions en ligne. Il supporte à la fois la sauvegarde en ligne et l'extension de base de données. Microsoft Clustered Server est supporté directement pour les implémentations à serveurs multiples; les autres solutions de cluster doivent être programmées manuellement. Les outils d'administration de base de données sont fournis à la fois en version native Windows et en version navigateur web.

18.5 Différences de fonctionnalités entre MaxDB et MySQL

La liste suivante est un court résumé des principales différences entre MaxDB et MySQL. Elle n'est pas complète.

- MaxDB fonctionne comme un système client/serveur. MySQL peut fonctionner comme un système client/serveur ou comme un système intégré.
- Il est possible que MaxDB ne fonctionne pas sur toutes les plates−formes supportées par MySQL. Par exemple, MaxDB ne fonctionne pas sur OS/2 d'IBM.
- MaxDB utilise un protocole réseau propriétaire pour la communication client/serveur alors que MySQL utilise soit TCP/IP (avec ou sans chiffrage SSL), soit des interfaces de connexion, ou des canaux de communication nommés (sous les systèmes de la famille Windows NT).
- MaxDB supporte les procédures stockées. Avec MySQL, les procédures stockées sont implémentées en version 5.0. Max DB supporte aussi les triggers par une extension SQL, qui est prévue pour MySQL 5.1. MaxDB contient un débogueur pour les langages de procédure stockée, peut déclencher sous−programmes imbriqués en cascade, et supporte les triggers multiples sur action et par ligne.
- MaxDB est livré avec des interfaces utilisateur en mode texte ou graphique, ou encore sur le web. MySQL est livré avec des interfaces utilisateurs en mode texte uniquement; une interface utilisateur graphique (`MySQL Control Center`) est distribué séparément. Les interfaces utilisateur sur le web pour MySQL sont offerts par des éditeurs tiers.
- MaxDB supporte un certain nombre d'interfaces de programmation qui sont aussi supportées par MySQL. Toutefois, MaxDB ne supporte pas RDO, ADO, ni .NET, qui sont toutes supportées par MySQL. MaxDB peut uniquement être intégré dans des applications C/C++.
- MaxDB contient des fonctionnalités administratives que MySQL n'a pas : la planification de tâches, les événements et alerte, et l'envoi de messages à une base de données administrateur sur signal d'alerte.

18.6 Interopérabilité entre MaxDB et MySQL

Les fonctionnalités suivantes seront incluses dans les versions `MaxDB` qui doivent sortir peu après la première version 7.5.00. Ces fonctionnalités assureront l'interopérabilité entre `MaxDB` et MySQL :

- Il y aura un proxy MySQL qui permettra de se connecter à `MaxDB` en utilisant le protocole MySQL. Ceci rend possible l'utilisation de programmes clients MySQL avec `MaxDB` , comme le client en ligne de commande `mysql` , l'utilitaire d'exportation ou le programme d'import `mysqlimport` . En utilisant `mysqldump` , on peut facilement exporter des données d'une base de donnée vers un autre système de base de données.
- La réplication entre MySQL et `MaxDB` sera supportée dans les deux sens. C'est à dire que MySQL ou `MaxDB` pourra être utilisé comme serveur maître de réplication. Le but à long terme est de faire converger et d'étendre la syntaxe de réplication de façon à ce que les deux systèmes de base de données utilisent la même syntaxe. Introduction .

18.7 Mots réservés de MaxDB

Comme MySQL, `MaxDB` a un certain nombre de mots réservés, qui ont une signification particulière. Normalement, ils ne peuvent pas être utilisés comme noms d'identifiants, comme les noms de tables ou de bases de donnés. Le tableau suivant liste les mots réservés dans `MaxDB` , et indique le contexte dans lequel ces mots sont utilisés. Il indique aussi s'ils ont une équivalence en MySQL ou non. Si une telle équivalence existe, la signification avec MySQL peut être identique ou différente par certains aspects. L'objectif principal est de montrer dans quelle mesure MaxDB diffère de MySQL; par conséquent cette liste n'est pas complète.

Pour la liste de mots réservés dans MySQL, consultez Est−ce que MySQL est sensible aux mots réservés ? .

Réservé par MaxDB	**Contexte d'utilisation dans** MaxDB	**équivalent MySQL**
@	Peut être préfixe à un identifiant, comme ``@table''	Non autorisé
ADDDATE()	fonction SQL	ADDDATE() ; nouveau en MySQL version 4.1.1
ADDTIME()	fonction SQL	ADDTIME() ; nouveau en MySQL version 4.1.1
ALPHA	fonction SQL	Rien de comparable
ARRAY	Type de donnée	Non implémenté
ASCII()	fonction SQL	ASCII() , mais implémenté avec une signification différente
AUTOCOMMIT	Transactions; ON par défaut	Transactions; OFF par défaut
BOOLEAN	types colonne; BOOLEAN n'accepte comme valeur que TRUE , FALSE , et NULL	BOOLEAN a été ajouté en MySQL version 4.1.0; c'est un synonyme de BOOL qui équivaut à TINYINT(1) . Il accepte les valeurs entières dans la même plage que TINYINT ainsi que la valeur NULL . TRUE et FALSE peuvent être utilisés comme alias de 1 et 0 .
CHECK	CHECK TABLE	CHECK TABLE ; similaire mais utilisation différente
COLUMN	type colonne	COLUMN ; mot parasite
CHAR()	fonction SQL	CHAR() ; syntaxe identique ; similaire, utilisation différente
COMMIT	Des validations implicites de transactions se produisent quand les requêtes de définition de données sont publiées	Des validations implicites de transactions se produisent quand les requêtes de définition de données sont publiées mais aussi avec d'autres commandes
COSH()	fonction SQL	Rien de comparable
COT()	fonction SQL	COT() ; syntaxe et implémentation identiques
CREATE	SQL, langage de définition des données	CREATE
DATABASE	fonction SQL	DATABASE() ; DATABASE est utilisé dans un contexte différent, par exemple CREATE DATABASE
DATE()	fonction SQL	CURRENT_DATE
DATEDIFF()	fonction SQL	DATEDIFF() ; nouveau en MySQL version 4.1.1
DAY()	fonction SQL	rien de comparable
DAYOFWEEK()	fonction SQL	DAYOFWEEK() ; le premier jour (1) par défaut est lundi avec MaxDB , et dimanche avec MySQL
DISTINCT	fonctions SQL AVG , MAX , MIN , SUM	DISTINCT ; mais utilisé dans un contexte différent : SELECT DISTINCT
DROP	alias de DROP INDEX	DROP INDEX ; similaire mais utilisation différente
EBCDIC()	fonction SQL	Rien de comparable
EXPAND()	fonction SQL	Rien de comparable
EXPLAIN	Optimisation	EXPLAIN ; similaire mais utilisation différente
FIXED()	fonction SQL	rien de comparable
FLOAT()	fonction SQL	Rien de comparable
HEX()	fonction SQL	HEX() ; similaire mais utilisation différente
INDEX()	fonction SQL	INSTR() ou LOCATE() ; similaire mais syntaxe et signification différentes
INDEX	USE INDEX , IGNORE INDEX et des optimisations similaires sont utilisées juste après SELECT , comme SELECT ... USE INDEX	USE INDEX , IGNORE INDEX et des optimisations similaires sont utilisées dans la clause FROM d'une requête SELECT , comme dans SELECT ... FROM ... USE INDEX
INITCAP()	fonction SQL	Rien de comparable
LENGTH()	fonction SQL	LENGTH() ; syntaxe identique mais implémentation légèrement différente
LFILL()	fonction SQL	Rien de comparable
LIKE	Comparaisons	LIKE ; mais LIKE que MaxDB fournit se rapproche plutôt du REGEX de MySQL
LIKE caractères de rapprochement	MaxDB supporte ``%'', ``_'', ``contrôle+souligné'', ``contrôle+flèche vers le haut'', ``*'', et ``?'' comme caractères de remplacement dans une comparaison LIKE	MySQL supporte ``%'', et ``_'' comme caractères de remplacement dans une comparaison LIKE
LPAD()	fonction SQL	LPAD() ; implémentation légèrement différente
LTRIM()	fonction SQL	LTRIM() ; implémentation légèrement différente
MAKEDATE()	fonction SQL	MAKEDATE() ; nouveau en MySQL version 4.1.1
MAKETIME()	fonction SQL	MAKETIME() ; nouveau en MySQL version 4.1.1
MAPCHAR()	fonction SQL	Rien de comparable
MICROSECOND()	fonction SQL	MICROSECOND() ; nouveau en MySQL version 4.1.1
NOROUND()	fonction SQL	Rien de comparable
NULL	types colonnes; comparaisons	NULL ; MaxDB supporte les valeurs spéciales NULL qui sont renvoyées par des opérations arithmétiques lors de dépassement de capacité ou lors des divisions par zéro; MySQL ne supporte pas de telles valeurs spéciales
PI	fonction SQL	PI() ; syntaxe et implémentation identiques, mais les parenthèses sont obligatoires

REF	type de donnée	Rien de comparable
RFILL()	fonction SQL	Rien de comparable
ROWNO	Prédicat dans la clause WHERE	Similaire à la clause LIMIT
RPAD()	fonction SQL	RPAD() ; implémentation légèrement différente
RTRIM()	fonction SQL	RTRIM() ; implémentation légèrement différente
SEQUENCE	CREATE SEQUENCE , DROP SEQUENCE	AUTO_INCREMENT ; concept similaire mais implémentation différente
SINH()	fonction SQL	Rien de comparable
SOUNDS()	fonction SQL	SOUNDEX() ; syntaxe légèrement différente
STATISTICS	UPDATE STATISTICS	ANALYZE ; concept similaire, mais implémentation différente
SUBSTR()	fonction SQL	SUBSTRING() ; implémentation légèrement différente
SUBTIME()	fonction SQL	SUBTIME() ; nouveau en MySQL version 4.1.1
SYNONYM	langage de définition de données: CREATE [PUBLIC] SYNONYM , RENAME SYNONYM , DROP SYNONYM	Rien de comparable
TANH()	fonction SQL	Rien de comparable
TIME()	fonction SQL	CURRENT_TIME
TIMEDIFF()	fonction SQL	TIMEDIFF() ; nouveau en MySQL version 4.1.1
TIMESTAMP()	fonction SQL	TIMESTAMP() ; nouveau en MySQL version 4.1.1
TIMESTAMP() comme argument de DAYOFMONTH() et DAYOFYEAR()	fonction SQL	Rien de comparable
TIMEZONE()	fonction SQL	Rien de comparable
TRANSACTION()	Renvoie l'identité de la transaction en cours	Rien de comparable
TRANSLATE()	fonction SQL	REPLACE() ; syntaxe et implémentation identiques
TRIM()	fonction SQL	TRIM() ; implémentation légèrement différente
TRUNC()	fonction SQL	TRUNCATE() ; syntaxe et implémentation légèrement différentes
USE	mysql interface en ligne de commande	USE
USER	fonction SQL	USER() ; syntaxe identique, mais implémentation légèrement différente, et les parenthèses sont obligatoires
UTC_DIFF()	fonction SQL	UTC_DATE() ; fournit un moyen de calculer le résultat de UTC_DIFF()
VALUE()	fonction SQL, alias pour COALESCE()	COALESCE() ; syntaxe et implémentation identiques
VARIANCE()	fonction SQL	Rien de comparable
WEEKOFYEAR()	fonction SQL	WEEKOFYEAR() ; nouveau en MySQL version 4.1.1

19 Données spatiales avec MySQL

MySQL 4.1 propose une extension de gestion des données spatiales, et des capacités de génération, stockage et analyse des données spatiales. Actuellement, ces fonctionnalités ne sont possibles qu'avec les tables MyISAM.

Ce chapitre couvre les sujets suivants :

- La base de l'extension de gestion des données spatiales est le modèle géométrique OpenGIS
- Formats de données pour représenter les données spatiales.
- Comme manipuler les données spatiales avec MySQL .
- Indexer les données spatiales avec MySQL .
- Différences avec les spécifications OpenGIS

19.1 Introduction à GIS

MySQL implémente l'extension spatiale en suivant les spécifications du Open GIS Consortium (OGC). C'est un consortium international de plus de 250 personnes, agences et universités qui participent au développement public des concepts de gestions des données spatiales. L'OGC dispose d'un site web http://www.opengis.org/ .

En 1997, l'Open GIS Consortium a publié un document intitulé OpenGIS (R) Simple Features Specifications For SQL : ce document propose plusieurs concepts pour ajouter le support des données spatiales aux serveurs de base de données SQL. Ces spécifications sont disponibles sur le site de Open GIS à l'adresse http://www.opengis.org/techno/implementation.htm : elle contient encore d'autres informations connexes à ce chapitre.

MySQL implémente une sous−partie de l'environnement **SQL avec des types géométriques** , proposé par OGC. Ce terme fait référence à un environnement SQL, disposant d'un jeu de types géométriques. Une colonne SQL géométrique est une colonne avec un type de données géométrique. Ces spécifications décrivent l'ensemble des types géométriques, ainsi que des fonctions permettant leur création et leur analyse.

Un **lieu géographique** représente tout ce qui dispose d'une localisation dans le monde. Un lieu peut être :

- Un entité. Par exemple, une montagne, un lac, une ville.
- Une région. Par exemple, les tropiques, une zone postale.
- Un endroit définissable. Par exemple, un carrefour, une place à la rencontre de deux rues.

Vous pouvez aussi rencontrer des documents qui utilisent le terme de **lieu géospatial** pour désigner les lieux géographiques.

Géométrie est un autre mot qui décrit un lieu géographique. Le sens original du mot **géométrie** se rapporte à une branche des mathématiques. Un autre sens se rapporte à la cartographie, désignant les lieux géométriques que les cartographes utilisent pour dessiner le monde.

Ce chapitre utiliser tous ces termes de manières synonyme : **lieu géographique** , **lieu géospatial** , **lieu** , or **géométrie** . Le terme le plus couramment utilisé ici sera **géométrie** .

Nous définirons une **géométrie** comme **un point ou un ensemble de points représentant un endroit dans le monde** .

19.2 Le modèle géométrique OpenGIS

Le jeu de types géométriques proposé par OGC dans ses spécifications `SQL with Geometry Types` , est basé sur le **modèle géométrique OpenGIS** . Dans ce modèle, chaque objet géométrique dispose des propriétés générales suivantes :

- Il est associé à un système de référence spatiales (`Spatial Reference System`), qui décrit l'origine des coordonnées de l'espace.
- Il appartient à une classe géométrique.

19.2.1 La hiérarchie des classes géométriques

La hiérarchie des classes géométriques est définie comme ceci :

- `Geometry` (non−instanciable)
  - `Point` (instanciable)
  - `Curve` (non−instanciable)
    - `LineString` (instanciable)
      - `Line`
      - `LinearRing`
  - `Surface` (non−instanciable)
    - `Polygon` (instanciable)
  - `GeometryCollection` (instanciable)
    - `MultiPoint` (instanciable)
    - `MultiCurve` (non−instanciable)
      - `MultiLineString` (instanciable)
    - `MultiSurface` (non−instanciable)
      - `MultiPolygon` (instanciable)

Certaines classes sont abstraites et non−instanciables. C'est à dire, il n'est pas possible de créer un objet de cette classe. Les autres classes sont instanciables, et on peut en créer des objets. Chaque classe a des propriétés, et les classes instanciables ont des assertions (des règles qui définissent des instances valides).

`Geometry` est la classe de base. C'est une classe abstraite. Les sous−classes instanciables de `Geometry` sont limitées à des objets de zéro, une ou deux dimensions, qui existent dans un espace bidimensionnel. Toutes les classes géométriques instanciables sont définies de façon à ce que les instances valides d'une classe géométrique soient topologiquement fermées (c'est à dire que l'objet géométrique inclut ses frontières).

La classe `Geometry` a les sous−classes de `Point` , `Curve` , `Surface` et `GeometryCollection` :

- `Point` représente un objet sans dimension.
- `Curve` représente un objet à une dimension, et a pour sous−classe `LineString` , avec les sous−classes `Line` et `LinearRing` .
- `Surface` représente les objets bidimensionnels, et a pour sous−classe `Polygon` .
- `GeometryCollection` dispose des classes de regroupement `MultiPoint` , `MultiLineString` et `MultiPolygon` , destinées aux groupes d'objets de zéro, une ou deux dimensions. Elle permet de modéliser les groupes de points `Points` , de lignes `LineStrings` et de polygones `Polygons` , respectivement. `MultiCurve` et `MultiSurface` sont présentées comme des super−classes abstraites, qui généralisent

les interfaces de regroupements, pour gérer les courbes `Curves` et les surfaces `Surfaces` .

`Geometry` , `Curve` , `Surface` , `MultiCurve` , et `MultiSurface` sont définies comme non−instanciables. Elles définissent un jeu de méthodes communes à leurs sous−classes, et sont inclues ici pour des raisons d'extensibilité.

`Point` , `LineString` , `Polygon` , `GeometryCollection` , `MultiPoint` , `MultiLineString` et `MultiPolygon` sont des classes instanciables.

19.2.2 Classe Geometry

`Geometry` est la classe racine de la hiérarchie. C'est une classe non−instanciable, mais elle dispose d'un grand nombre de propriétés qui sont communes à toutes les formes géométriques, dérivées de `Geometry` . Ces propriétés sont décrites dans la liste suivante. Les sous−classes ont leur propres propriétés spécifiques, définies ultérieurement.

propriétés de la classe `Geometry` Un objet `Geometry` a les propriétés suivantes :

- Son **type** . Chaque objet `Geometry` appartient à une des classes instanciables de la hiérarchie.
- Son **SRID** , ou identifiant de référence spatiale : `Spatial Reference Identifier` . Cette valeur spécifie le système de référence spatial (`Spatial Reference System`), qui décrit l'espace de coordonnées dans lequel l'objet est défini.
- Ses coordonnées `coordinates` dans le système de référence spatial, représentées par des nombres à virgule flottante en double précision (8 octets). Tous les objets non−vides contiennent au moins une paire de coordonnées (X,Y). Les formes géométriques vides ne contiennent pas de coordonnées.

  Les coordonnées sont relatives au `SRID` . Par exemple, dans différents systèmes de coordonnées, la distance entre deux objets peut varier même si les objets ont les mêmes coordonnées, car les distances **planes** et les distances **géocentriques** (système de coordonnées à la surface de la Terre) suivent deux géométries différentes.
- Son intérieur `interior` , sa frontière `boundary` et son extérieur `exterior` . Toutes les formes géométriques occupe une position dans l'espace. L'extérieur de la forme est l'espace qui n'est pas occupé par la forme. L'intérieur de la géométrie est l'espace occupé par la géométrie. La frontière est l'interface entre l'extérieur de la forme et son intérieur.
- Son `MBR` (Rectangle minimal d'enveloppe, `Minimum Bounding Rectangle`), appelé aussi enveloppe. C'est la forme géométrique la plus petite, formée par les coordonnées minimales et maximales (X,Y) :

  ```
  ((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
  ```
- L'attribut **simple** ou **non−simple** . Les objets géométriques de certains types, comme `LineString` , `MultiPoint` , `MultiLineString` sont simple ou non−simple. Chaque type dispose de ses propres assertions.
- L'attribut fermé (`closed`) ou non−fermé (`not closed`). Les objets géométriques de certains types, comme `LineString` , `MultiString` sont fermés ou non−fermés. Chaque type dispose de ses propres assertions.
- L'attribut vide (`empty`) ou non−vide(`not empty`). Une forme est vide si elle ne contient aucun point. L'extérieur, l'intérieur et la frontière d'une forme vide ne sont pas définis (ils sont représentés par la valeur `NULL`). Une forme vide est toujours simple, et a une surface de 0.
- Sa **dimension** . Une forme a une dimension de −{}1, 0, 1, ou 2 :
  - −{}1 représente une forme vide.
  - 0 représente les formes sans surface ni dimension.
  - 1 représente les formes avec une dimension non nulle, mais sans surface.
  - 2 représente les formes avec une dimension et une surface non−nulles.

Les `Point` sont de dimension zéro. Les lignes `LineString` sont de dimension un. Les polygones `Polygon` sont de dimension deux. Les dimensions des objets `MultiPoint` , `MultiLineString` et `MultiPolygon` sont les mêmes que les dimensions des objets dont ils sont composés.

19.2.3 Classe Point

Un `Point` est une forme géométrique qui représente un endroit dans un espace de coordonnées. **Exemples de** `Point`

- Imaginez une carte à grande échelle, avec de nombreuses villes. Un point représentera une ville.
- Sur une carte de ville, un point peut représenter un arrêt de bus.

Propriété du `Point`

- Abscisse, ou coordonnées X.
- Ordonnée, ou coordonnée Y.
- Un point `Point` est défini comme une forme avec zéro dimension.
- Le rectangle d'encadrement d'un `Point` est un rectangle vide.

19.2.4 Classe Curve

Une courbe `Curve` est une forme géométrique à une dimension, qui représente généralement une séquence de points. Une sous−classe particulière de `Curve` est l'interpolation entre deux points. `Curve` est une classe non−instanciable.

Propriétés de la classe `Curve`

- Les coordonnées de ses points.
- `Curve` est définie comme une forme à une dimension.
- Un objet `Curve` est simple si elle ne passe pas par le même point deux fois.
- Un objet `Curve` est fermé si son point de départ est le même que son point d'arrivée.
- La frontière d'un objet `Curve` est vide.
- La frontière d'un objet `Curve` non−fermé est ses deux points terminaux.
- Un objet `Curve` qui est simple et fermé est un objet `LinearRing` .

19.2.5 Classe LineString

Un objet `LineString` est un objet `Curve` avec une interpolation linéaire entre ses points. **Exemples de** `LineString`

- Sur une carte du monde, un objet `LineString` peut représenter les rivières.
- Sur une carte de ville, un objet `LineString` peut représenter les rues.

Propriété des objets `LineString`

- Coordonnées des segments de `LineString` , définis par des paires de points consécutifs.
- Un objet `LineString` est un objet `Line` s'il est constitué de deux points.
- Un objet `LineString` est un objet `LinearRing` s'il est fermé et simple.

19.2.6 Classe Surface

Un objet `Surface` est une forme géométrique à deux dimensions. C'est une classe non−instanciable. Sa seule sous−classe instanciable est la classe `Polygon` . **Propriété de** `Surface`

- Un objet `Surface` est défini comme une forme géométrique à deux dimensions.
- Les spécifications OpenGIS définissent un objet `Surface` comme une forme géométrique simple si elle est d'un seul tenant, qui est associé avec un seul extérieur, et aucune frontière intérieure.
- La frontière d'un objet `Surface` est l'ensemble des courbes fermées qui correspondent à ses frontières extérieures et intérieures.

19.2.7 Classe Polygon

Un polygone `Polygon` est une surface `Surface` plane avec plusieurs côtés. Il est défini par une seule frontière extérieure, au aucune ou plusieurs frontières intérieures. Chaque frontière intérieure définit un trou dans le polygone `Polygon` .

Exemple avec `Polygon`

- Sur une carte régionale, un objet `Polygon` peut représenter une forêt, un département, etc.

Assertions de `Polygon`

- La frontière d'un `Polygon` est constituée d'un ensemble d'objets `LinearRing` (c'est dire, des objets `LineString` qui sont simples et fermés), et qui forme sa frontière intérieure et extérieure.
- Deux frontières intérieures ne se coupent pas. Les courbes d'une frontière d'un objet `Polygon` peuvent se couper en un `Point` , mais uniquement en un point de tangence.
- Un polygone `Polygon` ne peut pas avoir de lignes coupées, d'extrusions ou de trous.
- L'intérieur de tous les `Polygon` est un ensemble de point connectés.
- L'extérieur d'un objet `Polygon` ayant un ou plusieurs trou n'est pas connecté. Chaque trou défini un composant connecté de l'extérieur.

Dans les assertions ci−dessus, les polygones sont des formes géométriques simples. Ces assertions font d'un objet `Polygon` une forme simple.

19.2.8 Classe GeometryCollection

Un objet `GeometryCollection` est une forme géométrique, représentée par le regroupement d'autres formes géométriques.

Tous les éléments dans un objet `GeometryCollection` doivent être placés dans le même système de référence spatiale (c'est à dire dans le même système de coordonnées). `GeometryCollection` n'impose aucune autre contrainte à ses éléments, même si les sous−classes de `GeometryCollection` décrites plus loin, en imposent d'autres. Les restrictions peuvent être basée sur :

- Le type d'élément (par exemple, un objet `MultiPoint` ne peut contenir que des objets `Point`).
- Dimension
- Contraintes sur le degré de recouvrement entre deux éléments.

19.2.9 Classe MultiPoint

Un objet `MultiPoint` est un regroupement d'objets de type `Point` . Les points ne sont pas connectés ou ordonnées de quelque manière que ce soit. **Exemples avec** `MultiPoint`

- Sur une carte du monde, un objet `Multipoint` peut représenter un archipel.
- Sur une carte de ville, un objet `Multipoint` peut représenter les différentes succursales d'une entreprise.

Propriétés de `MultiPoint`

- Un objet `MultiPoint` est un objet sans dimension.
- Un objet `MultiPoint` est simple si tous les points `Point` sont tous distincts (les paires de coordonnées sont toutes distinctes).
- La frontière d'un objet `MultiPoint` est vide.

19.2.10 Classe MultiCurve

Un objet `MultiCurve` est une forme géométrique composée d'objets `Curve` . `MultiCurve` est une classe non−instanciable. **Propriété de** `MultiCurve`

- Un objet `MultiCurve` est défini comme une forme géométrique a une dimension.
- Un objet `MultiCurve` est dit simple si et seulement si tous ses éléments sont simples, et que les seules intersections entre deux éléments interviennent sur les frontières des deux éléments.
- La frontière d'un objet `MultiCurve` est obtenue en appliquant la ``règle de l'union modulo 2'' (dite aussi, règle pair / impair) : Un point est une frontière d'un objet `MultiCurve` s'il fait partie d'un nombre impair de frontière d'élément de l'objet `MultiCurve` .
- Un objet `MultiCurve` est fermé si tous ses éléments sont fermés.
- La frontière d'un objet `MultiCurve` fermé est toujours vide.

19.2.11 Classe MultiLineString

Un objet `MultiLineString` est une forme géométrique de classe `MultiCurve` , composé uniquement d'objets `LineString` .

Exemples avec `MultiLineString`

- Sur une carte régionales, un objet `MultiLineString` représente un système de rivières, ou un autoroute.

19.2.12 Classe MultiSurface

Un objet `MultiSurface` est un groupe de surfaces. `MultiSurface` est une classe non−instanciable. Sa seule sous−classe instanciable est la classe `MultiPolygon` . **Assertions** `MultiSurface`

- Les intérieurs de deux surfaces d'un objet `MultiSurface` doivent avoir des intersections nulles.
- La frontière de deux éléments de l'objet `MultiSurface` doivent avoir en commun un nombre fini de points.

19.2.13 Classe MultiPolygon

Un objet `MultiPolygon` est un objet `MultiSurface` composé d'objets `Polygon` . **Exemple avec** `MultiPolygon`

* Sur une carte régionale, un objet `MultiPolygon` peut représenter un système de lacs.

Assertions `MultiPolygon`

* Les intérieurs de deux objets `Polygon` qui sont éléments d'un même objet `MultiPolygon` n'ont pas d'intersection.
* Les frontières de deux objets `Polygon` qui sont éléments d'un objet `MultiPolygon` ne peuvent que se toucher en un nombre fini de points, et non pas se recouvrir. Le recouvrement est déjà interdit par l'assertion précédente.
* Un objet `MultiPolygon` ne peut pas avoir de lignes coupées, d'extrusions ou de trous. Un objet `MultiPolygon` est un ensemble de point régulier, et fermé.
* L'intérieur d'un objet `MultiPolygon` composé de plus d'un objet `Polygon` n'est pas connecté. Le nombre de composants connectés de l'intérieur d'un objet `MultiPolygon` est égal au nombre d'objets `Polygon` dans l'objet `MultiPolygon` .

Propriétés `MultiPolygon`

* Un objet `MultiPolygon` est défini comme une forme géométrique à deux dimensions.
* La frontière d'un objet `MultiPolygon` est un ensemble de courbes fermées (des objets `LineString`) correspondants aux limites de ses objets `Polygon` composants.
* Chaque objet `Curve` de la frontière de l'objet `MultiPolygon` est dans la frontière d'un seul objet `Polygon` .
* Chaque objet `Curve` de la frontière d'un élément `Polygon` est dans la frontière de l'objet `MultiPolygon` .

19.3 Formats géométriques supportés

Cette section décrit les formats de données géométriques standard qui sont utilisés pour représenter des objets géométriques dans les requêtes. Il s'agit de :

* Le format Well−Known Text (WKT)
* Le format Well−Known binaire (WKB)

En interne, MySQL stocke les valeurs géométriques dans un format indépendant des formats WKT et WKB.

19.3.1 Format Well−Known Text (WKT)

La représentation Well−Known Text (WKT) est conçue pour faire des échanges au format ASCII.

Exemple de représentation d'objets WKT :

* Un `Point` :
  ```
  POINT(15 20)
  ```
  Notez que les coordonnées du point n'ont pas de séparateur virgule.
* Une ligne `LineString` de quatre points :
  ```
  LINESTRING(0 0, 10 10, 20 25, 50 60)
  ```
* Un polygone `Polygon` avec un anneau extérieur et un anneau intérieur :
  ```
  POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))
  ```

- Un groupe de points `MultiPoint` avec trois `Point` :

```
MULTIPOINT(0 0, 20 20, 60 60)
```

- Un groupe de lignes `MultiLineString` avec deux lignes `LineString` :

```
MULTILINESTRING((10 10, 20 20), (15 15, 30 15))
```

- Un groupe de polygone `MultiPolygon` avec deux polygones `Polygon` :

```
MULTIPOLYGON(((0 0,10 0,10 10,0 10,0 0)),((5 5,7 5,7 7,5 7, 5 5)))
```

- Un groupe géométrique `GeometryCollection` constitué de deux `Point` et d'une ligne `LineString` :

```
GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15, 20 20))
```

Une grammaire Backus−Naur qui spécifie les règles de générations formelles pour écrire des valeurs WKT est disponible dans les spécifications OGC, qui sont référencées au début de ce chapitre.

19.3.2 Le format Well−Known Binary (WKB)

La représentation Well−Known Binary (WKB) des valeurs géométriques est définies par les spécifications OpenGIS. Il est aussi défini comme le standard ISO ``SQL/MM Part 3: Spatial''.

WKB sert à échanger des données géométriques sous forme de flux binaires représenté par des `BLOB` contenant des données WKB.

WKB utilise des entiers non−signés d'un octet, de quatre octets et des nombres à virgules flottantes de précision double (IEEE 754 format). Un octet fait 8 bits.

Par exemple, la valeur WKB qui correspond à `POINT(1 1)` est constituée de la séquence suivante de 21 octets, représentée ici en code hexadécimal :

```
0101000000000000000000F03F000000000000F03F
```

La séquence peut être décomposée comme ceci :

```
 Byte order : 01
WKB type   : 01000000
X          : 000000000000F03F
Y          : 000000000000F03F
```

Voici le détail des composants :

- L'ordre des octets peut être 0 ou 1, pour indiquer un stockage little−endian ou big−endian. Les ordres little−endian et big−endian sont aussi connus sous le nom de `Network Data Representation` (NDR) et `External Data Representation` (XDR), respectivement.
- Le type WKB est un code qui indique le type géométrique. Les valeurs de 1 à 7 indiquent : `Point` , Ligne `LineString` , Polygone `Polygon` , Plusieurs points `MultiPoint` , Plusieurs lignes `MultiLineString` , Plusieurs polygones `MultiPolygon` , et Groupe géométrique `GeometryCollection` .
- Une valeur `Point` a des coordonnées X et Y, représentées par un nombre à virgule flottante, en double précision.

Les valeurs WKB des formes géométriques plus complexes sont représentées par des structures biens plus complexes, comme présenté dans les spécifications OpenGIS.

19.4 Créer une base de données avec les fonctionnalités géographiques

Cette section décrit les types de données que vous pouvez utiliser pour représenter des données géographiques dans MySQL, et les fonctions disponibles pour créer et lire les données spatiales.

19.4.1 Types de données géographiques MySQL

MySQL fournit un jeu de types de données qui correspondent aux classes du modèle géométrique OpenGIS. Certains de ces types contiennent des valeurs géométriques simples :

- `GEOMETRY`
- `POINT`
- `LINESTRING`
- `POLYGON`

`GEOMETRY` est le type le plus général. Il peut stocker une forme géométrique de n'importe quel type. Les autres types se restreignent a un type particulier de forme.

Les autres types de données permettent de gérer les groupes de formes géométriques :

- `MULTIPOINT`
- `MULTILINESTRING`
- `MULTIPOLYGON`
- `GEOMETRYCOLLECTION`

`GEOMETRYCOLLECTION` peut stocker un groupe quelconque de formes géométriques. Les autres types se restreignent à des formes géométriques particulières.

19.4.2 Créer des objets géographiques

Cette section décrit comment créer des valeurs en utilisant les fonctions Well−Known Text et Well−Known Binary qui sont définies dans le standard OpenGIS, et les fonctions spécifiques de MySQL.

19.4.2.1 Créer des objets géométriques avec les fonctions WKT

MySQL fournit de nombreuses fonctions qui prennent en arguments un `BLOB` contenant la représentation au format Well−Known Text et optionnellement, un système de référence (SRID), et retourne la forme géométrique correspondante.

`GeomFromText()` accepte des données WKT de n'importe quelle type de données géométriques comme premier argument. L'implémentation fournit aussi des fonctions de construction spécifiques à chaque forme géométrique.

GeomFromText(wkt[,srid])
GeometryFromText(wkt[,srid])
: Construit une forme géométrique à partir de sa représentation WKT et du SRID.

PointFromText(wkt[,srid])
: Construit un objet `POINT` à partir de sa représentation WKT et du SRID.

LineFromText(wkt[,srid])
LineStringFromText(wkt[,srid])
: Construit un objet `LINESTRING` à partir de sa représentation WKT et du SRID.

PolyFromText(wkt[,srid])
PolygonFromText(wkt[,srid])
: Construit un objet `POLYGON` à partir de sa représentation WKT et du SRID.

MPointFromText(wkt[,srid])
MultiPointFromText(wkt[,srid])
: Construit un objet `MULTIPOINT` à partir de sa représentation WKT et du SRID.

MLineFromText(wkt[,srid])
MultiLineStringFromText(wkt[,srid])
: Construit un objet `MULTILINESTRING` à partir de sa représentation WKT et du SRID.

MPolyFromText(wkt[,srid])
MultiPolygonFromText(wkt[,srid])
    Construit un objet `MULTIPOLYGON` à partir de sa représentation WKT et du SRID.
GeomCollFromText(wkt[,srid])
GeometryCollectionFromText(wkt[,srid])
    Construit un objet `GEOMETRYCOLLECTION` à partir de sa représentation WKT et du SRID.

Les spécifications OpenGIS décrivent aussi des fonctions optionnelles pour construire des `Polygon` et `MultiPolygon` basées sur les représentations WKT d'une collection d'anneaux ou d'objets `LineString` fermés. Ces objets peuvent avoir des intersections non vides. MySQL ne supporte pas encore ces fonctions :

BdPolyFromText(wkt,srid)
    Construit un objet `Polygon` à partir de la représentation WKT d'un objet `MultiLineString` , contenant un ensemble d'objets `LineString` fermés.
BdMPolyFromText(wkt,srid)
    Construit un objet `MultiPolygon` à partir de la représentation WKT d'un objet `MultiLineString` , contenant un ensemble d'objets `LineString` fermés.

19.4.2.2 Créer des objets géométriques avec les fonctions WKB

MySQL fournit de nombreuses fonctions qui prennent en arguments un `BLOB` contenant la représentation au format Well−Known Binary et optionnellement, un système de référence (SRID), et retourne la forme géométrique correspondante.

`GeomFromWKT()` accepte des données WKB de n'importe quelle type de données géométriques comme premier argument. L'implémentation fournit aussi des fonctions de construction spécifiques à chaque forme géométrique.

`GeomFromWKB(wkb[,srid])`
`GeometryFromWKB(wkt[,srid])`
    Construit une forme géométrique à partir de sa représentation WKB et du SRID.
`PointFromWKB(wkb[,srid])`
    Construit un objet `POINT` à partir de sa représentation WKB et du SRID.
`LineFromWKB(wkb[,srid])`
`LineStringFromWKB(wkb[,srid])`
    Construit un objet `LINESTRING` à partir de sa représentation WKB et du SRID.
`PolyFromWKB(wkb[,srid])`
`PolygonFromWKB(wkb[,srid])`
    Construit un objet `POLYGON` à partir de sa représentation WKB et du SRID.
`MPointFromWKB(wkb[,srid])`
`MultiPointFromWKB(wkb[,srid])`
    Construit un objet `MULTIPOINT` à partir de sa représentation WKB et du SRID.
`MLineFromWKB(wkb[,srid])`
`MultiLineStringFromWKB(wkb[,srid])`
    Construit un objet `MULTILINESTRING` à partir de sa représentation WKB et du SRID.
`MPolyFromWKB(wkb[,srid])`
`MultiPolygonFromWKB(wkb[,srid])`
    Construit un objet `MULTIPOLYGON` à partir de sa représentation WKB et du SRID.
`GeomCollFromWKB(wkb[,srid])`
`GeometryCollectionFromWKB(wkt[,srid])`
    Construit un objet `GEOMETRYCOLLECTION` à partir de sa représentation WKB et du SRID.

Les spécifications OpenGIS décrivent aussi des fonctions optionnelles pour construire des `Polygon` et `MultiPolygon` basées sur les représentations WKB d'une collection d'anneaux ou d'objets `LineString` fermés. Ces objets peuvent avoir des intersections non vides. MySQL ne supporte pas encore ces fonctions :

`BdPolyFromWKB(wkb,srid)`
    Construit un objet `Polygon` à partir de la représentation WKB d'un objet `MultiLineString` , contenant un ensemble d'objets `LineString` fermés.

`BdMPolyFromWKB(wkb,srid)`
: Construit un objet `MultiPolygon` à partir de la représentation WKB d'un objet `MultiLineString` , contenant un ensemble d'objets `LineString` fermés.

19.4.2.3 Création de formes géométriques avec les fonctions spécifiques de MySQL

Note : MySQL ne dispose pas encore de toutes les fonctions listées dans cette section.

MySQL fournit un jeu de fonction pratiques pour créer des représentations WKB. Ces fonctions sont décrites dans cette section, et sont spécifiques à MySQL : ce sont des extensions aux spécifications OpenGIS. Le résultat de ces fonctions sont des valeurs de type `BLOB` qui contiennent la représentation au format WKB des objets géométriques, sans leur SRID. Le résultat de ces fonctions peut être utilisé comme premier argument de toute fonction de la famille `GeomFromWKB()` .

Point(x,y)
: Construit un `Point` au format WKB, en utilisant ses coordonnées.

MultiPoint(pt1,pt2,...)
: Construit un objet `MultiPoint` au format WKB, en utilisant les arguments `Point` WKB. Si aucun argument n'est au format `WKBPoint` , la valeur retournée est `NULL` .

LineString(pt1,pt2,...)
: Construit un objet `LineString` au format WKB, à partir d'arguments `Point` WKB. Si aucun argument n'est au format `WKBPoint` , la valeur retournée est `NULL` . Si moins de deux arguments est fourni, la valeur retournée est `NULL` .

MultiLineString(ls1,ls2,...)
: Construit un objet `MultiLineString` au format WKB, en utilisant les objets `LineString` WBK. Si aucun argument n'est de classe `LineString` , la valeur retournée alors `NULL` .

Polygon(ls1,ls2,...)
: Construit un objet `Polygon` au format WKB, en utilisant les objets `LineString` WBK. Si aucun argument n'est de classe `LinearRing` , (c'est à dire, un objet `LineString` fermé et simple), la valeur retournée alors `NULL` .

MultiPolygon(poly1,poly2,...)
: Construit un objet `MultiPolygon` au format WKB, en utilisant les objets `Polygon` WBK. Si aucun argument n'est de classe `Polygon` , la valeur retournée alors `NULL` .

GeometryCollection(g1,g2,...)
: Construit un objet `GeometryCollection` au format WKB. Si aucun argument n'est au format valide WBK, la valeur retournée alors `NULL` .

19.4.3 Créer des colonnes géométriques

MySQL fournit une méthode standard pour créer des colonnes géographiques, avec les commandes `CREATE TABLE` ou `ALTER TABLE` , etc. Actuellement, les colonnes géométriques ne sont supportées que par les tables `MyISAM` .

- Utilisez la commande `CREATE TABLE` pour créer une colonne géométrique :

```
 mysql> CREATE TABLE geom (g GEOMETRY);
Query OK, 0 rows affected (0.02 sec)
```

- Utilisez la commande `ALTER TABLE` pour ajouter ou effacer une colonne géométrique dans une table existante :

```
 mysql> ALTER TABLE geom ADD pt POINT;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> ALTER TABLE geom DROP pt;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

19.4.4 Remplir des colonnes géométriques

Une fois que vous avez créé des colonnes géométriques, vous pouvez les remplir avec des données géographiques.

Les valeurs doivent être stockées dans un format géométrique interne, mais vous pouvez les convertir à ce format à partir des formats Well−Known Text (WKT) et Well−Known Binary (WKB). Les exemples suivants vous montre comment insérer des valeurs géométriques dans une table, en convertissant des valeurs WKT en un format géométrique.

Vous pouvez faire la conversion directement avec la commande `INSERT` :

```
 INSERT INTO geom VALUES (GeomFromText('POINT(1 1)'));

 SET &#64;g = 'POINT(1 1)';
INSERT INTO geom VALUES (GeomFromText(&#64;g));
```

Au besoin, la conversion peut avoir lieur avant la commande `INSERT` :

```
 SET &#64;g = GeomFromText('POINT(1 1)');
INSERT INTO geom VALUES (&#64;g);
```

Les exemples suivants illustre l'insertion de données plus complexes dans la table :

```
 SET &#64;g = 'LINESTRING(0 0,1 1,2 2)';
INSERT INTO geom VALUES (GeomFromText(&#64;g));

 SET &#64;g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
INSERT INTO geom VALUES (GeomFromText(&#64;g));

 SET &#64;g = 'GEOMETRYCOLLECTION(POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))';
INSERT INTO geom VALUES (GeomFromText(&#64;g));
```

Les exemples précédents utilisent tous la fonction `GeomFromText()` pour créer des valeurs géométriques. Vous pouvez aussi utiliser des fonctions spécifiques à chaque forme géométrique :

```
 SET &#64;g = 'POINT(1 1)';
INSERT INTO geom VALUES (PointFromText(&#64;g));

 SET &#64;g = 'LINESTRING(0 0,1 1,2 2)';
INSERT INTO geom VALUES (LineStringFromText(&#64;g));

 SET &#64;g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
INSERT INTO geom VALUES (PolygonFromText(&#64;g));

 SET &#64;g = 'GEOMETRYCOLLECTION(POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))';
INSERT INTO geom VALUES (GeomCollFromText(&#64;g));
```

Notez que si une application cliente veut utiliser des représentations WKB de valeur géométrique, elle est responsable d'envoyer des requêtes avec des valeurs WKB valides au serveur. Sinon, il y a de nombreux moyens de passer cette contrainte. Par exemple :

- Insertion d'un point `POINT(1 1)` avec sa valeur littérale héxadécimale :
  ```
  mysql> INSERT INTO geom VALUES
     -> (GeomFromWKB(0x0101000000000000000000F03F000000000000F03F));
  ```
- Une application ODBC veut envoyer une représentation WKB, en l'associant à une variable de requête en utilisant un argument de type `BLOB` :
  ```
  INSERT INTO geom VALUES (GeomFromWKB(?))
  ```
  D'autres interfaces de programmation peuvent supporter des mécanismes similaires.
- Dans un programme C, vous pouvez protéger les valeurs binaires en utilisant la fonction `mysql_real_escape_string()` et inclure le résultat dans une chaîne de requête, qui sera envoyée au serveur. `mysql_real_escape_string()` .

19.4.5 Lire des données géométriques

Les formes géométriques sont stockées dans une table, et peuvent être lues dans différents formats. Vous pouvez les convertir au format interne, WBK ou WBT.

19.4.5.1 Lire des données géométriques au format interne

Lire des données géométriques au format interne peut être utile lors des transferts de tables en tables :

```
CREATE TABLE geom2 (g GEOMETRY) SELECT g FROM geom;
```

19.4.5.2 Lire des données géométriques au format WBT

La fonction `AsText()` fournit une représentation textuelle d'une forme géométrique. Elle convertit la forme depuis le format interne en une chaîne au format WKT.

```
mysql> SELECT AsText(g) FROM geom;
+-------------------------+
| AsText(p1)              |
+-------------------------+
| POINT(1 1)              |
| LINESTRING(0 0,1 1,2 2) |
+-------------------------+
```

19.4.5.3 Lire des données géométriques au format WBK

La fonction `AsBinary()` fournit l'accès binaire aux formes géométriques. Elle convertit une forme géométrique, depuis le format interne en un `BLOB` contenant la représentation WKB.

```
SELECT AsBinary(g) FROM geom;
```

19.5 Analyser des données géographiques

Après avoir remplies les colonnes géographiques avec des valeurs, vous êtes prêts à les analyser. MySQL propose un jeu de fonctions pour effectuer différentes opérations sur des données géographiques. Ces fonctions peuvent être regroupées en plusieurs catégories principales, suivant le type de manipulations :

- Les fonctions qui convertissent les données géométriques en différents formats.
- Les fonctions qui fournissent des accès aux données qualitatives ou quantitatives d'un objet géométrique.
- Les fonctions qui décrivent les relations entre deux objets géométriques.
- Les fonctions qui créent des objets à partir d'autres objets existants.

Les fonctions d'analyse de données géographiques peuvent être utilisées dans plusieurs contextes, comme :

- Un programme SQL interactif, comme `mysql` ou `MySQLCC`
- Une application écrite dans un langage qui dispose du support des bibliothèques clientes MySQL.

19.5.1 Fonctions pour convertir les formes de format

MySQL supporte les fonctions suivantes pour convertir des formes géométriques entre les formats internes, WKT et WKB :

GeomFromText(wkt[,srid])

Convertit une chaîne au format WKT vers le format interne, et retourne le résultat. Des fonctions adaptées au type sont supportées comme `PointFromText()` et `LineFromText()` ; voyez Créer des objets géométriques avec les fonctions WKT .

GeomFromWKB(wkb[,srid])

Convertit une chaîne au format WKB vers le format interne, et retourne le résultat. Des fonctions adaptées au type sont supportées comme `PointFromWKB()` et `LineFromWKB()` ; voyez Créer des objets géométriques avec les fonctions WKB .

AsText(g)

Convertit une chaîne au format interne vers le format WKT, et retourne le résultat.

```
mysql> SET &#64;g = 'LineString(1 1,2 2,3 3)';
mysql> SELECT AsText(GeomFromText(&#64;g));
+--------------------------+
| AsText(GeomFromText(&#64;G)) |
+--------------------------+
| LINESTRING(1 1,2 2,3 3)  |
+--------------------------+
```

AsBinary(g)

Convertit une chaîne au format interne vers le format WKB, et retourne le résultat.

19.5.2 Fonction d'analyse des propriétés des formes Geometry

Chaque fonction de ce groupe prend une forme géométrique comme argument et représente un attribut qualitatif ou quantitatif de cette forme. Certaines fonctions sont spécifiques à une forme particulière. Certaines fonctions retournent `NULL` si l'argument n'est pas d'un type valide. Par exemple, `Area()` retourne `NULL` si le type de l'objet est ni `Polygon` ni `MultiPolygon` .

19.5.2.1 Fonctions générales d'analyse géométrique

Les fonctions de cette section n'ont pas de restriction sur les arguments, et acceptent toutes sortes de formes.

`GeometryType(g)`

Retourne le type de forme de `g` , sous forme de chaîne. Le nom correspond à l'une des sous−classes instanciable `Geometry` .

```
mysql> SELECT GeometryType(GeomFromText('POINT(1 1)'));
+------------------------------------------+
| GeometryType(GeomFromText('POINT(1 1)')) |
+------------------------------------------+
| POINT                                    |
+------------------------------------------+
```

`Dimension(g)`

Retourne le nombre de dimensions de l'objet `g` . Le résultat peut être −{}1, 0, 1 ou 2. La signification de ces valeurs est expliqué dans la section Classe `Geometry` .

```
mysql> SELECT Dimension(GeomFromText('LineString(1 1,2 2)'));
+------------------------------------------------+
| Dimension(GeomFromText('LineString(1 1,2 2)')) |
+------------------------------------------------+
|                                              1 |
+------------------------------------------------+
```

`SRID(g)`

Retourne un entier indiquant l'identifiant du système de coordonnées de la forme `g` .

```
mysql> SELECT SRID(GeomFromText('LineString(1 1,2 2)',101));
+-----------------------------------------------+
| SRID(GeomFromText('LineString(1 1,2 2)',101)) |
+-----------------------------------------------+
|                                           101 |
+-----------------------------------------------+
```

`Envelope(g)`

Retourne le rectangle enveloppe (`Minimum Bounding Rectangle` , ou `MBR`) de la forme `g` . Le résultat est retourné sous forme de polygone.

```
mysql> SELECT AsText(Envelope(GeomFromText('LineString(1 1,2 2)')));
+-------------------------------------------------------+
| AsText(Envelope(GeomFromText('LineString(1 1,2 2)'))) |
+-------------------------------------------------------+
| POLYGON((1 1,2 1,2 2,1 2,1 1))                        |
+-------------------------------------------------------+
```

Le polygone est défini par ses sommets :

```
POLYGON((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

Les spécifications OpenGIS définissent les fonctions suivantes, que MySQL n'implémente pas :

`Boundary(g)`
: Retourne une forme qui représente la frontière de `g` .

`IsEmpty(g)`
: Retourne 1 si la forme `g` est vide, et 0 si elle n'est pas vide. Elle retourne −{}1 si l'argument est `NULL` . Si la forme est vide, elle représente un ensemble de points vide.

`IsSimple(g)`
: Actuellement, cette fonction est un inutilisable et ne doit pas être employée. Lorsqu'elle sera fonctionnelle, elle suivra la définition du prochain paragraphe.

Retourne 1 si la forme géométrique `g` n'au aucune anomalie géométrique, telle que l'auto−intersection ou l'auto−tangence. `IsSimple()` retourne 0 si l'argument n'est pas simple, −{}1 si l'objet est `NULL` .

La description de chaque classe géométrique instanciable est donnée plus tôt dans ce chapitre, et inclut les conditions qui font qu'une forme est considérée comme simple ou pas.

19.5.2.2 Fonctions d'analyse des Point

Un objet `Point` est constitué de ses coordonnées X et Y, qui peuvent être obtenues comme ceci :

X(p)
: Retourne l'absisse du point `p` sous forme d'un nombre à virgule en double précision.

```
mysql> SELECT X(GeomFromText('Point(56.7 53.34)'));
+--------------------------------------+
| X(GeomFromText('Point(56.7 53.34)')) |
+--------------------------------------+
|                                 56.7 |
+--------------------------------------+
```

Y(p)
: Retourne l'ordonnée du point `p` sous forme d'un nombre à virgule en double précision.

```
mysql> SELECT Y(GeomFromText('Point(56.7 53.34)'));
+--------------------------------------+
| Y(GeomFromText('Point(56.7 53.34)')) |
+--------------------------------------+
|                                53.34 |
+--------------------------------------+
```

19.5.2.3 Fonctions d'analyse des lignes LineString

Une ligne `LineString` est constituée de `Point` . Vous pouvez extraire des points particuliers d'une ligne `LineString` , compter le nombre de point qu'elle contient, ou encore calculer sa longueur.

`EndPoint(ls)`
: Retourne le `Point` terminal de la ligne `LineString ls` .

```
mysql> SELECT AsText(EndPoint(GeomFromText('LineString(1 1,2 2,3 3)')));
+------------------------------------------------------------+
| AsText(EndPoint(GeomFromText('LineString(1 1,2 2,3 3)')))  |
+------------------------------------------------------------+
| POINT(3 3)                                                 |
+------------------------------------------------------------+
```

`GLength(ls)`
: Retourne la longueur de la ligne `ls` sous forme d'un nombre à virgule et double précision.

```
mysql> SELECT GLength(GeomFromText('LineString(1 1,2 2,3 3)'));
+--------------------------------------------------+
| GLength(GeomFromText('LineString(1 1,2 2,3 3)')) |
```

```
+-----------------------------------------------+
|                               2.8284271247462 |
+-----------------------------------------------+
```

`IsClosed(ls)`
: Retourne 1 si `ls` est fermée : c'est à dire si `StartPoint()` et `EndPoint()` sont identiques. Retourne 0 si `ls` n'est pas fermée, et −{}1 si l'argument passé est `NULL`.

```
mysql> SELECT IsClosed(GeomFromText('LineString(1 1,2 2,3 3)'));
+------------------------------------------------+
| IsClosed(GeomFromText('LineString(1 1,2 2,3 3)')) |
+------------------------------------------------+
|                                              0 |
+------------------------------------------------+
```

`NumPoints(ls)`
: Retourne le nombre de points dans la ligne `ls`.

```
mysql> SELECT NumPoints(GeomFromText('LineString(1 1,2 2,3 3)'));
+-------------------------------------------------+
| NumPoints(GeomFromText('LineString(1 1,2 2,3 3)')) |
+-------------------------------------------------+
|                                               3 |
+-------------------------------------------------+
```

`PointN(ls,n)`
: Retourne le `n` −ième point de la ligne `ls`. La numérotation des points commence à 1.

```
mysql> SELECT AsText(PointN(GeomFromText('LineString(1 1,2 2,3 3)'),2));
+--------------------------------------------------------+
| AsText(PointN(GeomFromText('LineString(1 1,2 2,3 3)'),2)) |
+--------------------------------------------------------+
| POINT(2 2)                                             |
+--------------------------------------------------------+
```

`StartPoint(ls)`
: Retourne le premier `Point` de la ligne `ls`.

```
mysql> SELECT AsText(StartPoint(GeomFromText('LineString(1 1,2 2,3 3)')));
+-----------------------------------------------------------+
| AsText(StartPoint(GeomFromText('LineString(1 1,2 2,3 3)'))) |
+-----------------------------------------------------------+
| POINT(1 1)                                                |
+-----------------------------------------------------------+
```

Les spécifications OpenGIS définissent aussi les fonctions suivantes, que MySQL n'implémente pas encore :

`IsRing(ls)`
: Retourne 1 si `ls` est un anneau : il faut que `ls` soit fermée (c'est à dire que `StartPoint()` et `EndPoint()` sont identiques), et qu'elle soit simple (ne passe pas par le même point plusieurs fois). Retourne 0 si `ls` n'est pas un anneau, et −{}1 si elle vaut `NULL`.

19.5.2.4 Fonctions d'analyse des lignes MultiLineString

`GLength(mls)`
: Retourne la longueur de l'objet `mls`. La longueur de l'objet `mls` est égale à la somme des longueur de ses éléments.

```
mysql> SELECT GLength(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))'));
+-------------------------------------------------------------+
| GLength(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))')) |
+-------------------------------------------------------------+
|                                             4.2426406871193 |
+-------------------------------------------------------------+
```

`IsClosed(mls)`
: Retourne 1 si `mls` est fermée (c'est à dire que `StartPoint()` et `EndPoint()` sont identique pour chaque objet `LineString` de `mls`). Retourne 0 si `mls` n'est pas fermée et −{}1 si l'objet est `NULL`.

```
mysql> SELECT IsClosed(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))'));
+--------------------------------------------------------------+
| IsClosed(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5 5))')) |
+--------------------------------------------------------------+
|                                                            0 |
+--------------------------------------------------------------+
```

19.5.2.5 Fonctions d'analyse des lignes Polygon

Area(poly)

Retourne un nombre à virgule en double précision représentant l'aire de l'objet `Polygon` `poly`, tel que mesuré dans son référentiel.

```
mysql> SET &#64;poly = 'Polygon((0 0,0 3,3 0,0 0),(1 1,1 2,2 1,1 1))';
mysql> SELECT Area(GeomFromText(&#64;poly));
+---------------------------+
| Area(GeomFromText(&#64;poly)) |
+---------------------------+
|                         4 |
+---------------------------+
```

NumInteriorRings(poly)

Retourne le nombre d'anneau intérieurs de `poly`.

```
mysql> SET &#64;poly =
    -> 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
mysql> SELECT AsText(ExteriorRing(GeomFromText(&#64;poly)));
+-------------------------------------------+
| AsText(ExteriorRing(GeomFromText(&#64;poly))) |
+-------------------------------------------+
| LINESTRING(0 0,0 3,3 3,3 0,0 0)           |
+-------------------------------------------+
```

InteriorRingN(poly,n)

Retourne le `n` −ième anneau intérieur de l'objet `Polygon poly` sous forme d'un objet `LineString`. Ring numbers begin at 1.

```
mysql> SET &#64;poly =
    -> 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
mysql> SELECT AsText(InteriorRingN(GeomFromText(&#64;poly),1));
+----------------------------------------------+
| AsText(InteriorRingN(GeomFromText(&#64;poly),1)) |
+----------------------------------------------+
| LINESTRING(1 1,1 2,2 2,2 1,1 1)              |
+----------------------------------------------+
```

InteriorRingN(poly,n)

Retourne le nombre d'anneaux intérieurs dans l'objet `Polygon poly`.

```
mysql> SET &#64;poly =
    -> 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
mysql> SELECT NumInteriorRings(GeomFromText(&#64;poly));
+---------------------------------------+
| NumInteriorRings(GeomFromText(&#64;poly)) |
+---------------------------------------+
|                                     1 |
+---------------------------------------+
```

19.5.2.6 Fonctions d'analyse des lignes MultiPolygon

`Area(mpoly)`

Retourne la surface de l'objet `MultiPolygon mpoly`, mesuré dans son référentiel.

```
mysql> SELECT Area(GeomFromText('MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1)))'));
+-----------------------------------------------------------------------------------+
| Area(GeomFromText('MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1)))')) |
+-----------------------------------------------------------------------------------+
|                                                                                 8 |
+-----------------------------------------------------------------------------------+
```

Les spécifications OpenGIS définissent aussi les fonctions suivantes, que MySQL n'implémente pas encore :

`Centroid(mpoly)`

Retourne le centre mathématique de l'objet `MultiPolygon mpoly`, sous la forme d'un `Point`. Le résultat n'est pas forcément dans l'objet `MultiPolygon`.

`PointOnSurface(mpoly)`

Retourne un point `Point` qui est dans l'objet `MultiPolygon mpoly`.

19.5.2.7 Fonctions d'analyse des lignes GeometryCollection

`NumGeometries(gc)`

Retourne le nombre de formes géométriques qui constituent l'objet `GeometryCollection gc`.

```
mysql> SELECT NumGeometries(GeomFromText('GeometryCollection(Point(1 1),LineString(2 2, 3 3))'));
+------------------------------------------------------------------------------+
| NumGeometries(GeomFromText('GeometryCollection(Point(1 1),LineString(2 2, 3 3))')) |
+------------------------------------------------------------------------------+
|                                                                            2 |
+------------------------------------------------------------------------------+
```

`GeometryN(gc,n)`
: Retourne le `n` −ième objet constituant l'objet `GeometryCollection gc`. La numérotation commence à 1.

```
mysql> SELECT AsText(GeometryN(GeomFromText('GeometryCollection(Point(1 1),LineString(2 2, 3 3))'),1));
+------------------------------------------------------------------------------------+
| AsText(GeometryN(GeomFromText('GeometryCollection(Point(1 1),LineString(2 2, 3 3))'),1)) |
+------------------------------------------------------------------------------------+
| POINT(1 1)                                                                         |
+------------------------------------------------------------------------------------+
```

19.5.3 Fonctions qui génèrent des formes géométriques à partir d'autres formes

19.5.3.1 Fonctions géométriques qui génèrent de nouvelles formes

Dans la section Fonction d'analyse des propriétés des formes `Geometry`, nous avons déjà discuté de certaines fonctions qui génèrent de nouvelles formes à partir de formes existantes :

- `Envelope(g)`
- `StartPoint(ls)`
- `EndPoint(ls)`
- `PointN(ls,n)`
- `ExteriorRing(poly)`
- `InteriorRingN(poly,n)`
- `GeometryN(gc,n)`

19.5.3.2 Opérateurs géométriques

OpenGIS propose d'autres fonctions qui génèrent des formes géométriques. Elles sont conçues pour servir d'opérateurs géométriques.

Ces fonctions ne sont pas encore implémentées par MySQL. Elles devraient arriver dans les prochaines versions.

`Intersection(g1,g2)`
: Retourne l'ensemble des points qui représentent l'intersection des deux formes `g1` et `g2`.

`Union(g1,g2)`
: Retourne l'ensemble des points qui représentent l'union des deux formes `g1` et `g2`.

`Difference(g1,g2)`
: Retourne l'ensemble des points qui représentent la différence des deux formes `g1` et `g2`.

`SymDifference(g1,g2)`
: Retourne l'ensemble des points qui représentent la différence symétrique des deux formes `g1` et `g2`.

`Buffer(g,d)`
: Retourne l'ensemble des points dont la distance à la forme `g` est inférieure ou égale à `d`.

`ConvexHull(g)`
: Retourne l'enveloppe convexe de la forme géométrique `g`.

19.5.4 Fonctions de tests des relations géométriques entre les formes

Les fonctions décrites dans ces fonctions prennent deux formes géométriques comme argument, et retourne des informations qualitatives ou quantitatives sur leur relation.

19.5.5 Relations avec les Rectangles enveloppes (MBRs)

MySQL fournit des fonctions qui permettent de tester les relations entre les rectangles enveloppes de deux formes géométriques `g1` et `g2` . Il s'agit de :

MBRContains(g1,g2)
: Retourne 1 ou 0 pour indiquer le rectangle enveloppe de `g1` contient celui de `g2` .

```
 mysql> SET &#64;g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
mysql> SET &#64;g2 = GeomFromText('Point(1 1)');
mysql> SELECT MBRContains(&#64;g1,&#64;g2), MBRContains(&#64;g2,&#64;g1);
----------------------+----------------------+
| MBRContains(&#64;g1,&#64;g2) | MBRContains(&#64;g2,&#64;g1) |
+----------------------+----------------------+
|                    1 |                    0 |
+----------------------+----------------------+
```

MBRWithin(g1,g2)
: Retourne 1 ou 0 pour indiquer le rectangle enveloppe de `g1` est à l'intérieur de `g2` .

```
 mysql> SET &#64;g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
mysql> SET &#64;g2 = GeomFromText('Polygon((0 0,0 5,5 5,5 0,0 0))');
mysql> SELECT MBRWithin(&#64;g1,&#64;g2), MBRWithin(&#64;g2,&#64;g1);
+--------------------+--------------------+
| MBRWithin(&#64;g1,&#64;g2) | MBRWithin(&#64;g2,&#64;g1) |
+--------------------+--------------------+
|                  1 |                  0 |
+--------------------+--------------------+
```

MBRDisjoint(g1,g2)
: Retourne 1 ou 0 pour indiquer les rectangles enveloppe de `g1` et `g2` sont disjoints (ils n'ont pas d'intersection).

MBREquals(g1,g2)
: Retourne 1 ou 0 pour indiquer le rectangle enveloppe de `g1` est le même que celui de `g2` .

MBRIntersects(g1,g2)
: Retourne 1 ou 0 pour indiquer le rectangle enveloppe de `g1` et celui de `g2` ont une intersection non vide.

MBROverlaps(g1,g2)
: Retourne 1 ou 0 pour indiquer le rectangle enveloppe de `g1` recouvre une partie de celui de `g2` .

MBRTouches(g1,g2)
: Retourne 1 ou 0 pour indiquer le rectangle enveloppe de `g1` touche celui de `g2` .

19.5.6 Fonctions qui testent les relations géométriques entre les formes

Les spécifications OpenGIS définissent les fonctions suivantes, que MySQL n'implémente pas encore. Elles devraient apparaître prochainement, dans les prochaines versions. Lorsqu'elles seront implémentées, elles fourniront le support complet des fonctions d'analyse spatiales, et non pas un simple support de la géométrie des enveloppes.

Ces fonctions opèrent toujours sur deux formes géométriques `g1` et `g2` .

Contains(g1,g2)
: Retourne 1 ou 0 suivant que `g1` contient complètement `g2` ou pas.

Crosses(g1,g2)
: Retourne 1 si `g1` rencontre `g2` . Retourne `NULL` si `g1` est un `Polygon` ou un `MultiPolygon` , ou si `g2` est un `Point` ou un groupe `MultiPoint` . Otherwise, returns 0.

  " **rencontre** " indique une relation entre deux formes, ayant les propriétés suivantes :

  - Les deux formes ont une intersection non vide.

Leur intersection est une forme géométrique qui a une dimension de moins que le nombre maximum de dimensions des deux formes `g1` et `g2` . L'intersection n'est pas égale à `g1` ou `g2` .

Disjoint(g1,g2)
: Retourne 1 ou 0 pour indiquer si `g1` est géométriquement disjoint de `g2` ou non.

Equals(g1,g2)
: Retourne 1 ou 0 pour indiquer que `g1` est géométriquement égal à `g2` , ou non.

Intersects(g1,g2)
: Retourne 1 ou 0, pour indiquer si `g1` a une intersection non vide avec `g2` ou pas.

Overlaps(g1,g2)
: Retourne 1 ou 0 pour indiquer sir `g1` recouvre `g2` ou pas. Le terme **recouvre** signifie que deux formes géométriques ont une intersection de même dimension que les formes initiales, mais différentes de ces formes.

Touches(g1,g2)
: Retourne 1 ou 0 pour indiquer si `g1` touche `g2` ou pas. Deux formes se **touchent** si leurs intérieurs ont une intersection vide, mais que l'une des deux frontières a une intersection non vide avec la frontière ou l'intérieur de l'autre.

Within(g1,g2)
: Retourne 1 ou 0 pour indiquer si `g1` est à l'intérieur de `g2` .

Distance(g1,g2)
: Retourne la distance la plus faible entre deux points des deux formes, sous forme d'un nombre à virgule et double précision.

Related(g1,g2,pattern_matrix)
: Retourne 1 ou 0, pour indiquer si la relation géométrique spécifiée par `pattern_matrix` existe entre les formes `g1` et `g2` . Retourne −{}1 si les arguments sont `NULL` . Le paramètre `pattern_matrix` est une chaîne. Ses spécifications seront détaillées lorsque la fonction sera codée.

19.6 Optimiser l'analyse géographique

Il est connu que les index accélèrent les recherches dans les bases de données non−géographiques. C'est aussi vrai avec les bases de données géographiques. Avec l'aide d'une grande variété d'index multi−dimensionnels qui ont été conçu pour cela, il est possible d'optimiser les recherches avec des index. Le plus classique est :

- Les requêtes de points, où on recherche les objets qui contiennent un point donné.
- Les requêtes de région, qui recherchent tous les objets ont des zones communes.

MySQL utilise des **R−Trees avec répartition quadratique** pour indexer les colonnes géographiques. Un index géographique est constitué en utilisant le MBR d'une forme géométrique. Dans la plupart des cas, le MBR est le rectangle minimum qui entoure une région. Pour les lignes horizontales ou verticales, le MBR est un rectangle généré dans les chaînes. Pour un point, le MBR est un rectangle dégénéré en un point.

19.6.1 Créer un index géométrique

MySQL peut créer des index géométriques en utilisant une syntaxe similaire à celle utilisée avec les index classiques, mais étendue avec l'attribut `SPATIAL` . Les colonnes géographiques doivent être déclarées comme `NOT NULL` . L'exemple suivant montre comment créer un index géographique :

- Avec `CREATE TABLE` :

```
mysql> CREATE TABLE geom (g GEOMETRY NOT NULL, SPATIAL INDEX(g));
```

- Avec `ALTER TABLE` :

```
mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
```

- Avec `CREATE INDEX` :

```
mysql> CREATE SPATIAL INDEX sp_index ON geom (g);
```

Pour effacer un index géométrique, utilisez `ALTER TABLE` ou `DROP INDEX` :

- Avec `ALTER TABLE` :

```
mysql> ALTER TABLE geom DROP INDEX g;
```

- Avec `DROP INDEX` :

```
mysql> DROP INDEX sp_index ON geom;
```

Exemple : supposons que la table `geom` contient plus de 32000 formes, qui sont stockées dans la colonne `g` , avec le type `GEOMETRY` . La table dispose aussi d'une colonne d'identifiant `fid` , de type `AUTO_INCREMENT` pour stocker des identifiants d'objet.

```
 mysql> SHOW FIELDS FROM geom;
+-------+----------+------+-----+---------+----------------+
| Field | Type     | Null | Key | Default | Extra          |
+-------+----------+------+-----+---------+----------------+
| fid   | int(11)  |      | PRI | NULL    | auto_increment |
| g     | geometry |      |     |         |                |
+-------+----------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

 mysql> SELECT COUNT(*) FROM geom;
+----------+
| count(*) |
+----------+
|    32376 |
+----------+
1 row in set (0.00 sec)
```

Pour ajouter un index géométrique à la colonne `g` , utilisez cette commande :

```
 mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
Query OK, 32376 rows affected (4.05 sec)
Records: 32376  Duplicates: 0  Warnings: 0
```

19.6.2 Utiliser un index géométrique

L'optimiseur vérifie si un index géométrique est disponible et peut être utilisé pour accélérer les requêtes qui utilisent des fonctions comme `MBRContains()` ou `MBRWithin()` dans les clauses `WHERE` . Par exemple, imaginons que vous devons trouver les objets qui sont dans un rectangle donné :

```
 mysql> SELECT fid,AsText(g) FROM geom WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000 16000,30000 15000))'),g);
+-----+-----------------------------------------------------------------------------+
| fid | AsText(g)                                                                   |
+-----+-----------------------------------------------------------------------------+
|  21 | LINESTRING(30350.4 15828.8,30350.6 15845,30333.8 15845,30333.8 15828.8)     |
|  22 | LINESTRING(30350.6 15871.4,30350.6 15887.8,30334 15887.8,30334 15871.4)     |
|  23 | LINESTRING(30350.6 15914.2,30350.6 15930.4,30334 15930.4,30334 15914.2)     |
|  24 | LINESTRING(30290.2 15823,30290.2 15839.4,30273.4 15839.4,30273.4 15823)     |
|  25 | LINESTRING(30291.4 15866.2,30291.6 15882.4,30274.8 15882.4,30274.8 15866.2) |
|  26 | LINESTRING(30291.6 15918.2,30291.6 15934.4,30275 15934.4,30275 15918.2)     |
| 249 | LINESTRING(30337.8 15938.6,30337.8 15946.8,30320.4 15946.8,30320.4 15938.4) |
|   1 | LINESTRING(30250.4 15129.2,30248.8 15138.4,30238.2 15136.4,30240 15127.2)   |
|   2 | LINESTRING(30220.2 15122.8,30217.2 15137.8,30207.6 15136,30210.4 15121)     |
|   3 | LINESTRING(30179 15114.4,30176.6 15129.4,30167 15128,30169 15113)           |
|   4 | LINESTRING(30155.2 15121.4,30140.4 15118.6,30142 15109,30157 15111.6)       |
|   5 | LINESTRING(30192.4 15085,30177.6 15082.2,30179.2 15072.4,30194.2 15075.2)   |
|   6 | LINESTRING(30244 15087,30229 15086.2,30229.4 15076.4,30244.6 15077)         |
|   7 | LINESTRING(30200.6 15059.4,30185.6 15058.6,30186 15048.8,30201.2 15049.4)   |
|  10 | LINESTRING(30179.6 15017.8,30181 15002.8,30190.8 15003.6,30189.6 15019)     |
|  11 | LINESTRING(30154.2 15000.4,30168.6 15004.8,30166 15014.2,30151.2 15009.8)   |
|  13 | LINESTRING(30105 15065.8,30108.4 15050.8,30118 15053,30114.6 15067.8)       |
| 154 | LINESTRING(30276.2 15143.8,30261.4 15141,30263 15131.4,30278 15134)         |
| 155 | LINESTRING(30269.8 15084,30269.4 15093.4,30258.6 15093,30259 15083.4)       |
| 157 | LINESTRING(30128.2 15011,30113.2 15010.2,30113.6 15000.4,30128.8 15001)     |
+-----+-----------------------------------------------------------------------------+
20 rows in set (0.00 sec)
```

Maintenant, vérifions comment cette requête est exécutée, avec la commande `EXPLAIN` :

```
 mysql> EXPLAIN SELECT fid,AsText(g) FROM geom WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000 16000,30000 15000))'),g);
+----+-------------+-------+-------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+-------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | geom  | range | g             | g    |      32 | NULL |   50 | Using where |
+----+-------------+-------+-------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Voyons ce qui se passe si nous n'avions pas utilisé d'index spatial :

```
 mysql> EXPLAIN SELECT fid,AsText(g) FROM g IGNORE INDEX (g) WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000 16000,30000 15000))'),g);
+----+-------------+-------+------+---------------+------+---------+------+-------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows  | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+-------+-------------+
|  1 | SIMPLE      | geom  | ALL  | NULL          | NULL |    NULL | NULL | 32376 | Using where |
+----+-------------+-------+------+---------------+------+---------+------+-------+-------------+
1 row in set (0.00 sec)
```

Exécutons la requête ci−dessus, en ignorant l'index spatial disponible :

```
 mysql> SELECT fid,AsText(g) FROM geom IGNORE INDEX (g) WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000 16000,30000 15000))'),g);
+-----+-------------------------------------------------------------------------+
| fid | AsText(g)                                                               |
+-----+-------------------------------------------------------------------------+
|   1 | LINESTRING(30250.4 15129.2,30248.8 15138.4,30238.2 15136.4,30240 15127.2) |
|   2 | LINESTRING(30220.2 15122.8,30217.2 15137.8,30207.6 15136,30210.4 15121)   |
|   3 | LINESTRING(30179 15114.4,30176.6 15129.4,30167 15128,30169 15113)         |
|   4 | LINESTRING(30155.2 15121.4,30140.4 15118.6,30142 15109,30157 15111.6)     |
|   5 | LINESTRING(30192.4 15085,30177.6 15082.2,30179.2 15072.4,30194.2 15075.2) |
|   6 | LINESTRING(30244 15087,30229 15086.2,30229.4 15076.4,30244.6 15077)       |
|   7 | LINESTRING(30200.6 15059.4,30185.6 15058.6,30186 15048.8,30201.2 15049.4) |
|  10 | LINESTRING(30179.6 15017.8,30181 15002.8,30190.8 15003.6,30189.6 15019)   |
|  11 | LINESTRING(30154.2 15000.4,30168.6 15004.8,30166 15014.2,30151.2 15009.8) |
|  13 | LINESTRING(30105 15065.8,30108.4 15050.8,30118 15053,30114.6 15067.8)     |
|  21 | LINESTRING(30350.4 15828.8,30350.6 15845,30333.8 15845,30333.8 15828.8)   |
|  22 | LINESTRING(30350.6 15871.4,30350.6 15887.8,30334 15887.8,30334 15871.4)   |
|  23 | LINESTRING(30350.6 15914.2,30350.6 15930.4,30334 15930.4,30334 15914.2)   |
|  24 | LINESTRING(30290.2 15823,30290.2 15839.4,30273.4 15839.4,30273.4 15823)   |
|  25 | LINESTRING(30291.4 15866.2,30291.6 15882.4,30274.8 15882.4,30274.8 15866.2) |
|  26 | LINESTRING(30291.6 15918.2,30291.6 15934.4,30275 15934.4,30275 15918.2)   |
| 154 | LINESTRING(30276.2 15143.8,30261.4 15141,30263 15131.4,30278 15134)       |
| 155 | LINESTRING(30269.8 15084,30269.4 15093.4,30258.6 15093,30259 15083.4)     |
| 157 | LINESTRING(30128.2 15011,30113.2 15010.2,30113.6 15000.4,30128.8 15001)   |
| 249 | LINESTRING(30337.8 15938.6,30337.8 15946.8,30320.4 15946.8,30320.4 15938.4) |
+-----+-------------------------------------------------------------------------+
20 rows in set (0.46 sec)
```

Lorsque l'index n'est pas utilisé, le temps d'exécution de la requête passe de 0.00 seconde à 0.46 second.Dans les prochaines versions, les index spatiaux seront aussi utilisés pour optimiser d'autres fonctions. Fonctions de tests des relations géométriques entre les formes .

19.7 MySQL compatibilité avec GIS

19.7.1 Les fonctionnalités de GIS que nous n'avons pas encore implémenté

Vues de meta−données
: Les spécifications OpenGIS proposent plusieurs meta−données supplémentaires. Par exemple, un système de vue appelé `GEOMETRY_COLUMNS` contient une description les colonnes géométriques, une ligne pour chaque colonne géométrique dans la base.

Les fonctions OpenGIS `Length()` sur les `LineString` et `MultiLineString` sont appelées `GLength()` en MySQL
: Le problème est que ce nom est en conflit avec la fonction SQL existante `Length()` qui calcule la taille d'une chaîne de caractères, et il n'est pas possible de faire la différence entre le contexte géographique ou textuel. Nous devons résoudre ce problème, ou trouver un autre nom à cette fonction.

20 Procédures stockées et fonctions

Les procédures stockées et les fonctions sont de nouvelles fonctionnalités de MySQL version 5.0. Une procédure stockées est un jeu de commandes SQL qui réside sur le serveur. Une fois qu'elle sont enregistrées, les clients n'ont pas besoin de soumettre chaque commande individuellement, mais peuvent les lancer d'un seul coup.

Les procédures stockées fournissent un gain de performances, car moins d'informations sont échangées entre le serveur et le client. En échange, cela augmente la charge du serveur, car ce dernier doit réaliser plus de travail. Souvent, il y a de nombreux clients, mais peut de serveurs.

Les procédures stockées permettent aussi l'utilisation de bibliothèques et de fonctions sur le serveur. Les langages de programmation modernes ont déjà intégré ce type de concept, et l'utilisation de ces langages de programmation externes reste valable et utile, en dehors de la base de données.

Quelques situations où les procédures stockées sont utiles :

- Lorsque plusieurs applications clientes sont écrites dans différents langages sur différentes plates−formes, et utilisent le serveur comme point d'interaction.
- Lorsque la sécurité est prioritaire. Les banques, par exemple, utilisent les procédures stockées pour toutes les opérations standards. Cela conduit à un environnement cohérent et sécurisé, car les procédures assurent que les opérations sont correctement faîtes et enregistrées. Dans une telle configuration, les applications et les utilisateurs n'ont aucun accès direct aux tables, mais passent par des procédures stockées pre−définies.

MySQL suit la syntaxe de la norme SQL:2003 pour les procédures stockées, qui est aussi utilisée par IBM dans DB2. La compatibilité avec les autres langages de procédures stockées, comme PL/SQL ou T−SQL sera ajouté ultérieurement.

L'implémentation des procédures stockées de MySQL est en cours de développement. Toutes les syntaxes décrites dans ce chapitre sont supportées, et les limitations ou extensions sont documentés lorsque c'est nécessaire.

Les procédures stockées requièrent la table `proc` dans la base `mysql` . Cette table est créée durant l'installation de MySQL 5.0. Si vous passez à MySQL 5.0 depuis une ancienne version, assurez−vous de bien mettre à jour vos tables de droits, et que la table `proc` existe. Mise à jour des tables de droits .

20.1 Syntaxe des procédures stockées

Les procédures stockées et les fonctions sont créées avec les commandes `CREATE PROCEDURE` et `CREATE FUNCTION` . Une procédure est appelée avec la commande `CALL` , et ne peut retourner de valeur que via les variables de retour. Les fonctions peuvent retourner une valeur scalaire, et être appelée depuis une commande, tout comme toute autre fonction. Les procédures stockées peuvent appeler une autre routine stockée. Une routine est une procédure stockée ou une fonction.

Actuellement, MySQL préserve uniquement le contexte de la base par défaut. C'est à dire que si vous utilisez la commande `USE dbname` dans une procédure, le contexte initial sera restauré à la fin de la procédure.

Une routine hérite de la base de données par défaut de l'utilisateur appelant, et donc, il est recommande de commencer les routines avec la commande `USE dbname` , ou de spécifier explicitement les tables et les bases de données qui sont utilisées : i.e. `base.table` .

MySQL supporte une extension très pratique qui permet d'utiliser des expressions régulières dans les commandes `SELECT` (c'est à dire, sans utiliser de curseur ou de variable locales). Le résultat d'une telle requête est simplement envoyé directement au client.

Plusieurs commandes `SELECT` génèreront plusieurs jeux de résultats, et le client doit utiliser la bibliothèques qui supporte les résultats multiples. Cela signifie que vous devez utiliser une bibliothèque cliente de version 4.1 ou plus récente.

La section suivante décrit la syntaxe à utiliser pour créer, modifier, détruire et appeler des procédures stockées.

20.1.1 Gérer des procédures stockées

20.1.1.1 CREATE PROCEDURE et CREATE FUNCTION

```
 CREATE PROCEDURE sp_name ([parameter[,...]])
[characteristic ...] routine_body

 CREATE FUNCTION sp_name ([parameter[,...]])
[RETURNS type]
[characteristic ...] routine_body

 paramètre :
  [ IN | OUT | INOUT ] param_name type

 type :
  Any valid MySQL data type

 characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | SQL SECURITY &#123;DEFINER | INVOKER&#125;
  | COMMENT string

 routine_body :
  Commande(s) SQL valide(s)
```

La clause `RETURNS` peut être spécifiée uniquement pour une `FUNCTION` . Elle sert à indiquer le type de retour de la fonction, et le corps de la fonction doit contenir une instruction `RETURN value` .

La liste de paramètre entre parenthèses est obligatoire. S'il n'y a pas de paramètre, une liste vide sous la forme `()` doit être utilisée. Chaque paramètre est un paramètre de type `IN` par défaut. Pour spécifier un autre type, utilisez les mots `OUT` ou `INOUT` avant le nom du paramètre. Spécifier `IN` , `OUT` ou `INOUT` n'est valable que pour une `PROCEDURE` .

L'instruction `CREATE FUNCTION` est utilisée dans les anciennes versions de MySQL pour créer des `UDF` (`User Defined Functions` , fonctions utilisateur). Ajout de nouvelles fonctions à MySQL . Les `UDF` sont toujours supportées, même avec la présence des procédures stockées. Une `UDF` peut être considérée comme une fonction stockée. Cependant, notez que les `UDF` et les fonctions stockées partagent le même espace de noms.

Un framework pour développer des procédures stockées externes sera prochainement présenté. Il permettra d'écrire des procédures stockées dans d'autres langages que SQL. Il est probable que l'un des premiers langages supportés sera PHP, car le moteur PHP est compact, compatible avec les threads et peut être facilement intégré. Comme ce framework sera public, il est probable que bien d'autres langages soient supportés.

Une fonction est considérée comme ``déterministe'' si elle retourne toujours le même résultat pour les mêmes paramètres d'entrée. Sinon, elle est considérée comme ``non déterministe''. L'optimiseur peut utiliser cette propriété. Actuellement, l'attribut `DETERMINISTIC` est accepté, mais il n'est pas encore utilisé.

L'attribut `SQL SECURITY` peut être utilisé pour spécifier si la routine doit être exécutée avec les droits de l'utilisateur qui l'a créé ou avec ceux de celui qui appelle la fonction. La valeur par défaut est `DEFINER` . Cette fonctionnalité est nouvelle en SQL:2003.

MySQL n'utilise pas le droit `GRANT EXECUTE` . Pour le moment, si une procédure `p1()` utilise la table `t1` , l'appelant doit avoir les droits sur la table `t1` afin que la procédure `p1()` puisse réussir.

MySQL stocke la configuration `SQL_MODE` en effet au moment de la création de la procédure, et l'utilisera toujours lors de l'exécution de la procédure.

La clause `COMMENT` est une extension MySQL, et peut servir à décrire la procédure stockée. Cette information est affichée par les commandes `SHOW CREATE PROCEDURE` et `SHOW CREATE FUNCTION` .

MySQL permet aux routines de contenir des commandes DDL, telle que `CREATE` et `DROP` , et des transactions SQL, comme `COMMIT` . Ce n'est pas obligatoire selon le standard et c'est donc une extension spécifique.

Note : Actuellement, les fonctions stockées `FUNCTION` s ne doivent pas contenir de références aux tables. Notez que cela inclut aussi les commandes `SET` , mais pas les commandes `SELECT` . Cette limitation sera supprimée aussitôt que possible.

L'exemple suivant est une procédure stockée simple, qui utilise un paramètre de sortie `OUT` . L'exemple utilise la commande `delimiter` du client `mysql` pour modifier le délimiteur de commande avant de définir la procédure. Cela permet au délimiteur `;` d'être utilisé dans le corps de la procédure, plutôt que d'être interprété par le client `mysql` .

```
 mysql> delimiter |

 mysql> CREATE PROCEDURE simpleproc (OUT param1 INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO param1 FROM t;
    -> END
    -> |
Query OK, 0 rows affected (0.00 sec)

 mysql> CALL simpleproc(&#64;a)|
Query OK, 0 rows affected (0.00 sec)

 mysql> SELECT &#64;a|
+------+
| &#64;a   |
+------+
| 3    |
+------+
1 row in set (0.00 sec)
```

L'exemple suivant est une fonction qui prend un paramètre, effectue une opération avec une fonction SQL, et retourne le résultat :

```
 mysql> delimiter |

 mysql> CREATE FUNCTION bonjour (s CHAR(20)) RETURNS CHAR(50)
    -> RETURN CONCAT('Bonjour, ',s,'!');
    -> |
Query OK, 0 rows affected (0.00 sec)

 mysql> SELECT bonjour('le monde')|
+---------------------+
| bonjour('le monde') |
+---------------------+
| Bonjour, le monde!  |
+---------------------+
1 row in set (0.00 sec)
```

20.1.1.2 ALTER PROCEDURE et ALTER FUNCTION

```
ALTER PROCEDURE | FUNCTION sp_name [characteristic ...]

characteristic:
    NAME newname
  | SQL SECURITY &#123;DEFINER | INVOKER&#125;
  | COMMENT string
```

Cette commande peut être utilisée pour renommer une procédure stockée ou une fonction, et pour en changer les caractéristiques. Plusieurs modifications peut être spécifiées dans une commande `ALTER PROCEDURE` et `ALTER FUNCTION` .

20.1.1.3 DROP PROCEDURE et DROP FUNCTION

```
DROP PROCEDURE | FUNCTION [IF EXISTS] sp_name
```

Cette commande sert à effacer une procédure stockée ou une fonction. C'est à dire que la routine spécifiée est supprimée du serveur.La clause `IF EXISTS` est une extension de MySQL. Elle permet d'éviter une erreur si la routine n'existe pas. Une alerte est alors produite, et peut être lue avec `SHOW WARNINGS` .

20.1.1.4 SHOW CREATE PROCEDURE et SHOW CREATE FUNCTION

```
SHOW CREATE PROCEDURE | FUNCTION sp_name
```

Cette commande est une extension MySQL. Similaire à `SHOW CREATE TABLE` , elle retourne la chaîne exacte qui permet de recréer la routine.

20.1.2 SHOW PROCEDURE STATUS et SHOW FUNCTION STATUS

```
SHOW PROCEDURE | FUNCTION STATUS [LIKE pattern]
```

Cette commande est une extension MySQL. Elle retourne les caractéristiques de la routine, comme son nom, type, créateur, ainsi que les dates de création et de modification. Si le paramètre pattern n'est pas fourni, les informations sur toutes les procédures stockées ou fonctions sont listées, en fonction de la commande exacte que vous utilisez.

20.1.3 CALL

```
CALL sp_name([parameter[,...]])
```

La commande `CALL` sert à appeler une routine qui a été définie précédemment avec `CREATE PROCEDURE` .

20.1.4 La commande composée BEGIN ... END

```
 [begin_label:] BEGIN
 statement(s)
END [end_label]
```

Les routines peuvent contenir des commandes multiples, en utilisant le bloc de commande `BEGIN ... END` .

`begin_label` et `end_label` doivent être identiques, s'ils sont spécifiés.

Notez bien que la clause optionnelle `[NOT] ATOMIC` n'est pas encore supportée. Cela signifie qu'il n'y a pas de début de transaction au début du bloc, et que la clause `BEGIN` sera utilisé sans affecté la transaction courante.

Les commandes multiples requièrent un client capable d'envoyer des requêtes contenant le caractère `';'` . C'est géré dans le client en ligne de commande `mysql` , avec la commande `delimiter` . En changeant le caractère de fin de requête `';'` pour le remplacer par `'|'` permet à `';'` d'être utilisé dans le corps de la routine.

20.1.5 La commande DECLARE

La commande `DECLARE` sert à définir différents objets locaux dans une routine : variables locales (Les variables de procédures stockées), conditions et gestionnaires (Conditions et gestionnaires), curseurs (Curseurs). Les commandes `SIGNAL` et `RESIGNAL` ne sont pas supportées pour le moment.

`DECLARE` ne peut être utilisé dans un bloc `BEGIN ... END` , et doit intervenir au début de la routine, avant tout autre commande.

20.1.6 Les variables dans les procédures stockées

Vous pouvez déclarer et utiliser des variables dans une routine.

20.1.6.1 DECLARE : déclarer une variable locale

```
DECLARE var_name[,...] type [DEFAULT value]
```

Cette commande sert à déclarer des variables globales. Le scope des variables est le bloc `BEGIN ... END` .

20.1.6.2 Commande d'affectation de variables SET

```
SET variable = expression [,...]
```

La commande `SET` des procédures stockées est une version étendue de la commande `SET` classique. Les variables référencées peuvent être déclarées dans le contexte de la routine ou comme variables globales.

La commande `SET` des procédures stockées est implémentée comme une sous−partie de la syntaxe `SET` . Cela permet la syntaxe étendue `SET a=x, b=y, ...` , où plusieurs types de variables (locales, serveur, globale ou session) sont mélangées. Cela permet aussi la combinaison de variables locales et d'option système qui n'ont de sens qu'au niveau global du serveur : dans ce cas, les options sont acceptées mais ignorées.

20.1.6.3 Syntaxe de SELECT ... INTO

```
SELECT column[,...] INTO variable[,...] table_expression
```

Cette syntaxe de `SELECT` stocke les colonnes sélectionnées dans des variables. Par conséquent, une seule ligne doit être lue. Cette commande est aussi extrêmement utile lorsqu'elle est utilisée avec des curseurs.

```
SELECT id,data INTO x,y FROM test.t1 LIMIT 1;
```

20.1.7 Conditions et gestionnaires

Certaines conditions peuvent exiger des gestions spécifiques. Ces conditions peuvent être liées à des erreurs, ou au déroulement de la routine.

20.1.7.1 DECLARE une condition

```
DECLARE condition_name CONDITION FOR condition_value

condition_value:
   SQLSTATE [VALUE] sqlstate_value
 | mysql_error_code
```

Cette commande spécifie les conditions qui demandent une gestion spécifique. Elle associe un nom avec une erreur spécifique. Ce nom peut être utilisé ultérieurement dans une commande `DECLARE HANDLER` . `DECLARE` pour les gestionnaires .

En plus des valeurs SQLSTATE, les codes d'erreur MySQL sont aussi supportés.

20.1.7.2 DECLARE un gestionnaire

```
DECLARE handler_type HANDLER FOR condition_value[,...] sp_statement

handler_type:
   CONTINUE
 | EXIT
 | UNDO

condition_value:
   SQLSTATE [VALUE] sqlstate_value
 | condition_name
 | SQLWARNING
 | NOT FOUND
 | SQLEXCEPTION
 | mysql_error_code
```

Cette commande spécifie les gestionnaires qui vont traiter une ou plusieurs conditions. Si une de ces conditions survient, le gestionnaire associé est appelé.Pour un gestionnaire `CONTINUE` , l'exécution de la routine courante continue après l'exécution du gestionnaire. Pour un gestionnaire `EXIT` , l'exécution de la routine est terminée. Le gestionnaire `UNDO` n'est pas encore supporté. Actuellement, `UNDO` se comporte comme `CONTINUE` .

- `SQLWARNING` est un raccourci pour toutes les codes `SQLSTATE` qui commencent par 01.
- `NOT FOUND` est un raccourci pour toutes les codes `SQLSTATE` qui commencent par 02.
- `EXCEPTION` est un raccourci pour toutes les codes `SQLSTATE` qui ne sont pas représenté par `SQLWARNING` ou `NOT FOUND` .

En plus des valeurs `SQLSTATE` , les codes d'erreur `MySQL` sont aussi supportés.

Par exemple :

```
 mysql> CREATE TABLE test.t (s1 int,primary key (s1));
Query OK, 0 rows affected (0.00 sec)

 mysql> delimiter |
```

```
 mysql> CREATE PROCEDURE handlerdemo ()
    -> BEGIN
    ->   DECLARE CONTINUE HANDLER FOR '23000' SET &#64;x2 = 1;
    ->   set &#64;x = 1;
    ->   INSERT INTO test.t VALUES (1);
    ->   set &#64;x = 2;
    ->   INSERT INTO test.t VALUES (1);
    ->   SET &#64;x = 3;
    -> END;
    -> |
Query OK, 0 rows affected (0.00 sec)

 mysql> CALL handlerdemo()|
Query OK, 0 rows affected (0.00 sec)

 mysql> SELECT &#64;x|
    +------+
    | &#64;x   |
    +------+
    | 3    |
    +------+
    1 row in set (0.00 sec)
```

Notez que `@x` vaut 3, ce qui signifie que `MySQL` a exécuté la procédure jusqu'à la fin. Si la ligne `DECLARE CONTINUE HANDLER FOR '23000' SET @x2 = 1;` était absente, `MySQL` aurait pris le chemin par défaut (`EXIT`) après l'échec du second `INSERT` , dû à la contrainte de `PRIMARY KEY` , et `SELECT @x` aurait retourné 2.

20.1.8 Curseurs

Des curseurs simples sont supportés dans les routines. La syntaxe est la même que dans le SQL intégré. Les curseurs sont actuellement assensible, sans scroll et en lecture seule. Les curseurs assensibles signifie que le curseur peut ou pas faire une copie de la table de résultat.

Par exemple :

```
 CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE a CHAR(16);
  DECLARE b,c INT;

   OPEN cur1;
  OPEN cur2;

   REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
       IF b < c THEN
          INSERT INTO test.t3 VALUES (a,b);
       ELSE
          INSERT INTO test.t3 VALUES (a,c);
       END IF;
    END IF;
  UNTIL done END REPEAT;

   CLOSE cur1;
  CLOSE cur2;
END
```

20.1.8.1 Déclaration des curseurs

```
 DECLARE cursor_name CURSOR FOR sql_statement
```

Plusieurs curseurs peuvent être définit dans une routine, mais chacun doit avoir un nom distinct.

20.1.8.2 Commande de curseur OPEN

```
OPEN cursor_name
```

Cette commande ouvre un curseur déclaré précédemment.

20.1.8.3 Commande de curseur FETCH

```
FETCH cursor_name
```

Cette commande lit la prochaine ligne (si elle existe), en utilisant un curseur ouvert, et avance le pointeur de curseur.

20.1.8.4 Commande de curseur CLOSE

```
CLOSE cursor_name
```

Cette commande clôt le curseur précédemment ouvert.

20.1.9 Instructions de contrôle

Les instructions `IF` , `CASE` , `LOOP` , `WHILE` , `ITERATE` et `LEAVE` sont toutes supportées.

Ces instructions peuvent contenir des commandes simples, ou des blocs de commandes `BEGIN ... END` . Les instructions peuvent être imbriquées.

Les boucles `FOR` ne sont pas supportées actuellement.

20.1.9.1 Commande IF

```
 IF search_condition THEN statement(s)
[ELSEIF search_condition THEN statement(s)]
...
[ELSE statement(s)]
END IF
```

`IF` implémente une instruction de condition simple. Si `search_condition` est vrai, la commande SQL correspondante est exécutée. Si `search_condition` est faux, la commande dans la clause `ELSE` est exécutée.

Notez aussi qu'il y a une fonction `IF()` . Instructions de contrôles .

20.1.9.2 Commande CASE

```
 CASE case_value
    WHEN when_value THEN statement
    [WHEN when_value THEN statement ...]
    [ELSE statement]
END CASE
```

ou :

```
 CASE
    WHEN search_condition THEN statement
    [WHEN search_condition THEN statement ...]
```

```
    [ELSE statement]
END CASE
```

`CASE` implémente une structure conditionnelle complexe. Si un des conditions `search_condition` est vraie, la commande SQL correspondante est exécutée. Si aucune condition n'est vérifiée, la commande SQL de la clause `ELSE` est exécuté.

Note : la syntaxe de la commande `CASE` à l'intérieure d'une procédure stockée diffère légèrement de l'expression SQL `CASE` . La commande `CASE` ne peut pas avoir de clause `ELSE NULL` , et l'instruction se termine avec `END CASE` au lieu de `END` . Instructions de contrôles .

20.1.9.3 Commande LOOP

```
 [begin_label:] LOOP
  statement(s)
END LOOP [end_label]
```

`LOOP` implémente une boucle, permettant l'exécution répétée d'un groupe de commande. Les commandes à l'intérieure de la boucle sont exécutée jusqu'à ce que la boucle se termine, généralement lorsqu'elle atteint la commande `LEAVE` . `begin_label` et `end_label` doivent être identiques, si les deux sont spécifiés.

20.1.9.4 Commande LEAVE

```
 LEAVE label
```

Cette commande sert à sortir d'une instruction de contrôle.

20.1.9.5 Commande ITERATE

```
 ITERATE label
```

`ITERATE` ne peut être utilisée qu'à l'intérieur d'une boucle `LOOP` , `REPEAT` ou `WHILE` . `ITERATE` signifie ``exécute encore une fois la boucle.''

For example:

```
 CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN ITERATE label1; END IF;
    LEAVE label1;
  END LOOP label1;
  SET &#64;x = p1;
END
```

20.1.9.6 Commande REPEAT

```
 [begin_label:] REPEAT
  statement(s)
UNTIL search_condition
END REPEAT [end_label]
```

Les commandese à l'intérieur d'une commande `REPEAT` sont répétées jusqu'à ce que la condition `search_condition` soit vraie. `begin_label` et `end_label` doivent être identiques, s'ils sont fournis.

Par exemple :

```
 mysql> delimiter |

 mysql> CREATE PROCEDURE dorepeat(p1 INT)
    -> BEGIN
    ->   SET &#64;x = 0;
    ->   REPEAT SET &#64;x = &#64;x + 1; UNTIL &#64;x > p1 END REPEAT;
    -> END
    -> |
Query OK, 0 rows affected (0.00 sec)

 mysql> CALL dorepeat(1000)|
Query OK, 0 rows affected (0.00 sec)

 mysql> SELECT &#64;x|
+------+
| &#64;x   |
+------+
| 1001 |
+------+
1 row in set (0.00 sec)
```

20.1.9.7 Syntaxe WHILE

```
 [begin_label:] WHILE search_condition DO
  statement(s)
END WHILE [end_label]
```

Les commandes dans l'instruction `WHILE` sont répétées tant que la condition `search_condition` est vraie. `begin_label` et `end_label` doivent être identiques, s'ils sont spécifiés.

Par exemple :

```
 CREATE PROCEDURE dowhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;

   WHILE v1 > 0 DO
    ...
    SET v1 = v1 - 1;
  END WHILE;
END
```

21 API MySQL

Ce chapitre décrit les interfaces disponibles pour MySQL, où les trouver, et comment les utiliser. L'API C est celle qui est couverte le plus en détail, puisqu'elle est développée par l'équipe MySQL, et sert de base à toutes les autres API.

21.1 Utilitaires de développement des programmes MySQL

Cette section décrit certains utilitaires que vous pouvez trouver utiles, lors du développement de programmes MySQL.

msql2mysql
: Un script Shell qui convertit un programme `mSQL` en MySQL. Il ne gère pas toutes les situations, mais c'est un bon début.

mysql_config
: Un script Shell qui produit les options nécessaires lors de la compilation de programmes MySQL.

21.1.1 msql2mysql , convertit des programmes mSQL vers MySQL

Initialement, l'API C de MySQL a été développée pour être très similaire à celle du serveur mSQL. A cause de cela, les programmes mSQL peuvent souvent être convertis facilement à utiliser MySQL, en changeant simplement les noms de l'API C.

L'utilitaire `msql2mysql` fait cette conversion, et transforme les appels C mSQL en leurs équivalents MYSQL.

`msql2mysql` convertit le fichier sur place : il est recommandé de faire une copie avant de lancer la conversion. Par exemple, utilisez `msql2mysql` comme ceci :

```
 shell> cp client-prog.c client-prog.c.orig
shell> msql2mysql client-prog.c
client-prog.c converted
```

Puis, examinez le fichier `client-prog.c` et faites toutes les modifications post−conversion nécessaires. `msql2mysql` utilise l'utilitaire `replace` pour faire les substitutions. L'utilitaire de remplacement `replace` .

21.1.2 mysql_config lit les options de compilations du client MySQL

`mysql_config` vous indique des informations pratiques pour compiler votre client MySQL et le connecter au serveur.

`mysql_config` supporte les options suivantes :

`--cflags`
: Options de compilations utilisées pour trouver les fichiers inclus.

`--include`
: Options du compilateur pour trouver les fichiers d'inclusion MySQL. (Normalement, il faut utiliser `--cflags` au lieu de cette commande)

`--libs`
: Les bibliothèques et options requises pour compiler avec la bibliothèque client MySQL.

`--libs_r`

Les bibliothèques et options requises pour la compilation avec la sécurité thread de la bibliothèque client MySQL.

`--socket`
: Le nom par défaut de la socket, défini lors de la configuration de MySQL.

`--port`
: Le numéro de port par défaut, défini lors de la configuration de MySQL.

`--version`
: Le numéro de version et la version de la distribution MySQL.

`--libmysqld-libs`
: Les bibliothèques et options requises pour compiler avec la bibliothèque intégrée MySQL.

Si vous exécutez `mysql_config` sans aucune option, il va afficher toutes les options qu'il supporte, ainsi que la valeur de toutes les options :

```
shell> mysql_config
Usage: /usr/local/mysql/bin/mysql_config [options]
Options:
  --cflags         [-I/usr/local/mysql/include/mysql -mcpu=pentiumpro]
  --include        [-I/usr/local/mysql/include/mysql]
  --libs           [-L/usr/local/mysql/lib/mysql -lmysqlclient -lz
                    -lcrypt -lnsl -lm -L/usr/lib -lssl -lcrypto]
  --libs_r         [-L/usr/local/mysql/lib/mysql -lmysqlclient_r
                    -lpthread -lz -lcrypt -lnsl -lm -lpthread]
  --socket         [/tmp/mysql.sock]
  --port           [3306]
  --version        [4.0.16]
  --libmysqld-libs [-L/usr/local/mysql/lib/mysql -lmysqld -lpthread -lz
                    -lcrypt -lnsl -lm -lpthread -lrt]
```

Vous pouvez utiliser `mysql_config` dans une ligne de commande pour inclure la valeur qui sera affichée par une option. Par exemple, pour compiler un client MySQL, utilisez `mysql_config` comme ceci :

```
CFG=/usr/local/mysql/bin/mysql_config
sh -c "gcc -o progname `$CFG --cflags` progname.c `$CFG --libs`"
```

Lorsque vous utilisez `mysql_config` de cette manière, assurez vous de l'invoquer entre des guillemets obliques ('`'). Cela indique que le Shell doit exécuter cette expression, et remplacer le résultat dans la commande.

21.2 API MySQL C

L'interface C est distribuée avec MySQL. Elle est inclues dans la bibliothèque `mysqlclient` et permet aux programmes écrits en C d'accéder à la base de données.

De nombreux client de la distribution source de MySQL sont écrits en C. Si vous recherchez des exemples d'illustration de l'utilisation de l'interface C, étudiez donc ces clients. Vous pouvez les trouver dans le dossier `clients` de la distribution source MySQL.

La plupart des autres interfaces clientes (sauf Connector/J) utilisent la bibliothèque `mysqlclient` pour communiquer avec le serveur MySQL. Cela signifie que, par exemple, que vous pouvez exploiter adroitement de nombreux autres variables qui sont utilisées par les autres programmes clients, car elles sont utilisées dans la bibliothèque. Voyez Clients MySQL et utilitaires , pour une liste de ces variables.

Le client a une taille maximale de buffer de communication. La taille du buffer qui est allouée au lancement 16 ko bytes, est automatiquement augmentée jusqu'à la taille maximum de 16 Mo. Comme la taille du buffer croit à la demande, vous ne consommerez pas de ressources supplémentaires en augmentant la limite maximum par défaut. Cette limite sert surtout à s'assurer de bien identifier des problèmes de requêtes erronées ou des paquets de communication.

Le buffer de communication doit être assez grand pour contenir une commande SQL complète (dans le sens du trafic allant vers le serveur), et une ligne complète (dans le sens du trafic allant vers le client). Chaque buffer de threads est dynamiquement agrandi pour gérer les requêtes jusqu'à la taille limite. Par exemple, si vous avez des valeurs de type `BLOB` qui contient jusqu'à 16 Mo de données, vous devez avoir un buffer de communication dont la limite doit être au minimum de 16 Mo, aussi bien sur le serveur que sur le client. La limite par défaut sur le client est de 16 Mo mais elle est de 1 Mo sur le serveur. Vous pouvez augmenter cette valeur en modifiant le paramètre `max_allowed_packet` lorsque le serveur est lancé. Choix des paramètres du serveur .

Le serveur MySQL réduit la taille des buffers de communication à `net_buffer_length` octets après chaque requête. Pour les clients, le buffer est associé à une connexion, et les ressources ne seront libérées que lorsque cette dernière sera refermée.

Pour la programmation avec les threads, voyez la section Comment programmer une client threadé . Pour la création d'applications qui incluent le serveur et le client dans le même programme, et ne communiquent avec aucun serveur externe, voyez libmysqld, la bibliothèque intégrée MySQL .

21.2.1 Types de données de l'API C

`MYSQL`
: Cette structure représente un gestionnaire de connexion à la base de données. Elle est utilisée dans la plupart des fonctions MySQL.

`MYSQL_RES`
: Cette structure représente le résultat d'une requête qui retourne des lignes (`SELECT` , `SHOW` , `DESCRIBE` , `EXPLAIN`). L'information retournée par une requête est appelée **jeu de résultats** dans le reste de cette section.

`MYSQL_ROW`
: C'est une représentation sûre pour les types d'une ligne de données. Elle est actuellement implémentée en tant que tableau de chaîne à octets comptés. (Vous ne pouvez la traiter en tant que chaîne terminée par une valeur nulle si les valeurs du champ peuvent contenir des données binaires, car de telles valeurs peuvent contenir elles−mêmes des octets nuls.) Les lignes sont obtenues en appelant `mysql_fetch_row()` .

`MYSQL_FIELD`
: Cette structure contient des informations à propos du champ, tel que son nom, son type, et sa taille. Ses membres sont décrit en plus de détails ici. Vous pouvez obtenir une structure `MYSQL_FIELD` pour chaque champ en appelant plusieurs fois `mysql_fetch_field()` . Les valeurs des champs ne font pas partie de la structure; elles sont contenues dans une structure `MYSQL_ROW` .

`MYSQL_FIELD_OFFSET`
: C'est une représentation sûre des types pour les index dans une liste de champs MySQL. (Utilisés par `mysql_field_seek()` .) Les index sont des numéros de champs, dans une ligne, commençant à zéro.

`my_ulonglong`
: Le type utilisé pour le nombre de lignes et pour `mysql_affected_rows()` , `mysql_num_rows()` , et `mysql_insert_id()` . Ce type fournit une échelle allant de `0` à `1.84e19` .Sur quelques systèmes, essayer d'écrire la valeur d'un type `my_ulonglong` ne fonctionnera pas. Pour écrire une telle valeur, convertissez là en `unsigned long` et utilisez un format d'impression `%lu` . Exemple :

```
printf ("Nombre de lignes : %lu\n", (unsigned long) mysql_num_rows(result));
```

La structure `MYSQL_FIELD` contient les membres listés ici :

`char * name`
: Le nom du champ, une chaîne terminée par une valeur nulle.

`char * table`
: Le nom de la table contenant ce champ, s'il n'est pas calculé. Pour les champs calculés, la valeur de `table` est une chaîne vide.

`char * def`
: La valeur par défaut de ce champ, en tant que chaîne terminée par une valeur nulle. Ce n'est définit que si vous utilisez `mysql_list_fields()` .

`enum enum_field_types type`

Le type du champ. La valeur de `type` peut être l'une des suivantes :

Valeur de type	Description du type
FIELD_TYPE_TINY	Champ TINYINT
FIELD_TYPE_SHORT	Champ SMALLINT
FIELD_TYPE_LONG	Champ INTEGER
FIELD_TYPE_INT24	Champ MEDIUMINT
FIELD_TYPE_LONGLONG	Champ BIGINT
FIELD_TYPE_DECIMAL	Champ DECIMAL ou NUMERIC
FIELD_TYPE_FLOAT	Champ FLOAT
FIELD_TYPE_DOUBLE	Champ DOUBLE ou REAL
FIELD_TYPE_TIMESTAMP	Champ TIMESTAMP
FIELD_TYPE_DATE	Champ DATE
FIELD_TYPE_TIME	Champ TIME
FIELD_TYPE_DATETIME	Champ DATETIME
FIELD_TYPE_YEAR	Champ YEAR
FIELD_TYPE_STRING	Champ chaîne de caractères (CHAR ou VARCHAR)
FIELD_TYPE_VAR_STRING	Champ VARCHAR
FIELD_TYPE_BLOB	Champ BLOB ou TEXT (utilisez max_length pour déterminer la taille maximale)
FIELD_TYPE_SET	Champ SET
FIELD_TYPE_ENUM	Champ ENUM
FIELD_TYPE_NULL	Champ de type NULL
FIELD_TYPE_CHAR	Désapprouvé, utilisez FIELD_TYPE_TINY à la place

Vous pouvez utiliser la macro `IS_NUM()` pour tester si un champ est de type numérique ou non. Passez la valeur de `type` à `IS_NUM()` et elle sera évaluée à TRUE si le champ est numérique :

```
if (IS_NUM(field->type))
   printf("Le champ est numérique\n");
```

`unsigned int length`

La taille du champ, comme spécifié dans la définition de la table.

`unsigned int max_length`

La longueur maximale du champ pour le jeu de résultats (la taille de la plus longue valeur de champ actuellement dans le jeu de résultat). Si vous utilisez `mysql_store_result()` ou `mysql_list_fields()` , cela contient la longueur maximale pour le champ. Si vous utilisez `mysql_use_result()` , la valeur de cette variable est zéro.

`unsigned int flags`

Les différents attributs sous forme de bits pour le champ. La valeur de `flags` peut avoir zéro ou plusieurs de ces bits suivants activés :

Valeur d'attribut	Description d'attribut
NOT_NULL_FLAG	Le champ ne peut être NULL
PRI_KEY_FLAG	Le champ fait parti d'une clef primaire
UNIQUE_KEY_FLAG	Le champ fait parti d'une clef unique
MULTIPLE_KEY_FLAG	Le champ fait parti d'une clef non-unique
UNSIGNED_FLAG	Le champ possède l'attribut UNSIGNED
ZEROFILL_FLAG	Le champ possède l'attribut ZEROFILL
BINARY_FLAG	Le champ possède l'attribut BINARY
AUTO_INCREMENT_FLAG	Le champ possède l'attribut AUTO_INCREMENT
ENUM_FLAG	Le champ est un ENUM (désapprouvé)
SET_FLAG	Le champ est un SET (désapprouvé)
BLOB_FLAG	Le champ est un BLOB ou TEXT (désapprouvé)
TIMESTAMP_FLAG	Le champ est un TIMESTAMP (désapprouvé)

L'utilisation des attributs `BLOB_FLAG` , `ENUM_FLAG` , `SET_FLAG` , et `TIMESTAMP_FLAG` est désapprouvé car ils indiquent un type de champ plutôt qu'un attribut de type de champ. Il est préférable de tester `field->type` avec `FIELD_TYPE_BLOB` , `FIELD_TYPE_ENUM` , `FIELD_TYPE_SET` , ou `FIELD_TYPE_TIMESTAMP` à la place.

L'exemple suivant illustre une utilisation typique de la valeur de `flags` :

```
if (field->flags & NOT_NULL_FLAG)
   printf("Le champ ne peut être nul\n");
```

Vous pouvez utiliser les différentes macros ci−dessous pour déterminer le statut booléen de la valeur de l'attribut :

Statut de l'attribut	Description
IS_NOT_NULL(flags)	Vrai si le champ est défini en tant que NOT NULL

IS_PRI_KEY(flags)	Vrai si le champ est une clef primaire
IS_BLOB(flags)	Vrai si le champ est un BLOB ou TEXT (désapprouvé; tester plutôt field->type)

unsigned int decimals

Le nombre de décimales pour les champs numériques.

21.2.2 Vue d'ensemble des fonctions de l'API C

Les fonctions disponibles dans l'API C sont listées ici et décrites en plus de détails dans la section suivante. Description des fonctions C .

Fonction	**Description**
mysql_affected_rows()	Retourne le nombre de lignes changées/effacées/insérés par le dernier UPDATE , DELETE , ou INSERT .
mysql_change_user()	Change l'utilisateur et la base de données pour une connexion ouverte.
mysql_character_set_name()	Retourne le nom du jeu de caractère de la connexion.
mysql_close()	Ferme une connexion au serveur.
mysql_connect()	Connecte à un serveur MySQL. Cette fonction est désapprouvée; utilisez mysql_real_connect() à la place.
mysql_create_db()	Crée une base de données. Cette fonction est désapprouvée, utilisez plutôt la commande SQL CREATE DATABASE .
mysql_data_seek()	Déplace le pointeur vers une ligne arbitraire dans le jeu de résultats de la requête.
mysql_debug()	Effectue un DBUG_PUSH avec la chaîne donnée.
mysql_drop_db()	Supprime une base de données. Cette fonction est désapprouvée, utilisez plutôt la commande SQL DROP DATABASE .
mysql_dump_debug_info()	Demande au serveur d'écrire les informations de débogage dans un fichier de log.
mysql_eof()	Détermine si la dernière ligne du jeu de résultats a été lue ou non. Cette fonction est désapprouvée, vous pouvez utiliser mysql_errno() ou mysql_error() à la place.
mysql_errno()	Retourne le numéro de l'erreur de la fonction appelée en dernier.
mysql_error()	Retourne le message d'erreur de la dernière fonction MySQL appelée.
mysql_escape_string()	Protège une chaîne en échappant les caractères spéciaux.
mysql_fetch_field()	Retourne le type du champ suivant dans la table.
mysql_fetch_field_direct()	Retourne le type d'une colonne, étant donné un numéro de champ.
mysql_fetch_fields()	Retourne un tableau avec toutes les structures de champs.
mysql_fetch_lengths()	Retourne la longueur de toutes les colonnes dans la ligne suivante.
mysql_fetch_row()	Récupère la ligne suivante dans le jeu de résultats.
mysql_field_seek()	Place le curseur de colonne sur une colonne précise.
mysql_field_count()	Retourne le nombre de colonnes dans le résultat pour la requête la plus récente.
mysql_field_tell()	Retourne la position du curseur de champs utilisé pour le dernier appel à mysql_fetch_field() .
mysql_free_result()	Libère la mémoire utilisée par un jeu de résultats.
mysql_get_client_info()	Retourne la version du client sous forme de chaîne.
mysql_get_client_version()	Retourne la version du client sous forme d'entier.
mysql_get_host_info()	Retourne une chaîne décrivant la connexion.
mysql_get_server_version()	Retourne le numéro de version du serveur sous forme d'entier (nouveau en 4.1).
mysql_get_proto_info()	Retourne la version du protocole utilisé par la connexion.
mysql_get_server_info()	Retourne la version du serveur.
mysql_info()	Retourne des informations à propos de la requête la plus récente.
mysql_init()	Récupère ou initialise une structure MYSQL .
mysql_insert_id()	Retourne l'identifié généré pour une colonne AUTO_INCREMENT par la dernière requête.
mysql_kill()	Termine un processus donné.
mysql_list_dbs()	Retourne les noms des bases de données répondant à une expression régulière simple.
mysql_list_fields()	Retourne les noms des champs répondants à une expression régulière simple.
mysql_list_processes()	Retourne une liste des processus courants du serveur.
mysql_list_tables()	Retourne les noms des tables répondants à une expression régulière simple.
mysql_num_fields()	Retourne le nombre de colonnes dans un jeu de résultats.
mysql_num_rows()	Retourne le nombre de lignes dans le jeu de résultats.
mysql_options()	Configure les options de connexion pour mysql_connect() .
mysql_ping()	Vérifie si la connexion au serveur a toujours lieu. Reconnecte si besoin.
mysql_query()	Exécute une requête SQL spécifiée en tant que chaîne terminée par un caractère nul.
mysql_real_connect()	Connecte à un serveur MySQL.
mysql_real_escape_string()	Protège les caractères spéciaux dans une chaîne utilisable dans une requête SQL, en prenant en compte le jeu de caractères courant de la connexion.
mysql_real_query()	Exécute une requête SQL spécifiée en tant que chaîne comptée.
mysql_reload()	Demande au serveur de recharger la table des droits.
mysql_row_seek()	Déplace le pointeur vers un ligne dans le jeu de résultats, en utilisant une valeur retournée par mysql_row_tell() .
mysql_row_tell()	Retourne la position du curseur de lignes.
mysql_select_db()	Sélectionne une base de données.
mysql_set_server_option()	Spécifie la valeur d'une option pour la connexion (comme multi-statements).
mysql_sqlstate()	Retourne l'erreur SQLSTATE de la dernière erreur.
mysql_shutdown()	Termine le serveur de base de données.
mysql_stat()	Retourne le statut du serveur dans une chaîne.

mysql_store_result()	Récupère le jeu de résultats complet dans le client.
mysql_thread_id()	Retourne l'identifiant du thread courant.
mysql_thread_safe()	Retourne 1 si le client est compilé avec la sécurité thread.
mysql_use_result()	Initialise une récupération ligne par ligne des résultats.
mysql_warning_count()	Retourne le nombre d'alertes générées par la dernière commande SQL.
mysql_commit()	Valide une transaction (nouveau en 4.1).
mysql_rollback()	Annule une transaction (nouveau en 4.1).
mysql_autocommit()	Active et désactive le mode d'auto validation (nouveau en 4.1).
mysql_more_results()	Vérifie si il n'y a plus de résultats (nouveau en 4.1).
mysql_next_result()	Retourne/initie le prochain résultat dans une commande multi−requête (nouveau en 4.1).

Pour vous connecter au serveur, appelez `mysql_init()` pour initialiser un gestionnaire de connexion, puis appelez `mysql_real_connect()` avec ce gestionnaire (avec d'autres informations tel que l'hôte, l'utilisateur et le mot de passe). Lors de la connexion, `mysql_real_connect()` définit l'option `reconnect` (quit fait partie de la structure MYSQL) à `1` . Cette option indique, dans le cas où une requête ne peut être exécutée à cause d'une perte de connexion, d'essayer de se reconnecter au serveur avant d'abandonner. Lorsque vous n'avez plus besoin de la connexion, appelez `mysql_close()` pour la clore.

Tant qu'une connexion est active, le client envoi des requêtes SQL au serveur à l'aide de `mysql_query()` ou `mysql_real_query()` . La différence entre les deux est que `mysql_query()` s'attend à ce que la requête soit spécifiée en tant que chaîne terminée par la chaîne nulle, tandis que `mysql_real_query()` attend une chaîne de longueur connue. Si la chaîne contient des données binaires (incluant l'octet nul), vous devez utiliser `mysql_real_query()` .

Pour chaque requête non−sélective (par exemple, `INSERT` , `UPDATE` , `DELETE`), vous pouvez trouver combien de lignes ont été mises à jour (affectées) en appelant `mysql_affected_rows()` .

Pour les requêtes `SELECT` , vous récupérez les lignes sélectionnées dans un jeu de résultat. (Notez que quelques commandes ont le même comportement que `SELECT` , dans le sens où elle renvoient des lignes. Cela inclut `SHOW` , `DESCRIBE` , et `EXPLAIN` . Elles doivent être traitées de la même façon que les requêtes `SELECT` .)

Il y a deux façons pour un client de gérer les jeux de résultats. Une méthode consiste à récupérer le jeu de résultat en entier et en une seule fois en appelant `mysql_store_result()` . Cette fonction obtient toutes les lignes retournées par la requête et les stocke dans le client. La seconde méthode consiste à initialiser une récupération ligne par ligne du jeu de résultats en appelant `mysql_use_result()` . Cette fonction initie la récupération, mais ne récupère actuellement aucune ligne à partir du serveur.

Dans les deux cas, vous accédez aux ligne en appelant `mysql_fetch_row()` . Avec `mysql_store_result()` , `mysql_fetch_row()` accède aux lignes qui ont déjà été récupérées à partir du serveur. Avec `mysql_use_result()` , `mysql_fetch_row()` récupère actuellement la ligne à partir du serveur. Les informations à propos de la taille des données dans chaque ligne est disponible en appelant `mysql_fetch_lengths()` .

Après avoir fini de traiter le jeu de résultats, appelez `mysql_free_result()` pour libérer la mémoire utilisée.

Les deux mécanismes de récupération sont complémentaires. Les programmes clients doivent utiliser l'approche qui leur convient le mieux. En pratique, les clients tendent plus à utiliser `mysql_store_result()` .

Un avantage de `mysql_store_result()` est que puisque toutes les lignes ont été récupérées dans le client, vous ne pouvez pas que accéder aux lignes séquentiellement, vous pouvez revenir en arrière ou avancer dans le jeu de résultats en utilisant `mysql_data_seek()` ou `mysql_row_seek()` pour changer la position de la ligne courante dans le jeu de résultats. Vous pouvez aussi trouver le nombre total des lignes en appelant `mysql_num_rows()` . D'un autre côté, les besoins en mémoire de `mysql_store_result()` peuvent être très grands pour les grands jeux de résultats et vous aurez des chances d'obtenir un manque de mémoire.

Un avantage de `mysql_use_result()` est que le client a besoin de moins de mémoire pour le jeu de résultats car il utilise une ligne à la fois (et puisque il y a moins de pertes de mémoire, `mysql_use_result()` peut être plus rapide). Les inconvénients sont que vous devez récupérer chaque ligne rapidement pour éviter de bloquer le serveur, vous n'avez pas d'accès aléatoires aux lignes dans le jeu de résultats (vous ne pouvez accéder aux lignes que séquentiellement), et vous ne savez pas combien de lignes comporte le jeu de résultats tant que vous ne les avez pas toutes récupérées. De plus, vous **devez** récupérer toutes les lignes même si vous trouvez entre−temps l'informations que vous cherchiez.

L'API permet aux clients de gérer correctement les requêtes (récupérant les lignes seulement en cas de besoin) sans savoir si la requête était un `SELECT` ou non. Vous pouvez faire cela en appelant `mysql_store_result()` après chaque `mysql_query()` (ou `mysql_real_query()`). Si l'appel au jeu de résultats réussi, la requête était un `SELECT` et vous pouvez lire les lignes. Sinon, appelez `mysql_field_count()` pour vérifier si un résultat aurait du être retourné. Si `mysql_field_count()` retourne zéro, la requête n'a pas retourné de données (cela indique que c'était un `INSERT` , `UPDATE` , `DELETE` , etc.), et ne devait pas retourner de lignes. Si `mysql_field_count()` est non−nul, la requête aurait du retourner des lignes, mais ne l'a pas fait. Cela indique que la requête était un `SELECT` qui a échoué. Reportez vous à la description de `mysql_field_count()` pour un exemple d'utilisation.

`mysql_store_result()` et `mysql_use_result()` vous permettent d'obtenir des informations à propos des champs qui constituent le jeu de résultat (le nombre de champs, leurs noms et types, etc.). Vous pouvez accéder aux informations du champ séquentiellement dans une ligne en appelant plusieurs fois `mysql_fetch_field()` , ou avec le numéro du champ dans la ligne en appelant `mysql_fetch_field_direct()` . La position courante du pointeur de champ peut être changée en appelant `mysql_field_seek()` . Changer le pointeur de champ affecte les appels suivants à `mysql_fetch_field()` . Vous pouvez aussi obtenir en une seule fois les informations sur les champs en appelant `mysql_fetch_fields()` .

Pour détecter les erreurs, MySQL fournit un accès aux informations des erreurs via les fonctions `mysql_errno()` et `mysql_error()` . Elles retournent le code de l'erreur et le message pour la dernière fonction invoquée qui aurait pu réussir ou échouer, vous permettant ainsi de déterminer les erreurs et leurs causes.

21.2.3 Description des fonctions de l'API C

Dans les descriptions suivantes, un paramètre ou retour de fonction `NULL` correspond au `NULL` dans le sens C du terme, et non dans le sens de la valeur `NULL` de MySQL.

Les fonctions qui retournent une valeur retournent la plupart du temps un pointeur ou un entier. Sauf en cas d'indications contraires, les fonctions retournant un pointeur retournent une valeur non−`NULL` pour indiquer un succès ou une valeur `NULL` pour indiquer une erreur, les fonctions retournant un entier retournent zéro pour indiquer un succès et une valeur non−nulle en cas d'erreur. Notez que ``non−nulle'' ne signifie rien de plus que cela. Sauf si la description de la fonction le dit, ne testez pas avec une valeur différente de zéro :

```
if (result)                   /* correct */
   ... erreur ...

if (result < 0)               /* incorrect */
   ... erreur ...

if (result == -1)             /* incorrect */
   ... erreur ...
```

Lorsqu'une fonction retourne une erreur, la section **Erreurs** du descriptif de la fonction liste les types d'erreurs possibles. Vous pouvez trouver celle qui est arrivée en appelant `mysql_errno()` . Une chaîne de caractères représentant l'erreur peut être obtenue en appelant `mysql_error()` .

21.2.3.1 mysql_affected_rows()

```
my_ulonglong mysql_affected_rows(MYSQL *mysql)
```

Description Retourne le nombre de lignes modifiées par la dernière commande `UPDATE` , supprimées par la dernière commande `DELETE` ou insérée par la dernière commande `INSERT` . Peut être appelée immédiatement après `mysql_query()` pour les commandes `UPDATE` , `DELETE` , ou `INSERT` . Pour la commande `SELECT` , `mysql_affected_rows()` fonctionne comme `mysql_num_rows()` .

Valeur de retour Un entier supérieur à zéro indique le nombre de lignes affectées ou sélectionnées. Zéro indique qu'aucun enregistrement n'a été mis à jour pour une requête `UPDATE` , qu'aucune lignes n'a correspondu à la clause `WHERE` dans la requête ou que celle ci n'a pas encore été exécutée. −{}1 indique que la requête a renvoyé une erreur ou que, pour une requête `SELECT` , `mysql_affected_rows()` a été appelée avant `mysql_store_result()` . Comme `mysql_affected_rows()` retourne une valeur non signée, vous pouvez comparer avec −{}1 en comparant la valeur retournée par `(my_ulonglong)-1` (ou par `(my_ulonglong)~0` , ce qui est la même chose).

Erreurs Aucune. **Exemple**

```
 mysql_query(&mysql,"UPDATE products SET cost=cost*1.25 WHERE group=10");
printf("%ld produits mis à jour",(long) mysql_affected_rows(&mysql));
```

Si on spécifie l'option `CLIENT_FOUND_ROWS` en se connectant à `mysqld` , `mysql_affected_rows()` retournera le nombre d'enregistrements correspondants à la clause `WHERE` pour une requête `UPDATE` .Notez que quand on utilise une commande `REPLACE` , `mysql_affected_rows()` retournera 2 si le nouvel enregistrement en a remplacé un ancien. 2 en retour car dans ce cas, l'ancienne ligne a été supprimé puis la nouvelle insérée.

21.2.3.2 mysql_change_user()

```
my_bool mysql_change_user(MYSQL *mysql, const char *user, const char *password, const char *db)
```

Description Change l'utilisateur et définit la base de données spécifiée par `db` en tant que base de données par défaut (courante) dans la connexion spécifiée par `mysql` . Pour les requêtes suivantes, cette base de données sera celle utilisée pour les références aux tables ne spécifiant pas explicitement une base de données.

Cette fonction a été introduite à la version 3.23.3 de MySQL.

`mysql_change_user()` échoue si l'utilisateur ne peut être authentifié ou s'il n'a pas le droit d'utiliser cette base de données. Dans ce cas, l'utilisateur et la base de données ne sont pas changés.

Le paramètre `db` peut être mis à `NULL` si vous ne voulez pas avoir de base de données par défaut.

Valeur de retour Zéro en cas de succès. Différent de zéro si une erreur se produit.

Erreurs Les mêmes que vous pouvez obtenir avec `mysql_real_connect()` .

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas été exécutées dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne réponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a été perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

`ER_UNKNOWN_COM_ERROR`
: Le serveur MySQL n'implémente pas cette commande (probablement un ancien

serveur)

`ER_ACCESS_DENIED_ERROR`
: L'utilisateur ou le mot de passe étaient erronés.

`ER_BAD_DB_ERROR`
: La base de données n'existe pas.

`ER_DBACCESS_DENIED_ERROR`
: L'utilisateur n'a pas le droit d'accéder à la base de données.

`ER_WRONG_DB_NAME`
: Le nom de la base de données était trop long.

Exemple

```
 if (mysql_change_user(&mysql, "user", "password", "new_database"))
&#123;
   fprintf(stderr, "Impossible de changer d'utilisateur.  Erreur : %s\n",
           mysql_error(&mysql));
&#125;
```

21.2.3.3 mysql_character_set_name()

```
const char *mysql_character_set_name(MYSQL *mysql)
```

Description Retourne le jeu de caractères par défaut de la connexion courante.

Valeur de retour Le jeu de caractères par défaut

Erreurs Aucune.

21.2.3.4 mysql_close()

```
void mysql_close(MYSQL *mysql)
```

Description Ferme la connexion ouverte précédemment. `mysql_close()` libère aussi le pointeur de connexion `mysql` , si celui−ci avait été alloué dynamiquement par `mysql_init()` ou `mysql_connect()` .

Valeur de retour Aucune.

Erreurs Aucune.

21.2.3.5 mysql_connect()

```
MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd)
```

Description Cette fonction est désapprouvée. Il est préférable d'utiliser `mysql_real_connect()` à la place.

`mysql_connect()` essaye d'établir une connexion à un serveur MySQL lancé sur `host` . `mysql_connect()` doit s'achever avec succès avant que vous ne puissiez exécuter l'une des autres fonctions de l'API, à l'exception de `mysql_get_client_info()` .

La signification des paramètres est la même que pour ceux de la fonction `mysql_real_connect()` à la différence que le paramètre de connexion peut être `NULL` . Dans ce cas, l'API C alloue automatiquement une mémoire pour la structure de connexion et la libère quand vous appelez `mysql_close()` . Le désavantage de cette approche est que vous ne pouvez pas récupérer les messages d'erreur si la connexion échoue. (Pour obtenir des informations sur les erreurs à partir de `mysql_errno()` ou `mysql_error()` , vous devez fournir un pointeur `MYSQL` valide.)

Valeur de retour La même que pour `mysql_real_connect()` .

Erreurs Les mêmes que pour `mysql_real_connect()` .

21.2.3.6 mysql_create_db()

```
int mysql_create_db(MYSQL *mysql, const char *db)
```

Description Crée la base de données nommée avec le paramètre `db` .

Cette fonction est désapprouvée. Il est préférable d'utiliser `mysql_query()` pour générer une requête SQL `CREATE DATABASE` à la place.

Valeur de retour Zéro si la base à été crée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas été exécutées dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne réponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a été perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

Exemple

```
 if(mysql_create_db(&mysql, "ma_base"))
&#123;
   fprintf(stderr, "Impossible de créer une nouvelle base de données. Erreur : %s\n",
           mysql_error(&mysql));
&#125;
```

21.2.3.7 mysql_data_seek()

```
void mysql_data_seek(MYSQL_RES *result, my_ulonglong offset)
```

Description Se déplace vers une ligne arbitraire d'un jeu de résultat de requête. Cela nécessite que la structure du jeu de résultat contienne la totalité du résultat de la requête, de ce fait `mysql_data_seek()` peut être utilisée en conjonction avec `mysql_store_result()` , mais pas avec `mysql_use_result()` .

L'index de la ligne doit être compris entre 0 et `mysql_num_rows(result)-1` .

Valeur de retour Aucune.

Erreurs Aucune.

21.2.3.8 mysql_debug()

```
void mysql_debug(const char *debug)
```

Description Provoque un `DBUG_PUSH` avec la chaîne donnée. `mysql_debug()` utilises la bibliothèque de débogage Fred Fish. Pour utiliser cette fonction vous devez compiler la bibliothèque client avec le support débogage. Déboguer un serveur MySQL . Déboguer un client MySQL .

Valeur de retour Aucune.

Erreurs Aucune.

Exemple L'appel montré ici fais générer à la bibliothèque du client un fichier de trace dans `/tmp/client.trace` sur la machine du client :

```
mysql_debug("d:t:O,/tmp/client.trace");
```

21.2.3.9 mysql_drop_db()

```
int mysql_drop_db(MYSQL *mysql, const char *db)
```

Description Supprime la base de données nommée avec le paramètre `db` .

Cette fonction est désapprouvée. Il est préférable d'utiliser `mysql_query()` pour générer une requête SQL `DROP DATABASE` à la place.

Valeur de retour Zéro si la base à été effacée avec succès. Différente de zéro si une erreur est survenue. **Erreurs**

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas été exécutées dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne réponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a été perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

Exemple

```
if(mysql_drop_db(&mysql, "ma_base"))
 fprintf(stderr, "Impossible de supprimer la base de données. Erreur : %s\n",
         mysql_error(&mysql));
```

21.2.3.10 mysql_dump_debug_info()

```
int mysql_dump_debug_info(MYSQL *mysql)
```

Description Demande au serveur d'écrire quelques informations de débogage dans le log. Pour que cela fonctionne, il faut que l'utilisateur ait le droit `SUPER` .

Valeur de retour Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas été exécutées dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne réponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a été perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

21.2.3.11 mysql_eof()

```
my_bool mysql_eof(MYSQL_RES *result)
```

Description Cette fonction est désapprouvée. Vous pouvez utiliser `mysql_errno()` ou `mysql_error()` à la place.

`mysql_eof()` détermine si la dernière ligne d'un jeu de résultats a été lue.

Si vous obtenez un jeu de résultats suite à un appel à `mysql_store_result()` , le client reçois le jeu entier en une seule opération. Dans ce cas, un retour `NULL` de la fonction `mysql_fetch_row()` signifie toujours que la fin du jeu de résultat a été atteinte et il n'est donc pas nécessaire d'appeler `mysql_eof()` . Lors d'une utilisation avec `mysql_store_result()` , `mysql_eof()` retournera toujours true.

D'un autre côté, si vous utilisez `mysql_use_result()` pour initialiser la récupération d'un jeu de résultats, les lignes sont obtenues du serveur une par une lors des appels successifs de `mysql_fetch_row()` . Puisque une erreur peut survenir à la connexion durant ce processus, une valeur de retour `NULL` de la part de `mysql_fetch_row()` ne signifie pas nécessairement que la fin du jeu de résultats a été atteinte normalement. Dans ce cas, vous pouvez utiliser `mysql_eof()` pour déterminer ce qui est arrivé. `mysql_eof()` retourne une valeur non−nulle si la fin du jeu de résultats a été atteinte et zéro en cas d'erreur.

Historiquement, `mysql_eof()` a vu le jour avant les fonctions d'erreurs standards de MySQL `mysql_errno()` et `mysql_error()` . Puisque ces fonctions fournissent les mêmes informations, leur utilisation est préférée à `mysql_eof()` , qui est maintenant désapprouvée. (En fait, elles fournissent plus d'informations, car `mysql_eof()` ne retourne que des valeurs booléennes alors que les fonctions d'erreurs indiquent les raisons des erreurs lorsqu'elles surviennent.)

Valeur de retour Zéro si aucune erreur n'est survenue. Autre chose dans le cas contraire.

Erreurs Aucune.

Exemple L'exemple suivant vous montre comment vous devez utiliser `mysql_eof()` :

```
 mysql_query(&mysql,"SELECT * FROM une_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
&#123;
    // traite les données
&#125;
if(!mysql_eof(result))  // mysql_fetch_row() failed due to an error
&#123;
    fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
&#125;
```

Vous pouvez reproduire la même chose avec les fonctions d'erreurs de MySQL :

```
 mysql_query(&mysql,"SELECT * FROM une_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
&#123;
    // traite les données
&#125;
if(mysql_errno(&mysql))  // mysql_fetch_row() ne marche pas à cause d'une erreur
&#123;
    fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
&#125;
```

21.2.3.12 mysql_errno()

```
unsigned int mysql_errno(MYSQL *mysql)
```

Description Pour la connexion spécifiée par `mysql` , `mysql_errno()` retourne le code de l'erreur pour l'appel le plus récent à une fonction de l'API qui peut réussir ou échouer. Un zéro en valeur de retour signifie qu'aucune erreur ne s'est produite. Les codes erreur du client sont listés dans le fichier d'entête MySQL (`errmsg.h`). Les codes erreur du serveur sont listés dans le fichier `mysqld_error.h` . Dans les sources de la distribution MySQL vous pouvez trouver la liste complète des messages d'erreur et le code qui leur est associé dans le fichier `Docs/mysqld_error.txt` . Les codes d'erreur serveurs sont listés dans Erreurs retournées .

Notez que certaines fonctions comme `mysql_fetch_row()` ne donne pas de valeur à `mysql_errno()` si elles réussissent.

En général, les fonctions qui doivent interroger le serveur pour obtenir des informations vont remettre à zéro `mysql_errno()` si elles réussissent.

Valeur de retour Un code d'erreur. Zéro si aucune erreur n'est survenue.

Erreurs Aucune.

21.2.3.13 mysql_error()

```
char *mysql_error(MYSQL *mysql)
```

Description Pour la connexion spécifiée par `mysql` , `mysql_error()` retourne le message d'erreur pour l'appel le plus récent à une fonction de l'API qui peut réussir ou échouer. Une chaîne vide (`""`) est retournée si aucune erreur n'est survenue. Cela signifie que les deux tests suivants sont équivalents :

```
 if(mysql_errno(&mysql))
&#123;
    // une erreur est survenue
&#125;

 if(mysql_error(&mysql)[0] != '\0')
&#123;
    // une erreur est survenue
&#125;
```

La langue des messages d'erreurs peut être changée en recompilant la bibliothèque du client MySQL. Actuellement, vous pouvez choisir les messages d'erreur parmi un choix de plusieurs langues. Messages d'erreurs non−anglophones .

Valeur de retour Une chaîne de caractères qui décrit l'erreur. Une chaîne vide si aucune erreur n'est survenue.

Erreurs Aucune.

21.2.3.14 mysql_escape_string()

Vous devez utiliser la fonction `mysql_real_escape_string()` à la place de celle ci !

Cette fonction est identique à `mysql_real_escape_string()` à l'exception faite que `mysql_real_escape_string()` prends deux identifiants de connexion comme premiers arguments et échappe la chaîne en se basant que le jeu de caractères courant. `mysql_escape_string()` ne prends pas d'identifiant de connexion et ne respecte pas le jeu de caractères courant.

21.2.3.15 mysql_fetch_field()

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)
```

Description Retourne la définition d'une colonne d'un jeu de résultats en tant que structure `MYSQL_FIELD` . Appelez cette fonction plusieurs fois pour obtenir des informations à propos de toutes les colonnes dans le jeu de résultat. `mysql_fetch_field()` retourne `NULL` quand il ne reste plus de champs.

`mysql_fetch_field()` est mis à zéro pour retourner des informations à propos du premier champ à chaque fois que vous exécutez une nouvelle requête `SELECT` . Le champ retourné par `mysql_fetch_field()` est aussi affecté par les appels à `mysql_field_seek()` .

Si vous avez appelé `mysql_query()` pour exécuter un `SELECT` sur une table mais n'avez pas appelé `mysql_store_result()` , MySQL retourne la longueur par défaut du BLOB (8 ko octets) si vous avez appelé `mysql_fetch_field()` pour obtenir la longueur d'un champ `BLOB` . (La taille 8K est choisie car MySQL ne connaît pas la longueur maximale du `BLOB` . Cela devrait être un jour paramétrable.) Une fois que vous avez récupéré le jeu de résultats, `field->max_length` contient la longueur de la plus grande valeur de cette colonne dans la requête spécifiée.

Valeur de retour La structure `MYSQL_FIELD` de la colonne courante. `NULL` s'il ne reste plus de colonnes.

Erreurs Aucune. **Exemple**

```
 MYSQL_FIELD *field;

 while((field = mysql_fetch_field(result)))
&#123;
    printf("nom du champ : %s\n", field->name);
&#125;
```

21.2.3.16 mysql_fetch_fields()

```
MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES *result)
```

Description Retourne un tableau de toutes les structures `MYSQL_FIELD` dans un jeu de résultats. Chaque structure fournit la définition de champ d'une colonne dans le jeu de résultats.

Valeur de retour Un tableau de structures `MYSQL_FIELD` pour toutes les colonnes dans le jeu de résultat.

Erreurs Aucune. **Exemple**

```
 unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

 num_fields = mysql_num_fields(result);
fields = mysql_fetch_fields(result);
for(i = 0; i < num_fields; i++)
&#123;
   printf("Le champ %u est %s\n", i, fields[i].name);
&#125;
```

21.2.3.17 mysql_fetch_field_direct()

```
MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *result, unsigned int fieldnr)
```

Description Etant donné un numéro de champ `fieldnr` pour une colonne dans un jeu de résultats, cette fonction retourne la définition de ce champ en tant que structure `MYSQL_FIELD` . Vous pouvez utiliser cette fonction pour obtenir la définition d'une colonne choisie arbitrairement. La valeur de `fieldnr` doit varier entre 0 et `mysql_num_fields(result)-1` .

Valeur de retour La structure `MYSQL_FIELD` pour la colonne spécifiée.

Erreurs Aucune. **Exemple**

```
 unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *field;

 num_fields = mysql_num_fields(result);
for(i = 0; i < num_fields; i++)
&#123;
    field = mysql_fetch_field_direct(result, i);
```

```
    printf("La champ %u est %s\n", i, field->name);
&#125;
```

21.2.3.18 mysql_fetch_lengths()

```
unsigned long *mysql_fetch_lengths(MYSQL_RES *result)
```

Description Retourne les longueurs des colonnes de la ligne courante dans le jeu de résultats. Si vous voulez copier les valeurs des champs, cette information sur la longueur est très utile pour l'optimisation, car vous pouvez éviter les appels à `strlen()` . De plus, si le jeu de résultat contient des données binaires, vous **devez** cette fonction pour déterminer la longueur des données, car `strlen()` retourne des résultats incorrects pour les champs contenant des caractères nuls.

La longueur des colonnes vides et des colonnes contenant la valeur `NULL` est zéro. Pour savoir comment distinguer ces cas, voyez la description de `mysql_fetch_row()` .

Valeur de retour Un tableau d'entiers longs non−signés représentant la taille de chaque colonne (n'incluant pas la caractère nul de terminaison). `NULL` si une erreur se produit.

Erreurs `mysql_fetch_lengths()` n'st valide que pour la ligne courante du jeu de résultats. Cette fonction retourne `NULL` si vous l'appelez avant d'appeler `mysql_fetch_row()` ou après avoir récupéré toutes les lignes du résultat.

Exemple

```
 MYSQL_ROW row;
unsigned long *lengths;
unsigned int num_fields;
unsigned int i;

 row = mysql_fetch_row(result);
if (row)
&#123;
    num_fields = mysql_num_fields(result);
    lengths = mysql_fetch_lengths(result);
    for(i = 0; i < num_fields; i++)
    &#123;
         printf("La colonne %u a %l octets de longueur.\n", i, lengths[i]);
    &#125;
&#125;
```

21.2.3.19 mysql_fetch_row()

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)
```

Description Récupère la ligne suivante d'un jeu de résultats. Lorsqu'elle est utilisée après `mysql_store_result()` , `mysql_fetch_row()` retourne `NULL` quand il n'y a plus de lignes a récupérer. Lorsqu'elle est utilisée après `mysql_use_result()` , `mysql_fetch_row()` retourne `NULL` quand il n'y a plus de lignes a récupérer ou qu'une erreur est rencontrée.

Le nombre de valeurs dans la ligne est donné par `mysql_num_fields(result)` . Si `row` contient la valeur de retour d'un appel à `mysql_fetch_row()` , les pointeurs sur les valeurs sont accédées de `row[0]` à `row[mysql_num_fields(result)-1]` . Les valeurs `NULL` de la ligne sont indiquées par des pointeurs `NULL` .

La longueur de la valeur du champ dans la ligne peut être obtenue en appelant `mysql_fetch_lengths()` . Les champs vides et les champs contenant `NULL` ont tous deux une longueur égale à zéro; vous pouvez les distinguer en vérifiant le pointeur sur la valeur du champ. Si le pointeur est `NULL` , le champ est `NULL` ; sinon, le champ est vide.

Valeur de retour Une structure `MYSQL_ROW` pour la prochaine ligne. `NULL` s'il n'y a plus de lignes a récupérer ou qu'une erreur survient.

Erreurs

`CR_SERVER_LOST`
: La connexion au serveur a été perdue durant la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue est survenue.

Exemple

```
 MYSQL_ROW row;
unsigned int num_fields;
unsigned int i;

 num_fields = mysql_num_fields(result);
while ((row = mysql_fetch_row(result)))
&#123;
   unsigned long *lengths;
   lengths = mysql_fetch_lengths(result);
   for(i = 0; i < num_fields; i++)
   &#123;
       printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");
   &#125;
   printf("\n");
&#125;
```

21.2.3.20 mysql_field_count()

`unsigned int mysql_field_count(MYSQL *mysql)`

Si vous utilisez une version de MySQL plus ancienne que la 3.22.24, vous devez utiliser `unsigned int mysql_num_fields(MYSQL *mysql)`.

Description Retourne le nombre de colonnes pour la requête la plus récente de la connexion.

L'utilisation normale de cette fonction est lorsque `mysql_store_result()` a retourné `NULL` (et que vous n'avez donc pas de pointeur sur jeu de résultats). Dans ce cas, vous pouvez appeler `mysql_field_count()` pour déterminer si `mysql_store_result()` aurait dû produire un résultat non−vide. Cela permet au programme client d'entreprendre les bonnes actions sans savoir si la requête était un `SELECT` (ou équivalent). L'exemple suivant illustre comment cela peut être fait.

Pourquoi est ce qu'après `mysql_query()` ait indiqué un résultat positif, `mysql_store_result()` retourne parfois `NULL` ? .

Valeur de retour Un entier non−signé représentant le nombre de champs dans un jeu de résultats.

Erreurs Aucune.

Exemple

```
 MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

 if (mysql_query(&mysql,query_string))
&#123;
    // erreur
&#125;
else // requête bonne, traitons les données qu'elle renvoit
&#123;
    result = mysql_store_result(&mysql);
    if (result)  // il y a des lignes
    &#123;
        num_fields = mysql_num_fields(result);
        // récupère les lignes, puis appele mysql_free_result(result)
    &#125;
    else  // mysql_store_result() n'a rien retourné; est-ce normal ?
    &#123;
        if(mysql_field_count(&mysql) == 0)
        &#123;
            // la requête ne retourne aucune donnée
            // (ce n'était pas un SELECT)
```

```
            num_rows = mysql_affected_rows(&mysql);
        &#125;
        else // mysql_store_result() aurait du retourner des données
        &#123;
            fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
        &#125;
    &#125;
&#125;
```

Une alternative est de remplacer l'appel à mysql_field_count(par mysql_errno(. Dans ce cas, vous vérifiez directement les erreurs à partir de mysql_store_result() plutôt qu'à partir de mysql_field_count() si la requête était un SELECT .

21.2.3.21 mysql_field_seek()

MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET offset)

Description Place le pointeur de champs à la position donnée. Le prochain appel à mysql_fetch_field() récupérera la définition du champ de la colonne associée à cet index.

Pour vous placer au début d'une ligne, passez 0 comme valeur d' offset .

Valeur de retour La dernière valeur de l'index de champ.

Erreurs Aucune.

21.2.3.22 mysql_field_tell()

MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)

Description Retourne la position du curseur de champ utilisé pour le dernier appel à mysql_fetch_field() . Cette valeur peut être utilisée en argument de mysql_field_seek() .

Valeur de retour L'indice courant du curseur de champ.

Erreurs Aucune.

21.2.3.23 mysql_free_result()

void mysql_free_result(MYSQL_RES *result)

Description Libère la mémoire alloué à un résultat avec mysql_store_result() , mysql_use_result() , mysql_list_dbs() , etc. Quand vous n'avez plus besoin d'un jeu de résultat, vous devez libérer la mémoire qu'il occupe en appelant mysql_free_result() .

Valeur de retour Aucune.

Erreurs Aucune.

21.2.3.24 mysql_get_client_info()

char *mysql_get_client_info(void)

Description Retourne une chaîne représentant la version de la bibliothèque du client.

Valeur de retour Une chaîne de caractères représentant la version de la bibliothèque du client.

Erreurs Aucune.

21.2.3.25 mysql_get_client_version()

```
unsigned long mysql_get_client_version(void)
```

Description Retourne un entier qui représente la version de la bibliothèque cliente. Cette valeur est au format XYYZZ où X est la version majeure, YY est la version publiée, et ZZ est le numéro de version de la version publiée. Par exemple, la valeur 40102 représente la version 4.1.2 .

Valeur retournée Un entier qui représente la version de la bibliothèque cliente MySQL.

Erreurs Aucune.

21.2.3.26 mysql_get_host_info()

```
char *mysql_get_host_info(MYSQL *mysql)
```

Description Retourne une chaîne de caractères décrivant le type de connexion actuellement utilisé, incluant le nom du serveur.

Valeur de retour Une chaîne de caractères représentant le nom du serveur et le type de connexion.

Erreurs Aucune.

21.2.3.27 mysql_get_proto_info()

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

Description Retourne la version du protocole utilisé par la connexion courante.

Valeur de retour Un entier non signé représentant la version du protocole utilisé par la connexion courante.

Erreurs Aucune.

21.2.3.28 mysql_get_server_info()

```
char *mysql_get_server_info(MYSQL *mysql)
```

Description Retourne une chaîne représentant le numéro de version du serveur.

Valeur de retour Une chaîne de caractères représentant le numéro de version du serveur.

Erreurs Aucune.

21.2.3.29 mysql_get_server_version()

```
unsigned long mysql_get_server_version(MYSQL *mysql)
```

Description Retourne le numéro de version du serveur, sous la forme d'un entier (nouveau en 4.1).

Valeurs retournées Un nombre, qui représente le numéro de version du serveur MySQL, au format suivant :

```
major_version*10000 + minor_version *100 + sub_version
```

Par exemple, 4.1.0 est retournée comme ceci : 40100.Ceci est pratique pour déterminer rapidement le numéro de version d'un serveur, dans un client, et connaître ainsi ses fonctionnalités.

Erreurs Aucune.

21.2.3.30 mysql_info()

```
char *mysql_info(MYSQL *mysql)
```

Description Récupère une chaîne de caractères fournissant des informations à propos de la requête exécutée le plus récemment, mais seulement pour celles listées ici. Pour les autres requêtes, mysql_info() retournera NULL .

Le format de la chaîne varie selon le type de requête, comme décrit ici. Les nombres présentés sont des exemples; la chaîne retournée contiendra les informations correspondantes à vos requêtes.

```
INSERT INTO ... SELECT ...
         Chaîne retournée : Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
         Chaîne retournée : Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
         Chaîne retournée : Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
         Chaîne retournée : Records: 3 Duplicates: 0 Warnings: 0
UPDATE
         Chaîne retournée : Rows matched: 40 Changed: 40 Warnings: 0
```

Notez que mysql_info() retourne une valeur non-nulle (NULL) pour les requêtes INSERT ... VALUES seulement si une liste de valeurs multiples est fournie à la requête. **Valeur de retour** Une chaîne de caractères représentant des informations additionnelles à propos de la dernière requête exécutée. NULL si aucune information n'est disponible pour la requête.

Erreurs Aucune.

21.2.3.31 mysql_init()

```
MYSQL *mysql_init(MYSQL *mysql)
```

Description Alloue ou initialise un objet MYSQL convenable pour mysql_real_connect() . Si mysql est un pointeur NULL , la fonction alloue, initialise et retourne un nouvel objet. Sinon, l'objet est initialisé et son adresse est retournée. Si mysql_init() alloue un nouvel objet, il sera libéré quand mysql_close() sera appelée pour clore la connexion.

Valeur de retour Un gestionnaire MYSQL* initialisé. NULL s'il n'y avait pas assez de mémoire pour allouer le nouvel objet.

Erreurs Si la mémoire est insuffisante, NULL est retourné.

21.2.3.32 mysql_insert_id()

```
my_ulonglong mysql_insert_id(MYSQL *mysql)
```

Description Retourne l'identifiant généré pour une colonne AUTO_INCREMENT par la dernière requête. Utilisez cette commande après avoir exécuté une requête INSERT sur une table qui contient un champ AUTO_INCREMENT .

Plus précisément, mysql_insert_id() est mis à jour dans ces conditions :

- Les commandes INSERT qui stockent une valeur dans une colonne de type AUTO_INCREMENT . C'est vrai si la valeur est automatiquement générée suivant le stockage de NULL ou 0 dans la colonne, ou une valeur explicite non-spéciale.
- Dans le cas des INSERT multiples, mysql_insert_id() retourne la **première** valeur AUTO_INCREMENT automatiquement générée : si aucune valeur n'est générée, elle retourne la **dernière** valeur explicitement insérée dans la colonne AUTO_INCREMENT .
- Les commandes INSERT qui génère une valeur AUTO_INCREMENT en insérant l'expression LAST_INSERT_ID(expr) dans une colonne.
- Les commandes INSERT qui génèrent une valeur AUTO_INCREMENT qui donne la valeur de LAST_INSERT_ID(expr) à une colonne.
- La valeur de mysql_insert_id() n'est pas affectée par les commandes telles que les commandes SELECT qui ne retournent pas de résultat.
- Si la commande précédent a retourné une erreur, la valeur de mysql_insert_id() est indéfinie.

Notez que mysql_insert_id() retourne 0 si la dernière requête n'a pas généré de valeur AUTO_INCREMENT . Si vous voulez garder cette valeur pour plus tard, assurez vous d'appeler mysql_insert_id() immédiatement après la requête ayant généré cette valeur.

mysql_insert_id() est mis à jour après l'exécution de requêtes INSERT et UPDATE qui génèrent une valeur AUTO_INCREMENT ou qui définissent la valeur d'une colonne à LAST_INSERT_ID(expr) . Fonctions diverses .

Notez aussi que la valeur de retour de la fonction SQL `LAST_INSERT_ID()` contient toujours la valeur d' `AUTO_INCREMENT` la plus à jour. Cette valeur n'est pas remise à zéro lors de l'exécution d'autre requêtes car elle est maintenue pour le serveur.

Valeur de retour La valeur de la colonne AUTO_INCREMENT qui a été mise à jour par la dernière requête. Retourne zéro si aucune requête n'avait eu lieu durant la connexion, ou si la dernière requête n'a pas mis à jour la valeur de la colonne AUTO_INCREMENT .

Erreurs Aucune.

21.2.3.33 mysql_kill()

```
int mysql_kill(MYSQL *mysql, unsigned long pid)
```

Description Demande au serveur de terminer le thread spécifié par pid .

Valeur de retour Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

```
CR_COMMANDS_OUT_OF_SYNC
        Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
        Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
        La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
        Une erreur inconnue s'est produite.
```

21.2.3.34 mysql_list_dbs()

```
MYSQL_RES *mysql_list_dbs(MYSQL *mysql, const char *wild)
```

Description Retourne une jeu de résultats se composant des noms des bases de données localisées sur le serveur qui correspondent à l'expression régulière spécifié par le paramètre wild . wild peut contenir les caractères spéciaux '%' ou '_' , ou peut être un pointeur NULL pour obtenir la liste de toutes les bases de données. Utiliser mysql_list_dbs() reviens à exécuter la requête SHOW databases [LIKE wild] .

Vous devez libérer le résultat avec `mysql_free_result()` .

Valeur de retour Un jeu de résultats MYSQL_RES en cas de succès. NULL si une erreur est survenue. **Erreurs**

```
CR_COMMANDS_OUT_OF_SYNC
        Les commandes n'ont pas été exécutées dans le bon ordre.
CR_OUT_OF_MEMORY
        Plus de mémoire.
CR_SERVER_GONE_ERROR
        Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
        La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
        Une erreur inconnue s'est produite.
```

21.2.3.35 mysql_list_fields()

```
MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char *wild)
```

Description Retourne un jeu de résultats consistant des noms de champs dans une table qui correspondent l'expression régulière simple spécifiée par la paramètre wild . wild peut contenir les caractères spéciaux '%' ou '_' , ou peut être un pointeur NULL pour correspondre à tous les champs. Utiliser mysql_list_fields() revient à exécuter la requête SHOW COLUMNS FROM nom_de_table [LIKE wild] .

Notez qu'il est recommandé d'utiliser `SHOW COLUMNS FROM nom_de_table` au lieu de / `mysql_list_fields()` .

Vous devez libérer le résultat avec mysql_free_result() .

Valeur de retour Un jeu de résultats `MYSQL_RES` en cas de succès. `NULL` sinon.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.36 mysql_list_processes()

```
MYSQL_RES *mysql_list_processes(MYSQL *mysql)
```

Description Retourne un jeu de résultat décrivant les threads courants du serveur. C'est le même genre d'informations renvoyé par mysqladmin processlist ou une requête SHOW PROCESSLIST .

Vous devez libérer le jeu de résultat avec mysql_free_result() .

Valeur de retour Un jeu de résultat MYSQL_RES en cas de succès. NULL si une erreur est survenue. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.37 mysql_list_tables()

```
MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)
```

Description Retourne un jeu de résultats consistant des noms de tables dans la base de données courante qui concordent avec l'expression régulière spécifié par le paramètre wild . wild peut contenir les caractères spéciaux '%' ou '_' , ou peut être un pointeur NULL pour obtenir toutes les tables. Faire appel à mysql_list_tables() revient à exécuter la requête SHOW tables [LIKE wild] .

Vous devez libérer le jeu de résultats avec mysql_free_result() .

Valeur de retour Un jeu d'enregistrements MYSQL_RES en cas de succès. NULL en cas d'erreurs. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.38 mysql_num_fields()

```
unsigned int mysql_num_fields(MYSQL_RES *result)
```

ou

```
unsigned int mysql_num_fields(MYSQL *mysql)
```

La seconde forme ne fonctionne plus à partir de la version 3.22.24 de MySQL. Pour passer un argument MYSQL* , vous devez utiliser la fonction unsigned int mysql_field_count(MYSQL *mysql) à la place.

Description Retourne le nombre de colonnes dans un jeu de résultats.

Notez que vous pouvez obtenir le nombre de colonnes soit à partir d'un pointeur sur résultat, soit d'un pointeur de connexion. Vous utiliserez le pointeur de connexion si mysql_store_result() ou mysql_use_result() ont retournés NULL (et que donc, vous n'avez pas de pointeur sur résultat). Dans ce cas, vous pouvez appeler mysql_field_count() pour déterminer si mysql_store_result() aurait du retourner un résultat non-vide. Cela permet au client d'effectuer les bonnes actions sans savoir si la requête était un SELECT (ou équivalent). L'exemple ci-dessous montre comment cela doit être utilisé.

Pourquoi est ce qu'après `mysql_query()` ait indiqué un résultat positif, `mysql_store_result()` retourne parfois `NULL` ? .

Valeur de retour Un entier non-signé représentant le nombre de champs dans un jeu de résultats.

Erreurs Aucune.

Exemple

```
 MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

 if (mysql_query(&mysql,query_string))
&#123;
    // erreur
&#125;
else // la requête fonctionne, on s'occupe des données
&#123;
    result = mysql_store_result(&mysql);
    if (result)  // il y a des lignes
    &#123;
        num_fields = mysql_num_fields(result);
        // recupérer les lignes, puis appeler mysql_free_result(result)
    &#125;
    else  // mysql_store_result() n'a rien retourné ! pourquoi ?
    &#123;
        if (mysql_errno(&mysql))
    &#123;
           fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
    &#125;
        else if (mysql_field_count(&mysql) == 0)
        &#123;
            // la requête ne retourne pas de données
            // (ce n'etait pas un SELECT)
            num_rows = mysql_affected_rows(&mysql);
        &#125;
    &#125;
&#125;
```

Une alternative (si vous savez que votre requête aurait du retourner des résultats) est de remplacer l'appel à mysql_errno(par un test sur la nullité de mysql_field_count(. Cela n'arrive que si un problème a été rencontré.

21.2.3.39 mysql_num_rows()

`my_ulonglong mysql_num_rows(MYSQL_RES *result)`

Description Retourne le nombre de lignes présentes dans le résultat.

L'utilisation de mysql_num_rows() dépend de si vous utilisez mysql_store_result() ou mysql_use_result() pour retourner le jeu résultat. Si vous utilisez mysql_store_result() , mysql_num_rows() peut être appelé immédiatement. Si vous utilisez mysql_use_result() , mysql_num_rows() ne retournera pas la valeur correcte tant que toutes les lignes du résultat n'auront pas été récupérées.

Valeur de retour Le nombre de lignes dans le résultat.

Erreurs Aucune.

21.2.3.40 mysql_options()

`int mysql_options(MYSQL *mysql, enum mysql_option option, const char *arg)`

Description Cette fonction peut être utilisée pour spécifier des options de connexion et modifier le comportement de la session courante. Cette fonction peut être appelée plusieurs fois pour définir plusieurs options.

mysql_options() doit être appelée après mysql_init() et avant mysql_connect() ou mysql_real_connect() .

L'argument `option` est l'option que vous voulez configurer; l'argument `arg` est la valeur pour cette option. Si l'option est un entier, `arg` doit pointer sur la valeur d'un entier.

Les valeurs possibles pour les options sont :

Option	**Type de l'argument**	**Fonction**
MYSQL_OPT_CONNECT_TIMEOUT	unsigned int *	Délai d'inactivité maximal permis.
MYSQL_OPT_READ_TIMEOUT	unsigned int *	Délai d'expiration pour les lectures depuis le serveur (fonctionne uniquement sur Windows sur les connexions TCP/IP)
MYSQL_OPT_WRITE_TIMEOUT	unsigned int *	Délai pour les écritures sur le serveur (fonctionne uniquement sur Windows sur les connexions TCP/IP)
MYSQL_OPT_COMPRESS	Non utilisé	Utiliser le protocole compressé client/serveur.
MYSQL_OPT_LOCAL_INFILE	pointeur optionnel sur uint	Si aucun pointeur n'est donné, ou que celui-ci pointe sur un unsigned int != 0 la commande LOAD LOCAL INFILE est activée.
MYSQL_OPT_NAMED_PIPE	Non utilisé	Utiliser les pipes nommés pour se connecter au serveur MySQL sur NT.
MYSQL_INIT_COMMAND	char *	Commande à exécuter lors de la connexion au serveur MySQL. Sera automatiquement re-exécutée lors des reconnexions.
MYSQL_READ_DEFAULT_FILE	char *	Lit les options à partir du fichier d'options nommé plutôt que de my.cnf .
MYSQL_READ_DEFAULT_GROUP	char *	Lit les options à partir du groupe spécifié dans le fichier d'options my.cnf ou le fichier spécifié par MYSQL_READ_DEFAULT_FILE .
MYSQL_OPT_PROTOCOL	unsigned int *	Type de protocole à utiliser. Doit être une des valeurs de mysql_protocol_type définies dans mysql.h .
MYSQL_SHARED_MEMORY_BASE_NAME	char*	Nom d'un objet de mémoire partagée pour communiquer avec le serveur. Doit être le même que l'option -shared-memory-base-name utilisé pour le serveur mysqld auquel vous voulez vous connecter.

Notez que le groupe client est toujours lu si vous utilisez MYSQL_READ_DEFAULT_FILE ou MYSQL_READ_DEFAULT_GROUP .

Le groupe spécifié dans le fichier des options peut contenir les options suivantes :

Option	**Description**
connect-timeout	Délai d'inactivité maximal permis en secondes. Sous Linux, ce délai est aussi utilisé lors de l'attente de la première réponse du serveur.
compress	Utiliser le protocole compressé client/serveur.
database	Se connecter à cette base de données si aucune base n'a été sélectionnée à la connexion.
debug	Options de débogage.
disable-local-infile	Interdit l'utilisation de LOAD DATA LOCAL .
host	Nom d'hôte par défaut.
init-command	Commande lors de la connexion au serveur MySQL. Sera automatiquement re-exécutée lors des reconnexions.
interactive-timeout	Revient à spécifier CLIENT_INTERACTIVE à mysql_real_connect() . mysql_real_connect().
local-infile[=(0|1)]	Si aucun argument, ou argument != 0, on permet alors l'utilisation de LOAD DATA LOCAL .
max_allowed_packet	Taille maximale de paquet que le client peut lire du serveur.
password	Mot de passe par défaut.
pipe	Utiliser les tunnels nommés pour se connecter à MySQL sur NT.
protocol=(TCP | SOCKET | PIPE | MEMORY)	Le protocole utilisé lors de la connexion au serveur (nouveau en version 4.1).
port	Port par défaut.
return-found-rows	Demande à mysql_info() de retourner les lignes trouvées au lieu des lignes mises à jour lors de l'utilisation de UPDATE .
shared-memory-base-name=name	Nom de l'objet en mémoire partagée à utiliser pour se connecter au serveur (par défaut, c'est "MySQL"). Nouveau en MySQL 4.1.
socket	Numéro de socket par défaut.
user	Utilisateur par défaut.

Notez que timeout a été remplacé par connect-timeout , mais que timeout fonctionne encore pour le moment.

Pour plus d'informations sur les fichiers d'options, reportez vous à Utiliser les fichiers d'options .

Valeur de retour Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Exemple

```
 MYSQL mysql;

 mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_COMPRESS,0);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"odbc");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
&#123;
    fprintf(stderr, "Impossible de se connecter à la base de données. Erreur : %s\n",
          mysql_error(&mysql));
&#125;
```

Ce qui précède demande au client d'utiliser le protocole compressé client/serveur et lit les options optionnelles de la section odbc dans le fichier my.cnf .

21.2.3.41 mysql_ping()

```
int mysql_ping(MYSQL *mysql)
```

Description Vérifie si la connexion au serveur est encore assurée. Si ce n'est pas le cas, une re−connexion automatique est tentée.

Cette fonction peut être utilisée par les clients qui restent inactifs longtemps, pour vérifier que le serveur n'a pas fermé la connexion et se re-connecter si nécessaire.

Valeur de retour Zéro si le serveur répond. Autre que zéro si une erreur est survenue.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Les commandes n'ont pas été exécutées dans le bon ordre.

CR_SERVER_GONE_ERROR
: Le serveur MySQL ne réponds pas.

CR_UNKNOWN_ERROR
: Une erreur inconnue s'est produite.

21.2.3.42 mysql_query()

```
int mysql_query(MYSQL *mysql, const char *query)
```

Description Exécute la requête SQL pointée par la chaîne terminée par null `query` . La requête doit se composer d'une seule opération. Vous ne devez pas ajouter de caractère de terminaison (`';'`) ou `\g` à la fin de la requête.

mysql_query() ne peut être utilisée pour les requêtes contenant des données bianaires, vous devez utiliser mysql_real_query() à la place. (LEs données binaires peuvent contenir le caractère '\0' , qui est interprété comme la fin de la chaîne requête.)

Si vous voulez savoir si la requête doit retourner un jeu de résultat ou non, vous pouvez utiliser `mysql_field_count()` pour vérifier. `mysql_field_count()` .

Valeur de retour Zéro si la requête a été effectuée avec succès. Différente de zéro si une erreur est survenue. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
: Les commandes n'ont pas été exécutées dans le bon ordre.

CR_SERVER_GONE_ERROR
: Le serveur MySQL ne réponds pas.

CR_SERVER_LOST
: La connexion au serveur a été perdue au cours la requête.

CR_UNKNOWN_ERROR
: Une erreur inconnue s'est produite.

21.2.3.43 mysql_real_connect()

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port,
const char *unix_socket, unsigned long client_flag)
```

Description mysql_real_connect() essaye de se connecter à une base de données MySQL tournant sur l'hôte. mysql_real_connect() doit se terminer correctement avant que vous ne puissiez aucune autre fonction de l'API, à l'exception de mysql_get_client_info() .

Les paramètres sont spécifiés comme suit :

- Le premier paramètre doit être l'adresse d'une structure MYSQL existante. Avant d'appeler mysql_real_connect() vous devez appeler mysql_init() pour initialiser la structure MYSQL . Vous pouvez changer un tas d'options de connexion en appelant mysql_options() . mysql_options() .
- La valeur de host peut être un nom de domaine ou une adresse IP. Si host est NULL ou égal à la chaîne "localhost" , une connexion à la machine local est essayée. Si le système supporte les sockets (Unix) ou les tunnels nommés (Windows), elles sont utilisées au lieu de TCP/IP pour se connecter au serveur.
- La paramètre user contient l'identifiant MySQL de l'utilisateur. Si user est NULL , l'utilisateur courant est sous-entendu. Avec Unix c'est l'utilisateur courant. Avec Windows ODBC, le nom de l'utilisateur courant doit être spécifié explicitement. Comment remplir les différents champs dans le programme d'administrateur ODBC .
- La paramètre passwd contient le mot de passe de user . Si passwd est NULL , seules les entrées de la table user pour l'utilisateur ayant un champ vide seront testées. Cela permet à l'administrateur de mettre en place le système de privilèges MySQL de façon à ce que les utilisateurs aient divers privilèges selon qu'ils aient spécifié ou pas de mot de passe.Note : N'essayez pas d'encrypter le mot de passe avant l'appel à mysql_real_connect() ; l'encryptage du mot de passe est gérée automatiquement par l'API du client.
- db est le nom de la base de données. Si db n'est pas NULL , la connexion changera la base par défaut en cette valeur.
- Si port est différent de 0, sa valeur sera utilisé comme port de connexion TCP/IP. Notez que le paramètre host détermine le

type de la connexion.

- Si unix_socket n'est pas NULL , la chaîne spécifie la socket ou le tunnel nommé à utiliser. Notez que le paramètre host détermine le type de la connexion.
- La valeur de client_flag est habituellement 0, mais peut être la combinaison des options suivantes dans des circonstances très spéciales :

Nom de l'option	**Description**
CLIENT_COMPRESS	Utilise le protocole compressé.
CLIENT_FOUND_ROWS	Retourne le nombre de lignes trouvées, et non de lignes affectées.
CLIENT_IGNORE_SPACE	Autorise les espaces après les noms de fonctions. Rend tous les noms de fonctions des mots réservés.
CLIENT_INTERACTIVE	Autorise interactive_timeout secondes (au lieu de wait_timeout secondes) d'innactivité avant de fermer la connexion.
CLIENT_LOCAL_FILES	Active le support de LOAD DATA LOCAL .
CLIENT_MULTI_STATEMENTS	Indique au serveur que le client peut envoyer des requêtes multiples (séparées par des ';'). Si cette option n'est pas configurée, les commandes multiples sont désactivées. Nouveau en 4.1.
CLIENT_MULTI_RESULTS	Indique au serveru que le client peut gérer des jeux de résultats multiples, issus de commandes multiples, ou de procédures stockées. C'est automatique si l'option CLIENT_MULTI_STATEMENTS est active. Nouveau en 4.1.
CLIENT_NO_SCHEMA	N'autorise pas la syntaxe db_name.tbl_name.col_name . Cela est fait pour ODBC. Il fait génèrer une erreur à l'analyseur si vous utilisez cette syntaxe, ce qui peut se réveler fort utile pour la chasse aux bogues dans les programmes ODBC.
CLIENT_ODBC	Le client est un client ODBC. Cela rend mysqld plus accueillant vis à vis de ODBC.
CLIENT_SSL	Utilise SSL (protocole encrypté).

Valeur de retour Un gestionnaire de connexion MYSQL* si la connexion a réussi, NULL si elle a échoué. Pour une connexion à succès, la valeur de retour est la même que la valeur du premier paramètre. **Erreurs**

CR_CONN_HOST_ERROR
Impossible de se connecter au serveur MySQL.

CR_CONNECTION_ERROR
Impossible de se connecter au serveur MySQL local.

CR_IPSOCK_ERROR
Impossible de créer une socket IP.

CR_OUT_OF_MEMORY
Plus de mémoire.

CR_SOCKET_CREATE_ERROR
Impossible de créer une socket UNIX.

CR_UNKNOWN_HOST
Impossible de trouver l'adresse IP de l'hôte.

CR_VERSION_ERROR
Une disparité de protocole a résulté de la tentative de connexion à un serveur avec une bibliothèque de client qui utilise une version différente du protocole. Cela peut arriver si vous utilisez une très vieille bibliothèque cliente pour vous connecter à un serveur qui n'a pas été démarré avec l'option --old-protocol .

CR_NAMEDPIPEOPEN_ERROR
Impossible de créer un tunnel nommé sur Windows.

CR_NAMEDPIPEWAIT_ERROR
Impossible d'attendre un tunnel nommé sur Windows.

CR_NAMEDPIPESETSTATE_ERROR
Impossible d'obtenir un gestionnaire de tunnel sur Windows.

CR_SERVER_LOST
Si connect_timeout > 0 et qu'il a fallu plus de connect_timeout secondes pour se connecter au serveur, ou que celui-ci n'a plus répondu durant l'exécution de init-command .

Exemple

```
 MYSQL mysql;

 mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
&#123;
    fprintf(stderr, "Impossible de se connecter à la base de données, erreur : %s\n",
          mysql_error(&mysql));
&#125;
```

En utilisant mysql_options() la bibliothèque MySQL lira les sections [client] et [your_prog_name] dans le fichier my.cnf ce qui assurera le bon fonctionnement de votre programme, même si quelqu'un a configuré MySQL d'une façon non-standard.

Notez que pendant la connexion, `mysql_real_connect()` configure l'option `reconnect` (partie de la structure `MYSQL`) à `1` . Cette option indique, dans le cas où une requête ne peut être exécutée à cause d'une déconnexion, d'essayer de se reconnecter au serveur avant d'abandonner.

21.2.3.44 mysql_real_escape_string()

`unsigned long mysql_real_escape_string(MYSQL *mysql, char *en, const char *de, unsigned long longueur)`

Description Cette fonction est utilisée pour créer une requête SQL légale que vous pouvez utiliser dans une commande SQL. Chaînes .

La string dans `de` est encodée en chaîne échappé SQL, prenom en compte le jeu de caractères de la connexion. Le résultat est placé dans `en` et un octet nul de terminaison est ajouté à la fin de celui−ci. Les caractères encodés sont `NUL` (ASCII 0), `'\n'` , `'\r'` , `'\'` , `'''` , `'"'` , et Ctrl−Z (Comment écrire les chaînes et les nombres ?). (En fait, MySQL a seulement besoin que l'anti−slash et le guillemet utilisé pour entourer la chaîne soient échappés. Cette fonction échappe les autre caractères pour les rendre plus facile à lire dans les fichiers de log.)

La chaîne pointée par de doit avoir une taille de longueur octets. Vous devez allouer à l'espace de en au moins longueur*2+1 octets. (Dans le pire des cas, chaque caractère devra être encodé en utilisant deux octets, et vous avez besoin de place pour l'octet nul de terminaison.) Lorsque mysql_escape_string() retourne un résultat, le contenu de en sera une chaîne terminée par un caractère nul. La valeur de retour est la longueur de la chaîne encodée, n'incluant pas le caractère nul de terminaison. **Exemple**

```
 char query[1000],*end;

 end = strmov(query,"INSERT INTO test_table values(");
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"C'est quoi ça",11);
*end++ = '\'';
*end++ = ',';
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"donnée binaire : \0\r\n",16);
*end++ = '\'';
*end++ = ')';

 if (mysql_real_query(&mysql,query,(unsigned int) (end - query)))
&#123;
   fprintf(stderr, "Impossible d'insérer la ligne, erreur : %s\n",
           mysql_error(&mysql));
&#125;
```

La fonction strmov() utilisée dans cet exemple est inclue dans la bibliothèque mysqlclient et fonctionne comme strcpy() mais retourne un pointeur sur le nul de fin du premier paramètre. **Valeur de retour** La longueur de la valeur passée dans to , n'incluant pas la caractère nul de fin de chaîne.

Erreurs Aucune.

21.2.3.45 mysql_real_query()

`int mysql_real_query(MYSQL *mysql, const char *query, unsigned long length)`

Description Exécute la requête SQL pointée par query , qui doit être une chaîne de caractères de length octets de longueur. La requête ne doit contenir qu'une seule commande. Vous ne devez pas ajouter de point virgule (';') ou \g à la fin de la requête.

Vous **devez** utiliser `mysql_real_query()` au lieu de `mysql_query()` pour les requêtes qui continent des données binaires, car celles−ci peuvent contenir le caractère `'\0'` . De plus, `mysql_real_query()` est plus rapide que `mysql_query()` car elle n'invoque pas `strlen()` sur la chaîne contenant la requête.

Si vous voulez savoir si la requête est censée retourner un jeu de résultat ou non, vous pouvez utiliser mysql_field_count() pour vérifier cela. mysql_field_count() .

Valeur de retour Zéro si la requête a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Les commandes n'ont pas été exécutées dans le bon ordre.

CR_SERVER_GONE_ERROR
: Le serveur MySQL ne réponds pas.

CR_SERVER_LOST
: La connexion au serveur a été perdue au cours la requête.

CR_UNKNOWN_ERROR
: Une erreur inconnue s'est produite.

21.2.3.46 mysql_reload()

`int mysql_reload(MYSQL *mysql)`

Description Demande au serveur MySQL de recharger les tables de droits. L'utilisateur soit avoir les droits RELOAD .

Cette fonction est déconseillée. Il est préférable d'utiliser `mysql_query()` pour exécuter une requête `FLUSH PRIVILEGES` à la place.

Valeur retournée Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.47 mysql_row_seek()

```
MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *result, MYSQL_ROW_OFFSET offset)
```

Description Déplace le curseur de ligne vers une ligne arbitraire dans un jeu de résultats de requête. Cela nécessite que le jeu de résultats contienne la totalité des lignes retournée par la requête, et donc, mysql_row_seek() ne peut être utilisée qu'en conjonction avec mysql_store_result() , et non avec mysql_use_result() .

La position doit être une valeur retournée par un appel à mysql_row_tell() ou à mysql_row_seek() . Cette valeur n'est pas un simple numéro de ligne; si vous voulez vous déplacer dans un jeu de résultats en utilisant le numéro d'une ligne, utilisez mysql_data_seek() .

Valeur de retour La position précédente du curseur de ligne. Cette valeur peut être passée à mysql_row_seek() .

Erreurs Aucune.

21.2.3.48 mysql_row_tell()

```
MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *result)
```

Description Retourne la position courante du pointeur de lignes pour le dernier appel à mysql_fetch_row() . Cette valeur peut être utilisée comme argument de mysql_row_seek() .

Vous ne devez utiliser mysql_row_tell() qu'après mysql_store_result() , et non après mysql_use_result() .

Valeur de retour La position courante du pointeur de ligne.

Erreurs Aucune.

21.2.3.49 mysql_select_db()

```
int mysql_select_db(MYSQL *mysql, const char *db)
```

Description Rend la base de données spécifiée par db la base par défaut (courante) pour la connexion spécifiée par mysql . Pour les requêtes suivantes, cette base de données sera utilisée comme référence pour les tables dont la base de données n'a pas été spécifiée explicitement.

mysql_select_db() échoue si l'utilisateur ne peut être reconnu ayant droit d'accès à la base de données.

Valeur de retour Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.50 mysql_set_server_option()

```
int mysql_set_server_option(MYSQL *mysql, enum enum_mysql_set_option option)
```

Description Active ou désactive une option de connexion. Le paramètre option peut prendre l'une des valeurs suivantes :

MYSQL_OPTION_MULTI_STATEMENTS_ON	Active le support des commandes multiples.
MYSQL_OPTION_MULTI_STATEMENTS_OFF	Désactive le support des commandes multiples.

Valeur retournée Zéro en cas de succes. Non nul si une erreur est survenue. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Les commandes ont été exécutées dans un ordre invalide.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
ER_UNKNOWN_COM_ERROR
Le serveur ne supporte pas mysql_set_server_option() (ce qui peut être le cas d'un serveur antérieur à la version 4.1.1) ou le serveur ne supporte pas l'option qui a été utilisée.

21.2.3.51 mysql_shutdown()

```
int mysql_shutdown(MYSQL *mysql)
```

Description Demande au serveur de base de données de se terminer. L'utilisateur connecté doit avoir le droit SHUTDOWN .

Valeur de retour Zéro si la commande a été effectuée avec succès. Différente de zéro si une erreur est survenue.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.52 mysql_sqlstate()

```
const char *mysql_sqlstate(MYSQL *mysql)
```

Description Retourne une chaîne termineé par null, contenant le code d'erreur SQLSTATE de la dernière erreur. Le code d'erreur est constitué de 5 caractères. '00000' signifie ``pas d'erreur''. Les valeurs sont spécifiées par les normes ANSI SQL et ODBC. Pour une liste des valeurs possibles, voyez Erreurs retournées .

Notez que les erreurs MySQL ne sont pas toutes associées à une erreur SQLSTATE. La valeur 'HY000' (erreur générale) est utilisée pour ces erreurs.

Cette fonction a été ajoutée en MySQL 4.1.1.

Valeur retournée Une chaîne terminée par null, qui contient le code d'erreur SQLSTATE.

Voir aussi mysql_errno() . mysql_error() . mysql_stmt_sqlstate() .

21.2.3.53 mysql_ssl_set()

```
int mysql_ssl_set(MYSQL *mysql, const char *key, const char *cert, const char *ca, const char *capath, const char *cipher)
```

Description mysql_ssl_set() sert à établire des connexions sécurisées par SSL. Elle doit être appelée avec mysql_real_connect() .

mysql_ssl_set() ne fait rien à moins que le support OpenSSL est activéé dans la bibliothèque cliente.

mysql est un gestionnaire de connexion, retourné par mysql_init() . Les autres paramètre sont les suivants :

- key est le chemin jusqu'au fichier de clé.
- cert est le chemin jusqu'au fichier de certificat.
- ca est le chemin jusqu'au fichier d'autorité de certification.
- capath est le chemin jusqu'au dossier qui contient les autorités de certifications SSL reconnus, au format PEM.
- cipher est une liste de chiffrements autorisés avec SSL.

Tous les paramètres SSL inutilisés doivent être fournis avec la valeur NULL .

Valeur retournée Cette fonction retourne toujours 0 . Si la configuration SSL est incorrecte, mysql_real_connect() va retourner une erreur lors de la tentative de connexion.

21.2.3.54 mysql_stat()

```
char *mysql_stat(MYSQL *mysql)
```

Description Retourne une chaîne de caractères contenant des informations similaires à celle fournies par la commande mysqladmin status . Cela inclus le temps de fonctionnement en secondes et le nombre de threads en cours d'exécution, questions, rechargement, et tables ouvertes.

Valeur de retour Une chaîne de caractères décrivant l'état du serveur. NULL si une erreur est survenue.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Les commandes n'ont pas été exécutées dans le bon ordre.

CR_SERVER_GONE_ERROR
: Le serveur MySQL ne réponds pas.

CR_SERVER_LOST
: La connexion au serveur a été perdue au cours la requête.

CR_UNKNOWN_ERROR
: Une erreur inconnue s'est produite.

21.2.3.55 mysql_store_result()

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

Description Vous devez appeler mysql_store_result() ou mysql_use_result() pour chaque requête qui récupère des données avec succès (SELECT , SHOW , DESCRIBE , EXPLAIN).

Vous n'avez pas à appeler mysql_store_result() ou mysql_use_result() pour d'autres requêtes, mais cela ne posera pas de problèmes ou ne ralentira pas vos scripts si vous appelez mysql_store_result() en tout cas. Vous pouvez savoir si la requête n'a pas renvoyé de résultat en vérifiant si mysql_store_result() retourne 0 (nous verrons cela plus tard).

Si vous voulez savoir si la requête devrait renvoyer un jeu de résultats ou non, vous pouvez utiliser mysql_field_count() pour vérifier. mysql_field_count() .

mysql_store_result() lit le résultat en entier et le stocke dans le client, alloue une structure MYSQL_RES , et place le résultat dans cette structure.

mysql_store_result() retourne un pointeur nul si la requête n'a pas retourné un jeu de résultats (si la requête était, par exemple, un INSERT).

mysql_store_result() retourne aussi un pointeur nul si la lecture à partir du jeu de résultats échoue. Vous pouvez vérifier la présence d'erreurs en regardant si mysql_error() ne retourne pas de pointeur nul, si mysql_errno() retourne <> 0, ou si mysql_field_count() retourne <> 0.

Un jeu de résultat vide est retourné si aucune ligne n'est retournée. (Un jeu de résultats vide diffère d'un pointeur nul en tant que valeur de retour.)

Une fois que vous avez appelé mysql_store_result() et obtenu un résultat qui n'est pas un pointeur nul, vous devez appeler mysql_num_rows() pour trouver combien de lignes contient le jeu de résultats.

Vous pouvez appeler mysql_fetch_row() pour récupèrer des lignes à partir du jeu de résultats, ou mysql_row_seek() et mysql_row_tell() pour obtenir ou changer la ligne courante dans le jeu de résultats.

Vous devez appeler mysql_free_result() une fois que vous n'avez plus besoin du résultat.

Pourquoi est ce qu'après mysql_query() ait indiqué un résultat positif, mysql_store_result() retourne parfois NULL ? .

Valeur de retour Une structure de résultat MYSQL_RES . NULL si une erreur survient.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_OUT_OF_MEMORY
Plus de mémoire.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.
CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.56 mysql_thread_id()

```
unsigned long mysql_thread_id(MYSQL *mysql)
```

Description Retourne l'identifiant du thread de la connexion courante. Cette valeur peut être utilisée comme argument de mysql_kill() pour terminer ce thread.

Si la connexion est perdue et que vous vous reconnectez via mysql_ping() , l'identifiant du thread changera. Cela signifie que cela ne sert à rien de récupérer l'identifiant du thread et de le sauvegarder pour l'utiliser plus tard. Vous devez le récupérer quand vous en avez besoin.

Valeur de retour L'identifiant du thread de la connexion courante.

Erreurs Aucune.

21.2.3.57 mysql_use_result()

```
MYSQL_RES *mysql_use_result(MYSQL *mysql)
```

Description Vous devez appeler mysql_store_result() ou mysql_use_result() pour chaque requête qui récupère des données avec succès (SELECT , SHOW , DESCRIBE , EXPLAIN).

mysql_use_result() initialise un jeu de résultats mais ne l'enregistre pas dans le client comme le fait mysql_store_result() . A la place, chaque ligne doit être récupéré manuellement à l'aide de la commande mysql_fetch_row() . Cela lit le résultat directement à partir du serveur sans l'enregistrer dans une table temporaire ou un tampon local, ce qui est plus rapide et utilise moins de mémoire que mysql_store_result() . Le client n'allouera de la mémoire que pour la ligne courante et un tampon de communication qui peut aller jusqu'à max_allowed_packet octets.

D'une autre côté, vous ne devez pas utiliser mysql_use_result() si vous faites beaucoup de traitements pour chaque ligne côté client, ou que le résultat est envoyé à un écran où l'utilisateur peut entrer ^S (arrêt défilement). Cela bloquera le serveur et empêchera les autres threads de mettre à jour n'importe quelle table à partir de laquelle les données sont lues.

Lors de l'utilisation de mysql_use_result() , vous devez exécuter mysql_fetch_row() jusqu'à ce que NULL soit retourné, sinon, les lignes non retournée seront inclues dans le jeu de résultat de votre prochaine requête. L'API C donnera l'erreur Commands out of sync; you can't run this command now si vous oubliez de le faire !

Vous ne devez pas utiliser mysql_data_seek() , mysql_row_seek() , mysql_row_tell() , mysql_num_rows() , ou mysql_affected_rows() avec un résultat retourné par mysql_use_result() , de même, vous ne devez pas exécuter d'autres requêtes tant que la commande mysql_use_result() n'est pas terminée. (Toutefois, après avoir récupéré toutes les lignes, mysql_num_rows() retournera correctement le nombre de lignes récupérées.)

Vous devez appeler mysql_free_result() lorsque vous n'avez plus besoin du jeu de résultats.

Valeur de retour Une structure de résultat MYSQL_RES . NULL si une erreur survient. **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Les commandes n'ont pas été exécutées dans le bon ordre.
CR_OUT_OF_MEMORY
Plus de mémoire.
CR_SERVER_GONE_ERROR
Le serveur MySQL ne réponds pas.

CR_SERVER_LOST
La connexion au serveur a été perdue au cours la requête.
CR_UNKNOWN_ERROR
Une erreur inconnue s'est produite.

21.2.3.58 mysql_warning_count()

```
unsigned int mysql_warning_count(MYSQL *mysql)
```

Description Retourne le nombre l'alertes générées durant l'exécution de la dernière commande SQL. Disponible depuis MySQL 4.1.

Valeur retournée Le nombre d'alertes.

Erreurs Aucune.

21.2.3.59 mysql_commit()

```
my_bool mysql_commit(MYSQL *mysql)
```

Description Valide la transaction courante. Disponible depuis MySQL 4.1

Valeurs retournées Zéro si la fonction réussit; non−nul si une erreur survient.

Erreurs Aucune

21.2.3.60 mysql_rollback()

```
my_bool mysql_rollback(MYSQL *mysql)
```

Description Annule la transaction courante. Disponible avec MySQL 4.1

Valeurs retournées Zéro si l'annulation a réussi, et non−nul si une erreur est survenue.

Erreurs Aucune.

21.2.3.61 mysql_autocommit()

```
my_bool mysql_autocommit(MYSQL *mysql, my_bool mode)
```

Description Active ou désactive le mode d'auto−validation (`autocommit`). Si le paramètre `mode` vaut 1, l'auto−validation est activée. Dans le cas où il vaut 0, l'auto−validation est désactivée. Disponible depuis MySQL 4.1

Valeurs retournées Zéro si la fonction réussit. Non nul si une erreur survient.

Erreurs Aucune.

21.2.3.62 mysql_more_results()

```
my_bool mysql_more_results(MYSQL *mysql)
```

Description Retourne TRUE si il y a d'autre résultats disponibles pour la requête courante, et si l'application doit appeler mysql_next_result() pour lire ces résultats. Disponible en MySQL 4.1

Valeurs retournées `TRUE` si d'autres résultats existent. `FALSE` si il n'y a plus d'autres résultats disponibles.

Notez que dans la plupart des cas, vous pouvez appeler mysql_next_result() pour voir s'il existe plus d'un résultat, et initier le prochain jeu de résultat si c'est le cas.

Gestion des requêtes multiples en C . mysql_next_result() .

Erreurs Aucune.

21.2.3.63 mysql_next_result()

```
int mysql_next_result(MYSQL *mysql)
```

Description S'il existe des résultats disponibles, mysql_next_result() va lire la prochaine ligne, et retourne son statut à l'application. Disponible depuis MySQL 4.1

Notez que vous devez appeler mysql_free_result() pour la précédente requête, si elle retournait un jeu de résultat.

Après avoir appelé mysql_next_result() l'état de la connexion est le même que si vous aviez appelé mysql_real_query() pour la requête suivante. Cela signifie que vous pouvez maintenant appeler mysql_store_result() , mysql_warning_count() , mysql_affected_rows() , etc. sur la connexion.

Si mysql_next_result() retourne une erreur, aucune autre commande ne pourra être exécuté, et il n'y a pas d'autres résultats à lire.

Gestion des requêtes multiples en C .

Valeurs retournées

Valeur retournée	Description
0	Requête réussie et il reste des résultats
-1	Requête réussie et il ne reste pas de résultats
>0	Une erreur est survenue

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Les commandes ont été exécutées dans un ordre invalide. Par exemple, si vous n'avez pas appelé mysql_use_result() avec un résultat précédent.

CR_SERVER_GONE_ERROR
: Le serveur MySQL s'est éteint.

CR_SERVER_LOST
: La connexion au serveur a été perdue durant la requête.

CR_UNKNOWN_ERROR
: Une erreur inconnue est survenue.

21.2.4 Fonctions C de commandes préparées

Depuis MySQL 4.1, vous pouvez aussi utiliser les commandes préparées en utilisant le gestionnaire 'MYSQL_STMT', qui supporte l'exécution de commandes multiples, avec liaison en entrée et sortie.

L'exécution de requêtes préparées est un moyen efficace d'exécuter une requête plus d'une fois. La requête est préparée, ou analysée une première fois. Elle est ensuite exécutée une ou plusieurs fois plus tard, en utilisant un pointeur de commande qui est retourné durant la préparation.

Un autre avantage des commandes préparées est qu'elles utilisent un protocole binaire qui rend les transferts entre le client et le serveur bien plus efficace que l'ancien protocole MySQL.

Les commandes préparées sont plus rapides que les exécutions directes de requêtes qui sont exécutées plus d'une fois, notamment parce que la requête est analysée une seule fois. Dans le cas de l'exécution directe, la requête est analysée à chaque fois. Les commandes préparées permettent de réduire le trafic réseau durant l'exécution, car seuls les données paramètres sont échangées.

21.2.5 Types de données de l'API C

Note : les requêtes préparées sont susceptibles d'être revues. Les informations ci−dessous sont destinées aux nouveaux utilisateurs, mais soyez prévenus qu'elles peuvent changer. Des modifications incompatibles ont été faites en MySQL 4.1.2. Voyez Description des fonctions C de commandes préparées pour plus de détails.

Les requêtes préparées utilisent principalement les deux structures MYSQL_STMT et MYSQL_BIND :

MYSQL_STMT
: Cette structure représente un pointeur de commande sur une commande préparée. Elle sert pour toutes les fonctions liées aux commandes.La commande est initialisée lorsque la requête est préparée en utilisant mysql_prepare() .

  La structure est utilisées par toutes les fonctions ultérieures, liées au commandes préparées.

  La structure MYSQL_STMT n'a pas de membres utilisables par l'application.

  Une connexion peut avoir de multiples pointeurs de commandes, et la limite ne dépend que des ressources systèmes.

MYSQL_BIND
: Cette structure est utilisée pour lier les paramètres avec des buffers, avec mysql_bind_param() , pour utilisation avec mysql_execute() , ainsi que pour lier les résultats avec des buffers via mysql_bind_result() m, lors de la lecture de données avec mysql_fetch() .

  La structure MYSQL_BIND contient les membres suivants : Chacun est utilisé pour la lecture et l'écriture, mais leur objectifs et parfois différent, en fonction du sens de communication.

  enum enum_field_types buffer_type
  : Le type de buffer. Les valeurs possibles pour buffer_type sont listées plus loin dans cette section. Pour l'envoi, buffer_type indique le type de valeur que vous allez fournir, lors de l'association de paramètres. Pour la réception, cela indique le type de valeur que vous pouvez lire dans le buffer de résultat.

  void *buffer
  : Pour l'envoi, c'est un pointeur sur le buffer où sont stockées les valeurs du paramètre de la requête. Pour la réception, c'est un pointeur sur la valeur retournée. Pour les types numériques, buffer pointe sur une variable avec le bon type C. Si vous associez la variable avec une colonne qui a un attribut UNSIGNED , la variable doit être de type C unsigned . Pour les colonnes de type date, buffer doit pointer sur une structure MYSQL_TIME . Pour les caractères et les chaînes binaires, buffer doit pointer sur un buffer de caractères.

  unsigned long buffer_length
  : La taille de *buffer en octets. Pour les caractères et les données binaires C, buffer_length spécifie la taille de *buffer à utiliser comme paramètre si il est utilisé avec mysql_bind_param() , ou la taille lue dans le résultat si il est utilisé avec mysql_bind_result() .

  long *length
  : Un pointeur sur une variable unsigned long qui indique le nombre réel d'octets stockés dans *buffer . length est utilisé pour les données de type caractères ou binaires. Pour l'envoi de données, length point sur une variable unsigned long qui indique la taille des valeurs des paramètres stockés dans *buffer ; cette valeur est utilisée par mysql_execute() . Si length est un pointeur null, le protocole suppose que les données caractères ou binaires sont terminées par null. Pour la lecture, mysql_fetch() place la taille de la valeur de la colonne retournée dans la variable length . length est ignorée pour les valeurs numériques et temporelles, car la taille de ces valeurs sont déterminées par la valeur de buffer_type .

  my_bool *is_null
  : Ce membre pointe sur une variable my_bool qui est true si la valeur est NULL , et false si elle n'est pas NULL . Pour l'envoi, donnez à *is_null la valeur de true pour indiquer que la valeur que vous émettez est NULL . Pour la réception, cette valeur sera true après avoir lu une ligne si le résultat est NULL .

MYSQL_TIME
: Cette structure est utilisée pour écrire et lire des données de type DATE, TIME et TIMESTAMP, directement avec le serveur. Cela se fait en donnant au membre buffer_type d'une structure MYSQL_BIND un des types temporels, et en donnant au membre buffer un pointeur sur une structure MYSQL_TIME . La structure MYSQL_TIME contient les membres suivants :

  unsigned int year
  : L'année.

  unsigned int month
  : Le mois de l'année.

  unsigned int day
  : Le jour du mois.

  unsigned int hour
  : L'heure du jour.

  unsigned int minute
  : La minute de l'heure.

  unsigned int second
  : La seconde de la minute.

  my_bool neg
  : Un booléen pour indiquer que le temps est négatif.

  unsigned long second_part
  : La partie décimale de la seconde. Ce membre est actuellement inutilisé.

  Seuls les membres d'une structure MYSQL_TIME qui s'appliquent à une valeur sont utilisés : les éléments year , month et day sont utilisés pour les types DATE , DATETIME et TIMESTAMP ; les éléments hour , minute et second sont utilisés por les types TIME , DATETIME et TIMESTAMP . Gestion en C des dates et heures .

La table suivante montre les valeurs permises, qui peuvent être spécifiées dans le membre buffer_type des structures MYSQL_BIND . La table montre aussi les types SQL qui correspondent à chaque type de buffer_type , et, pour les types numériques et temporels, le type C correspondant.

buffer_type **Valeur**	**Type SQL**	**Type C**
MYSQL_TYPE_TINY	TINYINT	char
MYSQL_TYPE_SHORT	SMALLINT	short int
MYSQL_TYPE_LONG	INT	long int
MYSQL_TYPE_LONGLONG	BIGINT	long long int
MYSQL_TYPE_FLOAT	FLOAT	float
MYSQL_TYPE_DOUBLE	DOUBLE	double
MYSQL_TYPE_TIME	TIME	MYSQL_TIME
MYSQL_TYPE_DATE	DATE	MYSQL_TIME
MYSQL_TYPE_DATETIME	DATETIME	MYSQL_TIME
MYSQL_TYPE_TIMESTAMP	TIMESTAMP	MYSQL_TIME
MYSQL_TYPE_STRING	CHAR	
MYSQL_TYPE_VAR_STRING	VARCHAR	
MYSQL_TYPE_TINY_BLOB	TINYBLOB/TINYTEXT	
MYSQL_TYPE_BLOB	BLOB/TEXT	
MYSQL_TYPE_MEDIUM_BLOB	MEDIUMBLOB/MEDIUMTEXT	
MYSQL_TYPE_LONG_BLOB	LONGBLOB/LONGTEXT	

Des conversions implicites de type peuvent survenir dans les deux directions de communication.

21.2.6 Présentation des fonctions de l'interface C

Note : les fonctions pour les commandes préparées est toujours en cours de développement. Les informations de cette section sont destinées aux premiers utilisateurs mais il faut être conscient que l'API peut changer. Certains changements sont incompatibles avec MySQL 4.1.2. Voyez Description des fonctions C de commandes préparées pour plus de détails.

Voici les fonctions disponibles pour les commandes préparées. Elles sont listées ici et détailles plus loin. Description des fonctions C de commandes préparées .

Fonction	**Description**
mysql_stmt_init()	Alloue la mémoire pour une structure MYSQL_STMT et l'initialise.
mysql_stmt_bind_param()	Associe un buffer avec une variable de requêtes, dans une commande préparée.
mysql_stmt_bind_result()	Lie les buffers de l'application avec les colonnes d'un résultat.
mysql_stmt_execute()	Exécute une commande préparée.
mysql_stmt_fetch()	Lit la prochaine ligne de données dans le résultat, et retourne toutes les données des colonnes liées.
mysql_stmt_fetch_column()	Lit les données d'une seule colonne, dans le résultat.
mysql_stmt_result_metadata()	Retourne les meta−données de la commande préparée, sous forme d'un jeu de résultat.
mysql_stmt_param_count()	Retourne le nombre de paramètres d'une commande préparée.
mysql_stmt_param_metadata()	Retourne les meta−données des paramètres, sous forme d'un jeu de résultat.
mysql_stmt_prepare()	Prépare une chaîne SQL pour l'exécution.
mysql_stmt_send_long_data()	Envoie de grandes données par parties.
mysql_stmt_affected_rows()	Retourne le nombre de lignes modifiées, effacées ou insérée dans la dernière requête UPDATE , DELETE ou INSERT .
mysql_stmt_insert_id()	Retourne l'identifiant généré par la colonne AUTO_INCREMENT de la dernière commande préparée.
mysql_stmt_close()	Libère une commande préparée de la mémoire.
mysql_stmt_data_seek()	Se place à une numéro de ligne arbitraire dans un jeu de résultat.
mysql_stmt_errno()	Retourne le numéro d'erreur de la dernière requête.
mysql_stmt_error()	Retourne le message d'erreur de la dernière requête.
mysql_stmt_free_result()	Libère les ressources allouées pour la commande.
mysql_stmt_num_rows()	Retourne le nombre total de lignes dans un jeu de résultat bufferisé.
mysql_stmt_reset()	Remet à zéro les buffers de la commande, sur le serveur.
mysql_stmt_row_seek()	Se place à un numéro de ligne, dans un résultat de commande, en utilisant la valeur retournée par mysql_stmt_row_tell() .
mysql_stmt_row_tell()	Retourne la position du curseur de ligne de la commande.
mysql_stmt_sqlstate()	Retourne le code d'erreur SQLSTATE de la dernière opération.
mysql_stmt_store_result()	Lit tout le résultat dans le client.
mysql_stmt_attr_set()	Modifie un attribut d'une commande préparée.
mysql_stmt_attr_get()	Lit la valeur d'un attribut d'une commande préparée.

Appelez `mysql_prepare()` pour préparer et initialiser la commande, puis appelez `mysql_bind_param()` pour fournir les données des paramètres, enfin appelez `mysql_execute()` pour exécuter la requête. Vous pouvez répéter `mysql_execute()` en modifiant les valeurs des paramètres des buffers respectifs via `mysql_bind_param()` .

Dans le cas où la requête est une commande SELECT , ou toute autre commande qui retourne un résultat, alors mysql_prepare() va aussi retourner les méta données de résultat sous la forme d'une structure MYSQL_RES avec mysql_prepare_result() .

Vous pouvez fournir les buffers de résultat avec mysql_bind_result() , pour que mysql_fetch() lise automatiquement les résultats dans les buffers. Cela est fait ligne par ligne.

Vous pouvez aussi envoyer le texte ou les données binaires au serveur en utilisant la fonction mysql_stmt_send_long_data() . mysql_stmt_send_long_data() .

Une fois que l'exécution de la commande est terminée, elle doit être supprimée avec mysql_stmt_close pour que toute les ressources allouées soient détruites.

Etapes d'exécution : Pour préparer et exécuter une commande, l'application :

- appelle mysql_prepare() et passe une chaîne contenant la commande SQL. Si la préparation réussi, mysql_prepare() retourne un pointeur de commande valide.
- Si la requête a un résultat, alors mysql_prepare_result retourne les méta informations de résultat.
- spécifie les valeurs de tous les paramètres de mysql_bind_param . Tous les paramètres doivent être fournis, sinon, cela générera une erreur, ou engendrera des résultats inattendus.
- appelle mysql_execute() pour exécuter la requête.
- Répète les étapes 2 et 3 autant que nécessaire, en modifiant les valeurs des paramètres, et en exécutant à nouveau la commande.
- Lie les buffers de données aux lignes de résultat, si la commande génère un résultat, en utilisant mysql_bind_result() .
- Lit les données dans les buffers, ligne par ligne, en appelant mysql_fetch() jusqu'à ce qu'il n'y ait plus de lignes.

Lorsque mysql_prepare() est appelé, dans le protocole client/serveur MySQL :

- Le serveur analyse la requête et envoie le statut OK au client en lui assignant un identifiant de commande. Il renvoie aussi le nombre total de paramètres, le nombre de colonnes et des meta−informations si un résultat est attendu. La syntaxe et la sémantique de la requête sont vérifiés durant cet appel.
- Le client utilise cet identifiant de commande pour les exécutions ultérieures, pour que le serveur identifie la commande dans le pool de commandes. Désormais, le client alloue un pointeur de commande avec cet identifiant, et le retourne à l'application.

Lorsque mysql_execute() est appelé, avec le protocole client/serveur MySQL :

- Le client utilise le pointeur de commande et envoie les paramètres au serveur.
- Le serveur identifie la commande en utilisant l'identifiant, et remplace les marqueurs de paramètres par leur valeur, puis il exécute la requête. Si cela conduit à un résultat, il est retourné au client, ou bien un statut OK, indiquant le nombre total de ligne affecté est retourné.

Lorsque mysql_fetch() est appelé, dans le protocole client/serveur MySQL :

- Le client lit les données dans le paquet, ligne par ligne, et les place dans les buffers de données, avec les conversions nécessaires. Si le type de buffer de l'application est le même que le type de champs, alors les conversions sont immédiates.

Vous pouvez lire les codes et messages d'erreur, ansi que les codes d'erreur SQLSTATE avec les fonctions mysql_stmt_errno() , mysql_stmt_error() et mysql_stmt_sqlstate() , respectivement.

21.2.7 Descriptions des fonctions C pour les requêtes préparées

Vous devez utiliser ces fonctions lorsque vous voulez préparer et exécuter des commandes.

Note : l'API des commandes préparées est en cours de révision. Ces informations sont destinées aux nouveaux utilisateurs, mais soyez conscients que l'API peut changer sans préavis.

En MySQL 4.1.2, les noms de plusieurs fonctions de commandes préparées ont changé :

Ancien nom	**Nouveau nom**
mysql_bind_param()	mysql_stmt_bind_param()
mysql_bind_result()	mysql_stmt_bind_result()
mysql_prepare()	mysql_stmt_prepare()
mysql_execute()	mysql_stmt_execute()
mysql_fetch()	mysql_stmt_fetch()
mysql_fetch_column()	mysql_stmt_fetch_column()
mysql_param_count()	mysql_stmt_param_count()
mysql_param_result()	mysql_stmt_param_metadata()
mysql_get_metadata()	mysql_stmt_result_metadata()
mysql_send_long_data()	mysql_stmt_send_long_data()

Toutes les fonctions qui utilisent une structure MYSQL_STMT sont préfixées par mysql_stmt_ .

De plus, en version 4.1.2, la signature de la fonction mysql_stmt_prepare() a changé en int mysql_stmt_prepare(MYSQL_STMT *stmt, const char *query, unsigned long length) . Pour créer un pointeur MYSQL_STMT , utilisez la fonction mysql_stmt_init() .

21.2.7.1 mysql_stmt_init()

MYSQL_STMT *mysql_stmt_init(MYSQL *mysql)

Description Crée une structure MYSQL_STMT.

Valeur retournées Un pointeur sur une structure MYSQL_STMT en cas de succès. NULL s'il n'y a plus de mémoire. **Erreurs**

CR_OUT_OF_MEMORY
: Out of memory : plus de mémoire.

21.2.7.2 mysql_stmt_bind_param()

my_bool mysql_stmt_bind_param(MYSQL_STMT *stmt, MYSQL_BIND *bind)

Description mysql_stmt_bind_param() sert à lire des données aux variables de requêtes dans une commande SQL, préparée avec mysql_stmt_prepare() . Elle utilise les structures MYSQL_BIND pour fournir les données. bind est l'adresse d'un tableau de structures MYSQL_BIND . La bibliothèque cliente attend un tableau contenant un élément pour chaque variable de requête '?' qui est présent dans la requête.

Supposez que vous ayez préparé la commande suivante :

```
INSERT INTO mytbl VALUES(?,?,?)
```

Lorsque vous liez les paramètres, le tableau de structures MYSQL_BIND doit contenir trois éléments, et peut être déclaré comme ceci :

```
MYSQL_BIND bind[3];
```

Les membres de chaque structure MYSQL_BIND doit être configuré comme décrit dans la section Types de données de commandes préparées C . **Return Values** Zéro, si l'association a réussi. Non-nul si une erreur est survenue. **Erreurs**

CR_INVALID_BUFFER_USE
: Indique si les données seront fournies par bloc et si le type de buffer n'est pas chaîne, ou binaire.

CR_UNSUPPORTED_PARAM_TYPE
: La conversion n'est pas supportée. Eventuellement, la valeur de buffer_type est invalide, ou n'est pas d'un type supporté.

CR_OUT_OF_MEMORY
: Plus de mémoire.

CR_UNKNOWN_ERROR
: Une erreur inconnue est survenue.

Exemple Pour une exemple avec mysql_stmt_bind_param() , voyez l'exemple de la fonction mysql_stmt_execute() .

21.2.7.3 mysql_stmt_bind_result()

my_bool mysql_stmt_bind_result(MYSQL_STMT *stmt, MYSQL_BIND *bind)

Description mysql_stmt_bind_result() is used to associate (bind) columns in the result set to data buffers and length buffers. When mysql_stmt_fetch() is called to fetch data, the MySQL client/server protocol places the data for the bound columns into the specified buffers.

All columns must be bound to buffers prior to calling mysql_stmt_fetch() . bind is the address of an array of MYSQL_BIND structures. The client library expects the array to contain an element for each column of the result set. Otherwise, mysql_stmt_fetch() simply ignores the data fetch. Also, the buffers should be large enough to hold the data values, because the protocol doesn't return data values in chunks.

A column can be bound or rebound at any time, even after a result set has been partially retrieved. The new binding takes effect the next time mysql_stmt_fetch() is called. Suppose that an application binds the columns in a result set and calls mysql_stmt_fetch() . The client/server protocol returns data in the bound buffers. Then suppose the application binds the columns to a different set of buffers. The protocol does not place data into the newly bound buffers until the next call to mysql_stmt_fetch() occurs.

To bind a column, an application calls mysql_stmt_bind_result() and passes the type, address, and the address of the length buffer. The members of each MYSQL_BIND element that should be set are described in Types de données de commandes préparées C .

Return Values Zero if the bind was successful. Non-zero if an error occurred. **Errors**

CR_UNSUPPORTED_PARAM_TYPE
: The conversion is not supported. Possibly the buffer_type value is illegal or is not one of the supported types.

CR_OUT_OF_MEMORY
Out of memory.
CR_UNKNOWN_ERROR
An unknown error occurred.

Example For the usage of mysql_stmt_bind_result() , refer to the Example from mysql_stmt_fetch() .

21.2.7.4 mysql_stmt_execute()

int mysql_stmt_execute(MYSQL_STMT *stmt)

Description mysql_stmt_execute() exécute la requête préparée, associée avec le pointeur 'stmt'. Les valeurs des marqueurs de paramètres seront envoyées au serveur durant cet appel, pour que le serveur remplace les marqueurs avec les nouvelles valeurs.

Si la commande est UPDATE, DELETE ou INSERT, le nombre total de lignes changées, modifiées ou insérées est accessible avec la fonction mysql_stmt_affected_rows . Si la requête retourne un résultat, alors vous devez appeler la fonction mysql_stmt_fetch() pour lire les données avant d'appeler tout autre fonction de traitement du résultat. Pour plus d'informations sur comment lire les données binaires, voyez aussi mysql_stmt_fetch() . **Valeurs retournées** Zéro si l'exécution a réussi. Non−zéro si une erreur est survenue. Le code d'erreur et le message peuvent être obtenus en appelant les fonctions mysql_stmt_errno() et mysql_stmt_error() . **Erreurs**

CR_COMMANDS_OUT_OF_SYNC
Commands were executed in an improper order : les commandes ont été exécutées dans un ordre invalide.
CR_OUT_OF_MEMORY
Out of memory : plus de mémoire.
CR_SERVER_GONE_ERROR
The MySQL server has gone away : le serveur MySQL s'est éteint.
CR_SERVER_LOST
The connection to the server was lost during the query : la connexion a été perdue durant la requête.
CR_UNKNOWN_ERROR
An unknown error occurred : une erreur inconnue est survenue.

Exemple L'exemple suivant explique l'utilisation de mysql_prepare , mysql_param_count , mysql_bind_param , mysql_stmt_execute et mysql_stmt_affected_rows() .

```
 MYSQL_BIND   bind[3];
MYSQL_STMT   *stmt;
ulonglong    affected_rows;
long         length;
unsigned int param_count;
int          int_data;
short        small_data;
char         str_data[50], query[255];
my_bool      is_null;

   /* Passe en mode d'auto commit */
  mysql_autocommit(mysql, 1);

  if (mysql_query(mysql,"DROP TABLE IF EXISTS test_table"))
  &#123;
    fprintf(stderr, "\n suppression de table a échoué");
    fprintf(stderr, "\n %s", mysql_error(mysql));
    exit(0);
  &#125;
  if (mysql_query(mysql,"CREATE TABLE test_table(col1 int, col2 varchar(50), \
                                                 col3 smallint,\
                                                 col4 timestamp(14))"))
  &#123;
    fprintf(stderr, "\n la création de table a échoué");
    fprintf(stderr, "\n %s", mysql_error(mysql));
    exit(0);
  &#125;

  /* Prepare une requête d'insertion de trois paramètres */
  strmov(query, "INSERT INTO test_table(col1,col2,col3) values(?,?,?)");
  if(!(stmt = mysql_prepare(mysql, query, strlen(query))))
  &#123;
    fprintf(stderr, "\n la prépartion de l\'insertion a échoué");
    fprintf(stderr, "\n %s", mysql_error(mysql));
    exit(0);
  &#125;
  fprintf(stdout, "\n la préparation de l\'insertion a réussi");

   /* Lit le nombre de paramètres de la requête */
  param_count= mysql_param_count(stmt);

   fprintf(stdout, "\n total parameters in insert: %d", param_count);
  if (param_count != 3) /* valide le nombre de paramètres */
  &#123;
    fprintf(stderr, "\n le nombre de paramètres retourné par MySQL est invalide");
    exit(0);
  &#125;
```

```
  /* Lie les données aux paramètres */

  /* INTEGER PART */
 bind[0].buffer_type= MYSQL_TYPE_LONG;
 bind[0].buffer= (char *)&int_data;
 bind[0].is_null= 0;
 bind[0].length= 0;

  /* STRING PART */
 bind[1].buffer_type= MYSQL_TYPE_VAR_STRING;
 bind[1].buffer= (char *)str_data;
 bind[1].buffer_length= sizeof(str_data);
 bind[1].is_null= 0;
 bind[1].length= 0;

  /* SMALLINT PART */
 bind[2].buffer_type= MYSQL_TYPE_SHORT;
 bind[2].buffer= (char *)&small_data;
 bind[2].is_null= &is_null;
 bind[2].length= 0;
 is_null= 0;
  /* Lie les buffers */
 if (mysql_bind_param(stmt, bind))
 &#123;
   fprintf(stderr, "\n param bind failed");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
 &#125;

  /* Spécifie les données */
 int_data= 10;             /* integer */
 strcpy(str_data,"MySQL"); /* string  */

  /* INSERT SMALLINT data as NULL */
 is_null= 1;

  /* Exécute la requête */
 if (mysql_stmt_execute(stmt))
 &#123;
   fprintf(stderr, "\n l\'exécution 1 a échoué");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
 &#125;

 /* Lit le nombre de lignes affectées */
 affected_rows= mysql_stmt_affected_rows(stmt);

  fprintf(stdout, "\n total affected rows: %lld", affected_rows);
 if (affected_rows != 1) /* validation du nombre de lignes affectées */
 &#123;
   fprintf(stderr, "\n nombre de lignes affectées par MySQL invalide");
   exit(0);
 &#125;

  /* Ré-exécute l'insertion, en modifiant les valeurs */
 int_data= 1000;
 strcpy(str_data,"La base de données Open Source la plus populaire");
small_data= 1000;         /* smallint */
 is_null= 0;               /* remet à zéro NULL */

  /* Exécute l'insertion : 2eme */
 if (mysql_stmt_execute(stmt))
 &#123;
   fprintf(stderr, "\n la deuxième exécution a échoué");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
 &#125;

 /* Lit le nombre total de lignes affectées */
 affected_rows= mysql_stmt_affected_rows(stmt);

  fprintf(stdout, "\n Nombre de lignes affectées : %lld", affected_rows);
 if (affected_rows != 1) /* valide le nombre de lignes affectées */
 &#123;
   fprintf(stderr, "\n Nombre de lignes affectées invalides");
   exit(0);
 &#125;

  /* Ferme la requête */
 if (mysql_stmt_close(stmt))
 &#123;
   fprintf(stderr, "\n erreur lors de la fermeture de la commande");
   fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
   exit(0);
 &#125;

 /* Efface la table */
 if (mysql_query(mysql,"DROP TABLE test_table"))
 &#123;
   fprintf(stderr, "\n suppression de table échouée");
   fprintf(stderr, "\n %s", mysql_error(mysql));
   exit(0);
```

```
  &#125;
  fprintf(stdout, "Bravo! les commandes préparées MySQL fonctionnent!!");
```

Note : pour des exemples complets sur l'utilisation des commandes préparées, voyez le fichier tests/client_test.c . Ce fichier est disponible dans la distribution source de MySQL,ou dans le serveur BitKeeper.

21.2.7.5 mysql_stmt_fetch()

int mysql_stmt_fetch(MYSQL_STMT *stmt)

Description mysql_stmt_fetch() retourne la ligne suivante dans le résultat. La fonction peut être appelée uniquement si le résultat existe, c'est à dire après mysql_stmt_execute() qui crée le résultat, ou après mysql_stmt_store_result() , qui est appelé après mysql_stmt_execute() pour mettre en buffer tout le résultat.

Si les lignes sont liées à des buffers avec mysql_stmt_bind_result() , la fonction retourne les données dans ces buffers pour toutes les colonnes de la ligne en cours, et les tailles sont retournées dans le pointeur de taille.

Notez que toutes les colonnes doivent être liées par l'application avant d'appeler mysql_stmt_fetch() .

Si les données lues contiennent la valeur NULL, alors la valeur is_null de MYSQL_BIND contiendra TRUE, 1, ou sinon, les données et leur longueur seront reoturnées dans les variables *buffer et *length , basées sur le type de buffer, spécifié par l'application. Tous les nombres ont une taille fixe, listée en octet ci−dessous. La taille des types chaînes dépend des données, comme indiqué dans data_length .

Type	**Length**
MYSQL_TYPE_TINY	1
MYSQL_TYPE_SHORT	2
MYSQL_TYPE_LONG	4
MYSQL_TYPE_LONGLONG	8
MYSQL_TYPE_FLOAT	4
MYSQL_TYPE_DOUBLE	8
MYSQL_TYPE_TIME	sizeof(MYSQL_TIME)
MYSQL_TYPE_DATE	sizeof(MYSQL_TIME)
MYSQL_TYPE_DATETIME	sizeof(MYSQL_TIME)
MYSQL_TYPE_TIMESTAMP	sizeof(MYSQL_TIME)
MYSQL_TYPE_STRING	data length
MYSQL_TYPE_VAR_STRING	data_length
MYSQL_TYPE_TINY_BLOB	data_length
MYSQL_TYPE_BLOB	data_length
MYSQL_TYPE_MEDIUM_BLOB	data_length
MYSQL_TYPE_LONG_BLOB	data_length

où *data_length ne vaut rien d'autre que 'la taille réelle des donées'. **Valeurs retournées**

Return Value	**Description**
0	Réussi. Les données ont été lues, et placées dans les buffers.
1	Une erreur est survenue. Le code d'erreur et le message d'erreur sont disponibles grâce aux fonctions mysql_stmt_errno() et mysql_stmt_error() .
100, MYSQL_NO_DATA	Il ne reste plus de données ou de lignes.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Commands were executed in an improper order : les commandes ont été exécutées dans un ordre invalide.

CR_OUT_OF_MEMORY
: Out of memory : plus de mémoire.

CR_SERVER_GONE_ERROR
: The MySQL server has gone away : Le serveur MySQL s'est éteint.

CR_SERVER_LOST
: The connection to the server was lost during the query : la connexion au serveur a été perdue durant la requête.

CR_UNKNOWN_ERROR
: An unknown error occurred : Une erreur inconnue est survenue.

CR_UNSUPPORTED_PARAM_TYPE
: If the buffer type is MYSQL_TYPE_DATE,DATETIME,TIME,or TIMESTAMP; and if the field type is not DATE, TIME, DATETIME or TIMESTAMP Le buffer est de type MYSQL_TYPE_DATE,DATETIME,TIME ou TIMESTAMP et le type de champs n'est pas DATE, TIME, DATETIME or TIMESTAMP.

Toutes les autres erreurs de conversions non supportées sont
: disponibles avec mysql_bind_result() .

Exemple L'exemple ci−dessous explique l'utilisation de mysql_get_metadata() , mysql_bind_result() et mysql_stmt_fetch() Cette exemple s'attend à lire les deux lignes insérées dans l'exemple de mysql_stmt_execute() .) La variable mysql est supposée être une connexion valide.

```
 #define STRING_SIZE 50

 #define SELECT_SAMPLE "SELECT col1, col2, col3, col4 FROM test_table"

 MYSQL_STMT    *stmt;
MYSQL_BIND    bind[4];
MYSQL_RES     *prepare_meta_result;
```

```
MYSQL_TIME    ts;
unsigned long length[4];
int           param_count, column_count, row_count;
short         small_data;
int           int_data;
char          str_data[STRING_SIZE];
my_bool       is_null[4];

 /* Prépare une commande SELECT pour lire les données dans la table test_table */
stmt = mysql_prepare(mysql, SELECT_SAMPLE, strlen(SELECT_SAMPLE));
if (!stmt)
&#123;
  fprintf(stderr, " mysql_prepare(), SELECT failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
&#125;
fprintf(stdout, " prepare, SELECT successful\n");

 /* Lit le nombre de paramètrs de la commande */
param_count= mysql_param_count(stmt);
fprintf(stdout, " total parameters in SELECT: %d\n", param_count);

 if (param_count != 0) /* validate parameter count */
&#123;
  fprintf(stderr, " invalid parameter count returned by MySQL\n");
  exit(0);
&#125;

 /* Lit les méta-données */
prepare_meta_result = mysql_get_metadata(stmt);
if (!prepare_meta_result)
&#123;
  fprintf(stderr, " mysql_get_metadata(), returned no meta information\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
&#125;

 /* Lit le nombre de colonnes de la requête */
column_count= mysql_num_fields(prepare_meta_result);
fprintf(stdout, " total columns in SELECT statement: %d\n", column_count);

 if (column_count != 4) /* validate column count */
&#123;
  fprintf(stderr, " invalid column count returned by MySQL\n");
  exit(0);
&#125;

 /* Exécute la requête SELECT */
if (mysql_execute(stmt))
&#123;
  fprintf(stderr, " mysql_execute(), failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
&#125;

 /* Lie les buffers de résultats pour les 4 colonnes avant de les lire */

 /* INTEGER COLUMN */
bind[0].buffer_type= MYSQL_TYPE_LONG;
bind[0].buffer= (char *)&int_data;
bind[0].is_null= &is_null[0];
bind[0].length= &length[0];

 /* STRING COLUMN */
bind[1].buffer_type= MYSQL_TYPE_VAR_STRING;
bind[1].buffer= (char *)str_data;
bind[1].buffer_length= STRING_SIZE;
bind[1].is_null= &is_null[1];
bind[1].length= &length[1];

 /* SMALLINT COLUMN */
bind[2].buffer_type= MYSQL_TYPE_SHORT;
bind[2].buffer= (char *)&small_data;
bind[2].is_null= &is_null[2];
bind[2].length= &length[2];

 /* TIMESTAMP COLUMN */
bind[3].buffer_type= MYSQL_TYPE_TIMESTAMP;
bind[3].buffer= (char *)&ts;
bind[3].is_null= &is_null[3];
bind[3].length= &length[3];

 /* Lit les résultats */
if (mysql_bind_result(stmt, bind))
&#123;
  fprintf(stderr, " mysql_bind_result() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
&#125;

 /* Maintenant, lis les résultats dans les buffers */
if (mysql_stmt_store_result(stmt))
```

```
&#123;
  fprintf(stderr, " mysql_stmt_store_result() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
&#125;

 /* Lit toutes les lignes */
row_count= 0;
fprintf(stdout, "Fetching results ...\n");
while (!mysql_stmt_fetch(stmt))
&#123;
  row_count++;
  fprintf(stdout, "  row %d\n", row_count);

   /* colonne 1 */
  fprintf(stdout, "   column1 (integer)  : ");
  if (is_null[0])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %d(%ld)\n", int_data, length[0]);

   /* colonne 2 */
  fprintf(stdout, "   column2 (string)   : ");
  if (is_null[1])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %s(%ld)\n", str_data, length[1]);

   /* colonne 3 */
  fprintf(stdout, "   column3 (smallint) : ");
  if (is_null[2])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %d(%ld)\n", small_data, length[2]);

   /* colonne 4 */
  fprintf(stdout, "   column4 (timestamp): ");
  if (is_null[3])
    fprintf(stdout, " NULL\n");
  else
    fprintf(stdout, " %04d-%02d-%02d %02d:%02d:%02d (%ld)\n",
                                               ts.year, ts.month, ts.day,
                                               ts.hour, ts.minute, ts.second,
                                               length[3]);
  fprintf(stdout, "\n");
&#125;

 /* Valide la ligne lue */
fprintf(stdout, " total rows fetched: %d\n", row_count);
if (row_count != 2)
&#123;
  fprintf(stderr, " MySQL failed to return all rows\n");
  exit(0);
&#125;

 /* Libère les méta-données de résultat */
mysql_free_result(prepare_meta_result);
 /* Ferme la commande */
if (mysql_stmt_close(stmt))
&#123;
  fprintf(stderr, " failed while closing the statement\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
&#125;
```

21.2.7.6 mysql_stmt_fetch_column()

`int mysql_stmt_fetch_column(MYSQL_STMT *stmt, MYSQL_BIND *bind, unsigned int column, unsigned long offset)`

A définir.

Description

Valeur retournée

Erreurs

21.2.7.7 mysql_stmt_result_metadata()

`MYSQL_RES *mysql_stmt_result_metadata(MYSQL_STMT *stmt)`

Description Si la fonction mysql_prepare() a généré un résultat, alors mysql_stmt_result_metadata() retourne les méta données de résultats sous la forme d'un structure MYSQL_RES , qui peut être utilisée ultérieurement pour traiter des méta informations, telles qu le nombre de champs et les informations individuelles de champs. Ce résultat peut être passé en argument à l'une des fonctions de champs suivantes, pour traiter les données :

- mysql_num_fields()
- mysql_fetch_field()
- mysql_fetch_field_direct()
- mysql_fetch_fields()
- mysql_field_count()
- mysql_field_seek()
- mysql_field_tell()
- mysql_free_result()

La structure de jeu de résultats doit être libérée une fois que vous en ave fini avec, grâce à la fonction mysql_free_result() . C'est similaire à la méthode pour libérer les ressources obtenus de mysql_store_result() .

Le jeu de résultats retourné par mysql_stmt_result_metadata() contient uniquement des méta−données. Il ne contient aucune ligne de résultat. Les lignes sont lues en utitilisant la ressource de commande, avec la fonction mysql_stmt_fetch() .

Valeurs retournées Une structure de type MYSQL_RES . NULL si aucune méta données n'existe pour la requête préparée. **Erreurs**

CR_OUT_OF_MEMORY
: Out of memory : plus de mémoire

CR_UNKNOWN_ERROR
: An unknown error occured : Une erreur inconnue est survenue.

Exemple Pour une illustration de la fonction mysql_stmt_result_metadata() , voyez l'exemple de la fonction mysql_stmt_fetch() .

21.2.7.8 mysql_stmt_param_count()

```
unsigned long mysql_stmt_param_count(MYSQL_STMT *stmt)
```

Description Retourne le nombre de marqueurs de paramètres présents dans la requête préparée.

Valeurs retournées Un entier non signé, représentant le nombre de paramètres dans la requête.

Erreurs Aucune.

Exemple Pour une illustration de la fonction mysql_stmt_param_count() , voyez l'exemple de la fonction mysql_stmt_execute() .

21.2.7.9 mysql_stmt_param_metadata()

```
MYSQL_RES *mysql_stmt_param_metadata(MYSQL_STMT *stmt)
```

A définir.

Description

Valeurs retournées

Erreurs

21.2.7.10 mysql_stmt_prepare()

```
int mysql_stmt_prepare(MYSQL_STMT *stmt, const char *query, unsigned long length)
```

Description Prépare la requête représentée par la chaîne terminée par NUL query , et retourne un pointeur de commande à utiliser ultérieurement pour les autres opérations. La requête doit contenir une commande SQL unique. Vous ne devez pas ajouter le point-virgule (';') ni \g de fin de requête.

L'application peut inclure une ou plusieurs variable de requête SQL, grâce au caractère point d'interrogation ('?'), placé dans la commande SQL, aux bons endroits.

Les variables de requêtes ne sont valides qu'à certaines places dans les commandes SQL. Par exemple, elles sont autorisées dans les listes VALUES() d'une commande INSERT (pour spécifier les valeurs des lignes), ou dans les clauses de comparaisons WHERE , pour spécifier une valeur de comparaison. Sinon, elles ne sont pas autorisées pour les identifiants (comme les noms de tables ou de colonnes), dans les listes de colonnes sélectionnées par la commande SELECT , ou pour spécifier un opérateur tel que = . Cette dernière restriction est due au fait qu'il serait impossible de déterminer le type de paramètre. En général, les variables ne sont autorisées que dans les commandes de manipulations de données (Data Manipulation Language (DML)), et non pas dans les commandes de définition des données (Data Defination Language (DDL) .

Les variables de requêtes doivent être liés par l'application à des variables, avec la fonction `mysql_stmt_bind_param()` avant exécution.

Valeur retournées Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Commands were executed in an improper order : les commandes ont été exécutées dans un ordre invalide.

CR_OUT_OF_MEMORY
: Out of memory : plus de mémoire.

CR_SERVER_GONE_ERROR
: The MySQL server has gone away : le serveur s'est éteint durant l'exécution de la requête.

CR_SERVER_LOST
: The connection to the server was lost during the query : la connexion au serveur a été perdue.

CR_UNKNOWN_ERROR
: An unknown error occurred : erreur inconnue.

Si la préparation échoue, c'est à dire si mysql_stmt_prepare() retourne NULL), un message d'erreur peut être obtenu en appelant mysql_error() . **Exemple** Pour une utilisation de mysql_stmt_prepare() , voyez l'exemple dans mysql_stmt_execute() .

21.2.7.11 mysql_stmt_send_long_data()

my_bool mysql_stmt_send_long_data(MYSQL_STMT *stmt, unsigned int parameter_number, const char *data, unsigned long length)

Description Permet à une application d'envoyer des données par parties au serveur. Cette fonction doit être utilisée pour envoyer des caractères ou du contenu binaire par partie dans une colonne (qui sera de type TEXT ou BLOB), avec un type de caractère ou de données binaires.

Le paramètre data est un pointeur sur un buffer qui contient les données pour le paramètre, représenté par parameter_number . Le paramètre length indique la quantité de données qui doit être envoyée, en octets.

Note : le prochain appel à `mysql_stmt_execute()` va ignorer les buffers de variables de requêtes, pour tous les paramètres qui ont été utilisé avec `mysql_stmt_send_long_data()` depuis le dernier appel à `mysql_stmt_execute()` ou `mysql_stmt_reset()` .

Si vous voulez remettre à zéro cette fonction, utilisez mysql_stmt_reset() . mysql_stmt_reset() .

Valeurs retournées Zéro si les données ont pu être envoyées au serveur. Non−nul si une erreur est survenue.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Commands were executed in an improper order : Les conmmandes ont été envoyées dans un ordre invalide.

CR_SERVER_GONE_ERROR
: The MySQL server has gone away : le serveur MySQL s'est éteind.

CR_OUT_OF_MEMORY
: Out of memory : plus de mémoire.

CR_UNKNOWN_ERROR
: An unknown error occured : une erreur inconnue s'est produite.

Exemple L'exemple ci−dessous explique comment envoyer des données par partie dans une colonne de type TEXT :

```
 #define INSERT_QUERY "INSERT INTO test_long_data(text_column) VALUES(?)"

MYSQL_BIND bind[1];
long       length;

 smtt = mysql_stmt_init(mysql);
if (!stmt)
&#123;
  fprintf(stderr, " mysql_stmt_init(), out of memory\n");
  exit(0);
&#125;
if (mysql_stmt_prepare(stmt, INSERT_QUERY, strlen(INSERT_QUERY)))
&#123;
  fprintf(stderr, "\n mysql_stmt_prepare(), INSERT failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
```

```
&#125;
 memset(bind, 0, sizeof(bind));
 bind[0].buffer_type= MYSQL_TYPE_STRING;
 bind[0].length= &length;
 bind[0].is_null= 0;

   /* Liaison des buffers */
if (mysql_stmt_bind_param(stmt, bind))
&#123;
  fprintf(stderr, "\n param bind failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
&#125;

    /* Envoi des données au serveur, par parties */
 if (!mysql_stmt_send_long_data(stmt,0,"MySQL",5))
&#123;
  fprintf(stderr, "\n send_long_data failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
&#125;

    /* Envoi des données suivantes */
 if (mysql_stmt_send_long_data(stmt,0," − The most popular open source database",40))
&#123;
  fprintf(stderr, "\n send_long_data failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
&#125;

    /* Exécution de la requête */
 if (mysql_stmt_execute(stmt))
&#123;
  fprintf(stderr, "\n mysql_stmt_execute failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
&#125;
```

21.2.7.12 mysql_stmt_affected_rows()

`my_ulonglong mysql_stmt_affected_rows(MYSQL_STMT *stmt)`

Description Retourne le nombre total de ligne modifiées par la dernière commande. Cette fonction peut être appelée immédiatement après la fonction mysql_execute() pour les commandes UPDATE, DELETE ou INSERT. Pour les commandes SELECT, mysql_stmt_affected() fonctionne comme mysql_num_rows().

Valeurs retournées Un entier supérieur à zéro indique le nombre de ligne affectées ou lues. Zéro indique qu'aucune ligne n'a été modifiées durant une commande UPDATE, ou qu'aucune ligne n'a vérifié la clause WHERE dans la requête, ou qu'aucune requête n'a été exécuté. −{}1 indique que la requête a retourné une erreur, ou que, pour une requête SELECT, mysql_stmt_affected_rows() a été appelé avant mysql_fetch(). Comme `mysql_stmt_affected_rows()` retourne une valeur non signée, vous pouvez surveiller la valeur −{}1 en analysant la valeur retournée par `(my_ulonglong)-1` (ou to `(my_ulonglong)~0` , qui est équivalent).

Erreurs Aucune.

Exemple Plus une illustration de `mysql_stmt_affected_rows()` voyez l'exemple de `mysql_stmt_execute()` .

21.2.7.13 mysql_stmt_insert_id()

`my_ulonglong mysql_stmt_insert_id(MYSQL_STMT *stmt)`

Description Retourne la valeur générée pour une colonne de type AUTO_INCREMENT par une requête préparée INSERT ou UPDATE . Utilisez cette fonction après avoir exécuté la commande INSERT sur la table qui contient la colonne AUTO_INCREMENT .

Voyez `mysql_insert_id()` pour plus de détails.

Valeurs retournées La valeur générée puor la colonne AUTO_INCREMENT qui était automatiquement générée ou explicitement donnée durant l'exécution de la requête, ou la valeur générée par la dernière fonction LAST_INSERT_ID(expr) . La valeur retournée est indéfinie si la commande n'a pas manipulé de valeur AUTO_INCREMENT .

Erreurs Aucune.

21.2.7.14 mysql_stmt_close()

my_bool mysql_stmt_close(MYSQL_STMT *)

Description Termine la commande préparée. mysql_stmt_close() va aussi désallouer le pointeur de commande alloué par stmt .

Si les résultats de la requête courante sont en attente, ou non lus, ils seront annulés. Le prochain appel pourra donc être exécuté.

Valeur retournée Zéro si la commande a pu être terminée. Une valeur non nulle si une erreur est survenue. **Erreurs**

CR_SERVER_GONE_ERROR
The MySQL server has gone away
CR_UNKNOWN_ERROR
An unkown error occured

Exemple Pour une illustration de la fonction mysql_stmt_close() voyez un exemple avec mysql_stmt_execute() .

21.2.7.15 mysql_stmt_data_seek()

void mysql_stmt_data_seek(MYSQL_STMT *stmt, my_ulonglong offset)

Description Place le pointeur de résultat à une ligne arbitaire. La valeur de offset est un numéro de ligne, et doit être dans l'intervalle 0 à mysql_stmt_num_rows(stmt)-1 .

Cette fonction impose que la structure du jeu de résultat soit entièrement téléchargée : mysql_stmt_data_seek() ne peut être utilisée qu'avec mysql_stmt_store_result() .

Valeurs retournées Aucune.

Erreurs Aucune.

21.2.7.16 mysql_stmt_errno()

unsigned int mysql_stmt_errno(MYSQL_STMT *stmt)

Description Pour la commande spécifiée par stmt , mysql_stmt_errno() retourne le code d'erreur de la plus récente fonction d'API appelée, qu'elle ait réussi ou échoué. Une valeur de zéro indique qu'il n'y a pas eu d'erreur. Les numéros d'erreurs clients sont listés dans le fichier d'entêtes errmsg.h . Les messages d'erreurs serveurs sont listés dans le fichier d'entêtes mysqld_error.h . Dans la distribution source de MySQLm vous pouvez trouver une liste complète des messages d'erreurs et de leur numéro, dans le fichier Docs/mysqld_error.txt . Les codes d'erreur du serveur sont aussi listées dans Erreurs retournées .

Valeurs retournées Une valeur représentant un code d'erreur. Zéro représente l'absence d'erreur.

Erreurs Aucune

21.2.7.17 mysql_stmt_error()

char *mysql_stmt_error(MYSQL_STMT *stmt)

Description Pour la commande spécifiée par stmt , mysql_stmt_error() retourne le message d'erreur de la fonction d'API la plus récemment appelée, qu'elle ait réussi ou pas. Une chaîne vide ("") est retournée si aucune erreur n'est survenue. Cela signifie que les instructions suivantes sont identiques :

```
 if (mysql_stmt_errno(stmt))
&#123;
  // une erreur est survenue
&#125;

 if (mysql_stmt_error(stmt))
&#123;
  // une erreur est survenue
&#125;
```

La langue utilisée pour les messages d'erreurs du client MySQL peuvent être modifiées à la compilation de la bibliothèque cliente MySQL. Actuellement, vous pouvez choisir les message d'erreur dans plusieurs langues.

Valeurs retournées Une chaîne de caractères qui décrit l'erreur. Une chaîne vide signifie qu'il n'y a pas eu d'erreur.

Erreurs Aucune

21.2.7.18 mysql_stmt_free_result()

```
my_bool mysql_stmt_free_result(MYSQL_STMT *stmt)
```

A définir.

Description

Valeur retournée Erreurs

21.2.7.19 mysql_stmt_num_rows()

```
my_ulonglong mysql_stmt_num_rows(MYSQL_STMT *stmt)
```

Description Retourne le nombre de lignes dans le jeu de résultat.

L'utilisation de `mysql_stmt_num_rows()` dépend du fait que vous avez utilisé ou non `mysql_stmt_store_result()` pour rapatrier l'intégralité du résultat dans le client.

Si vous utilisez mysql_stmt_store_result() , mysql_stmt_num_rows() peut être appelé immédiamtement.

Valeur retournée Le nombre de lignes dans le jeu de résultat.

Erreurs Aucun.

21.2.7.20 mysql_stmt_reset()

```
my_bool mysql_stmt_reset(MYSQL_STMT *stmt)
```

Description Remet la commande préparée sur le client et sur le serveur à son état juste après la prépartion. Actuellement, on s'en sert surtout pour remettre à zéro les données envoyées par `mysql_stmt_send_long_data()` .

Pour préparer à nouveau la commande pour une autre commande, utilisez mysql_stmt_prepare() .

Valeurs retournées Zéro si la commande a été remise à zéro. Non−nulle, si une erreur est survenue.

Erreurs

CR_COMMANDS_OUT_OF_SYNC
: Commands were executed in an improper order. : Les commandes ont été exécutées dans un ordre invalide.

CR_SERVER_GONE_ERROR
: The MySQL server has gone away : Le serveur s'est éteind.

CR_SERVER_LOST
: The connection to the server was lost during the query : La connexion au serveur a été perdue durant l'exécution.

CR_UNKNOWN_ERROR
: An unknown error occurred. : Une erreur inconnue est survenue.

21.2.7.21 mysql_stmt_row_seek()

```
MYSQL_ROW_OFFSET mysql_stmt_row_seek(MYSQL_STMT *stmt, MYSQL_ROW_OFFSET offset)
```

Description Place le pointeur de lignes à une position arbitraire dans le jeu de résultats. La valeur offset est un offset de ligne, qui doit être retourné par mysql_stmt_row_tell() ou par mysql_stmt_row_seek() . Cette valeur n'est pas un numéro de ligne : si vous voulez atteindre une ligne dans un résultat, à partir de son numéro, vous devez utiliser mysql_stmt_data_seek() .

Cette fonction requiert que le jeu de résultat entier soit téléchargé, et donc, mysql_stmt_row_seek() ne peut être utilisé qu'avec mysql_stmt_store_result() .

Valeur retournée La position précédente du curseur de ligne. Cette valeur peut être passée à mysql_stmt_row_seek() .

Erreurs Aucune.

21.2.7.22 mysql_stmt_row_tell()

```
MYSQL_ROW_OFFSET mysql_stmt_row_tell(MYSQL_STMT *stmt)
```

Description Retourne la position courante du pointeur de ligne. C'est la position à laquelle le dernier appel à mysql_fetch() l'a laissé. Cette valeur peut être utilisée comme argument avec mysql_stmt_row_seek() .

Vous ne devez utiliser mysql_stmt_row_tell() qu'après mysql_stmt_store_result() .

Valeur retournée La position courante du pointeur de lignes.

Erreurs Aucune.

21.2.7.23 mysql_stmt_sqlstate()

```
const char *mysql_stmt_sqlstate(MYSQL_STMT *stmt)
```

Description Pour la commande stmt , mysql_stmt_sqlstate() retourne une chaîne terminée par un null, contenant le code d'erreur SQLSTATE de la plus récente fonction de requête préparée qui ait été utilisée. Le code d'erreur est constitué de 5 caractères. "00000" signifie ``pas d'erreur''. Les valeurs sont spécifiées par les normes ANSI SQL et ODBC. Pour une liste des valeurs possibles, voyez Erreurs retournées .

Notez que toutes les erreurs MySQL ne sont pas associée à une erreur SQLSTATE. La valeur "HY000" (erreur générale) sert pour les erreurs orphelines.

Cette fonction a été ajoutée en MySQL 4.1.1.

Valeur retournée Une chaîne de caractères terminée par un null, contenant le code d'erreur SQLSTATE.

21.2.7.24 mysql_stmt_store_result()

```
int mysql_stmt_store_result(MYSQL_STMT *stmt)
```

Description Vous devez appeler la fonction mysql_stmt_store_result() pour chaque requête qui doit lire de données (SELECT , SHOW , DESCRIBE , EXPLAIN) et uniquement si vous voulez lire la totalité du résultat dans un buffer du client, pour que les appels suivants à mysql_fetch() retourne des données bufferisées.

Vous n'avez pas à appeler mysql_stmt_store_result() pour les requêtes suivantes, mais cela ne causera pas de ralentissement notable. Vous pouvez détecter si une requête n'a pas de résultat en vérifiant si mysql_prepare_result() retourne 0. Pour plus d'informations, voyez mysql_stmt_result_metadata() .

Note : MySQL ne calcule pas par défaut MYSQL_FIELD->max_length pour toutes les colonnes de mysql_stmt_store_result() car ce calcul ralentirait considérablement mysql_stmt_store_result() et la plupart des applications n'ont pas besoin de max_length . Si vous voulez max_length , vous pouvez appeler mysql_stmt_attr_set(MYSQL_STMT, STMT_ATTR_UPDATE_MAX_LENGTH, pour l'obtenir. mysql_stmt_attr_set() .

Valeurs retournées `Zéro` si les résultats sont mis en buffer correctement, et non−nul si une erreur survient.

Errors

CR_COMMANDS_OUT_OF_SYNC
: Commands were executed in an improper order : les commandes ont été exécutées dans un ordre invalide.

CR_OUT_OF_MEMORY
: Out of memory : plus de mémoire.

CR_SERVER_GONE_ERROR
: The MySQL server has gone away : le serveur MySQL s'est éteind.

CR_SERVER_LOST
: The connection to the server was lost during the query : la connexion au serveur MySQL s'est interrompue durant la commande.

CR_UNKNOWN_ERROR
: An unknown error occurred : une erreur inconnue est survenue.

21.2.7.25 mysql_stmt_attr_set()

`int mysql_stmt_attr_set(MYSQL_STMT *stmt, enum enum_stmt_attr_type option, const void *arg)`

Description Sert à modifier le comportement d'une commande. Cette fonction peut être appelée plusieurs fois.

L'argument `option` est le nom de l'option que vous voulez modifier; le pointeur `arg` doit pointer sur une variable qui contient la valeur de l'option. Si l'option est un entier, alors `arg` doit être un pointeur.

Valeurs possibles pour les options :

Option	**Type d'argument**	**Fonction**
STMT_ATTR_UPDATE_MAX_LENGTH	my_bool *	Si la valeur est 1 : modifie les meta-données MYSQL_FIELD->max_length dans mysql_stmt_store_result() .

Valeur retournée 0 si tout va bien. 1 si attr_type est inconnu.

Erreurs aucune

21.2.7.26 mysql_stmt_attr_get()

`int mysql_stmt_attr_get(MYSQL_STMT *stmt, enum enum_stmt_attr_type option, void *arg)`

Description Sert à lire la valeur courante pour un attribut de commande.

L'argument `option` est le nom de l'option que vous voulez lire; le pointeur `arg` doit pointer sur une variable qui contient la valeur de l'option. Si l'option est un entier, alors `arg` doit être un pointeur.

Voyez mysql_stmt_attr_set pour avoir la liste des options et leur type. mysql_stmt_attr_set() .

Valeur retournée `0` si tout va bien. `1` si `attr_type` est inconnu.

Erreurs aucune

21.2.8 Gestion des commandes multiples avec l'interface C

Depuis la version 4.1, MySQL supporte l'exécution de requêtes multiples dans une seule commande. Pour cela, vous devez activer l'option client `CLIENT_MULTI_QUERIES` lors de l'ouverture de la connexion.

Par défaut, `mysql_query()` ou `mysql_real_query()` ne retournent que le statut de la première requête, et les statuts suivants peut être obtenu avec `mysql_more_results()` et `mysql_next_result()` .

```
 /* Connexion au serveur, avec l'option CLIENT_MULTI_QUERIES */
mysql_real_query(..., CLIENT_MULTI_QUERIES);
```

```
  /* Exécution de plusieurs requêtes */
 mysql_query(mysql,"DROP TABLE IF EXISTS test_table;\
                   CREATE TABLE test_table(id int);\
                   INSERT INTO test_table VALUES(10);\
                   UPDATE test_table SET id=20 WHERE id=10;\
                   SELECT * FROM test_table;\
                   DROP TABLE test_table";
 while (mysql_more_results(mysql))
 &#123;
   /* Traitement de tous les résultats */
   mysql_next_result(mysql);
   ...
   printf("total affected rows: %lld", mysql_affected_rows(mysql));
   ...
   if ((result= mysql_store_result(mysql))
   &#123;
     /* Retourne un résultat, le traite */
   &#125;
 &#125;
```

21.2.9 Gestion des dates et horaires avec l'interface C

En utilisant le nouveau protocole binaire de MySQL 4.1 et plus récent, vous pouvez envoyer et recevoir les données de type (`DATE` , `TIME` , `DATETIME` et `TIMESTAMP`) avec la structure `MYSQL_TIME` . Les membres de cette structure sont décrits dans Types de données de commandes préparées C .

Afin d'envoyer les données, il faut utiliser une requête préparée avec la fonction `mysql_prepare()` . Avant d'appeler la fonction `mysql_execute()` , pour exécuter la commande, utilisez la procédure suivante pour préparer chaque donnée :

- Dans la structure MYSQL_BIND , associée aux données, assignez au membre buffer_type le type de données que vous envoyez. Pour DATE , TIME , DATETIME , ou TIMESTAMP , utilisez buffer_type MYSQL_TYPE_DATE , MYSQL_TYPE_TIME , MYSQL_TYPE_DATETIME , ou MYSQL_TYPE_TIMESTAMP , respectivement.
- Donnez au membre buffer de la structure MYSQL_BIND , l'adresse de la structure MYSQL_TIME dans laquelle vous avez stocké votre valeur temporelle.
- Remplissez les membres de la structure MYSQL_TIME qui sont adaptés au type de données que vous passez.

Utilisez `mysql_bind_param()` pour lier les données à la requête. Puis, appelez `mysql_execute()` .

Pour lire des données temporelles, la procédure est similaire, hormis le fait que vous donnez au membre buffer_type le type de donnée que vous attendez, et que buffer doit pointer sur l'adresse de la structure MYSQL_TIME qui va recevoir les données retournées. Utilisez mysql_bind_results() pour lier les buffers à la commande après avoir appelé mysql_execute() et avant de lire les résultats.

Voici un exemple qui réalise l'insertion de données `DATE` , `TIME` et `TIMESTAMP` . La variable `mysql` est supposée représenter une connexion valide.

```
 MYSQL_TIME  ts;
MYSQL_BIND  bind[3];
MYSQL_STMT  *stmt;

  strmov(query, "INSERT INTO test_table(date_field, time_field,
                                      timestamp_field) VALUES(?,?,?");

   stmt= mysql_prepare(mysql, query, strlen(query)));

   /* configure les trois buffers pour les trois paramètres */
  bind[0].buffer_type= MYSQL_TYPE_DATE;
  bind[0].buffer= (char *)&ts;
  bind[0].is_null= 0;
  bind[0].length= 0;
  ..
  bind[1]= bind[2]= bind[0];
  ..

   mysql_bind_param(stmt, bind);

   /* Fournit les données à envoyer dans la structure ts */
  ts.year= 2002;
  ts.month= 02;
  ts.day= 03;

   ts.hour= 10;
  ts.minute= 45;
  ts.second= 20;

   mysql_execute(stmt);
  ..
```

21.2.10 Description des fonctions threadées de C

Vous devez utiliser les fonctions suivantes quand vous voulez créer un client threadé. Comment programmer une client threadé .

21.2.10.1 my_init()

```
void my_init(void)
```

Description Cette fonction doit être appelée une fois dans le programme avant tout appel à une fonction MySQL. Cela initialise quelques variables globales dont MySQL a besoin. Si vous utilisez une bibliothèque client sûr pour les threads, cela appellera aussi `mysql_thread_init()` pour ce thread.

Ceci est automatiquement appelé par mysql_init() , mysql_server_init() et mysql_connect() .

Valeur de retour Aucune.

21.2.10.2 mysql_thread_init()

```
my_bool mysql_thread_init(void)
```

Description Cette fonction doit être appelée à chaque création de thread pour initialiser les variables spécifiques aux threads.

Elle est appelée automatiquement par `my_init()` et `mysql_connect()` .

Valeur de retour Aucune.

21.2.10.3 mysql_thread_end()

```
void mysql_thread_end(void)
```

Description Cette fonction doit être appelée avant pthread_exit() pour libérer la mémoire allouée part mysql_thread_init() .

Notez que cette fonction **n'est pas invoquée automatiquement** par la bibliothèque du client. Elle doit être invoquée explicitement pour éviter les pertes de mémoire.

Valeur de retour Aucune.

21.2.10.4 mysql_thread_safe()

```
unsigned int mysql_thread_safe(void)
```

Description Cette fonction indique si le client est compilé avec le support des threads (thread−safe).

Valeur de retour 1 indique que le client est thread−safe, 0 sinon.

21.2.11 Description des fonctions C du serveur embarqué

Vous devez utiliser les fonctions suivantes si vous voulez permettre à votre application d'être liée avec la bibliothèque du serveur embarqué MySQL. libmysqld, la bibliothèque intégrée MySQL .

Si le programme est lié avec `-lmysqlclient` au lieu de `-lmysqld` , ces fonctions ne font rien. Cela permet de choisir d'utiliser un serveur embarqué MySQL, ou un serveur tournant à part sans avoir à changer votre code.

21.2.11.1 mysql_server_init()

`int mysql_server_init(int argc, char **argv, char **groups)`

Description Cette fonction **doit** être appelée une fois dans le programme avant d'appeler toute autre fonction MySQL. Elle démarre le serveur et initialise tout sous-système (mysys , InnoDB , etc.) utilisé par le serveur. Si cette fonction n'est pas appelée, le programme plantera. Si vous utilisez le paquet DBUG fournit avec MySQL, vous devez exécuter cette fonction après avoir appelé MY_INIT() .

Les arguments `argc` et `argv` sont analogues aux arguments de `main()` . Le premier élément `argv` est ignoré (il contient le plus souvent le nom du programme). Par convenance, `argc` peut être `0` (zéro) si il n'y a aucun argument passé en ligne de commande pour le serveur.

La liste de mots terminée par NULL dans groups détermine les groupes dans les fichiers d'options qui seront actifs. Utiliser les fichiers d'options . Par convenance, groups peut être NULL , dans ce cas, les groupes [server] et [emedded] sont activés. **Exemple**

```
 #include <mysql.h>
#include <stdlib.h>

 static char *server_args[] = &#123;
  "ce_programme",       /* cette chaîne n'est pas utilisée */
  "--datadir=.",
  "--key_buffer_size=32M"
&#125;;
static char *server_groups[] = &#123;
  "embedded",
  "server",
  "this_program_SERVER",
  (char *)NULL
&#125;;

 int main(void) &#123;
  mysql_server_init(sizeof(server_args) / sizeof(char *),
                    server_args, server_groups);

   /* Utilisez les fonction de L'API MySQL ici */

   mysql_server_end();

   return EXIT_SUCCESS;
&#125;
```

Valeur de retour 0 en cas de succès, 1 si une erreur survient.

21.2.11.2 mysql_server_end()

`void mysql_server_end(void)`

Description Cette fonction **doit** être appelée une fois dans le programme après toutes les autres fonctions MySQL. Elle coupe le serveur incorporé.

Valeur de retour Aucune.

21.2.12 Questions courantes sur la bibliothèque C

21.2.12.1 Pourquoi est−ce que mysql_store_result() retourne parfois NULL après que mysql_query() ait réussi

Il est possible que `mysql_store_result()` retourne `NULL` après un appel à `mysql_query()` . Quand cela arrive, cela signifie que l'une des conditions suivantes a été remplie :

- Il y a eu un problème avec malloc() (par exemple, si la taille du résultat était trop importante).
- Les données n'ont pu être lues (erreur survenue à la connexion).
- La requête n'a retourné aucun résultat (par exemple, il s'agissait d'un INSERT , UPDATE , ou d'un DELETE).

Vous pouvez toujours vérifier si la requête devait bien fournir un résultat non vide en invoquant mysql_field_count() . Si mysql_field_count() retourne zéro, le résultat est vide et la dernière requête n'en retournait pas (par exemple, un INSERT ou un DELETE). Si mysql_field_count() retourne un résultat non nul, la requête aurait du produire un résultat non nul. Voyez la documentation de la fonction mysql_field_count() pour plus d'exemples.

Vous pouvez tester les erreurs en faisant appel à mysql_error() ou mysql_errno() .

21.2.12.2 Quels résultats puis−je obtenir d'une requête?

En plus des enregistrements retournés par une requête, vous pouvez obtenir les informations suivantes :

- mysql_affected_rows() retourne le nombre d'enregistrements affectés par la dernière requête INSERT , UPDATE , ou DELETE . Une exception est que si DELETE est utilisé sans clause WHERE , la table est re−créée vide, ce qui est plus rapide! Dans ce cas, mysql_affected_rows() retournera zéro comme nombre d'enregistrements affectés.
- mysql_num_rows() retourne le nombre d'enregistrements dans le résultat. Avec mysql_store_result() , mysql_num_rows() peut être utilisée dès que mysql_store_result() retourne un résultat. Avec mysql_use_result() , mysql_num_rows() ne doit être appelé qu'après avoir récupéré tous les enregistrements avec mysql_fetch_row() .
- mysql_insert_id() retourne l'ID généré par la dernière requête qui a inséré une ligne dans une table avec un index AUTO_INCREMENT . mysql_insert_id() .
- Quelques requêtes (LOAD DATA INFILE ... , INSERT INTO ... SELECT ... , UPDATE) retournent des informations additionnelles. Le résultat est renvoyé par mysql_info() . Regardez la documentation de mysql_info() pour plus d'informations sur le format de la chaîne retournée. mysql_info() retourne un pointeur NULL s'il n'y a pas d'informations additionelles.

21.2.12.3 Comment lire l'identifiant unique de la dernière ligne insérée

Si vous insérez une ligne dans une table qui contient une colonne ayant l'attribut AUTO_INCREMENT , vous pouvez obtenir le dernier identifiant généré en appelant la fonction mysql_insert_id() .

Vous pouvez aussi récupérer cet identifiant en utilisant la fonction LAST_INSERT_ID() dans une requête que vous passez à mysql_query() .

Vous pouvez vérifier qu'un index AUTO_INCREMENT est utilisé en exécutant le code suivant. Cela vérifiera aussi si la requête était un INSERT avec un index AUTO_INCREMENT :

```
 if (mysql_error(&mysql)[0] == 0 &&
    mysql_num_fields(result) == 0 &&
    mysql_insert_id(&mysql) != 0)
&#123;
    used_id = mysql_insert_id(&mysql);
&#125;
```

Pour plus d'informations, voyez la section mysql_insert_id() .Lorsqu'une nouvelle valeur AUTO_INCREMENT est générée, vous pouvez l'obtenir en utilisant la commande SELECT LAST_INSERT_ID() avec mysql_query() et en lisant la valeur dans le résultat obtenu.

Pour LAST_INSERT_ID() , l'identifiant généré par la dernière insertion est entretenu sur le serveur en se basant sur la connexion. Il ne sera pas changé par un autre client. Il ne changera pas non plus si vous mettez à jour une autre colonne AUTO_INCREMENT avec une valeur normale (ni NULL ni 0).

Si vous voulez utiliser l'identifiant généré pour une table et l'insérer dans une autre, vous pouvez utiliser les requêtes suivantes :

```
 INSERT INTO foo (auto,text)
    VALUES(NULL,'text');              # génère un identifiant en insérant NULL
INSERT INTO foo2 (id,text)
    VALUES(LAST_INSERT_ID(),'text');  # on l'utilise dans la seconde page
```

Notez que mysql_insert_id() retourne la valeur stockée dans une colonne AUTO_INCREMENT , que cette valeur ait été générée automatiquement en enregistrant NULL ou 0 ou une valeur explicite. LAST_INSERT_ID() retourne les valeurs générées automatiquement par AUTO_INCREMENT . Si vous stockez une valeur explicite, autre que NULL ou 0 , cela n'affecte pas le résultat de LAST_INSERT_ID() .

21.2.12.4 Problèmes lors de la liaison avec l'API C

Lors de la liaison avec l'API C, l'erreur suivante peut survenir sur quelques systèmes :

```
 gcc −g −o client test.o −L/usr/local/lib/mysql −lmysqlclient −lsocket −lnsl

 Undefined        first referenced
 symbol          in file
floor            /usr/local/lib/mysql/libmysqlclient.a(password.o)
```

```
ld: fatal: Symbol referencing errors. No output written to client
```

Si cela se produit sur votre système, vous devez inclure la bibliothèque mathématique en ajoutant -lm à la fin de la ligne de compilation/liaison.

21.2.13 Compiler les clients

Si vous compilez des clients MySQL que vous avez écrits vous−même, ils doivent être liés en utilisant l'option `-lmysqlclient -lz` de la commande de liaison. Vous aurez peut−être besoin de spécifier l'option `-L` pour dire au programme ou trouver les bibliothèques. Par exemple, si la bibliothèque est installée dans `/usr/local/mysql/lib` , utilisez `-L/usr/local/mysql/lib -lmysqlclient -lz` dans votre commande.

Pour les clients qui utilisent les fichiers d'entêtes de MySQL, vous aurez besoin de spécifier une option -I lors de leur compilation (par exemple, -I/usr/local/mysql/include), pour que le programme puisse les trouver.

Pour rendre ce qui précède plus simple sur Unix, nous avons fourni le script `mysql_config` . `mysql_config` , lit les options de compilations pour les clients MySQL .

Vous pouvez l'utiliser pour compiler un client MySQL comme ceci :

```
 CFG=/usr/local/mysql/bin/mysql_config
sh -c "gcc -o progname `$CFG --cflags` progname.c `$CFG --libs`"
```

sh -c est nécessaire pour s'assurer que le Shell ne traitera pas le résultat de mysql_config comme un seul mot.

21.2.14 Comment faire un client MySQL threadé

La bibliothèque cliente est presque compatible avec les threads. Le problème le plus important est les routines de `net.c` qui lisent les sockets, et qui ne sont pas compatibles avec les interruptions. Cela a été fait en imaginant que vous souhaitiez vos propres alarmes, qui pourraient interrompre une lecture trop longue. Si vous installez des gestionnaires d'interruption pour l'alarme `SIGPIPE` , la gestion des sockets devraient être compatible avec les threads.

Dans les anciennes versions binaires que nous distribuions sur notre site web, (http://www.mysql.com/), les bibliothèques clientes étaient normalement compilées avec l'option de compatibilité avec les threads (les exécutables Windows sont par défaut compatible avec les threads). Les nouvelles distributions binaires doivent disposer des deux bibliothèques, compatibles ou non avec les threads.

Pour obtenir une client threadé où vous pouvez interrompre le client avec d'autres threads, mettre des délais d'expiration lors des discussions avec le serveur MySQL, vous devriez utiliser les bibliothèques `-lmysys` , `-lmystrings` et `-ldbug` , ainsi que `net_serv.o` que le serveur utilise.

Si vous n'avez pas besoin des interruption ou des expirations, vous pouvez compiler simplement une bibliothèque compatible avec les threads, (mysqlclient_r) et l'utiliser. Interface C . Dans ce cas, vous n'avez pas à vous préoccuper du fichier net_serv.o ou des autres bibliothèques MySQL.

Lorsque vous utiliser un client threadé et que vous souhaitez utiliser des délais d'expiration et des interruptions, vous pouvez faire grand usage des routines du fichier `thr_alarm.c` . Si vous utiliser des routines issues de la bibliothèque `mysys` , la seule chose à penser est de commencer par utiliser `my_init()` ! Description des fonctions thréadées en C .

Toutes les fonctions, hormis mysql_real_connect() sont compatibles avec les threads par défaut. Les notes suivantes décrivent comment compiler une bibliothèque cliente compatible avec les threads. Les notes ci-dessous, écrites pour mysql_real_connect() s'appliquent aussi à mysql_connect() , mais comme mysql_connect() est obsolète, vous devriez utiliser mysql_real_connect()).

Pour rendre `mysql_real_connect()` compatible avec les threads, vous devez recompiler la bibliothèque cliente avec cette commande :

```
 shell> ./configure --enable-thread-safe-client
```

Cela va créer une bibliothèque cliente compatible avec les threads libmysqlclient_r . Supposons que votre système d'exploitation dispose d'une fonction gethostbyname_r() compatible avec les threads. Cette bibliothèque est compatible avec les threads pour chaque connexion. Vous pouvez partager une connexion entre deux threads, avec les limitations suivantes :

- Deux threads ne peuvent pas envoyer de requêtes simultanées au serveur MySQL, sur la même connexion. En particulier, vous

devez vous assurer qu'entre mysql_query() et mysql_store_result() , aucun autre thread n'utilise la même connexion.
- De nombreux threads peuvent accéder à différents résultats qui sont lus avec mysql_store_result() .
- Si vous utilisez mysql_use_result , vous devez vous assurer qu'aucun autre thread n'utilise la même connexion jusqu'à ce qu'elle soit refermée. Cependant, il vaut bien mieux pour votre client threadé qu'ils utilisent mysql_store_result() .
- Si vous voulez utiliser de multiples threads sur la même connexion, vous devez avoir un verrou mutex autour de vos fonctions mysql_query() et mysql_store_result() . Une fois que mysql_store_result() est prêt, le verrou peut être libéré et d'autres threads vont pouvoir utiliser la connexion.
- Si vous programmez avec les threads POSIX, vous pouvez utiliser les fonctions pthread_mutex_lock() et pthread_mutex_unlock() pour poser et enlever le verrou mutex.

Vous devez savoir ce qui suit si vous avez un thread qui appel une fonction MySQL qui n,a pas crée de connexion à la base MySQL :

Lorsque vous appelez mysql_init() ou mysql_connect() , MySQL va créer une variable spécifique au thread qui est utilisée par la libaririe de débogage (entre autres).

Si vous appelez une fonction MYSQL, avant que le thread n'ai appelé mysql_init() ou mysql_connect() , le thread ne va pas avoir les variables spécifiques en place, et vous risquez d'obtenir un core dump tôt ou tard.

Pour faire fonctionner le tout proprement, vous devez suivre ces étapes :

- Appeler my_init() au début du programme, si il appelle une autre fonction MySQL, avant d'appeler mysql_real_connect() .
- Appeler mysql_thread_init() dans le gestionnaire de threads avant d'appeler une autre fonction MySQL.
- Dans le thread, appelez mysql_thread_end() avant d'appeler pthread_exit() . Cela va libérer la mémoire utiliser par les variables spécifiques MySQL.

Vous pouvez rencontrer des erreurs à cause des symboles non définis lors du link de votre client avec libmysqlclient_r . Dans la plupart des cas, c'est parce que vous n'avez pas inclus la bibliothèque de threads dans la ligne de compilation.

21.2.15 libmysqld , la bibliothèque du serveur embarqué MySQL

21.2.15.1 Vue d'ensemble de la bibliothèque du serveur embarqué MySQL

La bibliothèque embarquée MySQL rend possible l'accès à un serveur MySQL complet, depuis une application. Le principal avantage est l'amélioration des performances, et une gestion bien plus simple des applications.

Les API sont identiques pour la version embarquée et la version client/serveur. Pour changer les anciennes applications threadées, et les faire utiliser la bibliothèque embarquée, vous devez simplement ajouter deux appels aux fonctions suivantes :

Fonction	Quand l'utiliser
mysql_server_init()	Doit être appelée avant toute autre fonction MySQL, et de préférence très tôt dans la fonction main() .
mysql_server_end()	Doit être appelée avant que votre programme ne se termine.
mysql_thread_init()	Doit être appelée dans chaque thread que vous créez, qui aura accès à MySQL.
mysql_thread_end()	Doit être appelée avant d'appeler pthread_exit()

Puis, vous devez compiler votre code avec libmysqld.a au lieu de libmysqlclient.a .

Les fonctions ci−dessus mysql_server_xxx sont aussi inclues dans la bibliothèque libmysqlclient.a pour vous permettre de changer facilement entre les versions de la bibliothèque embarquée et celle de la bibliothèque client/serveur, en compilant simplement la bonne bibliothèque. mysql_server_init() .

21.2.15.2 Compiler des programmes avec libmysqld

Pour avoir la bibliothèque libmysqld vous devez configurer MySQL avec l'option --with-embedded-server .

Quand vous liez votre programme avec libmysqld , vous devez aussi inclure les bibliothèques pthread spécifiques au système et quelques bibliothèques que le serveur MySQL utilise. Vous pouvez obtenir la liste complète des bibliothèques en exécutant mysql_config --libmysqld-libs .

Les options correctes pour compiler et lier un programme threadé doivent être utilisées, même si vous n'appelez pas directement une fonction de threads dans votre code.

21.2.15.3 Restrictions lors de l'utilisation du serveur embarqué MySQL

Le serveur embarqué possède les limitations suivantes :

- Pas de support pour les tables ISAM. (Ceci est principalement fait pour rendre la bibliothèque plus petite)
- Pas de fonctions définies par l'utilisateur (UDF).
- Pas de traçage de pile lors des vidages de mémoire (core dump).
- Pas de support pour les RAID internes. (Cela n'est normalement pas requis vu que la plupart des systèmes d'exploitation supportent aujourd'hui les grands fichiers).
- Vous pouvez le configurer en tant que serveur ou maître (pas de réplication).
- vous ne pouvez vous connecter au serveur embarqué à partir d'un processus externe avec les sockets ou TCP/IP.

Quelques unes de ces limitations peuvent être changée en éditant le fichier mysql_embed.h et recompilant MySQL.

21.2.15.4 Utilisation de fichiers d'options avec le serveur embarqué

Voici la manière recommandée d'utiliser les fichiers d'options pour que le passage des applications client/serveur vers une application où MySQL est embarqué soit plus facile. Utiliser les fichiers d'options .

- Placez les options communes dans la section [server] . Elles seront lues par les deux versions de MySQL.
- Placez les options spécifiques au client/serveur dans la section [mysqld] .
- Placez les options spécifiques à MySQL embarqué dans la section [embedded] .
- Placez les options spécifiques aux applications dans une section [NomApplication_SERVER] .

21.2.15.5 Choses à faire pour le serveur embarqué (TODO)

- Nous allons proposer des options pour se débarrasser de quelques parties de MySQL pour rendre la bibliothèque plus petite.
- Il y a encore beaucoup d'optimisations de vitesse à faire.
- Les erreurs sont écrites dans stderr. Nous ajouterons une options pour spécifier un fichier pour cela.
- Nous devons changer InnoDB pour qu'il n'y ait plus tant de sorties lors de l'utilisation du serveur embarqué.

21.2.15.6 Un exemple simple de serveur embarqué

Ce programme exemple et son makefile devraient fonctionner sans changements sur un système Linux ou FreeBSD. Pour les autres systèmes d'exploitation, quelques petits changements seront requis. Cet exemple est là pour vous donner assez de détails pour comprendre le problème, sans avoir en tête qu'il s'agit d'une partie nécessaire d'une application réelle.

Pour essayer ccet exemple, créez un dossier `test_libmysqld` au même niveau que le dossier des sources mysql−4.0. Sauvegardez le fichier source `test_libmysqld.c` et `GNUmakefile` dans le dossier, puis exécutez GNU `make` à l'intérieur du répertoire `test_libmysqld` .

test_libmysqld.c

```
 /*
 * Un simple exemple de client, utilisant la bibliothèque du serveur embarqué MySQL
 */

 #include <mysql.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

 MYSQL *db_connect(const char *dbname);
void db_disconnect(MYSQL *db);
void db_do_query(MYSQL *db, const char *query);

 const char *server_groups[] = &#123;
  "test_libmysqld_SERVER", "embedded", "server", NULL
&#125;;

 int
main(int argc, char **argv)
&#123;
  MYSQL *one, *two;

   /* mysql_server_init() doit être appelée avant toute autre fonction
   * mysql.
   *
   * Vous pouvez utiliser mysql_server_init(0, NULL, NULL), et cela initialisera
   * le serveur en utilisant groups = &#123;
```

```
   *   "server", "embedded", NULL
   *  &#125;.
   *
   * Dans votre fichier $HOME/.my.cnf, vous voudrez sûrement mettre :

 [test_libmysqld_SERVER]
language = /chemin/vers/la/source/de/mysql/sql/share/english

    * Vous pouvez, bien sûr, modifier argc et argv avant de les passer
   * à cette fonction. Ou vous pouvez en créer de nouveaux de la manière
   * que vous souhaitez. Mais tout les arguments dans argv (à part
   * argv[0], qui est le nom du programme) doivent être des options valides
   * pour le serveur MySQL.
   *
   * Si vous liez ce client avec la bibliothèque mysqlclient normale,
   * cette fonction n'est qu'un bout de code qui ne fait rien.
   */
  mysql_server_init(argc, argv, (char **)server_groups);

   one = db_connect("test");
  two = db_connect(NULL);

   db_do_query(one, "SHOW TABLE STATUS");
  db_do_query(two, "SHOW DATABASES");

   mysql_close(two);
  mysql_close(one);

   /* Cela doit être appelé après toutes les autres fonctions mysql */
  mysql_server_end();

   exit(EXIT_SUCCESS);
&#125;

 static void
die(MYSQL *db, char *fmt, ...)
&#123;
  va_list ap;
  va_start(ap, fmt);
  vfprintf(stderr, fmt, ap);
  va_end(ap);
  (void)putc('\n', stderr);
  if (db)
    db_disconnect(db);
  exit(EXIT_FAILURE);
&#125;

 MYSQL *
db_connect(const char *dbname)
&#123;
  MYSQL *db = mysql_init(NULL);
  if (!db)
    die(db, "mysql_init a échoué : pas de mémoire");
  /*
   * Notez que le client et le serveur utilisent des noms de groupes séparés.
   * Ceci est critique, car le serveur n'acceptera pas les options du client
   * et vice versa.
   */
  mysql_options(db, MYSQL_READ_DEFAULT_GROUP, "test_libmysqld_CLIENT");
  if (!mysql_real_connect(db, NULL, NULL, NULL, dbname, 0, NULL, 0))
    die(db, "mysql_real_connect a échoué : %s", mysql_error(db));

   return db;
&#125;

 void
db_disconnect(MYSQL *db)
&#123;
  mysql_close(db);
&#125;

 void
db_do_query(MYSQL *db, const char *query)
&#123;
  if (mysql_query(db, query) != 0)
    goto err;

   if (mysql_field_count(db) > 0)
  &#123;
    MYSQL_RES   *res;
    MYSQL_ROW    row, end_row;
    int num_fields;

     if (!(res = mysql_store_result(db)))
      goto err;
    num_fields = mysql_num_fields(res);
    while ((row = mysql_fetch_row(res)))
    &#123;
      (void)fputs(">> ", stdout);
      for (end_row = row + num_fields; row < end_row; ++row)
        (void)printf("%s\t", row ? (char*)*row : "NULL");
      (void)fputc('\n', stdout);
```

```
    &#125;
    (void)fputc('\n', stdout);
  &#125;
  else
    (void)printf("Lignes affectées : %lld\n", mysql_affected_rows(db));

   return;

 err:
  die(db, "db_do_query a échoué : %s [%s]", mysql_error(db), query);
&#125;
```

GNUmakefile

```
 # On suppose que MySQL est installé dans /usr/local/mysql
inc      := /usr/local/mysql/include/mysql
lib      := /usr/local/mysql/lib

 # Si vous n'avez pas encore installé MySQL, essayez plutôt ceci
#inc      := $(HOME)/mysql-4.0/include
#lib      := $(HOME)/mysql-4.0/libmysqld

 CC       := gcc
CPPFLAGS := -I$(inc) -D_THREAD_SAFE -D_REENTRANT
CFLAGS   := -g -W -Wall
LDFLAGS  := -static
# Vous pouvez changer -lmysqld en -lmysqlclient pour utiliser
# la bibliothèque client/serveur
LDLIBS    = -L$(lib) -lmysqld -lz -lm -lcrypt

 ifneq (,$(shell grep FreeBSD /COPYRIGHT 2>/dev/null))
# FreeBSD
LDFLAGS += -pthread
else
# Linux
LDLIBS += -lpthread
endif

 # Cela fonctionne pour les programes de tests sur un simple fichier
sources := $(wildcard *.c)
objects := $(patsubst %c,%o,$(sources))
targets := $(basename $(sources))

 all: $(targets)

 clean:
    rm -f $(targets) $(objects) *.core
```

21.2.15.7 Licence du serveur embarqué

Le code source de MySQL est couvert par la licence GNU GPL (Licence GNU). L'un des effets de cela est que tout programme qui inclut, par liaison avec `libmysqld` , le code source de MySQL doit être réalisé en tant que logiciel libre (sous une licence compatible avec la GPL).

Nous encourageons tout le monde à promouvoir le logiciel libre en réalisant du code sous la licence GPL ou une autre licence compatible. Pour ceux qui ne peuvent le faire, une autre option est d'acheter la licence commerciale pour le code `MySQL` chez `MySQL AB` . Pour plus de détails, voyez Licences MySQL .

21.3 Support ODBC avec MySQL

MySQL fournit un support de ODBC via le programme `MyODBC` . Ce chapitre vous apprendra à installer `MyODBC` , et à l'utiliser. Vous trouverez aussi une liste de programmes connus pour leur utilisation de `MyODBC` .

21.3.1 Comment installer MyODBC

`MyODBC` 2.50 est un pilote 32−bit ODBC 2.50 avec un niveau de spécification 0 (avec le niveau 1 et 2 de proposés) pour connecter une application compatible ODBC à MySQL. `MyODBC` fonctionne sur Windows 9x/Me/NT/2000/XP et la plupart des plate−formes Unix. `MyODBC` 3.51 est une version améliorée avec les spécifications de niveau 1 de ODBC 3.5x (API noyau complète + fonctionnalités du niveau 2).

MyODBC est Open Source , et vous pouvez trouver la version la plus récente sur http://www.mysql.com/downloads/api−myodbc.html . Notez que les version 2.50.x sont licencées LGPL tandis que les versions 3.51.x sont licencées GPL .

Si vous avez des problèmes avec MyODBC et que votre programme fonctionne aussi avec OLEDB, essayez le pilote OLEDB.

Normalement, vous n'avez besoin d'installer MyODBC que sur les machines Windows. Vous avez besoin d'installer MyODBC sous Unix si vous avez un programme tel que ColdFusion qui fonctionne sur les machines Unix et utilise ODBC pour se connecter aux bases de données.

Si vous voulez installer MyODBC sur un ordinateur Unix, vous aurez aussi besoin d'un gestionnaire ODBC . MyODBC est connu pour fonctionner avec la plupart des gestionnaires ODBC d'Unix.

Pour installer MyODBC sur Windows, vous devez télécharger le fichier .zip de MyODBC approprié, le décompresser avec WinZIP ou un programme similaire et exécuter le fichier SETUP.EXE .

Sur Windows/NT/XP vous pouvez obtenir l'erreur suivante durant l'installation de MyODBC :

```
 An error occurred while copying C:\WINDOWS\SYSTEM\MFC30.DLL. Restart
Windows and try installing again (before running any applications which
use ODBC)
```

Le problème dans ce cas est qu'un autre programme utilise ODBC et du fait de l'architecture Windows, vous ne pouvez pas installer de nouveau pilote ODBC avec le programme d'installation de Microsoft ODBC. Dans la plupart des cas, vous pouvez continuer en cliquant juste sur Ignore pour copier le reste des fichiers MyODBC et l'installation finale devrait fonctionner. Si ce n'est pas le cas, la solution est de redémarrer votre machine en mode ``safe mode`` (faites le en appuyant sur F8 juste avant que votre machine ne démarre Windows), installez MyODBC , et redémarrez en mode normal.

- Pour créer une connexion à un ordinateur Unix depuis un ordinateur Windows, avec une application ODBC (une qui ne supporte pas MySQL nativement), vous devez installer MyODBC sur l'ordinateur Windows.
- L'utilisateur et la machine Windows doivent avoir les droits d'accès au serveur MySQL situé sur la machine Unix. vous pouvez configurer cela avec la commande GRANT . Syntaxe de GRANT et REVOKE .
- Vous devez créer une entrée DNS ODBC comme suit :
    - Ouvrez le panneau de configuration de Windows.
    - Double−cliquez sur l'icône Sources de données ODBC (32 bits).
    - Cliquez sur le volet User DSN.
    - Cliquez sur le bouton Add.
    - Sélectionnez MySQL dans l'écran Create New Data Source et cliquez sur le bouton Terminer.
    - L'écran de configuration par défaut du pilote MySQL est affiché. Comment remplir les différents champs dans le programme d'administrateur ODBC .
- Démarrez maintenant votre application et sélectionnez le pilote ODBC avec les DSN que vous avez spécifié dans l'administrateur ODBC.

Notez que d'autres options de configuration sont présentes dans l'écran de MySQL (traçage, se connecter automatiquement, etc.), vous pouvez les essayer en cas de problèmes.

21.3.2 Comment remplir les différents champs dans le programme d'administrateur ODBC

Il y a trois possibilités pour indiquer le nom du serveur sous Windows95 :

- Utiliser l'adresse IP du serveur.
- Ajouter un fichier \windows\lmhosts avec les informations suivantes :
  ```
  ip nomhote
  ```
  Pour exemple :
  ```
  194.216.84.21 mon_nom_hote
  ```
- Configurer le PC pour qu'il utilise DNS.

Exemple de valeurs dans l' installation ODBC :

```
 Windows DSN name:   test
Description:        Ceci est ma base de données
MySql Database:     test
Server:             194.216.84.21
User:               monty
Password:           mon_pass
Port:
```

La valeur du champ Windows DSN name est n'importe quel nom qui est unique pour votre installation de ODBC sur Windows. Vous n'avez pas besoin de spécifier des valeurs pour les champs Server , User , Password , et Port dans l'écran d'installation d'ODBC. Toutefois, si vous le faites, ces valeurs seront celles utilisées par défaut par la suite lorsque vous tenterez d'établir une connexion. Vous pourrez changez les options à ce moment là.

Si le numéro du port n'est pas donné, le port par défaut (3306) est utilisé.

Si vous spécifiez l'option Read options from C:\my.cnf , les groupes client et odbc seront lus à partir du fichier C:\my.cnf . Vous pouvez utiliser toutes les options utilisables par mysql_options() . mysql_options() .

21.3.3 Paramètres de connexion de MyODBC

Il est possible de spécifier les paramètres suivants à `MyODBC` dans la section `[Servername]` du fichier `ODBC.INI` ou bien, avec l'argument `InConnectionString` dans l'appel de `SQLDriverConnect()` .

Paramètre	Valeur par défaut	Commentaire
user	ODBC (sous Windows)	Le nom d'utilisateur utilisé pour se connecter à MySQL.
server	localhost	Le nom d'hôte du serveur MySQL.
database		La base de données par défaut.
option	0	Un entier avec lequel vous spécifier le mode de fonctionnement de MyODBC . Voir ci−dessous.
port	3306	Le port TCP/IP à utiliser si le server n'est pas localhost .
stmt		Une commande qui sera exécutée au moment de la connexion à MySQL .
password		Le mot de passe pour le serveur server , et l'utilisateur user .
socket		La socket ou le pipe Windows à utiliser.

L'argument option est utilisé pour indiquer à `MyODBC` que le client n'est pas 100% compatible avec ODBC. Sous Windows, il est possible de configurer cette option en activant les options dans l'écran de connexion, mais il est aussi possible de le configurer dans ce paramètre. Les options suivantes sont listées dans l'ordre dans lequel elles apparaissent dans l'écran `MyODBC` :

Bit	Description
1	Le client ne peut gérer le fait que MyODBC retourne la taille réelle de la colonne.
2	Le client ne peut gérer que MySQL retourne le nombre de ligne affecté. Si cette option est activée, alors MySQL retourne ' found rows ' à la place. Il faut avoir MySQL 3.21.14 ou plus récent pour profiter de cette fonctionnalité.
4	Ecrit un fichier de débogage dans c:\myodbc.log. Cela revient au même que le code MYSQL_DEBUG=d:t:O,c::\myodbc.log dans le fichier AUTOEXEC.BAT
8	Ne limite pas les paquets pour les résultats et les paramètres.
16	Ne pas poser de questions, même si le pilote le veut.
32	Simule un pilote ODBC 1.0 dans certains contexte.
64	Ignore l'utilisation du nom de base de données dans la syntaxe ' base.table.colonne '.
128	Force l'utilisation du gestionnaire de curseur ODBC (expérimental).
256	Désactive l'utilisation de la lecture étendue (expérimental).
512	Complète les champs CHAR jusqu'à contenance.
1024	SQLDescribeCol() retourne les noms complets de colonne.
2048	Utilise le protocole de communication client/serveur
4096	Indique au serveur qu'il doit ignorer les espaces après les noms de fonction, et avant le '(' (nécessaire pour PowerBuilder). Cela va faire de tous les noms de fonctions des mots clés.
8192	Connexion, avec les pipes nommés, à mysqld sur un machine NT.
16384	Change les colonnes LONGLONG en colonne INT (certaines applications ne peuvent pas gérer les LONGLONG).
32768	Retourne 'user' comme Table_qualifier et Table_owner dans les tables SQLTables (expérimental)
65536	Lit les paramètres des groupes client et odbc dans my.cnf
131072	Ajoute des vérifications de sécurité (ne devrait pas être nécessaire, mais...)

Si vous voulez combiner des options, vous devez additionner les options ci−dessus. Par exemple, activer l'option 12 (4+8), active le déboguage sans limite de paquets.

La bibliothèque par défaut MYODBC.DLL est compilée pour des performances optimales. Si vous voulez déboguer MyODBC (par exemple, pour activer le traçage), vous devriez utiliser MYODBCD.DLL . Pour installer ce ficher, copiez MYODBCD.DLL à la place de la bibliothèque installé MYODBC.DLL .

21.3.4 Comment reporter les problèmes avec ODBC

`MyODBC` a été testé avec Access, Admndemo.exe, C++−Builder, Borland Builder 4, Centura Team Developer (anciennement Gupta SQL/Windows), ColdFusion (on Solaris et NT avec pack de service 5), Crystal Reports, DataJunction, Delphi, ERwin, Excel, iHTML, FileMaker Pro, FoxPro, Notes 4.5/4.6, SBSS, Perl DBD−ODBC, Paradox, Powerbuilder, Powerdesigner 32 bit, VC++, et Visual Basic.

Si vous avez connaissance de n'importe quelle autre application qui utilise MyODBC , merci d'envoyer un mail à sur les listes de diffusion! Les listes de diffusion MySQL .

Avec quelques programmes, vous pouvez avoir l'erreur suivante : `Another user has modifies the record that you have modified` . Dans la plupart des cas, le problème peut être résolu en suivant l'une des méthodes suivantes :

- ajouter une clef primaire à la table s'il n'en existe pas déjà une.
- Ajouter une colonne timestamp s'il n'y en a pas déjà une.
- N'utilisez que des champs réels doubles. Certains programmes peuvent se bloquer lorsqu'ils comparent des réels simples.

Si ce qui précède ne vous aide pas, vous devez tracer `MyODBC` et essayer de voir pourquoi les choses vont de travers.

21.3.5 Programmes qui fonctionnent avec MyODBC

La majorité des programmes devraient fonctionner avec `MyODBC` , mais tous ceux que nous avons listés ici sont ceux que nous avons testé nous même, ou dont nous avons reçu la confirmation de la part d'utilisateurs :

Programme
Commentaire

Access

Pour le faire fonctionner avec Access :

Si vous utilisez Access 2000, vous devez installer la dernière Microsoft MDAC (Microsoft Data Access Components) (version 2.6 ou plus récente) depuis http://www.microsoft.com/data/ . Cela va corriger le bug suivant d'Access : lorsque vous exportez des données vers MySQL, les noms de la table et des colonnes ne sont pas spécifiés. Un autre moyen de pallier à ce bug est de passer en MyODBC version 2.50.33 et MySQL version 3.23.x, qui, ensemble, forment un correctif à ce bug!

Vous devriez aussi installer le Microsoft Jet 4.0 Service Pack 5 (SP5) disponible à http://support.microsoft.com/support/kb/articles/Q239/1/14.ASP . Cela va corriger certains cas ou les colonnes étaient marquées comme effacées (`#deleted#`) dans Access.

Notez que si vous utilisez MySQL version 3.22, vous devez appliquer le patch MDAC et utiliser MyODBC 2.50.32 ou 2.50.34 et plus récent pour corriger ce bug.

Pour toutes les versions d'Access, vous devez activer l'option MyODBC appelée Return matching rows . Pour Access 2.0, vous devez aussi activer Simulate ODBC 1.0 .

Vous devez avoir un timestamp dans toutes les tables que vous voulez pouvoir modifier. Pour une meilleure portabilité, TIMESTAMP(14) ou un simple TIMESTAMP est recommandé au lieu de TIMESTAMP(X) .

Vous devez avoir une clé primaire dans la table. Sinon, les nouvelles lignes et les lignes modifiées peuvent être marquées comme effacées #DELETED# .

N'utilisez que les champs DOUBLE . Access échoue lors des comparaisons entre les champs en simple précision. Le symptôme est généralement que les nouvelles lignes ou les lignes modifiées sont marquées comme effacées (#DELETED#), ou que vous ne pouvez plus trouver les nouvelles lignes ou les lignes modifiées.

Si vous reliez des tables via MyODBC, dont l'une des colonnes est de type BIGINT , les résultats sont affichés comme effacés (#DELETED). La solution est :

Avoir au moins une colonne de type TIMESTAMP comme type de données, et de préférence TIMESTAMP(14) .

Vérifiez l'option de connexion 'Change BIGINT columns to INT' dans l'administrateur ODBC DSN.

Effacez le lien de table dans Access, et recréez le.

Les anciennes lignes seront toujours marquées comme #DELETED# , mais les nouvelles lignes seront affichées correctement.

Si vous butez toujours dans l'erreur Another user has changed your data après avoir ajouté une colonne TIMESTAMP , le truc suivant peut vous aider : N'utilisez pas la vue de table au format tableur. A la place, créez un formulaire avec les champs que vous souhaitez, et utilisez ce formulaire comme vue de la table. Vous devez utiliser la propriété DefaultValue pour la colonne TIMESTAMP , avec la valeur NOW() . C'est une bonne idée de cacher la colonne TIMESTAMP à vos utilisateurs, pour qu'ils ne soient pas perturbés.

Dans certains cas, Access peut générer des requêtes SQL illégales, que MySQL ne peut comprendre. Vous pouvez corriger cela en sélectionnant "Query|SQLSpecific|Pass-Through" dans le menu Access.

Access sur NT va indiquer que les colonnes BLOB sont des OLE OBJECTS . Si vous voulez avoir des colonnes de type MEMO à la place, vous devez changer le type de la colonne en TEXT avec ALTER TABLE .

Access ne peut pas toujours gérer correctement les colonnes DATE . Sis vous avez des problèmes avec, changez ces colonnes en type DATETIME .

Si vous avec une colonne Access de type BYTE , Access va essayer de l'exporter sous la forme d'un TINYINT au lieu d'un TINYINT UNSIGNED . Cela va vous donner des problèmes si vous avez des valeurs supérieures à 127!

ADO

Lorsque vous programmez avec l'API ADO et MyODBC vous devez faire attention aux propriétés par défaut qui ne sont pas supportées par MySQL. Par exemple, utiliser la propriété CursorLocation Property comme adUseServer va retourner des valeurs de −1 pour RecordCount Property . Pour avoir la véritable valeur, vous devez utiliser la valeur adUseClient , comme dans le code VB ci-contre :

```
 Dim myconn As New ADODB.Connection
Dim myrs As New Recordset
Dim mySQL As String
Dim myrows As Long

 myconn.Open "DSN=MyODBCsample"
mySQL = "SELECT * from user"
myrs.Source = mySQL
Set myrs.ActiveConnection = myconn
```

```
myrs.CursorLocation = adUseClient
myrs.Open
myrows = myrs.RecordCount

 myrs.Close
myconn.Close
```

Un autre palliatif est d'utiliser la commande SELECT COUNT(*) , pour connaître le nombre exact de lignes.

Active server pages (ASP)
: Vous devez utiliser l'option Return matching rows .

Applications BDE
: Pour faire fonctionner ces applications, vous devez activer les options Don't optimize column widths et Return matching rows .

Borland Builder 4
: Lorsque vous démarrez une requête, vous pouvez utiliser la propriété Active ou utiliser la méthode Open . Notez que Active va commencer en générant automatiquement une requête SELECT * FROM ... qui peut ne pas être pratique avec de grosses tables!

ColdFusion (On Unix)
: Les informations suivantes sont reprises de la documentation ColdFusion :

Utilisez les informations suivantes pour configurer le serveur ColdFusion Server sous Linux pour utiliser le pilote unixODBC avec MyODBC pour les serveurs MySQL. Allaire a vérifié que MyODBC version 2.50.26 fonctionne avec MySQL version 3.22.27 et ColdFusion pour Linux. (toute nouvelle version doit aussi fonctionner). Vous pouvez télécharger MyODBC à http://www.mysql.com/downloads/api−myodbc.html

ColdFusion version 4.5.1 vous permet d'utiliser l'administrateur ColdFusion pour ajouter des serveurs MySQL. Toutefois, le pilote n'est pas inclus dans ColdFusion version 4.5.1. Auparavant, le pilote MySQL était disponible dans le menu déroulant. Vous devez compiler et copier le pilote MyODBC dans /opt/coldfusion/lib/libmyodbc.so .

Le dossier Contrib contient le programme mydsn−xxx.zip qui vous permet de compiler et supprimer des fichiers d'enregistrement DSN pour MyODBC sur les applications Coldfusion.

DataJunction
: Vous devez le modifier pour qu'il exporte des VARCHAR plutôt que des ENUM , car il exporte ces derniers d'une manière qui pose des problèmes à MySQL.

Excel
: Fonctionne. Quelques conseils :

  Si vous avez des problèmes avec les dates, essayez de les sélectionner comme des chaînes, en utilisant la fonction CONCAT() . Par exemple :

  ```
  select CONCAT(rise_time), CONCAT(set_time)
      from sunrise_sunset;
  ```

  Les valeurs lues sous forme de chaînes seront correctement relues par Excel97.

  Le but de la fonction CONCAT() est de faire croire à ODBC que cette colonne est de type ``string''. Sans la fonction CONCAT() , ODBC sait que la colonne est de type heure, et Excel ne le comprendra pas.

  Notez qu'il y a un bug dans Excel, car il convertit automatiquement une chaîne en heure. Cela serait bien si la source était un fichier texte, mais c'est totalement idiot si la source est une connexion ODBC qui fournit les types exacts.

Word
: Pour lire des données depuis MySQL vers des documents Word ou Excel, vous devez utiliser le pilote MyODBC et l'aide Add−in Microsoft Query.Par exemple, pour créer une base de données avec une table contenant 2 colonnes de type texte :

  Insérez des lignes avec le client mysql .
  Créez un fichier DSN en utilisant le gestionnaire ODBC, par exemple, my pour la base ci−dessus.
  Ouvrez Word.
  Créez un nouveau document vide.
  Utilisez la barre "Database", pressez sur le bouton d'insertion.
  Pressez sur le bouton "Get Data".
  Sur la droite de l'écran, choisissez le bouton "Ms Query".
  Dans ce écran, créez une nouvelle source de données avec "New Data Source" et en utilisant le fichier DSN my .
  Sélectionner une nouvelle requête.
  Sélectionnez les colonnes que vous souhaitez.
  Faîtes votre filtre.
  Faîtes votre tri.
  Sélectionnez "Return Data" de Microsoft Word.
  Cliquez sur "Finish".
  Cliquez sur le bouton "Insert data" et sélectionnez les lignes.
  Cliquez sur "OK" et vous verrez vous lignes dans votre document Word.

odbcadmin
: Programme de test pour ODBC.

Delphi
: Vous devez utiliser BDE version 3.2 ou plus récent. Utilisez l'option Don't optimize column width lors de la connexion à MySQL.

De plus, voici un code Delphi pratique, pour configurer une entrée ODBC et une entrée BDE pour MyODBC (l'entrée BDE requiert le BDE Alias Editor qui est gratuit sur les pages "Delphi Super Page" près de chez vous. (Merci à Bryan Brunton bryan@flesherfab.com pour cela) :

```
fReg:= TRegistry.Create;
 fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
 fReg.WriteString('Database', 'Documents');
 fReg.WriteString('Description', ' ');
 fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
 fReg.WriteString('Flag', '1');
 fReg.WriteString('Password', '');
 fReg.WriteString('Port', ' ');
 fReg.WriteString('Server', 'xmark');
 fReg.WriteString('User', 'winuser');
 fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
 fReg.WriteString('DocumentsFab', 'MySQL');
 fReg.CloseKey;
 fReg.Free;

  Memo1.Lines.Add('DATABASE NAME=');
 Memo1.Lines.Add('USER NAME=');
 Memo1.Lines.Add('ODBC DSN=DocumentsFab');
 Memo1.Lines.Add('OPEN MODE=READ/WRITE');
 Memo1.Lines.Add('BATCH COUNT=200');
 Memo1.Lines.Add('LANGDRIVER=');
 Memo1.Lines.Add('MAX ROWS=-1');
 Memo1.Lines.Add('SCHEMA CACHE DIR=');
 Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
 Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
 Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
 Memo1.Lines.Add('SQLQRYMODE=');
 Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
 Memo1.Lines.Add('ENABLE BCD=FALSE');
 Memo1.Lines.Add('ROWSET SIZE=20');
 Memo1.Lines.Add('BLOBS TO CACHE=64');
 Memo1.Lines.Add('BLOB SIZE=32');

  AliasEditor.Add('DocumentsFab','MySQL',Memo1.Lines);
```

C++ Builder
: Testé avec BDE version 3.0. Le seul problème connu est que lorsque la structure d'une table change, les champs de requêtes ne changent pas. BDE ne semble pas reconnaître les clés primaires, mais uniquement l'index PRIMARY, même si cela n'a jamais été un problème.

Vision
: Vous devriez utiliser l'option Return matching rows .

Visual Basic
: Pour être capable de modifier une table, vous devez définir une clé primaire.

Visual Basic avec ADO ne peut pas gérer les grands entiers. Cela signifie que les requêtes comme `SHOW PROCESSLIST` ne vont pas fonctionner correctement. Le correctif consiste à ajouter l'option `OPTION=16834` dans la chaîne de connexion ODBC pour d'utiliser l'option `Change BIGINT columns to INT` de MyODBC. Vous pouvez aussi utiliser l'option `Return matching rows` .

VisualInterDev
: Si vous rencontrez l'erreur [Microsoft][ODBC Driver Manager] Driver does not support this parameter , la raison est que vous avez un grand entier BIGINT dans votre résultat. Essayez d'utiliser l'option Change BIGINT columns to INT de MyODBC.

Visual Objects
: Vous devez utiliser l'option Don't optimize column widths .

21.3.6 Comment obtenir la valeur d'une colonne AUTO_INCREMENT avec ODBC

Un problème récurrent est d'obtenir la dernière valeur générée automatiquement par une commande `INSERT` . Avec ODBC, vous pouvez procéder de cette façon (en supposons que `auto` est un champ `AUTO_INCREMENT`):

```
 INSERT INTO foo (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

Ou, si vous voulez juste insérer cette valeur dans une autre table :

```
 INSERT INTO foo (auto,text) VALUES(NULL,'text');
INSERT INTO foo2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

Comment puis-je obtenir l'identifiant unique de la dernière ligne insérée? .Pour quelques applications utilisant ODBC (du moins Delphi et Access), la requête suivante peut être utilisée pour trouver une ligne insérée dernièrement :

```
 SELECT * FROM nom_de_table WHERE auto IS NULL;
```

21.3.7 Rapporter des problèmes avec MYODBC

Si vous rencontrez des difficultés avec MyODBC , commencez par faire un fichier de log avec le gestionnaire ODBC (le fichier de log que vous obtenez en demande les logs de ODBCADMIN) et un log MyODBC .

Pour obtenir un fichier de log MyODBC , vous devez faire ceci :

- Assurez vous que vous utilisez myodbcd.dll et non pas myodbc.dll . Le moyen le plus facile pour le faire est d'obtenir myodbcd.dll dans la distribution MYODBC et de le copier à la place de myodbc.dll , qui est probablement dans le dossier C:\windows\system32 ou C:\winnt\system32 .Notez que vous voudrez probablement récupérer votre vieux fichier myodbc.dll lorsque vous aurez fini de tester, car il est bien plus rapide que myodbcd.dll .
- Activez l'option ` Trace MyODBC ' dans l'écran de configuration de MyODBC . Le fichier de log sera écrit dans le fichier C:\myodbc.log .Si l'option de trace n'est pas recommandée lorsque vous retournez dans l'écran précédent, cela signifie que vous n'utilisez pas myodbcd.dll (voir ci-dessus).
- Démarrez votre application, et faîtes la planter.

Vérifiez le fichier de trace MyODBC , pour essayer de comprendre ce qui ne va pas. Vous devriez être capable de trouver les requêtes émises en recherchant la chaîne >mysql_real_query dans le fichier myodbc.log .

Vous devriez aussi essayer de dupliquer la requête dans le client mysql ou admndemo pour voir si le problème vient de MYODBC ou MySQL.

Si vous trouvez quelques chose d'incorrect, n'envoyez que les lignes pertinentes (maximum, 40 lignes) à myodbc@lists.mysql.com . N'envoyez jamais le fichier de log MYODBC ou ODBC complet!

Si vous êtes incapables de trouver une erreur, la dernière option est de faire une archive (tar ou zip) qui contienne le fichier de trace MYODBC , le fichier de log ODBC, et un fichier README qui contienne une description du problème. Vous pouvez envoyer le tout à ftp://support.mysql.com/pub/mysql/secret/ . Seuls nous, à MYSQL AB, pourront accéder à ces fichiers, et nous seront très respectueux de vos données.

Si vous pouvez créer un problème qui reproduit le problème, essayez de l'uploader aussi!

Si le programme fonctionne avec d'autres serveurs SQL, vous devriez faire un log ODBC où vous faîtes exactement la même chose sur les autres serveurs SQL.

N'oubliez jamais que plus vous nous fournissez d'explication, plus nous pourront vous aider!

21.4 Connexion Java/MySQL (JDBC)

Il y a 2 pilotes JDBC supportés pour MySQL :

- MySQL Connector/J de MySQL AB, implémenté 100% Java natif. Ce produit était connu sous le nom de pilote mm.mysql . Vous pouvez télécharger MySQL Connector/J depuis l'URL http://www.mysql.com/products/connector-j/ .
- Le pilote Resin JDBC, qui est disponible sur l'URL http://www.caucho.com/projects/jdbc-mysql/index.xtp .

Pour de la documentation, consultez celle de JDBC et des pilotes pour les fonctionnalités relatives à MySQL.

La documentation de MySQL Connector/J est disponible en ligne, sur le site de MySQL AB à http://dev.mysql.com/doc/ .

21.5 API PHP pour MySQL

PHP est un langage côté serveur, qui peut être utilisé pour créer des pages web dynamiques. Il fournit un support d'accès à plusieurs systèmes de bases de données, dont MySQL fait partie. PHP peut être utilisé en tant que programme séparé, ou être compilé en tant que module pour le serveur web Apache.

La distribution et documentation sont disponibles sur le site web de PHP (http://www.php.net/).

21.5.1 Problèmes fréquents avec MySQL et PHP

- Error: "Maximum Execution Time Exceeded" C'est une limitation de PHP; Editez le fichier php.ini et changez le temps maximal d'exécution d'un script de 30 secondes à plus, selon vos besoins. C'est aussi une bonne idée de doubler la quantité de RAM allouée par script en la changeant à 16MB.
- Error: "Fatal error: Call to unsupported or undefined function mysql_connect() in .." Cela signifie que votre version de PHP n'est pas compilée avec le support de MySQL. Vous pouvez soit compiler un module dynamique MySQL et le charger dans PHP, ou bien recompiler PHP avec le support natif de MySQL. Ceci est décrit en détails dans le manuel PHP
- Error: "undefined reference to `uncompress'" Cela signifie que la bibliothèque cliente est compilée avec support d'un protocole client/serveur compressé. La solution est d'ajouter -lz à la fin lors de la liaison avec -lmysqlclient .

21.6 API Perl pour MySQL

DBI est une interface générique à plusieurs systèmes de bases de données. Cela signifie que vous pouvez écrire un script qui fonctionne parfaitement avec plusieurs systèmes différents sans y apporter aucun changement. Vous avez besoin de définir un pilote de base de données (DataBase Driver : DBD) pour chaque système. Pour MySQL, ce pilote se nomme DBD::mysql .

Perl DBI est maintenant l'interface recommandée pour Perl. Elle remple une ancienne interface appelée mysqlperl , qui doit être abandonnée.

Vous pouvez trouver les dernières informations relatives à DBI dans la section Commentaires sur l'installation de Perl .

Plus de détails sur DBI sont disponibles en ligne de commande, sur Internet ou en version imprimée.

- Une fois que vous avez installé DBI et DBD::mysql , vous pouvez utiliser la commande perldoc pour obtenir plus d'informations à propos de DBI .
```
 shell> perldoc DBI
shell> perldoc DBI::FAQ
shell> perldoc DBD::mysql
```
  Vous pouvez aussi utiliser les outils pod2man , pod2html , etc., pour convertir la documentation en différents formats.
- Pour plus d'informations sur le module DBI de Perl5, visitez la page web de DBI : http://dbi.perl.org/
- Pour une documentation imprimée, voyez le guide officiel DBI : Programming the Perl DBI (Alligator Descartes et Tim Bunce, O'Reilly &Associates, 2000). Les informations sur le livre sont disponibles sur le site de DBI : http://dbi.perl.org/ .Pour des informations qui sont spécifiques à DBI et MySQL, voyez MySQL and Perl for the Web (Paul DuBois, New Riders, 2001). Le site web de ce livre est http://www.kitebird.com/mysql-perl/ .

21.7 Interface MySQL C++

MySQL Connector/C++ (or MySQL++) est l'interface officielle de MySQL pour le C++. Plus d'informations sont disponibles sur http://www.mysql.com/products/mysql++/ .

21.7.1 Borland C++

Vous pouvez compiler la source Windows de MySQL avec Borland C++ 5.02. (La source Windows n'inclut que les fichiers issus de Microsoft VC++, pour Borland C++ vous devrez créer les fichiers de projet par vous-même.)

Un problème connu avec Borland C++ est qu'il utilise un alignement de structures différent de VC++. Cela signifie que vous aurez des problèmes si vous essayez d'utiliser la bibliothèque par défaut libmysql.dll (qui a été compilée avec VC++) avec Borland C++. Vous pouvez faire ce qui suit pour éviter ce problème.

- Vous pouvez utiliser les bibliothèques MySQL statiques pour Borland C++ que vous trouverez sur http://www.mysql.com/downloads/os-win32.html .
- Appelez mysql_init() avec NULL comme argument, et non une structure MYSQL pre-allouée.

21.8 MySQL Python API

MySQLdb fournit le support MySQL pour Python , compatible avec l'API de base de données Python version 2.0. Elle est disponible à l'URL http://sourceforge.net/projects/mysql-python/ .

21.9 MySQL Tcl API

MySQLtcl est une API simple pour accéder au serveur MySQL depuis le langage de programmation Tcl. Il est disponible sur http://www.xdobry.de/mysqltcl/ .

21.10 Couche MySQL pour Eiffel

Eiffel MySQL est une interface avec le langage de programmation Eiffel, écrit par Michael Ravits. Il est disponible sur http://efsa.sourceforge.net/archive/ravits/mysql.htm .

22 Gestion des erreurs avec MySQL

Ce chapitre décrit comment MySQL gère les erreurs.

22.1 Retours d'erreurs

Voici la liste des erreurs que vous pouvez rencontrer lorsque vous utilisez MySQL sur un serveur ayant le support de la langue locale.

Les colonnes Name et Error Code correspondent aux définitions placées dans le code source MySQL : include/mysqld_error.h

La colonne SQLSTATE correspond aux définitions du code source MySQL : include/sql_state.h

Le code d'erreur SQLSTATE n'apparaîtra que si vous utilisez MySQL version 4.1. Les codes d'erreurs SQLSTATE ont été ajouté pour assure la compatibilité avec X/Open / ANSI / ODBC.

Un message d'erreur est disponible dans le fichier de messages d'erreurs : share/english/errmsg.sys

Comme les mises à jours sont fréquentes, il est possible que les sources ci-dessus contiennent d'autres codes d'erreur.

Nom	**Code d'erreur**	**SQLSTATE**
ER_HASHCHK	1000	HY000
ER_NISAMCHK	1001	HY000
ER_NO	1002	HY000
ER_YES	1003	HY000
ER_CANT_CREATE_FILE	1004	HY000
ER_CANT_CREATE_TABLE	1005	HY000
ER_CANT_CREATE_DB	1006	HY000
ER_DB_CREATE_EXISTS	1007	HY000
ER_DB_DROP_EXISTS	1008	HY000
ER_DB_DROP_DELETE	1009	HY000
ER_DB_DROP_RMDIR	1010	HY000
ER_CANT_DELETE_FILE	1011	HY000
ER_CANT_FIND_SYSTEM_REC	1012	HY000
ER_CANT_GET_STAT	1013	HY000
ER_CANT_GET_WD	1014	HY000
ER_CANT_LOCK	1015	HY000
ER_CANT_OPEN_FILE	1016	HY000
ER_FILE_NOT_FOUND	1017	HY000
ER_CANT_READ_DIR	1018	HY000
ER_CANT_SET_WD	1019	HY000
ER_CHECKREAD	1020	HY000
ER_DISK_FULL	1021	HY000
ER_DUP_KEY	1022	23000
ER_ERROR_ON_CLOSE	1023	HY000
ER_ERROR_ON_READ	1024	HY000
ER_ERROR_ON_RENAME	1025	HY000
ER_ERROR_ON_WRITE	1026	HY000
ER_FILE_USED	1027	HY000
ER_FILSORT_ABORT	1028	HY000
ER_FORM_NOT_FOUND	1029	HY000
ER_GET_ERRNO	1030	HY000
ER_ILLEGAL_HA	1031	HY000
ER_KEY_NOT_FOUND	1032	HY000
ER_NOT_FORM_FILE	1033	HY000
ER_NOT_KEYFILE	1034	HY000
ER_OLD_KEYFILE	1035	HY000
ER_OPEN_AS_READONLY	1036	HY000
ER_OUTOFMEMORY	1037	HY001
ER_OUT_OF_SORTMEMORY	1038	HY001
ER_UNEXPECTED_EOF	1039	HY000
ER_CON_COUNT_ERROR	1040	08004
ER_OUT_OF_RESOURCES	1041	08004
ER_BAD_HOST_ERROR	1042	08S01

ER_HANDSHAKE_ERROR	1043	08S01
ER_DBACCESS_DENIED_ERROR	1044	42000
ER_ACCESS_DENIED_ERROR	1045	42000
ER_NO_DB_ERROR	1046	42000
ER_UNKNOWN_COM_ERROR	1047	08S01
ER_BAD_NULL_ERROR	1048	23000
ER_BAD_DB_ERROR	1049	42000
ER_TABLE_EXISTS_ERROR	1050	42S01
ER_BAD_TABLE_ERROR	1051	42S02
ER_NON_UNIQ_ERROR	1052	23000
ER_SERVER_SHUTDOWN F	1053	08S01
ER_BAD_FIELD_ERROR	1054	42S22
ER_WRONG_FIELD_WITH_GROUP	1055	42000
ER_WRONG_GROUP_FIELD	1056	42000
ER_WRONG_SUM_SELECT	1057	42000
ER_WRONG_VALUE_COUNT	1058	21S01
ER_TOO_LONG_IDENT	1059	42000
ER_DUP_FIELDNAME	1060	42S21
ER_DUP_KEYNAME	1061	42000
ER_DUP_ENTRY	1062	23000
ER_WRONG_FIELD_SPEC	1063	42000
ER_PARSE_ERROR	1064	42000
ER_EMPTY_QUERY	1065	42000
ER_NONUNIQ_TABLE	1066	42000
ER_INVALID_DEFAULT	1067	42000
ER_MULTIPLE_PRI_KEY	1068	42000
ER_TOO_MANY_KEYS	1069	42000
ER_TOO_MANY_KEY_PARTS	1070	42000
ER_TOO_LONG_KEY	1071	42000
ER_KEY_COLUMN_DOES_NOT_EXITS	1072	42000
ER_BLOB_USED_AS_KEY	1073	42000
ER_TOO_BIG_FIELDLENGTH	1074	42000
ER_WRONG_AUTO_KEY	1075	42000
ER_READY	1076	00000
ER_NORMAL_SHUTDOWN	1077	00000
ER_GOT_SIGNAL	1078	00000
ER_SHUTDOWN_COMPLETE	1079	00000
ER_FORCING_CLOSE	1080	08S01
ER_IPSOCK_ERROR	1081	08S01
ER_NO_SUCH_INDEX	1082	42S12
ER_WRONG_FIELD_TERMINATORS	1083	42000
ER_BLOBS_AND_NO_TERMINATED	1084	42000
ER_TEXTFILE_NOT_READABLE	1085	HY000
ER_FILE_EXISTS_ERROR	1086	HY000
ER_LOAD_INFO	1087	HY000
ER_ALTER_INFO	1088	HY000
ER_WRONG_SUB_KEY	1089	HY000
ER_CANT_REMOVE_ALL_FIELDS	1090	42000
ER_CANT_DROP_FIELD_OR_KEY	1091	42000
ER_INSERT_INFO	1092	HY000
ER_UPDATE_TABLE_USED	1093	HY000
ER_NO_SUCH_THREAD	1094	HY000
ER_KILL_DENIED_ERROR	1095	HY000
ER_NO_TABLES_USED	1096	HY000
ER_TOO_BIG_SET	1097	HY000
ER_NO_UNIQUE_LOGFILE	1098	HY000
ER_TABLE_NOT_LOCKED_FOR_WRITE	1099	HY000
ER_TABLE_NOT_LOCKED	1100	HY000
ER_BLOB_CANT_HAVE_DEFAULT	1101	42000
ER_WRONG_DB_NAME	1102	42000
ER_WRONG_TABLE_NAME	1103	42000
ER_TOO_BIG_SELECT	1104	42000
ER_UNKNOWN_ERROR	1105	HY000
ER_UNKNOWN_PROCEDURE	1106	42000
ER_WRONG_PARAMCOUNT_TO_PROCEDURE	1107	42000
ER_WRONG_PARAMETERS_TO_PROCEDURE	1108	HY000
ER_UNKNOWN_TABLE	1109	42S02
ER_FIELD_SPECIFIED_TWICE	1110	42000
ER_INVALID_GROUP_FUNC_USE	1111	42000
ER_UNSUPPORTED_EXTENSION	1112	42000
ER_TABLE_MUST_HAVE_COLUMNS	1113	42000

ER_RECORD_FILE_FULL	1114	HY000
ER_UNKNOWN_CHARACTER_SET	1115	42000
ER_TOO_MANY_TABLES	1116	HY000
ER_TOO_MANY_FIELDS	1117	HY000
ER_TOO_BIG_ROWSIZE	1118	42000
ER_STACK_OVERRUN	1119	HY000
ER_WRONG_OUTER_JOIN	1120	42000
ER_NULL_COLUMN_IN_INDEX	1121	42000
ER_CANT_FIND_UDF	1122	HY000
ER_CANT_INITIALIZE_UDF	1123	HY000
ER_UDF_NO_PATHS	1124	HY000
ER_UDF_EXISTS	1125	HY000
ER_CANT_OPEN_LIBRARY	1126	HY000
ER_CANT_FIND_DL_ENTRY	1127	HY000
ER_FUNCTION_NOT_DEFINED	1128	HY000
ER_HOST_IS_BLOCKED	1129	HY000
ER_HOST_NOT_PRIVILEGED	1130	HY000
ER_PASSWORD_ANONYMOUS_USER	1131	42000
ER_PASSWORD_NOT_ALLOWED	1132	42000
ER_PASSWORD_NO_MATCH	1133	42000
ER_UPDATE_INFO	1134	HY000
ER_CANT_CREATE_THREAD	1135	HY000
ER_WRONG_VALUE_COUNT_ON_ROW	1136	21S01
ER_CANT_REOPEN_TABLE	1137	HY000
ER_INVALID_USE_OF_NULL	1138	42000
ER_REGEXP_ERROR	1139	42000
ER_MIX_OF_GROUP_FUNC_AND_FIELDS	1140	42000
ER_NONEXISTING_GRANT	1141	42000
ER_TABLEACCESS_DENIED_ERROR	1142	42000
ER_COLUMNACCESS_DENIED_ERROR	1143	42000
ER_ILLEGAL_GRANT_FOR_TABLE	1144	42000
ER_GRANT_WRONG_HOST_OR_USER	1145	42000
ER_NO_SUCH_TABLE	1146	42S02
ER_NONEXISTING_TABLE_GRANT	1147	42000
ER_NOT_ALLOWED_COMMAND	1148	42000
ER_SYNTAX_ERROR	1149	42000
ER_DELAYED_CANT_CHANGE_LOCK	1150	HY000
ER_TOO_MANY_DELAYED_THREADS	1151	HY000
ER_ABORTING_CONNECTION	1152	08S01
ER_NET_PACKET_TOO_LARGE	1153	08S01
ER_NET_READ_ERROR_FROM_PIPE	1154	08S01
ER_NET_FCNTL_ERROR	1155	08S01
ER_NET_PACKETS_OUT_OF_ORDER	1156	08S01
ER_NET_UNCOMPRESS_ERROR	1157	08S01
ER_NET_READ_ERROR	1158	08S01
ER_NET_READ_INTERRUPTED	1159	08S01
ER_NET_ERROR_ON_WRITE	1160	08S01
ER_NET_WRITE_INTERRUPTED	1161	08S01
ER_TOO_LONG_STRING	1162	42000
ER_TABLE_CANT_HANDLE_BLOB	1163	42000
ER_TABLE_CANT_HANDLE_AUTO_INCREMENT	1164	42000
ER_DELAYED_INSERT_TABLE_LOCKED	1165	HY000
ER_WRONG_COLUMN_NAME	1166	42000
ER_WRONG_KEY_COLUMN	1167	42000
ER_WRONG_MRG_TABLE	1168	HY000
ER_DUP_UNIQUE	1169	23000
ER_BLOB_KEY_WITHOUT_LENGTH	1170	42000
ER_PRIMARY_CANT_HAVE_NULL	1171	42000
ER_TOO_MANY_ROWS	1172	42000
ER_REQUIRES_PRIMARY_KEY	1173	42000
ER_NO_RAID_COMPILED	1174	HY000
ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE	1175	HY000
ER_KEY_DOES_NOT_EXITS	1176	HY000
ER_CHECK_NO_SUCH_TABLE	1177	42000
ER_CHECK_NOT_IMPLEMENTED	1178	42000
ER_CANT_DO_THIS_DURING_AN_TRANSACTION	1179	25000
ER_ERROR_DURING_COMMIT	1180	HY000
ER_ERROR_DURING_ROLLBACK	1181	HY000
ER_ERROR_DURING_FLUSH_LOGS	1182	HY000
ER_ERROR_DURING_CHECKPOINT	1183	HY000
ER_NEW_ABORTING_CONNECTION	1184	08S01

ER_DUMP_NOT_IMPLEMENTED	1185	HY000
ER_FLUSH_MASTER_BINLOG_CLOSED	1186	HY000
ER_INDEX_REBUILD	1187	HY000
ER_MASTER	1188	HY000
ER_MASTER_NET_READ	1189	08S01
ER_MASTER_NET_WRITE	1190	08S01
ER_FT_MATCHING_KEY_NOT_FOUND	1191	HY000
ER_LOCK_OR_ACTIVE_TRANSACTION	1192	HY000
ER_UNKNOWN_SYSTEM_VARIABLE	1193	HY000
ER_CRASHED_ON_USAGE	1194	HY000
ER_CRASHED_ON_REPAIR	1195	HY000
ER_WARNING_NOT_COMPLETE_ROLLBACK	1196	HY000
ER_TRANS_CACHE_FULL	1197	HY000
ER_SLAVE_MUST_STOP	1198	HY000
ER_SLAVE_NOT_RUNNING	1199	HY000
ER_BAD_SLAVE	1200	HY000
ER_MASTER_INFO	1201	HY000
ER_SLAVE_THREAD	1202	HY000
ER_TOO_MANY_USER_CONNECTIONS	1203	42000
ER_SET_CONSTANTS_ONLY	1204	HY000
ER_LOCK_WAIT_TIMEOUT	1205	HY000
ER_LOCK_TABLE_FULL	1206	HY000
ER_READ_ONLY_TRANSACTION	1207	25000
ER_DROP_DB_WITH_READ_LOCK	1208	HY000
ER_CREATE_DB_WITH_READ_LOCK	1209	HY000
ER_WRONG_ARGUMENTS	1210	HY000
ER_NO_PERMISSION_TO_CREATE_USER	1211	42000
ER_UNION_TABLES_IN_DIFFERENT_DIR	1212	HY000
ER_LOCK_DEADLOCK	1213	40001
ER_TABLE_CANT_HANDLE_FULLTEXT	1214	HY000
ER_CANNOT_ADD_FOREIGN	1215	HY000
ER_NO_REFERENCED_ROW	1216	23000
ER_ROW_IS_REFERENCED	1217	23000
ER_CONNECT_TO_MASTER	1218	08S01
ER_QUERY_ON_MASTER	1219	HY000
ER_ERROR_WHEN_EXECUTING_COMMAND	1220	HY000
ER_WRONG_USAGE	1221	HY000
ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT	1222	21000
ER_CANT_UPDATE_WITH_READLOCK	1223	HY000
ER_MIXING_NOT_ALLOWED	1224	HY000
ER_DUP_ARGUMENT	1225	HY000
ER_USER_LIMIT_REACHED	1226	42000
ER_SPECIFIC_ACCESS_DENIED_ERROR	1227	HY000
ER_LOCAL_VARIABLE	1228	HY000
ER_GLOBAL_VARIABLE	1229	HY000
ER_NO_DEFAULT	1230	42000
ER_WRONG_VALUE_FOR_VAR	1231	42000
ER_WRONG_TYPE_FOR_VAR	1232	42000
ER_VAR_CANT_BE_READ	1233	HY000
ER_CANT_USE_OPTION_HERE	1234	42000
ER_NOT_SUPPORTED_YET	1235	42000
ER_MASTER_FATAL_ERROR_READING_BINLOG	1236	HY000
ER_SLAVE_IGNORED_TABLE	1237	HY000
ER_WRONG_FK_DEF	1238	42000
ER_KEY_REF_DO_NOT_MATCH_TABLE_REF	1239	HY000
ER_CARDINALITY_COL	1240	21000
ER_SUBSELECT_NO_ 1_ROW	1241	21000
ER_UNKNOWN_STMT_HANDLER	1242	HY000
ER_CORRUPT_HELP_DB	1243	HY000
ER_CYCLIC_REFERENCE	1244	HY000
ER_AUTO_CONVERT	1245	HY000
ER_ILLEGAL_REFERENCE	1246	42S22
ER_DERIVED_MUST_HAVE_ALIAS	1247	42000
ER_SELECT_REDUCED	1248	01000
ER_TABLENAME_NOT_ALLOWED_HERE	1249	42000
ER_NOT_SUPPORTED_AUTH_MODE	1250	08004
ER_SPATIAL_CANT_HAVE_NULL	1251	42000
ER_COLLATION_CHARSET_MISMATCH	1252	42000
ER_SLAVE_WAS_RUNNING	1253	HY000
ER_SLAVE_WAS_NOT_RUNNING	1254	HY000
ER_TOO_BIG_FOR_UNCOMPRESS	1255	HY000

ER_ZLIB_Z_MEM_ERROR	1256	HY000
ER_ZLIB_Z_BUF_ERROR	1257	HY000
ER_ZLIB_Z_DATA_ERROR	1258	HY000
ER_CUT_VALUE_GROUP_CONCAT	1259	HY000
ER_WARN_TOO_FEW_RECORDS	1260	01000
ER_WARN_TOO_MANY_RECORDS	1261	01000
ER_WARN_NULL_TO_NOTNULL	1262	01000
ER_WARN_DATA_OUT_OF_RANGE	1263	01000
ER_WARN_DATA_TRUNCATED	1264	01000
ER_WARN_USING_OTHER_HANDLER	1265	01000
ER_CANT_AGGREGATE_COLLATIONS	1266	42000
ER_DROP_USER	1267	42000
ER_REVOKE_GRANTS	1268	42000
ER_CANT_AGGREGATE_3COLLATIONS	1269	42000
ER_CANT_AGGREGATE_NCOLLATIONS	1270	42000
ER_VARIABLE_IS_NOT_STRUCT	1271	HY000
ER_UNKNOWN_COLLATION	1272	HY000
ER_SLAVE_IGNORED_SSL_PARAMS	1273	HY000
ER_SERVER_IS_IN_SECURE_AUTH_MODE	1274	HY000
ER_WARN_FIELD_RESOLVED	1275	HY000
ER_BAD_SLAVE_UNTIL_COND	1276	HY000
ER_MISSING_SKIP_SLAVE	1277	HY000
ER_UNTIL_COND_IGNORED	1278	HY000

22.2 Messages d'erreurs

Voici la liste des messages d'erreurs qui peuvent apparaître si vous appelez MySQL, quelque soit la langue utilisée par l'hôte.

Code d'erreur	**Message d'erreur**
1000	hashchk
1001	isamchk
1002	NO
1003	YES
1004	Can't create file '%s' (errno: %d)
1005	Can't create table '%s' (errno: %d)
1006	Can't create database '%s' (errno: %d)
1007	Can't create database '%s'; database exists
1008	Can't drop database '%s'; database doesn't exist
1009	Error dropping database (can't delete '%s', errno: %d)
1010	Error dropping database (can't rmdir '%s', errno: %d)
1011	Error on delete of '%s' (errno: %d)
1012	Can't read record in system table
1013	Can't get status of '%s' (errno: %d)
1014	Can't get working directory (errno: %d)
1015	Can't lock file (errno: %d)
1016	Can't open file: '%s' (errno: %d)
1017	Can't find file: '%s' (errno: %d)
1018	Can't read dir of '%s' (errno: %d)
1019	Can't change dir to '%s' (errno: %d)
1020	Record has changed since last read in table '%s'
1021	Disk full (%s). Waiting for someone to free some space...
1022	Can't write, duplicate key in table '%s'
1023	Error on close of '%s' (errno: %d)
1024	Error reading file '%s' (errno: %d)
1025	Error on rename of '%s' to '%s' (errno: %d)
1026	Error writing file '%s' (errno: %d)
1027	'%s' is locked against change
1028	Sort aborted
1029	View '%s' doesn't exist for '%s'
1030	Got error %d from storage engine
1031	Table storage engine for '%s' doesn't have this option
1032	Can't find record in '%s'
1033	Incorrect information in file: '%s'
1034	Incorrect key file for table: '%s'; try to repair it
1035	Old key file for table '%s'; repair it!
1036	Table '%s' is read only
1037	Out of memory. Restart daemon and try again (needed %d bytes)

1038	Out of sort memory. Increase daemon sort buffer size
1039	Unexpected eof found when reading file '%s' (errno: %d)
1040	Too many connections
1041	Out of memory; Check if mysqld or some other process uses all available memory. If not you may have to use 'ulimit' to allow mysqld to use more memory or you can add more swap space
1042	Can't get hostname for your address
1043	Bad handshake
1044	Access denied for user: '%s'@'%s' to database '%s'
1045	Access denied for user: '%s'@'%s' (Using password: %s)
1046	No Database Selected
1047	Unknown command
1048	Column '%s' cannot be null
1049	Unknown database '%s'
1050	Table '%s' already exists
1051	Unknown table '%s'
1052	Column: '%s' in %s is ambiguous
1053	Server shutdown in progress
1054	Unknown column '%s' in '%s'
1055	'%s' isn't in GROUP BY
1056	Can't group on '%s'
1057	Statement has sum functions and columns in same statement
1058	Column count doesn't match value count
1059	Identifier name '%s' is too long
1060	Duplicate column name '%s'
1061	Duplicate key name '%s'
1062	Duplicate entry '%s' for key %d
1063	Incorrect column specifier for column '%s'
1064	%s near '%s' at line %d
1065	Query was empty
1066	Not unique table/alias: '%s'
1067	Invalid default value for '%s'
1068	Multiple primary key defined
1069	Too many keys specified. Max %d keys allowed
1070	Too many key parts specified. Max %d parts allowed
1071	Specified key was too long. Max key length is %d
1072	Key column '%s' doesn't exist in table
1073	BLOB column '%s' can't be used in key specification with the used table type
1074	Too big column length for column '%s' (max = %d). Use BLOB instead
1075	Incorrect table definition; There can only be one auto column and it must be defined as a key
1076	%s: ready for connections. Version: '%s' socket: '%s' port: %d
1077	%s: Normal shutdown
1078	%s: Got signal %d. Aborting!
1079	%s: Shutdown Complete
1080	%s: Forcing close of thread %ld user: '%s'
1081	Can't create IP socket
1082	Table '%s' has no index like the one used in CREATE INDEX. Recreate the table
1083	Field separator argument is not what is expected. Check the manual
1084	You can't use fixed rowlength with BLOBs. Please use 'fields terminated by'
1085	The file '%s' must be in the database directory or be readable by all
1086	File '%s' already exists
1087	Records: %ld Deleted: %ld Skipped: %ld Warnings: %ld
1088	Records: %ld Duplicates: %ld
1089	Incorrect sub part key. The used key part isn't a string, the used length is longer than the key part or the storage engine doesn't support unique sub keys
1090	You can't delete all columns with ALTER TABLE. Use DROP TABLE instead
1091	Can't DROP '%s'. Check that column/key exists
1092	Records: %ld Duplicates: %ld Warnings: %ld
1093	You can't specify target table '%s' for update in FROM clause
1094	Unknown thread id: %lu
1095	You are not owner of thread %lu
1096	No tables used
1097	Too many strings for column %s and SET
1098	Can't generate a unique log-filename %s.(1-999)
1099	Table '%s' was locked with a READ lock and can't be updated
1100	Table '%s' was not locked with LOCK TABLES
1101	BLOB/TEXT column '%s' can't have a default value
1102	Incorrect database name '%s'
1103	Incorrect table name '%s'
1104	The SELECT would examine more rows than MAX_JOIN_SIZE. Check your WHERE and use SET SQL_BIG_SELECTS=1 or SET SQL_MAX_JOIN_SIZE=# if the SELECT is ok
1105	Unknown error
1106	Unknown procedure '%s'

1107	Incorrect parameter count to procedure '%s'
1108	Incorrect parameters to procedure '%s'
1109	Unknown table '%s' in %s
1110	Column '%s' specified twice
1111	Invalid use of group function
1112	Table '%s' uses an extension that doesn't exist in this MySQL version
1113	A table must have at least 1 column
1114	The table '%s' is full
1115	Unknown character set: '%s'
1116	Too many tables. MySQL can only use %d tables in a join
1117	Too many columns
1118	Too big row size. The maximum row size for the used table type, not counting BLOBs, is %ld. You have to change some fields to TEXT or BLOBs
1119	Thread stack overrun: Used: %ld of a %ld stack. Use 'mysqld −O thread_stack=#' to specify a bigger stack if needed
1120	Cross dependency found in OUTER JOIN. Examine your ON conditions
1121	Column '%s' is used with UNIQUE or INDEX but is not defined as NOT NULL
1122	Can't load function '%s'
1123	Can't initialize function '%s'; %s
1124	No paths allowed for shared library
1125	Function '%s' already exist
1126	Can't open shared library '%s' (errno: %d %s)
1127	Can't find function '%s' in library'
1128	Function '%s' is not defined
1129	Host '%s' is blocked because of many connection errors. Unblock with 'mysqladmin flush−hosts'
1130	Host '%s' is not allowed to connect to this MySQL server
1131	You are using MySQL as an anonymous users and anonymous users are not allowed to change passwords
1132	You must have privileges to update tables in the mysql database to be able to change passwords for others
1133	Can't find any matching row in the user table
1134	Rows matched: %ld Changed: %ld Warnings: %ld
1135	Can't create a new thread (errno %d). If you are not out of available memory, you can consult the manual for a possible OS−dependent bug
1136	Column count doesn't match value count at row %ld
1137	Can't reopen table: '%s'
1138	Invalid use of NULL value
1139	Got error '%s' from regexp
1140	Mixing of GROUP columns (MIN(),MAX(),COUNT()...) with no GROUP columns is illegal if there is no GROUP BY clause
1141	There is no such grant defined for user '%s' on host '%s'
1142	%s command denied to user: '%s'@'%s' for table '%s'
1143	%s command denied to user: '%s'@'%s' for column '%s' in table '%s'
1144	Illegal GRANT/REVOKE command. Please consult the manual which privileges can be used
1145	The host or user argument to GRANT is too long
1146	Table '%s.%s' doesn't exist
1147	There is no such grant defined for user '%s' on host '%s' on table '%s'
1148	The used command is not allowed with this MySQL version
1149	You have an error in your SQL syntax. Check the manual that corresponds to your MySQL server version for the right syntax to use
1150	Delayed insert thread couldn't get requested lock for table %s
1151	Too many delayed threads in use
1152	Aborted connection %ld to db: '%s' user: '%s' (%s)
1153	Got a packet bigger than 'max_allowed_packet'
1154	Got a read error from the connection pipe
1155	Got an error from fcntl()
1156	Got packets out of order
1157	Couldn't uncompress communication packet
1158	Got an error reading communication packets
1159	Got timeout reading communication packets
1160	Got an error writing communication packets
1161	Got timeout writing communication packets
1162	Result string is longer than max_allowed_packet
1163	The used table type doesn't support BLOB/TEXT columns
1164	The used table type doesn't support AUTO_INCREMENT columns
1165	INSERT DELAYED can't be used with table '%s' because it is locked with LOCK TABLES
1166	Incorrect column name '%s'
1167	The used storage engine can't index column '%s'
1168	All tables in the MERGE table are not identically defined
1169	Can't write, because of unique constraint, to table '%s'
1170	BLOB/TEXT column '%s' used in key specification without a key length
1171	All parts of a PRIMARY KEY must be NOT NULL; If you need NULL in a key, use UNIQUE instead
1172	Result consisted of more than one row
1173	This table type requires a primary key
1174	This version of MySQL is not compiled with RAID support
1175	You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column

1176	Key '%s' doesn't exist in table '%s'
1177	Can't open table
1178	The storage engine for the table doesn't support %s
1179	You are not allowed to execute this command in a transaction
1180	Got error %d during COMMIT
1181	Got error %d during ROLLBACK
1182	Got error %d during FLUSH_LOGS
1183	Got error %d during CHECKPOINT
1184	Aborted connection %ld to db: '%s' user: '%s' host: `%s' (%s)
1185	The storage engine for the table does not support binary table dump
1186	Binlog closed, cannot RESET MASTER
1187	Failed rebuilding the index of dumped table '%s'
1188	Error from master: '%s'
1189	Net error reading from master
1190	Net error writing to master
1191	Can't find FULLTEXT index matching the column list
1192	Can't execute the given command because you have active locked tables or an active transaction
1193	Unknown system variable '%s'
1194	Table '%s' is marked as crashed and should be repaired
1195	Table '%s' is marked as crashed and last (automatic?) repair failed
1196	Some non−transactional changed tables couldn't be rolled back
1197	Multi−statement transaction required more than 'max_binlog_cache_size' bytes of storage. Increase this mysqld variable and try again
1198	This operation cannot be performed with a running slave, run STOP SLAVE first
1199	This operation requires a running slave, configure slave and do START SLAVE
1200	The server is not configured as slave, fix in config file or with CHANGE MASTER TO
1201	Could not initialize master info structure, more error messages can be found in the MySQL error log
1202	Could not create slave thread, check system resources
1203	User %s has already more than 'max_user_connections' active connections
1204	You may only use constant expressions with SET
1205	Lock wait timeout exceeded; Try restarting transaction
1206	The total number of locks exceeds the lock table size
1207	Update locks cannot be acquired during a READ UNCOMMITTED transaction
1208	DROP DATABASE not allowed while thread is holding global read lock
1209	CREATE DATABASE not allowed while thread is holding global read lock
1210	Wrong arguments to %s
1211	'%s'@'%s' is not allowed to create new users
1212	Incorrect table definition; all MERGE tables must be in the same database
1213	Deadlock found when trying to get lock; Try restarting transaction
1214	The used table type doesn't support FULLTEXT indexes
1215	Cannot add foreign key constraint
1216	Cannot add or update a child row: a foreign key constraint fails
1217	Cannot delete or update a parent row: a foreign key constraint fails
1218	Error connecting to master: %s
1219	Error running query on master: %s
1220	Error when executing command %s: %s
1221	Wrong usage of %s and %s
1222	The used SELECT statements have a different number of columns
1223	Can't execute the query because you have a conflicting read lock
1224	Mixing of transactional and non−transactional tables is disabled
1225	Option '%s' used twice in statement
1226	User '%s' has exceeded the '%s' resource (current value: %ld)
1227	Access denied. You need the %s privilege for this operation
1228	Variable '%s' is a LOCAL variable and can't be used with SET GLOBAL
1229	Variable '%s' is a GLOBAL variable and should be set with SET GLOBAL
1230	Variable '%s' doesn't have a default value
1231	Variable '%s' can't be set to the value of '%s'
1232	Wrong argument type to variable '%s'
1233	Variable '%s' can only be set, not read
1234	Wrong usage/placement of '%s'
1235	This version of MySQL doesn't yet support '%s'
1236	Got fatal error %d: '%s' from master when reading data from binary log
1237	Slave SQL thread ignored the query because of replicate−*−table rules
1238	Wrong foreign key definition for '%s': %s
1239	Key reference and table reference doesn't match
1240	Operand should contain %d column(s)
1241	Subquery returns more than 1 row
1242	Unknown prepared statement handler (%ld) given to %s
1243	Help database is corrupt or does not exist
1244	Cyclic reference on subqueries
1245	Converting column '%s' from %s to %s

1246	Reference '%s' not supported (%s)
1247	Every derived table must have it's own alias
1248	Select %u was reduced during optimisation
1249	Table '%s' from one of SELECT's can not be used in %s
1250	Client does not support authentication protocol requested by server; consider upgrading MySQL client
1251	All parts of a SPATIAL KEY must be NOT NULL
1252	COLLATION '%s' is not valid for CHARACTER SET '%s'
1253	Slave is already running
1254	Slave has already been stopped
1255	Too big size of uncompressed data. The maximum size is %d. (probably, length of uncompressed data was corrupted)
1256	ZLIB: Not enough memory available for zlib
1257	ZLIB: Not enough room in the output buffer for zlib (probably, length of uncompressed data was corrupted)
1258	ZLIB: Input data was corrupted for zlib
1259	%d line(s) was(were) cut by group_concat()
1260	Row %ld doesn't contain data for all columns
1261	Row %ld was truncated; It contained more data than there where input columns
1262	Data truncated, NULL supplied to NOT NULL column '%s' at row %ld
1263	Data truncated, out of range for column '%s' at row %ld
1264	Data truncated for column '%s' at row %ld
1265	Using storage engine %s for table '%s'
1266	Illegal mix of collations (%s,%s) and (%s,%s) for operation '%s'
1267	Can't drop one or more of the requested users
1268	Can't revoke all privileges, grant for one or more of the requested users
1269	Illegal mix of collations (%s,%s), (%s,%s), (%s,%s) for operation '%s'
1270	Illegal mix of collations for operation '%s'
1271	Variable '%s' is not a variable component (Can't be used as XXXX.variable_name)
1272	Unknown collation: '%s'
1273	SSL parameters in CHANGE MASTER are ignored because this MySQL slave was compiled without SSL support. They can be used later when MySQL slave with SSL will be started.
1274	Server is running in --secure-auth mode, but '%s@%s' has a password in the old format; please change the password to the new format
1275	Field or reference '%s%s%s%s%s' of SELECT #%d was resolved in SELECT #%d
1276	Wrong parameter or combination of parameters for START SLAVE UNTIL
1277	It is recommended to use --skip-slave-start when doing step-by-step replication with START SLAVE UNTIL. Otherwise you will get problems if you get an unexpected slave's mysqld restart
1278	SQL thread is not to be started so UNTIL options are ignored
1279	Incorrect index name '%s'
1280	Incorrect catalog name '%s'
1281	Query cache failed to set size %lu, new query cache size is %lu
1282	Column '%s' cannot be part of FULLTEXT index
1283	Unknown key cache '%s'
1284	MySQL is started in --skip-name-resolve mode. You need to restart it without this switch for this grant to work
1285	Unknown table engine '%s'
1286	'%s' is deprecated. Use '%s' instead.
1287	The target table %s of the %s is not updatable.
1288	The '%s' feature was disabled. You need MySQL built with '%s' define to have it working
1289	MySQL is started in --skip-grant-tables mode. You can't use this command
1290	Can't create a %s from within another stored routine
1291	%s %s already exists
1292	%s %s does not exist
1293	Failed to DROP %s %s
1294	Failed to CREATE %s %s
1295	%s with no matching label: %s
1296	Redefining label %s
1297	End-label %s without match
1298	Referring to uninitialized variable %s
1299	SELECT in a stored procedure must have INTO
1300	RETURN is only allowed in a FUNCTION
1301	Statements like SELECT, INSERT, UPDATE (and others) are not allowed in a FUNCTION
1302	The update log is deprecated and replaced by the binary log. SET SQL_LOG_UPDATE has been ignored.
1303	The update log is deprecated and replaced by the binary log. SET SQL_LOG_UPDATE has been translated to SET SQL_LOG_BIN.
1304	Query execution was interrupted
1305	Wrong number of arguments for %s %s, expected %u, got %u
1306	Undefined CONDITION: %s
1307	No RETURN found in FUNCTION %s
1308	FUNCTION %s ended without RETURN
1309	Cursor statement must be a SELECT
1310	Cursor SELECT must not have INTO
1311	Undefined CURSOR: %s
1312	Cursor is already open
1313	Cursor is not open
1314	Undeclared variable: %s

1315	Wrong number of FETCH variables
1316	No data to FETCH
1317	Duplicate parameter: %s
1318	Duplicate variable: %s
1319	Duplicate condition: %s
1320	Duplicate cursor: %s
1321	Failed to ALTER %s %s
1322	Subselect value not supported

23 Etendre MySQL

23.1 Rouages de MySQL

Ce chapitre décrit un grand nombre de notions que vous devez connaître lorsque vous travaillez sur le code de MySQL. Si vous envisagez de contribuer au développement de MySQL, que vous voulez accéder à du code ultra récent, ou que vous souhaitez simplement vous tenir au courant du développement, suivez les instructions de la section Installer à partir des sources de l'arbre de développement . Si vous êtes intéressés par les rouages internes de MySQL, il est recommandé de vous inscrire à notre liste de diffusion interne `internals` . Cette liste est relativement calme. Pour les détails d'inscription, voyez Les listes de diffusion MySQL . Tous les développeurs de MySQL AB sont sur la liste `internals` et nous aidons ceux qui travaillent sur le code de MySQL. Utilisez cette liste pour poser des questions sur le code, et partagez les correctifs que vous voulez voir intégré au projet MySQL.

23.1.1 Threads MySQL

Le serveur MySQL crée les threads suivants :

- Le thread de connexion TCP/IP, qui gère toutes les demandes de connexion, et crée un nouveau thread dédié pour gérer l'identification et le traitement des requêtes SQL, pour chaque connexion.
- Sur Windows NT, il y a un thread appelé gestionnaire de pipe, qui effectue le même travail que le thread de gestion des demandes de connexion, sur les demandes de connexion par pipe.
- Le threads de signal gère tous les signaux. Ce thread gère aussi normalement les alertes et les appels à process_alarm() pour forcer les délais d'expiration des connexions qui sont inactives.
- Si mysqld est compilé avec l'option -DUSE_ALARM_THREAD , un thread dédié aux alarmes est créé. Il est uniquement utilisé sur certains systèmes où il y a des problèmes avec la fonction sigwait() ou si vous voulez utiliser la fonction thr_alarm() dans des applications qui n'ont pas de thread dédié aux signaux.
- Lors de l'utilisation de l'option --flush_time=# , un thread dédié est créé pour vider les tables de la mémoire à intervalle régulier.
- Chaque connexion a son propre thread dédié.
- Chaque table différente sur laquelle des INSERT DELAYED sont pratiquées reçoit un thread.
- Si vous utilisez l'option --master-host , un thread de réplication sera démarré pour lire et appliquer les modifications du maître.

`mysqladmin processlist` affiche uniquement les threads de connexion, ceux de `INSERT DELAYED` et ceux de réplication.

23.1.2 Suite de test de MySQL

Jusqu'à récemment, notre suite de test principale étaient basée sur des données propriétaires de client, et pour cette raison, il n'a jamais été publié. Le seul système de test public actuel est notre script `crash-me` , qui est un script Perl `DBI/DBD` qui se trouve dans le dossier `sql-bench` , et divers tests qui font parti du dossier `tests` . Le manque d'une suite de test publique et standardisée rend difficile à nos utilisateurs et nos développeurs les possibilités de tests de régressions. Pour corriger ce problème, nous avons créé un nouveau système de tests qui est inclus dans les distributions source et binaires, à partir de la version 3.23.29.

Le jeu de test actuel ne couvre pas toutes les situations en MySQL, mais il permet d'identifier les bogues les plus courants lors de requêtes SQL, les problèmes de bibliothèques et aussi les problèmes de réplication. Notre but final est de lui faire couvrir 100% du code. Nous apprécions les contributions à notre suite de test. Vous pouvez notamment fournir des test qui examine certaines fonctions critiques de votre système, et qui assureront que les futures versions de MySQL le prennent en compte.

23.1.2.1 Exécuter la suite de tests MySQL

Le système de tests est constitué d'un interpréteur de langage de tests (`mysqltest`), un script Shell qui exécute tous les scripts (`mysql-test-run`), les cas de tests réels, écrits dans un langage spécial de tests, et leur résultats attendus. Pour exécuter ces tests sur votre système après une compilation, tapez `make test` ou `mysql-test/mysql-test-run` depuis la racine de la distribution. Si vous avez installé une distribution binaire, `cd` jusqu'au dossier d'installation (par exemple, `/usr/local/mysql`), et exécutez `scripts/mysql-test-run` . Tous les tests doivent réussir. Si ce n'est pas le cas, vous devriez essayer de trouver pourquoi, et faire un rapport de bogues à MySQL. Rapporter des bogues dans la suite de tests .

Si vous avez une copie de mysqld qui fonctionne sur la machine où vous voulez faire des tests, vous n'avez pas à l'arrêter, tant qu'elle n'utilise pas les ports 9306 et 9307 . Si l'un de ces ports est pris, vous devriez éditer le script mysql-test-run et changer les valeurs des ports des maîtres et esclaves, en les remplaçant par des ports libres.

Vous pouvez exécuter des tests individuels avec `mysql-test/mysql-test-run test_name` .

Si l'un des tests échoue, vous devriez tester mysql-test-run avec l'option --force pour vérifier si aucun autre test n'échoue.

23.1.2.2 Améliorer la suite de tests MySQL

Vous pouvez utiliser le langage de `mysqltest` pour écrire vos propres cas de tests. Malheureusement, nous n'avons pas encore écrit une documentation complète pour ce logiciel, et nous prévoyons de le faire rapidement. Vous pouvez, toutefois, utiliser les cas de tests actuels comme exemples. Les points suivants devraient vous mettre le pied à l'étrier.

- Les tests sont situé dans mysql-test/t/*.test
- Un cas de tests est constitué de commandes terminées par un ; , et est similaire aux données d'entrées du client mysql . Une commande est par défaut une commande envoyée au serveur MySQL, à moins qu'il ne soit reconnu comme une commande interne (par exemple, sleep).
- Toutes les requêtes qui produisent des résultats, comme SELECT , SHOW , EXPLAIN , etc., doivent être précédées par @/path/to/result/file . Le fichier contient alors les résultats attendus. Un moyen simple pour générer le résultat du fichier est d'exécuter mysqltest -r < t/test-case-name.test depuis le dossier de tests mysql-test , puis d'éditer le fichier résultant, si nécessaire, pour ajuster le contenu. Dans ce cas, soyez très prudent lors de l'ajout ou la suppression de caractères invisibles : assurez vous de ne changer que du texte, ou d'effacer des lignes. Vous pouvez utiliser od -c pour vous assurer que votre éditeur n'a pas perturbé le fichier durant l'édition. Bien sur, nous espérons que vous n'aurez jamais a éditer le résultat du fichier mysqltest -r ca vous n'avez à faire cela que lorsque vous découvrez un bug.
- Pour être cohérent avec votre configuration, vous devriez placer les fichiers de résultats dans le dossier mysql-test/r et les nommer test_name.result . Si le test produit plus qu'un résultat, vous devez utiliser test_name.a.result , test_name.b.result , etc.
- Si une commande retourne une erreur, vous devez, sur la ligne de la commande, le spécifier avec --error error-number . Le numéro d'erreur peut être une liste d'erreurs possibles, séparées par des virgules ',' .
- Si vous écrivez un test de réplication, vous devez, sur la première ligne du fichier de test, ajouter le code source include/master-slave.inc; . Pour passer entre le maître et l'esclave, utilisez connection master; et connection slave; . Si vous avez besoin d'utiliser une connexion alternative, vous pouvez utiliser connection master1; pour le maître, et connection slave1; pour l'esclave.
- Si vous avez besoin d'une boucle, vous pouvez utiliser ceci :

```
 let $1=1000;
while ($1)
&#123;
 # votre requête ici
 dec $1;
&#125;
```

- Pour faire une pause entre les requêtes, utilisez la commande sleep . Elle supporte les fraction de secondes, ce qui vous permet d'utiliser sleep 1.3; , pour attendre 1,3 secondes.
- Pour exécuter l'esclave avec des options additionnelles pour votre cas de tests, ajoutez les au format ligne de commande dans mysql-test/t/test_name-slave.opt . Pour le maître, ajoutez les dans mysql-test/t/test_name-master.opt .
- Si vous avez une question sur la suite de tests, ou que vous avez un test à proposer, envoyez le par email à sur la liste interne. Les listes de diffusion MySQL . Comme la liste n'accepte pas les attachements, vous devriez les placer sur le serveur FTP : ftp://support.mysql.com/pub/mysql/Incoming/

23.1.2.3 Rapporter des bugs dans la suite de tests MySQL

Si votre version de MySQL ne passe pas un teste, vous devez faire ceci :

- N'envoyez pas de rapport de bug avant d'avoir étudier au maximum les raisons possibles de l'échec! Lorsque vous le faîtes, utilisez le programme mysqlbug , pour que nous puissions obtenir un maximum d'informations sur votre système et la version de MySQL . Comment rapporter des bogues ou problèmes .
- Assurez vous d'inclure le résultat de mysql-test-run , ainsi que le contenu de tous les fichiers .reject du dossier mysql-test/r .
- Si un test de la suite échoue, vérifiez si le test échoue aussi en l'exécutant seul :

```
 cd mysql-test
mysql-test-run --local test-name
```

  Si cela échoue, alors vous devriez configurer MySQL avec --with-debug et exécuter mysql-test-run avec l'option --debug . Si cela échoue aussi, envoyez le fichier de trace var/tmp/master.trace à ftp://support.mysql.com/pub/mysql/secret pour que nous puissions l'examiner. N'oubliez pas d'inclure une description complète de votre système, ainsi que de la version de l'exécutable mysqld , et de sa compilation.
- Essayez d'exécuter mysql-test-run avec l'option --force pour voir si il n'y a pas d'autres tests qui échouent.
- Si vous avez compilé MySQL vous-même, vérifiez notre manuel, ainsi que les notes de compilations pour votre plate-forme, ou bien, utilisez à la place un des exécutables que nous avons compilé pour vous, disponibles à http://www.mysql.com/downloads/ . Toutes nos versions exécutables doivent passer la suite de tests.
- Si vous obtenez une erreur, comme Result length mismatch ou Result content mismatch , cela signifie que le résultat de la suite de tests n'a pas la taille attendue. Cela peut être un bug de MySQL, ou que votre version de MySQL fournit un résultat d'une autre taille, dans certaines circonstances.Les résultats de tests qui ont échoués sont placés dans un fichier avec le

même nom de base que le fichier de test, et avec l'extension .reject . Si votre test échoue, faites un diff sur les deux fichiers. Si vous ne pouvez pas voir où ils diffèrent, examinez ces deux fichiers avec od -c , et vérifiez leur tailles respectives.
- Si un test échoue totalement, vous devriez vérifier les fichiers de log dans le dossier mysql-test/var/log , pour avoir des indices sur ce qui a échoué.
- Si vous avez compilé MySQL avec le débogage, vous pouvez essayer de le déboger en exécutant mysql-test-run avec --gdb et/ou --debug . Créer des fichiers de traçage .Si vous n'avez pas compilé MySQL pour le débogage, vous devriez essayer de le faire. Spécifiez simplement l'option --with-debug dans le script de configure ! Installer MySQL à partir des sources .

23.2 Ajouter des fonctions à MySQL

Il y a deux méthodes pour ajouter des fonctions à MySQL :

- Vous pouvez ajouter la fonction grâce à l'interface de fonctions utilisateur (UDF). Les fonctions utilisateur sont ajoutées et supprimées dynamiquement avec les commandes CREATE FUNCTION et DROP FUNCTION . Syntaxe de CREATE FUNCTION/DROP FUNCTION .
- Vous pouvez ajouter une fonction sous la forme native (intégrée) d'une fonction MySQL. Les fonctions natives sont compilées dans mysqld et sont disponibles en permanence.

Chaque méthode a ses avantages et inconvénients :

- Si vous écrivez une fonction utilisateur, vous devez installer le fichier objet en plus du serveur lui-même. Si vous compilez votre fonction dans le serveur, vous n'avez pas ce problème.
- Vous pouvez ajouter des UDF à une distribution binaire de MySQL. Les fonctions natives requièrent une modification de la distribution source.
- Si vous mettez à jour votre distribution MySQL, vous pouvez continuer à utiliser vos fonctions précédemment installées. Pour les fonctions natives, vous devez refaire les modifications du code à chaque mise à jour.

Quelque soit la méthode que vous utilisez pour ajouter de nouvelles fonctions, ces fonctions pourront être utilisées comme des fonctions natives telles que ABS() ou SOUNDEX() .

23.2.1 Syntaxe de CREATE FUNCTION/DROP FUNCTION

```
CREATE [AGGREGATE] FUNCTION nom_fonction RETURNS &#123;STRING|REAL|INTEGER&#125;
      SONAME nom_librairie_partagée

DROP FUNCTION nom_fonction
```

Une fonction définie par un utilisateur (UDF) est une méthode pour intégrer une fonction qui fonctionne de la même façon qu'une fonction native de MySQL, comme ABS() et CONCAT() .

AGGREGATE est une nouvelle option pour MySQL version 3.23. Une fonction AGGREGATE fonctionne exactement comme une fonction native comme SUM ou COUNT() .

CREATE FUNCTION enregistre le nom de la fonction, le type, et le nom des bibliothèques partagées dans la table mysql.func . Vous devez avoir les droits INSERT et DELETE dans la base mysql pour créer et supprimer les fonctions.

Toutes les fonctions actives sont rechargées chaque fois que le serveur démarre, sauf si vous démarrez mysqld avec l'option --skip-grant-tables . Dans ce cas, l'utilisation des UDF n'est pas prise en compte et les UDFs ne sont pas disponibles. (Une fonction active est une fonction qui peut être chargée avec CREATE FUNCTION et supprimée par REMOVE FUNCTION).

Concernant l'écriture des UDFs, Ajout de nouvelles fonctions à MySQL . Pour que le mécanisme des fonctions UDF fonctionne, les fonctions doivent être écrites en C ou C++, votre système doit supporter le chargement dynamique et vous devez avoir compilé mysqld dynamiquement (pas statiquement).

Notez que pour faire fonctionner AGGREGATE , vous devez avoir une table mysql.func qui contient la colonne type . Si ce n'est pas le cas, vous devez exécuter le script mysql_fix_privilege_table pour résoudre ce problème.

23.2.2 Ajouter une nouvelle fonction définie par l'utilisateur (UDF)

Pour que le mécanisme UDF fonctionne, les fonctions doivent êtres écrites en C ou C++ et votre système doit supporter le chargement dynamique. Les sources de MySQL incluent un fichier sql/udf_example.cc qui définit 5 nouvelles fonctions. Consultez ce fichier pour voir comment marchent les conventions d'appels des UDF .

Pour que mysqld puisse utiliser les fonctions UDF , vous devez configurer MySQL avec l'option --with-mysqld-ldflags=-rdynamic . La raison est que sur diverses plates-formes, (Linux inclus) vous pouvez charger une bibliothèque dynamique (avec dlopen()) depuis un programme statique lié, que vous pouvez obtenir si vous utilisez l'option --with-mysql-ldflags=-all-static . Si vous voulez utiliser une UDF qui nécessite un accès aux symboles de mysqld (comme l'exemple methaphone dans sql/udf_example.cc qui utilise default_charset_info), vous devez lier le programme avec -rdynamic (voir man dlopen).

Pour chaque fonction que vous voulez utiliser dans SQL, vous devez définir les fonctions correspondantes en C (ou C++). Dans la discussion ci−dessous, le nom `` xxx '' est utilisé comme un exemple de nom de fonction. Pour faire la différence entre l'usage de SQL et de C/C++, XXX() (majuscules) indique l'appel d'une fonction SQL et xxx() (minuscules) indique l'appel d'une fonction C/C++.

Les fonctions C/C++ que vous écrivez pour l'implémentation de l'interface de XXX() sont :

xxx() *(requis)*
: La fonction principale. C'est là où le résultat de la fonction est calculé. La correspondance entre le type de SQL et le type retourné par votre fonction C/C++ est affiché ci-dessous :

SQL type	**C/C++ type**
STRING	char *
INTEGER	long long
REAL	double

xxx_init() *(optionnel)*
: La fonction d'initialisation de xxx() . Elle peut-être utilisée pour :
  - Vérifier le nombre d'arguments de XXX() .
  - Vérifier que les arguments correspondent aux types requis ou indiquer à MySQL de contraindre des arguments aux types que vous voulez quand la fonction principale est appelée.
  - Allouer la mémoire requise pour la fonction principale.
  - Spécifier la longueur maximale de la sortie.
  - Spécifier (pour les fonctions REAL) le nombre maximal de décimales.
  - Spécifier si le résultat peut-être NULL .

xxx_deinit() *(optionnel)*
: La terminaison de la fonction xxx() . Elle doit libérer toute la mémoire allouée par l'initialisation de la fonction.

Quand une requête SQL fait appel à XXX() , MySQL appelle l'initialisation de la fonction xxx_init() , pour laisser exécuter n'importe quelle action exigée, telle que la vérification d'arguments ou l'allocation de mémoire.

Si xxx_init() retourne une erreur, la requête SQL est annulée avec un message d'erreur et la fonction principale et la fonction de terminaison ne sont pas appelées. Autrement, la fonction principale xxx() est appelée une fois pour chaque ligne. Après que toutes les lignes aient été traitées, la fonction de terminaison xxx_deinit() est appelée pour procéder aux nettoyages requis.

Pour les fonctions d'agrégat (comme SUM()), vous pouvez également ajouter les fonctions suivantes :

xxx_reset() *(requise)*
: Remet la somme à zéro et insère l'argument en tant que valeur initiale pour un nouveau groupe.

xxx_add() *(requise)*
: Ajoute l'argument à l'ancienne somme.

Quand vous utilisez les UDF d'agrégat, MySQL opère comme suit :Toutes les fonctions doivent être compatibles avec les threads (et non pas simplement la fonction principale, mais aussi les fonctions d'initialisation et de terminaison). Cela signifie que vous ne pouvez pas allouer de variables globales ou statiques. Si vous avez besoin de mémoire, allouez-la avec la fonction xxx_init() et libérez la avec xxx_deinit() .

- Appeler xxx_init() pour laisser la fonction d'agrégat allouer la mémoire dont elle aura besoin pour stocker les résultats.
- Trier la table en accord avec la clause GROUP BY .
- Pour la première ligne dans un nouveau groupe, appeler la fonction xxx_reset() .
- Pour chaque ligne appartenant à un même groupe, appeler la fonction xxx_add() .
- Quand le groupe change ou lorsque la dernière ligne a été traitée, appeler xxx() pour obtenir le résultat de l'agrégat.
- Répéter 3-5 tant que toutes les lignes n'ont pas été traitées.
- Appeler xxx_deinit() pour libérer la mémoire allouée.

Toutes les fonctions doivent être sûrs pour les threads (pas seulement la fonction principale, mais les fonctions d'initialisation et de terminaison également). Cela signifie que vous n'êtes pas autorisés à allouer une variable globale ou statique qui change ! Si vous avez besoin de mémoire, vous devez l'allouer avec la fonction xxx_init() et la libérer avec xxx_deinit() .

23.2.2.1 Fonctions utilisateur : appeler des fonctions simples

La fonction principale doit être déclarée comme illustrée ici. Notez que le type de retour et les paramètres diffèrent, suivant que vous voulez déclarer une fonction SQL XXX() qui retournera une STRING , un INTEGER ou un REAL dans la commande CREATE FUNCTION :

Pour les fonctions de chaînes STRING :

```
char *xxx(UDF_INIT *initid, UDF_ARGS *args,
          char *result, unsigned long *length,
          char *is_null, char *error);
```

Pour les fonctions d'entiers INTEGER :

```
long long xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Pour les fonctions de nombres à virgule flottante REAL :

```
double xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Les fonctions d'initialisation et de terminaison sont déclarées comme ceci :

```
my_bool xxx_init(UDF_INIT *initid, UDF_ARGS *args, char *message);

void xxx_deinit(UDF_INIT *initid);
```

Le paramètre initid est passé aux trois fonctions. Il pointe sur une structure UDF_INIT qui est utilisée pour communiquer des informations entre les fonctions. Les membres de la structure UDF_INIT sont ceux qui sont listés ci-dessous. La fonction d'initialisation doit préparer les membres qu'elle veut, et notamment leur donner une valeur initiale (pour utiliser les valeurs par défaut, laissez le membre intact) :

my_bool maybe_null
: xxx_init() doit remplacer maybe_null par 1 si xxx() peut retourner NULL . La valeur par défaut est 1 si l'un des arguments n'est déclaré comme maybe_null .

unsigned int decimals
: Le nombre de décimales. La valeur par défaut est le nombre maximum de décimales dans l'argument passé à la fonction. Par exemple, vis la fonction reçoit 1.34 , 1.345 et 1.3 , ce nombre sera 3, car 1.345 a 3 décimales.

unsigned int max_length
: La taille maximale de la chaîne résultat. La valeur par défaut dépend du type de résultat de la fonction. Pour les fonctions de chaînes, la valeur par défaut est la taille du plus grand argument. Pour les fonctions entières, la valeur est de 21 chiffres. Pour les fonctions à nombre à virgule flottante, la valeur est de 13, plus le nombre de décimales indiquées par initid->decimals . Pour les fonctions numériques, la taille inclut le signe et le séparateur décimal.

  Si vous voulez retourner un BLOB , vous devez donner à ce membre la valeur de 65 ko ou 16 Mo; cet espace mémoire ne sera pas alloué, mais utilisé pour décider quel type de colonne utiliser, si il y a un besoin de stockage temporaire.

char *ptr
: Un pointeur que la fonction peut utiliser pour ses besoins propres. Par exemple, la fonction peut utiliser initid->ptr pour transférer de la mémoire allouée entre les trois fonctions. En xxx_init() , allouez de la mémoire, et assignez la à ce pointeur :

  ```
  initid->ptr = allocated_memory;
  ```

  En xxx() et xxx_deinit() , utilisez initid->ptr pour exploiter ou supprimer la mémoire.

23.2.2.2 Appeler des fonctions utilisateurs pour les groupements

Voici une description des différentes fonctions que vous devez définir pour réaliser des calculs sur des regroupements, avec une fonction utilisateur :

Notez que ce qui suit n'est pas demandé ou utilisé par MySQL 4.1.1. Vous pouvez conserver cette définition pour assurer la compatibilité entre MySQL 4.0 et MySQL 4.1.1.

```
char *xxx_reset(UDF_INIT *initid, UDF_ARGS *args,
                char *is_null, char *error);
```

Cette fonction est appelée lorsque MySQL trouve la première ligne dans un nouveau groupe. Dans cette fonction, vous devez remettre à zéro des variables internes de sommaire, puis indique le nouvel argument comme premier membre du nouveau groupe.Dans de nombreuses situations, cela se fait en interne en remettant à zéro toutes les variables, et en appelant xxx_add() .

Cette fonction n'est demandée que par MySQL 4.1.1 et plus récent :

```
char *xxx_clear(UDF_INIT *initid, char *is_null, char *error);
```

Cette fonction est appelée à chaque fois qu'une ligne qui appartient à un groupe est trouvée, hormis la première ligne. Durant cette fonction, vous devez ajouter les données dans votre variable interne de sommaire.

Vous pouvez utiliser le pointeur error pour stocker un octet si quelque chose n'a pas fonctionné.

```
char *xxx_add(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Cette fonction est appelée pour toutes les lignes du même groupe, hormis pour la première ligne. Dans cette dernière, vous devez ajouter la valeur dans UDF_ARGS pour vos variables internes.

La fonction xxx() doit être déclarée de manière identique à celle d'un fonction utilisateur simple. Appel des UDF .

Cette fonction est appelée lorsque toutes les lignes d'un groupe ont été traitées. Vous ne devez normalement pas accéder à la variable args ici, mais retourner votre valeur, à partir des valeurs du sommaire interne.

Tous les traitements des arguments de `xxx_reset()` et `xxx_add()` doivent être fait d'une manière similaire à celle des fonctions `UDF` normales. Récupération des arguments .

La gestion de la valeur retournée par xxx() doit être identique à celle d'une fonction utilisateur classique. Valeurs retournées et gestion d'erreurs .

Le pointeur argument de `is_null` et `error` sont les mêmes pour tous les appels de `xxx_reset()` , `xxx_add()` et `xxx()` . Vous pouvez utiliser ces valeur pour vous rappeler si vous avez rencontré une erreur, ou si la fonction `xxx()` doit retourner `NULL` . Notez que vous ne devez pas stocker de chaîne dans `*error` ! C'est un conteneur d'un seul octet!

is_null est remis à zéro pour chaque (avant d'appeler xxx_reset() . error n'est jamais remis à zéro.

Si `isnull` ou `error` sont modifiés après `xxx()` , alors MySQL va retourner `NULL` comme résultat de la fonction de groupement.

23.2.2.3 Traitement des arguments

Le paramètre `args` pointe sur une structure `UDF_ARGS` qui dispose des membre suivants :

unsigned int arg_count
: Le nombre d'arguments. Vérifiez cette valeur dans la fonction d'initialisation, si vous voulez que votre fonction soit appelée avec un nombre particulier d'arguments. Par exemple :

```
 if (args->arg_count != 2)
&#123;
    strcpy(message,"XXX() requires two arguments");
    return 1;
&#125;
```

enum Item_result *arg_type
: Le type de chaque argument. Les valeurs possibles pour chaque type sont STRING_RESULT , INT_RESULT et REAL_RESULT .

Pour s'assurer que les arguments sont d'un type donné, et retourner une erreur dans le cas contraire, vérifiez le tableau `arg_type` durant la fonction d'initialisation. Par exemple :

```
 if (args->arg_type[0] != STRING_RESULT ||
    args->arg_type[1] != INT_RESULT)
&#123;
    strcpy(message,"XXX() requires a string and an integer");
    return 1;
&#125;
```

Comme alternative à l'imposition d'un type particulier pour les arguments des fonctions, vous pouvez utiliser la fonction pour qu'elle modifie le type des arguments et donne aux valeurs de arg_type le type que vous souhaitez. Cela fait que MYSQL va forcer les arguments à un type donnée, pour chaque appel de la fonction xxx() . Par exemple, pour forcer le type des deux premiers arguments en chaîne et entier, vous pouvez utiliser la fonction d'initialisation xxx_init() :

```
 args->arg_type[0] = STRING_RESULT;
args->arg_type[1] = INT_RESULT;
```

char **args
: args->args communique les informations à la fonction d'initialisation, ainsi que la nature des arguments avec laquelle elle a été appelée. Pour un argument constant i , args->args[i] pointe sur la valeur de l'argument. Voir plus bas pour les instructions d'accès à cette valeur. Pour les valeurs non constantes, args->args[i] vaut 0 . Un argument constant est une expression qui utilise des constantes, comme 3 ou 4*7-2 ou SIN(3.14) . Un argument non-constant est une expression qui fait référence aux valeurs qui peuvent changer de ligne en ligne, par exemple des noms de colonnes ou des fonctions qui sont appelées avec des arguments non-constants.

Pour chaque invocation de la fonction principale, `args->args` contient les arguments réels qui sont passés à la ligne qui sera traitée.

Les fonctions peuvent faire référence à un argument i comme ceci :

Un argument de type STRING_RESULT est donné sous la forme d'un pointeur de chaîne, plus une longueur, pour permettre la gestion des données binaires ou des données de taille arbitraire. Le contenu des chaînes est disponible avec l'expression args->args[i] et la taille de la chaîne est args->lengths[i] . Ne supposez pas que les chaînes sont terminés par le caractère nul.

Pour un argument de type INT_RESULT , vous devez transtyper la valeur args->args[i] en valeur long long :

```
 long long int_val;
int_val = *((long long*) args->args[i]);
```

Pour un argument de type REAL_RESULT , vous devez transtyper la valeur args->args[i] en valeur double :

```
 double    real_val;
real_val = *((double*) args->args[i]);
```

unsigned long *lengths
: Pour une fonction d'initialisation, le tableau lengths indique la taille maximale des chaînes pour chaque argument. Vous ne devez pas les modifier. Pour chaque appel de la fonction principale, lengths contient la taille réelle de toutes les chaînes arguments qui sont passé pour la ligne traitée. Pour les arguments de type INT_RESULT ou REAL_RESULT , lengths contient toujours la taille maximale de l'argument (comme pour la fonction d'initialisation).

23.2.2.4 Valeurs de retour et gestion d'erreurs.

La fonction d'initialisation doit retourner 0 si aucune erreur ne s'est produite et 1 sinon. Si une erreur s'est produite, xxx_init() doit stocker un message se terminant par un NULL dans le paramètre message . Le message sera retourné au client. La taille du tampon du message est de MYSQL_ERRMSG_SIZE caractères, mais vous devez essayer de garder une taille de message inférieure à 80 caractères, sinon, il remplit la largeur d'un écran de terminal standard.

La valeur de retour de la fonction principale xxx() est la valeur de la fonction, pour les fonctions long long et double . Une fonction de chaîne de caractères doit retourner un pointeur vers le résultat et stocker la taille de la chaîne de caractères dans l'argument length .

Affectez cette valeur au contenu et à la longueur de la valeur retournée. Par exemple :

```
 memcpy(result, "chaîne retournée", 16);
*length = 16;
```

Le tampon result qui est passé à la fonction a une taille de 255 bits. Si votre résultat dépasse ceci, ne vous inquiétez pas de l'allocation de mémoire pour ce résultat.

Si votre fonction de chaînes de caractères a besoin de retourner une chaîne de caractères plus grande que 255 bits, vous devez allouer de l'espace pour cela avec malloc() dans votre fonction xxx_init() . Vous pouvez stocker la mémoire allouée dans le buffer ptr de la structure UDF_INIT pour être ré−utilisée par les appels futurs de xxx() . Appel des UDF .

Pour indiquer une valeur de retour NULL dans la fonction principale, mettez is_null à 1 :

```
 *is_null = 1;
```

Pour indiquer une erreur retournée dans la fonction principale, mettez le paramètre error à 1 :

```
 *error = 1;
```

Si xxx() met *error à 1 pour chaque ligne, la valeur de la fonction est NULL pour la ligne en question et pour chaque ligne suivante traitée par le processus dans lequel XXX() est invoqué. (xxx() ne sera même pas appelé pour les lignes suivantes.) **Remarque** : dans les versions antérieures à 3.22.10, vous devez définir *error et *is_null :

```
 *error = 1;
*is_null = 1;
```

23.2.2.5 Compiler et installer des fonctions utilisateurs

Les fichiers qui implémentent des fonctions utilisateurs doivent être compilés et installés sur le même hôte que celui du serveur. Ce processus est décrit plus bas, avec le fichier udf_example.cc qui est inclut dans les sources MySQL . Ce fichier contient les fonctions suivantes :

- metaphon() retourne la version métaphone de la chaîne en argument. C'est une technique proche du soundex, mais elle est bien plus optimisée pour l'anglais.
- myfunc_double() retourne la moyenne des codes ASCII des caractères de la chaîne passée en argument.
- myfunc_int() retourne la somme de tailles des arguments.
- sequence([const int]) retourne une séquence, commençant à partir du nombre choisit ou 1, si aucun nombre n'a été fourni.
- lookup() retourne l'adresse IP numérique d'un hôte.
- reverse_lookup() retourne le nom d'hôte pour une adresse IP. Cette fonction peut être appelée avec une chaîne au format "xxx.xxx.xxx.xxx" ou quatre nombres.

Un fichier dynamiquement chargé doit être compilé sous la forme d'un objet partagé, grâce à une commande comme celle−ci :

```
 shell> gcc −shared −o udf_example.so myfunc.cc
```

Vous pouvez facilement trouver les options correctes pour la compilation en exécutant cette commande dans le dossier sql de votre installation source :

```
 shell> make udf_example.o
```

Vous devez exécuter une commande de compilation similaire à celle que le make affiche, sauf que vous devrez supprimer l'option −c près de la fin de la ligne, et ajouter −o udf_example.so à la fin de la ligne. Sur certains systèmes, vous devrez aussi supprimer −c de la commande).

Une fois que vous compilez un objet partagés contenant des fonctions utilisateurs, vous devez les installer, et prévenir le serveur MYSQL. Compiler un objet partagé avec `udf_example.cc` produit un fichier qui s'appelle `udf_example.so` (le nom exact peut varier suivant la plate−forme). Copiez ce fichier dans l'un des dossiers utilisé par `ld` , tel que `/usr/lib` . Par exemple, `/etc/ld.so.conf` .

Sur de nombreux systèmes, vous pouvez faire pointer la variable d'environnement LD_LIBRARY ou LD_LIBRARY_PATH pour qu'elle pointe dans le dossier où vous avez vos fichiers de fonctions. Le manuel de dlopen vous indiquera quelle variable utiliser sur votre système. Vous devriez indiquer cette valeur dans les options de démarrage de mysql.server et safe_mysqld , et redémarrer mysqld .

Après que la bibliothèque ait été installée, indiquez à `mysqld` ces nouvelles fonctions avec ces commandes :

```
 mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_int RETURNS INTEGER SONAME "udf_example.so";
mysql> CREATE FUNCTION lookup RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION reverse_lookup
    ->        RETURNS STRING SONAME "udf_example.so";
mysql> CREATE AGGREGATE FUNCTION avgcost
    ->        RETURNS REAL SONAME "udf_example.so";
```

Les fonctions peuvent être effacées plus tard avec DROP FUNCTION :

```
 mysql> DROP FUNCTION metaphon;
mysql> DROP FUNCTION myfunc_double;
mysql> DROP FUNCTION myfunc_int;
mysql> DROP FUNCTION lookup;
mysql> DROP FUNCTION reverse_lookup;
mysql> DROP FUNCTION avgcost;
```

Les commandes CREATE FUNCTION et DROP FUNCTION modifient la table système func dans la base mysql . Le nom de la fonction, son type et le nom de la bibliothèque partagée sont alors sauvés dans la table. Vous devez avoir les droits de INSERT et DELETE dans la base mysql pour ajouter et effacer des fonctions.Vous ne devez pas utiliser la commande CREATE FUNCTION pour ajouter une fonction qui a déjà été créée. Si vous devez reinstaller une fonction, vous devez la supprimer avec la commande DROP FUNCTION puis la reinstaller avec CREATE FUNCTION . Vous devrez faire cela, par exemple, si vous recompilez une nouvelle version de votre fonction, pour que mysqld utilise cette nouvelle version. Sinon, le serveur va continuer à utiliser l'ancienne version.

Les fonctions actives sont rechargées à chaque fois que le serveur démarre, à moins que vous ne démarriez le serveur `mysqld` avec l'option `--skip-grant-tables` . Dans ce cas, l'initialisation des fonctions utilisateurs sont ignorées, et ces fonctions sont inutilisables. Une fonction active doit avoir été créée avec `CREATE FUNCTION` et pas supprimée avec `DROP FUNCTION` .

23.2.3 Ajouter de nouvelles fonctions natives

La procédure pour ajouter une nouvelle fonction native est décrite ici. Notez que vous ne pouvez ajouter de fonctions natives à une distribution binaire car la procédure implique la modifications du code source de MySQL. Vous devez compiler MySQL vous−même à partir d'une distribution des sources. Notez aussi que si vous migrez vers une autre version de MySQL (par exemple, quand une nouvelle version est réalisée), vous devrez répéter la procédure avec la nouvelle version.

Pour ajouter une nouvelle fonction native à MySQL, suivez les étapes suivantes :

- Ajoutez une ligne dans lex.h qui définit le nom de la fonction dans le tableau sql_functions[] .
- Si le prototype de la fonction est simple (elle prend zéro, un, deux ou trois arguments), vous devez spécifier dans lex.h SYM(FUNC_ARG#) (où # est le nombre d'arguments) en tant que second argument dans le tableau sql_functions[] et ajouter une fonction qui crée un objet fonction dans item_create.cc . Regardez "ABS" et create_funcs_abs() pour un exemple.

  Si le prototype de la fonction est compliqué (par exemple, elle prend un nombre variable d'arguments), vous devez ajouter deux lignes à `sql_yacc.yy` . Une qui indique le symbole preprocesseur que `yacc` doit définir (cela doit être ajouté au début du fichier). Puis définir les paramètres de la fonction et un ``item'' avec ces paramètres à la règle `simple_expr` . Pour un exemple, vérifiez toutes les occurrences de `ATAN` dans `sql_yacc.yy` pour voir comment cela se fait.
- Dans item_func.h , déclarez une classe héritant de Item_num_func ou de Item_str_func , selon que votre fonction retourne une nombre ou un chaîne.
- Dans le fichier item_func.cc , ajoutez l'une des déclaration suivantes, selon que vous définissez une fonction numérique ou de chaîne de caractères :

  ```
   double   Item_func_newname::val()
  longlong Item_func_newname::val_int()
  String  *Item_func_newname::Str(String *str)
  ```

  Si vous héritez votre objet de l'un des éléments standards (comme Item_num_func) vous n'aurez probablement qu'à définir l'une des fonctions décrites ci−dessus et laisser l'objet parent prendre soin des autres fonctions. Par exemple, la classe

Item_str_func définit une fonction val() qui exécute atof() sur la valeur retournée par ::str() .

- Vous devez aussi probablement définir la fonction objet suivante :

```
void Item_func_newname::fix_length_and_dec()
```

Cette fonction doit au moins calculer max_length en se basant sur les arguments donnés. max_length est le nombre maximal de caractères que la fonction peut retourner. Cette fonction doit aussi définir maybe_null = 0 si la fonction principale ne peut pas retourner une valeur NULL . La fonction peut vérifier si l'un de ses arguments peut retourner NULL en vérifiant la variable maybe_null des arguments. Vous pouvez regarder Item_func_mod::fix_length_and_dec pour avoir un exemple concret.

Toutes les fonctions doivent être sûres pour les threads (en d'autres termes, n'utilisez aucune variable statique ou globale dans la fonction sans les protéger avec mutex).

Si vous voulez retourner NULL , à partir de ::val() , ::val_int() ou ::str() vous devez mettre null_value à 1 et retourner 0.

Pour les fonctions de l'objet `::str()` , il y a d'autres considérations à prendre en compte :

- L'argument String *str fournit un tampon de chaîne qui peut être utilisé pour contenir le résultat. (Pour plus d'informations à propos du type String , regardez le fichier sql_string.h .)
- La fonction ::str() doit retourner la chaîne contenant le résultat ou (char*) 0 si celui-ci est NULL .
- Aucune des fonctions de chaînes n'essaye d'allouer de mémoire tant que ce n'est pas nécessaire !

23.3 Ajouter une nouvelle procédure à MySQL

Avec MySQL, vous pouvez définir une procédure en C++ qui accède et modifie les données dans la requête avant que celle−ci ne soit envoyée au client. La modification peut être faite ligne par ligne ou au niveau de `GROUP BY` .

Nous avons crée une procédure d'exemple avec la version 3.23 de MySQL pour vous montrer comment cela fonctionne.

De plus, nous vous recommandons de jeter un oeil à `mylua` . Avec cela, vous pouvez utiliser le langage LUA pour charger une dynamiquement une procédure dans `mysqld` .

23.3.1 La procédure Analyse

analyse([max elements,[max memory]])

Cette procédure est définie dans le fichier `sql/sql_analyse.cc` . Elle examine les résultats de vos requêtes et en retourne une analyse :

- max elements (256 par défaut) est le nombre maximal de valeurs distinctes que analyse retiendra par colonne. C'est utilisé par analyse pour vérifier si le type optimal de la colonne ne serait pas le type ENUM .
- max memory (8192 par défaut) est le maximum de mémoire que analyse doit allouer par colonne quand elle essaye de trouver toutes les valeurs distinctes.

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

23.3.2 Ecrire une procédure

Pour le moment, la seule documentation à ce sujet est constituée par les sources.

Vous pourrez trouver toutes les informations sur les procédures en examinant les fichiers suivants :

- sql/sql_analyse.cc
- sql/procedure.h
- sql/procedure.cc
- sql/sql_select.cc

24 Problèmes et erreurs communes

Ce chapitre liste quelques problèmes et erreurs communes que les utilisateurs rencontreront. Vous apprendrez à déterminer d'où vient le problème, et comment le résoudre. Vous trouverez aussi les solutions appropriées à quelques problèmes communs.

24.1 Comment déterminer ce qui pose problème

Lorsque vous faite face à un problème, la première chose à faire et de trouver quel programme / pièce de l'équipement pose ce problème :

- Si vous avez l'un des symptômes suivants, alors il est probable que cela soit un problème matériel (mémoire, carte mère, processeur, ou disque dur) ou un problème de noyau (kernel) :
  - Le clavier ne fonctionne pas. Cela peut être vérifié en pressant la touche de verrouillage des majuscules. Si la lumière des majuscules ne s'allume pas, vous devez remplacer votre clavier. (Avant de le faire, redémarrez votre ordinateur après avoir vérifié les câbles du clavier.)
  - Le curseur de la souris ne bouge pas.
  - La machine ne répond pas à un ping externe.
  - D'autres programmes ne fonctionnent pas correctement.
  - Votre système a redémarré sans que vous vous y attendiez (un programme corrompu appartenant à un utilisateur ne devrait **jamais** être capable de couper votre système).

  Dans ce cas, vous devez commencer par vérifier tout vos câbles et démarrer quelques outils de diagnostic pour vérifier votre matériel ! Vous devez aussi regarder s'il existe des patches, mises à jours, ou packs de services pour votre système d'exploitation qui pourraient résoudre votre problème. Vérifiez aussi que vos bibliothèques (comme glibc) sont à jour.

  Il est toujours bon d'utiliser une machine avec de la mémoire ECC pour découvrir les problèmes de mémoire assez tôt !
- Si votre clavier est bloqué, vous pouvez réparer cela en vous identifiant sur votre machine à partir d'une autre machine et exécutant kbd_mode -a .
- Examinez votre fichier de log système (/var/log/messages ou similaire) pour connaître les raisons de vos problèmes. Si vous pensez que le problème vient de MySQL, vous devez aussi examiner les fichiers de log de MySQL. Le log des mises à jour .
- Si vous ne pensez pas avoir de problèmes au niveau du matériel, vous devez trouver quel programme pose problème.

  Essayez en utilisant top , ps , taskmanager , ou des programmes similaires, pour voir quel programme utilise trop de ressources ou bloque la machine.
- Vérifiez avec top , df , ou un programme similaire si vous n'avez plus de mémoire, d'espace disque, trop de fichiers ouverts ou un problème avec une autre ressource critique.
- Si le problème vient d'un processus, vous pouvez toujours essayer de le terminer. S'il ne veut pas se terminer, c'est probablement un bogue du système d'exploitation.

Si après tout cela vous pensez encore que le problème vient du serveur MySQL ou du client MySQL, il est temps de préparer un rapport de bogue pour notre liste de diffusion ou notre équipe de support. Dans ce rapport, essayez de donner la description la plus détaillée possible du comportement du système et de ce que vous pensez qu'il se passe. Vous devez aussi mentionner pourquoi est-ce que vous pensez que le problème vient de MySQL. Prenez en considération toutes les situations décrites dans ce chapitre. Décrivez les problèmes exactement comme ils surviennent sur votre système. Utilisez la méthode "copier/coller" pour les affichages et les messages d'erreurs provenant des programmes ou des fichiers de log.

Essayez de décrire en détail quel est le programme qui ne fonctionne pas et tous les symptômes que vous voyez ! Nous avons déjà reçu beaucoup de rapports de bogue qui disaient juste "le système ne marche pas". Cela ne nous fournit aucune information à propos du problème.

Si un programme échoue, il est toujours utile de savoir :

- Est-ce que le programme en question a causé une erreur de segmentation (ore dump) ?
- Est-ce que le programme consomme toutes les ressources processeur ? Vérifiez avec top Laissez le programme fonctionne un bout de temps, il se peut qu'il soit entrain de traiter une tâche lourde.
- Si c'est le serveur mysqld qui pose problème, pouvez vous essayer un mysqladmin -u root ping ou mysqladmin -u root processlist ?
- Que dit un programme client (essayez avec mysql , par exemple) quand vous essayez de vous connecter au serveur MySQL ? Le programme se bloque-t-il ? Obtenez vous un retour quelconque ?

Lors de l'envoi d'un rapport de bogue, vous devez respecter les règles définies dans ce manuel. Poser des questions ou rapporter des bogues .

24.2 Erreurs communes rencontrées avec MySQL

Cette section couvre les erreurs les plus fréquemment rencontrées par les utilisateurs. Vous trouverez ici une description de ces erreurs et un moyen de les corriger.

24.2.1 Erreur Access denied

Une erreur `Access denied` peut avoir de nombreuses causes. Souvent, le problème est relié aux comptes MySQL sur le serveur, qui autorisent la connexion des clients. Causes possibles de l'erreur `Access denied` . Comment le système de droits fonctionne .

24.2.2 Erreur Can't connect to [local] MySQL server

Un client MySQL sous Unix peut se connecter au serveur `mysqld` de deux façons différentes : sockets Unix, qui se connectent via un fichier du système de fichiers (`/tmp/mysqld.sock` par défaut) ou TCP/IP, qui se connecte via un port. Les sockets Unix sont plus rapides que TCP/IP mais ne peuvent être utilisée que pour des connexions locales. Les sockets sont utilisées si vous ne spécifiez pas de nom d'hôte ou si vous spécifiez le nom d'hôte spécial `localhost` .

Sur Windows, si le serveur `mysqld` tourne sur 9x/Me, vous ne pouvez vous connecter qu'avec TCP/IP. Si le serveur tourne sur NT/2000/XP et que mysqld a été démarré avec l'option `--enable-named-pipe` , vous pouvez aussi vous connectez avec un tunnel nommé. Son nom est MySQL. Si vous ne spécifiez pas un nom d'hôte lors de la connexion à `mysqld` , un client MySQL essayera d'abord de se connecter au tunnel nommé, et si cela ne marche pas il se connectera au port TCP/IP. Vous pouvez forcer l'utilisation des tunnels nommés sous Windows en utilisant `.` en tant que nom d'hôte.

L'erreur (2002) Can't connect to ... signifie généralement qu'il n'y a aucun serveur MySQL qui tourne sur la machine ou que vous utiliser un mauvais fichier de socket ou un port erroné quand vous essayez de vous connecter au serveur mysqld .

Commencez par vérifier (en utilisant `ps` ou le gestionnaire de tâches sous Windows) qu'il y a un processus nommé `mysqld` sur votre serveur ! S'il n'y en a aucun, vous devrez en démarrer un. Lancer et résoudre des problèmes avec MySQL .

Si un processus mysqld est actif, vous pouvez tester le serveur avec l'une des connexions suivantes (le port et le chemin vers la socket peuvent être différents chez vous, bien sûr) :

```
 shell> mysqladmin version
shell> mysqladmin variables
shell> mysqladmin -h `hostname` version variables
shell> mysqladmin -h `hostname` --port=3306 version
shell> mysqladmin -h 'ip de votre hôte' version
shell> mysqladmin --socket=/tmp/mysql.sock version
```

Notez l'utilisation des guillemets obliques plutôt que les simples avec la commande hostname ; cela provoque la substitution de hostname par la valeur courante du nom d'hôte de la machine dans la commande mysqladmin .Voilà quelques raisons pouvant entraîner l'erreur Can't connect to local MySQL server :

- mysqld ne fonctionne pas.
- Vous utilisez un système qui utilise les pthreads MIT. Si vous utilisez un système qui n'a pas le support natif des threads, mysqld utilises le paquet MIT-pthreads . Plate-formes supportées par MySQL . Toutefois, toutes les versions de MIT-pthreads ne supportent pas les sockets Unix. Sur un système qui ne supporte pas les sockets vous devez toujours spécifier le nom d'hôte explicitement lors de la connexion au serveur. Utilisez cette commande pour vérifier la connexion au serveur :

  ```
  shell> mysqladmin -h `hostname` version
  ```
- Quelqu'un a effacé le fichier de socket Unix que mysqld utilise (/tmp/mysqld.sock par défaut). Vous avez peut-être une tâche cron qui efface la socket MySQL (par exemple, une tâche qui supprime les anciens fichiers du dossier /tmp). Vous pouvez toujours exécuter mysqladmin version et vérifier que la socket que mysqladmin tente d'utiliser existe vraiment. La solution dans ce cas est de modifier la tâche cron pour qu'elle n'efface plus mysqld.sock ou de placer la socket quelque part d'autre. Comment protéger /tmp/mysql.sock .
- Vous avez démarré mysqld avec l'option --socket=/chemin/vers/socket . Si vous changez le chemin vers la socket vous devez aussi en notifier les clients. Vous pouvez le faire en fournissant le chemin vers la socket en argument au client. Comment protéger /tmp/mysql.sock .
- Vous utilisez Linux et un thread s'est terminé (core dumped). Dans ce cas, vous devez aussi terminer les autres threads mysqld (par exemple, avec le script mysql_zap avant de pouvoir démarrer un nouveau serveur MySQL. Que faire si MySQL crashe constamment .
- Vous n'avez peut-être pas les privilèges de lecture et écriture sur le dossier contenant la socket ou sur la socket elle-même. Dans ce cas, vous devez changer les droits sur ce dossier / fichier ou redémarrer mysqld pour qu'il prenne en compte un dossier auquel vous avec accès.

Si vous obtenez l'erreur `Can't connect to MySQL server on un_hôte` , vous pouvez essayer ce qui suit pour trouver le problème :

- Vérifiez que le serveur fonctionne en faisant telnet votre-nom-d-hôte port-tcp-ip et pressez la touche Enter plusieurs fois. Si il y a un serveur MySQL qui tourne sur ce port, vous devriez obtenir une réponse contenant le numéro de version du serveur. Si vous obtenez une erreur proche de telnet: Unable to connect to remote host: Connection refused , c'est qu'il n'y a pas de serveur tournant sur le port donné.
- Essayez de vous connecter au démon mysqld sur la machine locale et vérifiez le port TCP/IP de la configuration de mysqld (variable port) avec mysqladmin variables .

- Vérifiez que votre serveur mysqld n'est pas configuré avec l'option --skip-networking .

24.2.3 Erreur Client does not support authentication protocol

MySQL 4.1 utilise un protocole d'identification basé sur un algorithme de hashage, qui est incompatible avec celui des anciens clients. Si vous passez d'une ancienne version en version 4.1, et que vous essayez de vous connecter au serveur avec un vieux client, vous allez rencontrer ce message d'erreur :

```
shell> mysql
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

Pour résoudre ce problème, vous devez :

- Passer tous les programmes clients en version 4.1.1, ou plus récent.
- Utiliser un compte qui a un ancien mot de passe, lorsque vous vous connectez avec un client pre−4.1.
- Remettre un mot de passe ancien format pour les clients pre−4.1 :Ceci est fait avec la commande SET PASSWORD et la fonction OLD_PASSWORD() :

  ```
  mysql> SET PASSWORD FOR
      -> 'some_user'&#64;'some_host' = OLD_PASSWORD('newpwd');
  ```

  Alternativement, utilisez UPDATE et FLUSH PRIVILEGES :

  ```
  mysql> UPDATE mysql.user SET Password = OLD_PASSWORD('newpwd')
      -> WHERE Host = 'some_host' AND User = 'some_user';
  mysql> FLUSH PRIVILEGES;
  ```
- Dire au serveur qu'il doit utiliser l'ancien algorithme de hashage :
  - Démarrez mysqld avec --old-passwords .
  - Donnez un mot de passe à tous les utilisateurs qui ont un hash long de mot de passe. Vous pouvez les trouver comme ceci :

    ```
    SELECT * FROM mysql.user WHERE LENGTH(password) > 16;
    ```

Pour plus d'information sur l'identification et le hashage, voyez Chiffrement des mots de passe en MySQL 4.1 .

24.2.4 Echec de saisie du mot de passe avec le client interactif

Les clients MySQL demandent le mot de passe en ligne de commande lorsque l'option --password ou -p n'a pas de valeur :

```
shell> mysql -u user_name -p
Enter password:
```

Sur certains systèmes, vous vous apercevrez que votre mot de passe fonctionne lorsqu'il est spécifié dans un fichier de configuration, ou bien en ligne de commande, mais pas interactivement, lorsque l'invite Enter password: est proposée. Cela survient lorsque la bibliothèque système limite la taille des mots de passe à 8 caractères (généralement). C'est un problème lié à la bibliothèque système, et non à MySQL. Pour pallier ce problème, changez votre mot de passe MySQL pour qu'il fasse 8 ou moins de caractères, ou bien placez votre mot de passe dans votre fichier d'options.

24.2.5 Erreur Host '...' is blocked

Si vous obtenez cette erreur :

```
Host 'hostname' is blocked because of many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

cela veut dire que mysqld a reçu trop de (max_connect_errors) tentatives de connexions à l'hôte 'hostname' qui ont été interrompus en plein milieu. Après max_connect_errors requêtes échouées, mysqld pense qu'il se passe quelque chose de mauvais (comme une attaque de la part d'un pirate), et bloque le serveur pour les prochaines connexions jusqu'à ce que quelqu'un exécute la commande mysqladmin flush-hosts . Variables sytème du serveur .Par défaut, mysqld bloque le serveur après 10 connexions erronées. Vous pouvez facilement changer ce comportement en démarrant le serveur avec ces arguments :

```
shell> safe_mysqld -O max_connect_errors=10000 &
```

Notez que si vous recevez ce message pour un hôte en particulier, vous devriez vous assurer qu'il n'y a pas de problèmes de connexions TCP/IP depuis cet hôte. Si vos connexions TCP/IP ne marchent pas, il ne servira à rien d'augmenter la valeur de la variable max_connect_errors !

24.2.6 Erreur Too many connections

Si vous obtenez l'erreur `Too many connections` en essayant de vous connecter à MySQL, cela signifie qu'il y a déjà `max_connections` clients connectés au serveur `mysqld` .

Si vous avez besoin de plus de connexion que par défaut (100), vous devez redémarrer `mysqld` avec une plus grande valeur pour la variable `max_connections` .

Notez que mysqld permet actuellement à (max_connections +1) clients de se connecter. La dernière connexion est réservée à l'utilisateur ayant le privilège SUPER . En ne donnant pas ce privilège aux utilisateurs normaux (ils ne devraient pas en avoir besoin), un administrateur avec ce privilège peut se connecter et utiliser SHOW PROCESSLIST pour trouver ce qui pose problème. Syntaxe de SET et SHOW .

Le nombre maximal de connexion MySQL dépend de la qualité de la bibliothèque des threads sur une plate−forme donnée. Linux et Solaris devraient être capables de supporter jusqu'à 500−1000 connexion simultanées, cela dépend évidemment de la quantité de RAM que vous avez et de ce que font les clients.

24.2.7 Erreur Out of memory

Si vous lancez une requête et que vous obtenez l'erreur suivante :

```
 mysql: Out of memory at line 42, 'malloc.c'
mysql: needed 8136 byte (8k), memory in use: 12481367 bytes (12189k)
ERROR 2008: MySQL client ran out of memory
```

notez que cette erreur parle du client MySQL client mysql . La raison de cette erreur est simplement que le client n'a pas la mémoire suffisante pour stocker le résultat en entier.Pour remédier à ce problème, vérifiez d'abord que votre requête est correcte. Est-ce normal qu'elle retourne autant de lignes? Si oui, vous pouvez utiliser mysql −−quick , qui utilise mysql_use_result() pour récupérer les résultats. Cela sollicitera moins le client (mais plus le serveur).

24.2.8 Erreur MySQL server has gone away

Cette section couvre aussi l'erreur `Lost connection to server during query` .

Le plus souvent, l'erreur MySQL server has gone away se produit lorsque le serveur a dépassé le délai d'attente d'actions et a clos la connexion. Par défaut, le serveur clos la connexion après 8 heures si rien n'est arrivé. Vous pouvez changer cette limite en configurant le paramètre wait_timeout lorsque vous démarrez mysqld .

Une autre raison de recevoir l'erreur `MySQL server has gone away` est d'avoir emis un ``close'' sur votre connexion puis d'avoir essayé d'actionner une autre commande alors que la connexion était close.

Si vous avez un script, vous n'avez qu'à lancer la requête à nouveau pour que le client se reconnecte automatiquement.

Vous obtiendrez normalement les codes erreurs suivants dans ce cas (qui est indépendant du système d'exploitation) :

Code erreur	**Description**
CR_SERVER_GONE_ERROR	Le client ne peut envoyer de commandes au serveur.
CR_SERVER_LOST	Le client n'a pas obtenu d'erreur en contactant le serveur, mais n'a pas obtenu de réponse complète à la question posée.

Par défaut, le serveur ferme la connexion après 8 heures si rien ne se passe durant ce temps. Vous pouvez modifier la limite de temps en changeant la variable `wait_timeout` lorsque vous lancez `mysqld` . Variables sytème du serveur .

Si vous avez un script, il suffit d'émettre une nouvelle requête pour que le client se reconnecte automatiquement.

Vous obtiendrez aussi cette erreur si quelqu'un à terminé le processus avec `kill #idprocessus#` .

Vous pouvez vérifier si le serveur MySQL est encore en marche en exécutant mysqladmin version et examinant la date de mise en route. Si le problème est que mysqld a crashé, vous devriez vous concentrer sur la résolution du problème. Vous devez dans ce cas commencer par vérifier si emmètre la même requête fera à nouveau crasher MySQL. Que faire si MySQL crashe constamment .

Vous pouvez aussi obtenir ces erreurs si vous envoyez une requête incorrecte ou trop grande au serveur. Si `mysqld` reçoit un paquet trop large ou mal ordonné, il suppose que quelque chose s'est mal passé au niveau du client et ferme la connexion. Si vous avez besoin de grande requêtes (par exemple, si vous travaillez avec de grandes colonnes `BLOB`) vous pouvez augmenter la taille limite des requêtes en démarrant `mysqld` avec l'option `-O max_allowed_packet=#` (1M par défaut). Le surplus de mémoire est alloué à la demande, ce qui fait que `mysqld` n'utilisera de la mémoire que lorsque vous emmétrez une grande requête ou qu'il aura à retourner de grandes réponses ! Plus d'informations sur la configuration de la taille des paquets sont disponibles dans la section Erreur `Packet too large` .

Si vous voulez rapporter un bogue concernant ce problème, merci d'inclure les informations suivantes :

- MySQL a-t-il planté ? (Vous pouvez le savoir en regardant le fichier hostname.err . Que faire si MySQL crashe constamment .
- Si une requête spécifique fait planter mysqld et que les tables concernées ont bien été vérifiées avec CHECK TABLE avant l'exécution, pouvez-vous faire une batterie de tests ? Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .
- Quelle est la valeur de la variable wait_timeout dans le serveur MySQL ? mysqladmin variables vous donnera une réponse
- Avez-vous essayé de démarrer mysqld avec --log et vérifié si la requête apparaît bien dans le log ?

Poser des questions ou rapporter des bogues .

24.2.9 Erreur Packet too large

Lorsqu'un client MySQL ou le serveur `mysqld` reçoit un paquet plus grand que `max_allowed_packet` octets, il provoque une erreur `Packet too large` et ferme la connexion.

En MySQL 3.23 le plus gros paquet possible est 16M (à cause des limites du protocole client/serveur). En MySQL 4.0.1 et plus, cela n'est plus limité que par la quantité de mémoire que vous avez sur votre serveur (cela va théoriquement à un maximum de 2G).

Un paquet de communication est une simple commande SQL envoyée au serveur, ou une simple ligne renvoyée au client.

Lorsqu'un client MySQL ou que le serveur `mysqld` reçoit un paquet plus grand que `max_allowed_packet` octets, il provoque une erreur `Packet too large` et ferme la connexion. avec quelques clients, vous pouvez aussi obtenir l'erreur `Lost connection to MySQL server during query` si le paquet est trop grand.

Notez que le client et le serveur ont chacun leur propre variable max_allowed_packet . Si vous voulez gérer les gros paquets, vous devrez changer cette variable côté client et côté serveur.

Si utilisez le client `mysql` , la variable `max_allowed_packet` vaut par défaut 16Mo. C'est aussi la valeur maximale avant MySQL 4.0. Pour lui donner une valeur maximale plus grande, depuis 4.0, lancez `mysql` comme ceci :

```
mysql> mysql --max_allowed_packet=32M
```

Cela fixe une taille de paquet à 32Mo.

La valeur par défaut de `max_allowed_packet` est 1Mo. Vous pouvez augmenter cette valeur si le serveur doit gérer de grosses requêtes (par exemple, si vous travaillez avec de grandes colonnes `BLOB`). Par exemple, pour modifier la variable en lui donnant une valeur de 16 MO, lancez le serveur comme ceci :

```
mysql> mysqld --max_allowed_packet=16M
```

Avant MySQL 4.0, utilisez cette syntaxe :

```
mysql> mysqld --set-variable=max_allowed_packet=16M
```

Vous pouvez aussi utiliser le fichier d'options pour spécifier la valeur de max_allowed_packet . Par exemple, pour spécifier une valeur de 16MO, ajoutez la ligne suivante dans le fichiers d'options :

```
 [mysqld]
max_allowed_packet=16M
```

Avant MySQL 4.0, utilisez cette syntaxe :

```
 [mysqld]
set-variable = max_allowed_packet=16M
```

Il n'est pas dangereux d'augmenter cette valeur étant donné que la mémoire n'est alloué que lorsque besoin en est. Cette variable est plus une précaution pour capturer les paquets erronés qui circulent entre le client et le serveur. Elle sert aussi a vous assurer que vous n'utilisez pas accidentellement de gros paquets qui consommeront toute la mémoire.Si vous utilisez le client mysql , vous pouvez spécifier un plus grand tampon en démarrant le client avec mysql --set-variable=max_allowed_packet=8M . Les autres clients ont différentes méthodes pour configurer cette variable. Notez que --set-variable est désapprouvée depuis MySQL 4.0, utilisez --max-allowed-packet=8M à la place. Vous pouvez utiliser le fichier d'options pour augmenter la taille de max_allowed_packet dans mysqld . Par exemple, si vous vous attendez à stocker la totalité d'un MEDIUMBLOB dans une table, vous aurez besoin de démarrer le serveur avec l'option set-variable=max_allowed_packet=16M .

Vous pouvez aussi rencontrer d'étranges problèmes avec les gros paquets si vous utilisez les grands blob, mais que vous n'avez pas donné à `mysqld` l'accès à assez de mémoire pour gérer ces requêtes. Si vous pensez être dans ce cas, essayez d'ajouter `ulimit -d 256000` au début du script `safe_mysqld` et redémarrez `mysqld` .

24.2.10 Erreurs de communication / Connexion annulée

A partir de `MySQL 3.23.40` vous n'obtenez l'erreur `Aborted connection` que si vous démarrez `mysqld` avec `--warnings` .

Si vous trouvez des erreurs comme celle qui suit dans vos logs d'erreurs :

```
 010301 14:38:23  Aborted connection 854 to db: 'users' user: 'josh'
```

Le log d'erreurs .Cela signifie qu'un problème est survenu :

- Le programme client n'a pas appelé mysql_close() avant de quitter.
- Le client a été inactif plus de wait_timeout ou interactive_timeout secondes sans aucune requête. Variables sytème du serveur .
- L'exécution du programme client s'est terminée soudainement au milieu d'un transfert.

Lorsque ce qui précède arrive, la variable `Aborted_clients` est incrémentée.

La variable serveur Aborted_connects est incrémentée lorsque :

- Un paquet de connexion ne contient pas la bonne information.
- L'utilisateur n'avait pas les droits d'accès à la base de données.
- L'utilisateur a utilisé un mot de passe erroné.
- Il a fallu plus de connect_timeout secondes pour obtenir un paquet de communication. Variables sytème du serveur .

Notez que ce qui précède peut indiquer que quelqu'un essaye de s'introduire dans votre base de données !

Autres raisons pour les problèmes de clients échoués / connexions interrompues :

- L'utilisation du protocole Ethernet sous Linux, que le Duplex soit intégral (Full-Duplex) ou partiel (Half Duplex) avec Linux. La plupart des pilotes Ethernet de Linux ont ce bogue. Vous pouvez tester ce bogue en transférant un énorme fichier via FTP d'une machine à l'autre. Si le transfert est saccadé (succession de transfert-pause-transfert-pause) alors vous vivez le syndrôme du duplex de Linux. La seule solution est de changer le mode de Duplex sur les deux cartes réseau et Hub/Switch en duplex partiel ou duplex intégral et de faire des tests pour connaître la meilleure configuration.
- Quelques problèmes avec la bibliothèque de threads qui causent des interruptions de lectures.
- Mauvaise configuration TCP/IP.
- Les câbles, concentrateurs ou commutateurs Ethernet défectueux. On peut le diagnostiquer aisément en remplacement le matériel.
- max_allowed_packet est trop petit ou les requêtes ont besoin de plus de mémoire que celle que vous avez alloué à mysqld . Erreur Packet too large .

24.2.11 Erreur The table is full

Il y a différents cas où vous pouvez obtenir cette erreur :

- Vous utilisez une ancienne version de MySQL (avant 3.23.0) quand une table temporaire en mémoire devient plus grande que tmp_table_size octets. Pour éviter ce problème, vous pouvez utiliser l'option -O tmp_table_size=# pour faire augmenter la taille des tables temporaires à mysqld ou utiliser l'option SQL BIG_TABLES avant d'exécuter la requête qui pose problème. Syntaxe de SET .

  Vous pouvez aussi démarrer `mysqld` avec l'option `--big-tables` . Cela revient à utiliser `BIG_TABLES` pour toutes les requêtes.

  Dans la version 3.23 de MySQL, les tables temporaires en mémoire seront automatiquement changées en tables physique MyISAM après qu'elles n'aient dépassé tmp_table_size .

- Vous utilisez des tables InnoDB et avez dépassé leur taille. Dans ce cas, la solution est d'augmenter les tailles des tables. Ajout et suppression des fichiers de logs et de données InnoDB .
- Vous utilisez des tables ISAM ou MyISAM sur un système d'exploitation qui ne supporte pas les fichiers de plus de 2G et vous avez atteint cette limite dans le fichier de données ou d'index. Variables sytème du serveur .
- Vous utilisez des tables MyISAM et la taille des données ou de l'index est plus grande que celle que MySQL a alloué aux pointeurs. (Si vous ne spécifiez pas MAX_ROWS à CREATE TABLE MySQL n'allouera que des pointeurs supportant 4G de données).Vous pouvez obtenir la taille maximale des données / index en faisant :

```
SHOW TABLE STATUS FROM database LIKE 'nom_de_table';
```

  or using myisamchk -dv database/nom_de_table .Si le problème vient de là, vous pouvez le corriger en faisant quelque chose se rapprochant de :

```
ALTER TABLE nom_de_table MAX_ROWS=1000000000 AVG_ROW_LENGTH=nnn;
```

  Vous n'avez besoin de spécifier AVG_ROW_LENGTH que pour les tables avec des champs BLOB/TEXT car dans ce cas, MySQL ne peut optimiser l'espace requis en se basant uniquement sur le nombre de lignes.

24.2.12 Erreur Can't create/write to file

Si vous obtenez une erreur de ce type pour quelques requêtes :

```
Can't create/write to file '\\sqla3fe_0.ism'.
```

cela signifie que MySQL ne peut créer de fichier temporaire pour le jeu de résultats dans le dossier temporaire défini. (L'erreur précédente est typique de Windows, et le message d'erreur Unix est similaire.) La solution est de démarrer mysqld avec l'option --tmpdir=chemin ou d'ajouter à votre fichier d'options. Par exemple, pour spécifier le dossier C:\temp , utilisez ces lignes :

```
[mysqld]
tmpdir=C:/temp
```

Le dossier C:\temp doit exister au préalable. Utiliser les fichiers d'options .

Vérifiez aussi le code erreur que vous obtenez avec perror . Une autre raison peut être une erreur de disque saturé.

```
shell> perror 28
Error code  28:  No space left on device
```

24.2.13 Erreur du client Commands out of sync

Si vous obtenez l'erreur Commands out of sync; you can't run this command now , le problème vient du fait que vous appelez les fonctions dans le mauvais ordre dans votre code !

Cela peut se produire, par exemple, si vous utilisez mysql_use_result() et essayez d'exécuter une nouvelle requête avant d'avoir appelé mysql_free_result() . Cela peut aussi se produire si vous essayez d'exécuter deux requêtes qui retournent des données dans appeler mysql_use_result() ou mysql_store_result() entre les deux.

24.2.14 Erreur Ignoring user

Si vous obtenez l'erreur suivante :

```
Found wrong password for user: 'some_user@some_host'; ignoring user
```

cela signifie que lors du démarrage de mysqld ou lorsqu'il a rechargé les tables de permissions, il a trouvé une entrée dans la table user avec un mot de passe invalide. De ce fait, l'entrée est tout simplement ignorée par le système de droits.

Causes possibles et solutions pour ce problème :

- Vous faites peut-être tourner une nouvelle version de mysqld avec une vielle table user . Vous pouvez vérifier cela en exécutant mysqlshow mysql user pour voir si le champ du mot de passe est plus petit que 16 caractères. Si c'est le cas, vous pouvez le corriger en exécutant le script scripts/add_long_password .
- L'utilisateur a un ancien mot de passe (8 caractères) et vous n'avez pas démarré mysqld avec l'option --old-protocol . Mettez à jour le mot de passe dans la table user ou redémarrez mysqld avec --old-protocol .
- Vous avez spécifié un mot de passe dans la table user sans passer par la fonction PASSWORD() . Utilisez mysql pour mettre à jour l'utilisateur dans la table user avec un nouveau mot de passe. Assurez-vous d'utiliser la fonction PASSWORD() :

```
mysql> UPDATE user SET password=PASSWORD('votre mot de passe')
    ->             WHERE user='XXX';
```

24.2.15 Erreur Table 'xxx' doesn't exist

Si vous obtenez l'erreur suivante, cela signifie qu'aucune table portant le nom xxx n'existe dans la base de données courante.

```
 Table 'tbl_name' doesn't exist
Can't find file: 'tbl_name' (errno: 2)
```

Dans certains cas, les tables pourraient exister, mais vous n'utilisez pas les références correctes :

- Comme MySQL utilise des dossiers et des fichiers pour stocker les bases de données et les tables, les noms sont sensibles à la casse si le serveur fonctionne sur un système d'exploitation qui est sensible à la casse.
- Même pour les systèmes de fichiers qui ne sont pas sensibles à la casse, comme sur Windows, toutes les références dans une requêtes doivent être faîtes dans la même casse.

Vous pouvez vérifiez les tables disponibles avec la commande SHOW TABLES . Syntaxe de SET et SHOW .

24.2.16 Erreur Can't initialize character set xxx

Si vous obtenez l'erreur suivante :

```
 MySQL Connection Failed: Can't initialize character set xxx
```

Cela signifie l'une des choses suivantes :

- Le jeu de caractères est un jeu multi−octets et votre client ne les supporte pas.Dans ce cas, vous devez recompiler le client avec --with-charset=xxx ou avec --with-extra-charsets=xxx . Les options habituelles de configure .

  Tous les binaires standards de MySQL sont compilés avec --with-extra-character-sets=complex qui active le support de tous les jeux de caractères multi−octets. Le jeu de caractères utilisé pour les données et le stockage .
- Le jeu de caractères est un simple jeu de caractères qui n'est pas compilé dans mysqld et les fichiers de définition du jeu ne sont pas à l'endroit où le client si attend.

  Dans ce cas vous avez besoin de :

  - Recompiler le client avec le support du jeu de caractères. Les options habituelles de configure .
  - Spécifier au client où les fichiers de définition du jeu de caractères se situent. Pour beaucoup de clients, vous pouvez le faire avec l'option --character-sets-dir=chemin-vers-dossier-jeu-caractères .
  - Copier les fichiers de définition du jeu de caractères dans le dossier où le client s'attend à les trouver.

24.2.17 Fichier non trouvé

Si vous obtenez ERROR '...' not found (errno: 23) , Can't open file: ... (errno: 24) , ou toute autre erreur avec errno 23 ou errno 24 de la part de MySQL, cela signifie que vous n'avez pas alloué assez de descripteurs de fichiers à MySQL. Vous pouvez utiliser l'utilitaire perror pour obtenir une description de ce que veut dire l'identifiant de l'erreur :

```
 shell> perror 23
File table overflow
shell> perror 24
Too many open files
shell> perror 11
Resource temporarily unavailable
```

Le problème ici est que mysqld essaye de garder trop de fichiers ouverts en même temps. Vous pouvez soit demander à mysqld de ne pas ouvrir autant de fichiers simultanément ou augmenter le nombre de descripteurs de fichiers alloués à mysqld .

Pour dire à mysqld de garder moins de fichiers ouverts en même temps, vous pouvez rendre le cache de tables plus petit en utilisant l'option -O table_cache=32 de safe_mysqld (la valeur par défaut est 64). Réduire la valeur de max_connections réduira aussi le nombre de fichiers ouverts (90 comme valeur de défaut).

Pour changer le nombre de descripteurs de fichiers alloués à mysqld , vous pouvez utiliser l'option --open-files-limit=# de safe_mysqld ou -O open-files-limit=# de mysqld . Syntaxe de SHOW VARIABLES . La façon la plus facile de faire cela est d'ajouter cette option dans votre fichiers d'options. Utiliser les fichiers d'options . Si vous avec une ancienne version de mysqld qui ne le supporte pas, vous pouvez éditer le script safe_mysqld . Il y a une ligne commentée ulimit -n 256 dans le script. Vous pouvez enlever le caractère '#' pour décommenter cette ligne, et changer le nombre 256 pour affecter le nombre de descripteurs de fichiers alloués à mysqld .

`ulimit` (et `open-files-limit`) peuvent augmenter le nombre de descripteurs de fichiers, mais seulement jusqu'à la limite imposée par le système d'exploitation. Il y a aussi une limite 'matérielle' qui ne peut être dépassée que si vous démarrez `safe_mysqld` ou `mysqld` en tant que root (souvenez−vous juste que vous devez aussi utiliser l'option `--user=...` dans ce cas). Si vous avez besoin de repousser les limites du système d'exploitation pour les descripteurs de fichiers disponibles pour chaque processus, consultez la documentation de votre système.

Note : Notez que si vous démarrez le Shell tcsh , ulimit ne fonctionnera pas ! tcsh retournera aussi des valeurs incorrectes si vous atteignez la limite courante ! Dans ce cas, vous devez démarrer safe_mysqld avec sh !

24.3 Notes relatives à l'installation

24.3.1 Problèmes lors de la liaison avec la bibliothèque du client MySQL

Si vous liez votre programme et que vous obtenez des erreurs pour des symboles non-référencés qui commencent par mysql_ , comme ce qui suit :

```
 /tmp/ccFKsdPa.o: In function `main':
/tmp/ccFKsdPa.o(.text+0xb): undefined reference to `mysql_init'
/tmp/ccFKsdPa.o(.text+0x31): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x57): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x69): undefined reference to `mysql_error'
/tmp/ccFKsdPa.o(.text+0x9a): undefined reference to `mysql_close'
```

vous pouvez réparer cela en ajoutant -Ldir_path-lmysqlclient dans votre ligne de liaison. Pour déterminer le dossier correct, utilisez cette commande :

```
shell> mysql_config --libs
```

Si vous obtenez une erreur `undefined reference` pour la fonction `uncompress` ou `compress` , ajoutez `-lz` à la fin de votre ligne de liaison et essayez à nouveau !

Si vous obtenez des erreurs undefined reference pour des fonctions qui devraient exister sur votre système, comme connect , vérifiez la page de manuel de la fonction en question, pour les bibliothèques que vous devez ajouter à la ligne de liaison !

Si vous obtenez une erreur `undefined reference` pour des fonctions inexistantes sur votre système, ressemblant à ce qui suit :

```
mf_format.o(.text+0x201): undefined reference to `__lxstat'
```

cela signifie que votre bibliothèque est compilé sur un système qui n'est pas à 100% compatible avec le votre. Dans ce cas, vous devez obtenir la dernière distribution des sources de MySQL et compiler vous-mêmes. Installer MySQL à partir des sources .

Si vous essayez de faire fonctionner un programme et que vous obtenez des erreurs pour des symboles non−référencés qui commencent par `mysql_` ou une erreur disant que la bibliothèque `mysqlclient` ne peut être trouvée, cela signifie que votre système n'arrive pas à trouver la bibliothèque partagée `libmysqlclient.so` .

La solution est de dire à votre système de chercher les bibliothèques partagées là où la bibliothèque est située avec l'une des méthodes suivantes :

- Ajouter le chemin vers le dossier où se situe libmysqlclient.so dans la variable d'environnement LD_LIBRARY_PATH .
- Ajouter le chemin vers le dossier où se situe libmysqlclient.so dans la variable d'environnement LD_LIBRARY .
- Copiez le fichier libmysqlclient.so à un endroit où votre système le cherche, comme dans le dossier /lib , et mettez à jour les informations de la bibliothèque partagée en exécutant ldconfig .

Un autre moyen de résoudre ce problème est de lier votre programme statiquement, avec -static , ou en effaçant les bibliothèques dynamiques de MySQL avant de lier votre code. Dans le second cas vous devez vous assurer qu'aucun autre programme n'utilise les bibliothèques dynamiques !

24.3.2 Comment exécuter MySQL comme un utilisateur normal

Le serveur MySQL mysqld peut être démarré par n'importe quel utilisateur. Afin de changer l'utilisateur qui fait tourner mysqld en l'utilisateur Unix nom_utilisateur , vous devez faire ceci :

- Stoppez le serveur si il fonctionne (utilisez mysqladmin shutdown).
- Changez le propriétaire du dossier et des fichiers de bases pour qu'il soit nom_utilisateur . Il faut que cet utilisateur ait les droits d'écriture et de lecture (vous pourriez avoir à faire cette manipulation en tant que root Unix) :
  ```
  shell> chown −R nom_utilisateur /path/to/mysql/datadir
  ```
  Si les dossier ou les fichiers de données de MySQL sont des liens symboliques, vous devez vous assurer de pouvoir suivre ces lignes, et de changer les propriétaires des fichiers et dossiers sur lesquels ils pointent. L'option chown −R de chown peut ne pas suivre les liens symboliques.
- Démarrez le serveur avec l'utilisateur nom_utilisateur , ou bien, si vous utilisez MySQL version 3.22 ou plus récent, démarrez mysqld en tant que root Unix, et utilisez l'option −−user=nom_utilisateur . mysqld va alors changer automatiquement d'utilisateur pour utiliser nom_utilisateur avant d'accepter les connexions.
- Pour démarrer le serveur sous le nom d'utilisateur automatiquement au moment du démarrage du système, ajouter une ligne user qui spécifie le nom de l'utilisateur que le groupe de [mysqld] est du même groupe que le fichier d'options /etc/my.cnf ou le fichier d'options my.cnf dans le dossier de données du serveur. Par exemple :
  ```
  [mysqld]
  user=nom_utilisateur
  ```

A ce moment, votre processus mysqld doit fonctionner normalement sous le nom de l'utilisateur Unix nom_utilisateur . Une chose n'a pas changé : les droits dans les tables de droits de MySQL. Par défaut, juste après avoir exécuté le script d'installation des tables de droits mysql_install_db , l'utilisateur MySQL root est le seul utilisateur du système avec les droits de créer et de détruire les bases. A moins que vous n'ayez changé ces droits, ils ont toujours cours. Cela ne va pas vous empêcher d'accéder à MySQL en tant que root MySQL, même si vous n'êtes pas connecté en tant que root Unix. Spécifiez simplement l'option −u root au programme client.

Notez qu'accéder à MySQL en tant que root , en fournissant l'option −u root en ligne de commande, n'a rien a voir avec MySQL qui fonctionne avec les droits de root Unix, ou d'un autre utilisateur Unix. Les droits d'accès et les noms d'utilisateurs MySQL sont complètement séparé des noms d'utilisateurs et des mots de passes Unix. Le seul rapport avec les utilisateurs Unix est que si vous ne fournissez pas l'option −u lorsque vous démarrez votre client, le client va essayer de se connecter à MySQL avec votre nom d'utilisateur Unix.

Si votre serveur Unix n'est pas sécurisé, il est recommandé de donner un mot de passe à l'utilisateur MySQL root dans les tables de droits. Sinon, n'importe quel utilisateur ayant un compte sur cette machine va pouvoir accéder au compte root avec l'option mysql −u root nom_base et faire ce qu'il veut. Configuration et tests consécutifs à l'installation .

24.3.3 Problèmes avec les permissions sur fichiers

Si vous avez des problèmes avec les droits sur fichiers, par exemple, si mysql génère l'erreur suivante lorsque vous créez une table :

```
ERROR: Can't find file: 'path/with/filename.frm' (Errcode: 13)
```

alors la variable d'environnement UMASK est peut-être mal configurée lorsque mysqld est démarré. La valeur par défaut de umask est 0660 . Vous pouvez corriger ce comportement en démarrant safe_mysqld de la façon suivante :

```
shell> UMASK=384  # = 600 en octal
shell> export UMASK
shell> /chemin/vers/safe_mysqld &
```

Par défaut, MySQL créera les dossiers des bases de données et de RAID avec 0700 comme type de permissions. Vous pouvez modifier ce comportement en définissant la variable UMASK_DIR . Si vous le faite, les nouveaux dossiers seront crées en combinant UMASK et UMASK_DIR . Par exemple, si vous voulez donner un accès de groupe à tout les nouveaux dossiers, vous pouvez faire :

```
shell> UMASK_DIR=504  # = 770 en octal
shell> export UMASK_DIR
shell> /chemin/vers/safe_mysqld &
```

A partir de la version 3.23.25, MySQL suppose que les valeurs de UMASK et UMASK_DIR sont en octal si elles commencent par un zéro.

Variables d'environnement MySQL .

24.4 Notes relatives à l'administration

24.4.1 Comment réinitialiser un mot de passe Root oublié

Si vous n'avez jamais configuré un mot de passe root pour MySQL, le serveur n'en demendera jamais un pour toutes les connexions de cet utilisateur. Il est recommandé de toujours assigner un mot de passe à chaque utilisateur. Problèmes généraux de sécurité .

Si vous avez configuré un mot de passe pour l'utilisateur root , mais que vous l'avez oublié, vous pouvez en choisir un nouveau en suivant la procédure suivante :

La procédure sous Windows :

- Identifiez vous sur le système en tant qu'administrateur.
- Stoppez le serveur MySQL s'il fonctionnait. Pour un serveur en fonctionnement en tant que service Windows, il faut aller dans le gestionnaire de services :

  ```
  Start Menu -> Control Panel -> Administrative Tools -> Services
  ```

  Puis, trouver le service MySQL dans la liste, et arrêtez le.Si votre serveur ne fonctionne pas comme un service, essayer d'utiliser le gestionnaire de tâches pour l'arrêter.
- Ouvrez une fenêtre de console DOS :

  ```
  Start Menu -> Run -> cmd
  ```

- Nous supposerons ici que vous avez installé MySQL dans le dossier C:\mysql . Si vous l'avez installé ailleurs, ajustez les commandes.A l'invite de commandes, exécutez cette commande :

  ```
  C:\> C:\mysql\bin\mysqld-nt --skip-grant-tables
  ```

  Cela va relancer le serveur dans un mode spécial, qui ne vérifie pas les droits dans les tables.
- Gardez la première console ouverte, et ouvrez-en une seconde, et exécutez la commande suivante (une commande par ligne) :

  ```
  C:\> C:\mysql\bin\mysqladmin -u root
          flush-privileges password "newpwd"
  C:\> C:\mysql\bin\mysqladmin -u root -p shutdown
  ```

  Remplacez `` newpwd '' par le mot de passe root que vous voulez utiliser. La seconde commande va vous demander d'enter le mot de passe pour identification d'accès. Entrez le mot de passe que vous avez assigné dans la première commande.
- Stoppez le serveur MySQL, et relancez le comme d'habitude. Si vous lancez MySQL comme un service, démarrez le depuis le gestionnaire de services. Si vous le lancez manuellement, utilisez votre commande habituelle.
- Vous devriez pouvoir vous connecter en utilisant le nouveau mot de passe.

Dans un environnement Unix, la procédure pour redéfinir le mot de passe root est le suivant :

- Connectez vous sur votre système en tant que root Unix, ou avec le compte qui fait tourner le démon mysqld .
- Repérez le fichier .pid qui contient l'identifiant du processus du serveur. Le chemin et le nom exact de ce fichier dépendent de votre distribution, nom de serveur et configuration. Les chemins classiques sont : /var/lib/mysql/ , /var/run/mysqld/ et /usr/local/mysql/data/ . Généralement, le nom du fichier est suivi de l'extension .pid et commence avec mysqld ou le nom de votre serveur.Terminez le serveur mysqld en lui envoyant une commande kill (pas un kill -9), en utilisant le numéro d'identifiant de processus que vous venez de lire dans le fichier .pid .

  ```
  shell> kill `cat /dossier-donnees-mysql/hote.pid`
  ```

  Vous devez être l'utiliateur Unix root ou l'utilisateur qui fait tourner mysqld pour pouvoir le faire.
- Redémarrez mysqld avec l'option --skip-grant-tables .

  ```
  shell> mysqld_safe --skip-grant-tables &
  ```

- Choisissez un nouveau mot de passe avec la commande mysqladmin password :

  ```
  shell> mysqladmin -u root password 'nouveaumotdepasse'
  ```

  Remplacez `` nouveaumotdepasse '' avec le mot de passe root que vous souhaitez.
- Après cela, vous devriez pouvoir vous connecter avec le nouveau mot de passe.

Alternativement, sur toutes les plate-formes, vous pouvez aussi choisir le nouveau mot de passe en utilisant le client mysql :

- Stoppez et redémarrez mysqld avec l'option --skip-grant-tables comme décrit plus haut.
- Connectez vous au serveur mysqld avec :

  ```
  shell> mysql -u root mysql
  ```

- Exécutez la commande suivante dans le client mysql :

  ```
  mysql> UPDATE user SET Password=PASSWORD('nouveaumotdepasse')
      ->             WHERE User='root';
  mysql> FLUSH PRIVILEGES;
  ```

- Après cela, vous devriez pouvoir vous connecter avec le nouveau mot de passe.

24.4.2 Que faire si MySQL plante constamment ?

Toutes les versions de MySQL sont testées sur plusieurs plates−formes avant leur publication. Cela ne signifie pas qu'elles sont exemptées de bogues, cela signifie juste que si il y a des bogues, il y en a très peu et sont durs à trouver. Si vous avez un problème, cela nous aidera toujours si vous essayez de trouver d'où vient exactement le plantage système, et vous aurez plus de chances de le voir résolu rapidement.

D'abord, vous devez essayer de trouver si le problème vient du démon mysqld qui se termine, ou s'il est lié à votre client. Vous pouvez savoir depuis combien de temps le serveur mysqld tourne en exécutant mysqladmin version . Si mysqld s'est terminé, vous trouverez sûrement la raison dans le fichier de log d'erreurs. Le log d'erreurs .

Sur certains systèmes, vous pouvez trouver dans ce fichier une trace de la pile, au moment où mysqld s'est arrêté, que vous pouvez étudier avec resolve_back_stack . Utilisation d'un traçage de la pile . Notez que les valeurs des variables écrites dans .err peuvent ne pas être toujours correctes.

Plusieurs plantages de MySQL sont causés par des fichiers de données ou d'index corrompus. MySQL écrira les données sur le disque avec un appel système à write() , après chaque requête et avant d'en notifier le client. (Cela n'est pas vrai si vous utilisez delay_key_write , auquel cas seul les données sont écrites.) Cela signifie que les données sont intègres même si mysqld plante, puisque le système d'exploitation s'assurera que les données non sorties du tampon ne sont pas enregistrée sur le disque. Vous pouvez forcer MySQL à se synchroniser avec le disque après chaque requête en démarrant mysqld avec --flush .

Ce qui précède signifie que normalement, vous ne devriez obtenir de tables corrompues que si :

- Quelqu'un ou quelque chose a coupé mysqld ou la machine au milieu d'une mise à jour.
- Vous avez trouvé un bogue dans mysqld qui le termine au milieu d'une mise à jour.
- Quelqu'un manipule les fichiers de données ou d'index en dehors de mysqld sans verrouiller proprement les tables.
- Si vous faites tourner plusieurs serveurs mysqld avec les mêmes données sur un système qui ne gère pas bien les verrous de fichiers (normalement gérés par le démon lockd) ou que vous le faites avec --skip-external-locking
- Vous avez un fichier de données ou d'index corrompu qui contient des données faussées ce qui amène mysqld à confusion.
- Vous avez trouvé un bogue dans le système de stockage des données. Cela parait impossible, mais sait-on jamais ? Dans ce cas, essayez de changer le type de fichier pour qu'il soit pris en charge par un autre gestionnaire de bases de données en utilisant ALTER TABLE sur une copie réparée de la table !

Parce qu'il est très difficile de savoir pourquoi quelque chose plante, essayez d'abord de voir si les choses qui marchent pour les autres ne fonctionnent pas chez vous. Merci d'essayer les différentes choses suivantes :

- Utilisez mysqld --log et essayez de déterminer à partir des informations du log si une requête spécifique fait planter le serveur. Plus de 95% de tous les bogues sont liés à une requête spécifique ! Normalement, c'est la dernière requête dans le fichier de log avant que MySQL n'ait redémarré. Le log général de requêtes . Si vous pouvez faire planter MySQL à plusieurs reprise avec une requête, même après avoir vérifié toutes les tables avant de l'exécuter, alors vous avez trouvé le bogue et vous devez faire un rapport de bogue pour nous en avertir ! Comment rapporter des bogues ou problèmes .
- Essayer d'effectuer une batterie de tests que nous pourrons utiliser pour reproduire le problème. Faire une batterie de tests lorsque vous faites face à un problème de table corrompue .
- Essayez d'exécuter le test inclus du dossier mysql-test et les benchmarks MySQL. Suites de tests MySQL . Ils devraient tester plutôt bien MySQL. Vous pouvez aussi ajouter ce code au benchmarks pour simuler votre application ! Les benchmarks peuvent être trouvés dans le répertoire sql-bench dans la distribution des sources ou, pour une distribution binaire, dans le répertoire sql-bench de votre dossier d'installation MySQL.
- Essayez fork_test.pl . Il est situé dans le dossier tests de la distribution source.
- Si vous configurez MySQL pour le débogage, il sera plus facile d'obtenir des informations à propos des erreurs possibles si quelque chose se passe mal. Reconfigurez MySQL avec l'option --with-debug ou --with-debug=full de configure puis recompilez. Déboguer un serveur MySQL .
- Configurer MySQL pour le débogage inclus un outil d'allocation de mémoire qui peut trouver quelques erreurs. Il fournit aussi beaucoup d'informations sur ce qui se passe.
- Avez-vous appliqué les derniers patches de votre système d'exploitation ?
- Utilisez l'option --skip-external-locking de mysqld . Sur quelques systèmes, le gestionnaire des verrous lockd ne fonctionne pas convenablement; l'option --skip-external-locking dit à mysqld de ne pas utiliser de pas utiliser de verrous externes. (Cela signifie que vous ne pouvez pas faire tourner deux serveurs mysqld sur les mêmes données et que vous devez faire attention si vous utilisez myisamchk , mais il peut être instructif d'essayer cette option comme test.)
- Avez-vous essayé mysqladmin -u root processlist lorsque mysqld semble fonctionner mais ne répond plus ? Quelquefois, mysqld n'est pas comateux, même si vous le croyez. Le problème peut-être que toutes les connexions sont utilisées, ou qu'il y a quelques problèmes avec les verrous internes. mysqladmin processlist devra normalement être en mesure d'effectuer une connexion même dans ce cas, et peut fournir des informations utiles à propos du nombre de connexions courantes et de leur statut.
- Exécutez la commande mysqladmin -i 5 status ou mysqladmin -i 5 -r status ou dans une fenêtre séparée pour produire des statistiques pendant que vous exécutez vos autres requêtes.
- Essayez ce qui suit :
  - Démarrez mysqld à partir de gdb (ou d'un autre débogueur). Déboguer mysqld sous gdb .
  - Exécutez vos scripts de tests.
  - Affichez le traçage et les variables locales aux trois niveaux les plus bas. Avec gdb vous pouvez le faire avec les commandes suivantes lorsque mysqld s'est planté à l'intérieur de gdb :

    ```
     backtrace
    info local
    up
    info local
    up
    info local
    ```

    Avec gdb vous pouvez aussi savoir quels threads existent avec info threads et en prendre un avec thread # , où # est l'identifiant du thread.
- Essayez de simuler votre application avec un script Perl pour forcer MySQL à planter ou à avoir un comportement défectueux.
- Envoyez un rapport de bogue normal. Comment rapporter des bogues ou problèmes . Soyez le plus précis possible et donnez plus de détails que d'habitude. Puisque MySQL fonctionne pour beaucoup de personnes, il se peut que le plantage résulte de quelque chose de spécifique à votre système (par exemple, une erreur liée à la particularité de vos bibliothèques système).
- Si vous avez des problèmes avec des tables à lignes de longueurs dynamiques et que vous n'utilisez pas de colonnes BLOB/TEXT (mais seulement des colonnes VARCHAR), vous pouvez essayer de changer tous les VARCHAR en CHAR avec ALTER TABLE . Cela forcera MySQL à utiliser des lignes de tailles fixes. Les lignes à tailles fixées prennent un peu plus d'espace, mais sont plus tolérants aux corruptions !Le code courant des lignes dynamiques est utilisé chez MySQL AB depuis au moins 3 ans sans aucun problème, mais par nature, les lignes à longueur dynamique sont plus exposées aux erreurs, il est donc bon d'essayer ce qui précède pour voir si cela vous aide !

24.4.3 Comment MySQL gère un disque plein

Lorsqu'il n'y a plus d'espace disque, MySQL fait ce qui suit :

- Il vérifie chaque minute pour voir s'il y a assez d'espace pour écrire la ligne courante. Si oui, il continue comme si rien ne s'était passé.
- Chaque 6 minutes, il ajoute un avertissement dans le fichier de log à propos de la condition du disque.

Pour contourner ce problème, vous pouvez effectuer les actions suivantes :

- Pour continuer, il suffit juste d'avoir assez d'espace disque pour insérer tous les enregistrements.
- Pour annuler le thread, vous devez lui envoyer un mysqladmin kill . Le thread sera annulé la prochaine fois qu'il vérifiera le disque (dans 1 minute).
- Notez que d'autres threads peuvent être en train d'attendre pour accéder à la table qui a causé le problème de disque plein. Si vous avez beaucoup de threads `` locked '', terminer le thread qui a causé la défaillance permettra aux autres threads de continuer.

Les exceptions pour le comportement suivant sont lorsque vous utilisez REPAIR ou OPTIMIZE ou lorsque les index sont créés dans un batch après l'exécution de LOAD DATA INFILE ou d'un ALTER TABLE .

Toutes les commandes précédentes risquent d'utiliser de gros fichiers temporaires, qui pourraient perturber le reste du système s'ils n'étaient pas supprimés. Si MySQL obtient une erreur de disque plein lors de l'exécution d'une des commandes précédentes, il effacera les gros fichiers temporaires et marquera la table comme corrompue (à part pour `ALTER TABLE` , où l'ancienne table sera restaurée).

24.4.4 Où MySQL stocke les fichiers temporaires ?

MySQL utilise la valeur de la variable d'environnement `TMPDIR` comme chemin du dossier où stocker les fichiers temporaires. Si vous n'avez pas de variable `TMPDIR` , MySQL utilise alors le dossier système par défaut, qui est normalement `/tmp` , `/var/tmp` ou `/usr/tmp` . Si le support qui contient votre dossier temporaire est trop petit, modifiez le script `safe_mysqld` pour configurer `TMPDIR` , et lui faire désigner un dossier où vous aurez la place. Vous pouvez aussi configurer ce dossier avec l'option `--tmpdir` de `mysqld` .

Depuis MySQL 4.1, l'option `--tmpdir` peut prendre une liste de plusieurs chemins, qui seront utilisés alternativement. Les chemins doivent être séparés par des deux−points (`':'`) sous Unix et par des points−virgules (`';'`) sous Windows, NetWare et OS/2. **Note :** pour répartir la charge entre plusieurs disques physiques, les chemins doivent être dirigés vers des disques **physiques** disctincts, et non pas des partitions différentes du même disque.

Si le serveur MySQL est configuré comme un esclave de réplication, vous ne devez pas faire pointer le chemin de --tmpdir vers un dossier en mémoire vive, ou vers un dossier qui sera vidé au démarrage du serveur. Un esclave de réplication doit être capable de retrouver ses fichiers temporaires lors du redémarrage du serveur, de manière à pouvoir reprendre la réplication des tables temporaires ou des opérations de LOAD DATA INFILE . Si les fichiers du dossier temporaire sont perdus après redémarrage, la réplication échouera.

MySQL crée tous les fichiers temporaires sous forme de fichier cachés. Cela garantit que les fichiers temporaires seront supprimés lorsque `mysqld` est terminé. L'inconvénient d'utiliser les fichiers cachés est que vous ne verrez pas que le dossier temporaire est gros, et qu'ils risquent de remplir votre dossier temporaire.

Lors des tris avec les clauses ORDER BY ou GROUP BY , MySQL utilise normalement deux dossiers temporaires. L'espace disque maximal nécessaire est :

```
 (taille de ce qui est trié + taille du pointeur de base)
* nombre de lignes trouvées
* 2
```

taille du pointeur de base vaut généralement 4, mais peut croître dans le futur pour les tables réellement grandes.Pour certaines requêtes SELECT , MySQL crée aussi des tables temporaires SQL. Elles ne sont pas cachées, et portent un nom du type SQL_* .

`ALTER TABLE` crée une table temporaire dans le même dossier que la table originale.

24.4.5 Comment protéger ou changer le fichier socket /tmp/mysql.sock

Le dossier par déafut pour le fichier de socket Unix que le serveur utilise pour les communications locales est `/tmp/mysql.sock` . Cela peut poser des problèmes, car sur certaines versions d'Unix, tout le monde peut effacer les fichiers dans le dossier `/tmp` .

Sur la plupart des versions d'Unix, vous pouvez protéger votre dossier /tmp pour que les fichiers ne puissent être effacés que par leur propriétaire ou le super utilisateur (root). Pour cela, utilisez le sticky sur le dossier /tmp en vous connectant en tant que root et en exécutant la commande suivante :

```
shell> chmod +t /tmp
```

Vous pouvez vérifier que le bit sticky est actif en exécutant ls -ld /tmp . Si le dernier bit de permission est t , il l'est.

Vous pouvez changer l'endroit où MySQL utilise / place le fichier de socket de la façon suivante :

- Spécifiez le chemin dans un fichier d'options globales ou locales. Par exemple, placez dans /etc/my.cnf :
  ```
  [client]
  socket=chemin-vers-fichier-socket

  [mysqld]
  socket=chemin-vers-fichier-socket
  ```
  Utiliser les fichiers d'options .
- Spécifiez cela en ligne de commande à safe_mysqld et à la plupart des clients avec l'option --socket=chemin-vers-fichier-socket .
- Spécifiez le chemin vers la socket dans la variable d'environnement MYSQL_UNIX_PORT .
- Définissez le chemin avec l'option de configure --with-unix-socket-path=chemin-vers-fichier-socket . Les options habituelles de configure .

Vous pouvez vérifier que la socket fonctionne avec cette commande :

```
shell> mysqladmin --socket=/chemin/vers/socket version
```

24.4.6 Problèmes de fuseaux horaires

Si vous avez un problème avec SELECT NOW() qui retournerait des valeurs en GMT et non votre temps local, vous devez configurer la variable d'environnement TZ et la mettre sur votre fuseau horaire courant. Cela peut être fait pour l'environnement dans lequel le serveur fonctionne, par exemple, dans safe_mysqld ou mysql.server . Variables d'environnement MySQL .

24.5 Problèmes relatifs aux requêtes

24.5.1 Sensibilité à la casse dans les recherches

Par défaut, les recherchez de MySQL ne sont pas sensibles à la casse (cependant, il existe des jeux de caractères qui ne sont jamais insensibles à la casse, comme czech).

Cela signifie que si vous recherchez avec `nom_colonne LIKE 'a%'` , vous aurez toutes les valeurs de la colonne qui commencent par un `A` ou un `a` . Si vous voulez que cette recherche soit sensible à la casse, utilisez par exemple `INSTR(nom_colonne, "A")=1` pour vérifier un préfixe. Utilisez `STRCMP(nom_colonne, "A") = 0` si la valeur de la colonne doit être exactement `"A"` .

Les opérations de comparaisons simples (>=, >, = , < , <= , tri et groupement) sont basées sur la ``valeur de tri'' de chaque caractère. Les caractères avec la même valeur de tri (comme (comme 'E' , 'e' et 'é')) sont considérés comme le même caractère !

Dans les anciennes versions de MySQL les comparaisons avec `LIKE` étaient effectuées sur la majuscule de chaque caractère (E == e mais E <> é). Dans les nouvelles versions `LIKE` fonctionne comme les autres opérateurs de comparaison.

Si vous voulez qu'une colonne soit toujours traitée de façon sensible à la casse, déclarez là en tant que BINARY . Syntaxe de CREATE TABLE .

Si vous utilisez des données chinoises avec l'encodage `big5` , vous devez rendre toutes les colonnes de chaînes `BINARY` . Cela fonctionne car l'ordre de tri de l'encodage `big5` est basé sur l'ordre des codes ASCII. Depuis MySQL 4.1, vous pouvez explicitement déclarer une colonne avec le jeu de caractères `big5` :

```
CREATE TABLE t (name CHAR(40) CHARACTER SET big5);
```

24.5.2 Problèmes avec l'utilisation des colonnes DATE

Le format d'une valeur de DATE est 'YYYY-MM-DD' . En accord avec ANSI SQL, aucun autre format n'est autorisé. Vous devez utiliser ce format dans les UPDATE et les clauses WHERE des requêtes SELECT . Par exemple :

```
mysql> SELECT * FROM nom_de_table WHERE date >= '1997-05-05';
```

MySQL convertit automatiquement une date en nombre si la date est utilisée dans un contexte numérique (et vice versa). Il est aussi assez intelligent pour permettre une forme ``relaxée'' lors des mises à jour et dans les clauses WHERE qui comparent une date et une colonne TIMESTAMP , DATE , ou DATETIME . (Forme relaxée signifie que n'importe quel caractère de ponctuation peut être utilisé en tant que séparateurs des parties. Par exemple, '1998-08-15' et '1998#08#15' sont équivalents.) MySQL peut convertir une chaîne ne contenant aucun séparateur (comme '19980815'), en supposant qu'elle a un sens pour une date.La date spéciale '0000-00-00' peut être stockée et récupérée en tant que '0000-00-00'. Lors de l'utilisation d'une date '0000-00-00' avec MyODBC , elle sera automatiquement convertie en NULL à partir de la version 2.50.12 de MyODBC , car ODBC ne peut gérer ce type de dates.

Puisque MySQL effectue les conversions décrites plus haut, ce qui suit fonctionnera :

```
mysql> INSERT INTO nom_de_table (idate) VALUES (19970505);
mysql> INSERT INTO nom_de_table (idate) VALUES ('19970505');
mysql> INSERT INTO nom_de_table (idate) VALUES ('97-05-05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('1997.05.05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('1997 05 05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('0000-00-00');

mysql> SELECT idate FROM nom_de_table WHERE idate >= '1997-05-05';
mysql> SELECT idate FROM nom_de_table WHERE idate >= 19970505;
mysql> SELECT MOD(idate,100) FROM nom_de_table WHERE idate >= 19970505;
mysql> SELECT idate FROM nom_de_table WHERE idate >= '19970505';
```

Toutefois, ce qui suit ne fonctionnera pas :

```
mysql> SELECT idate FROM nom_de_table WHERE STRCMP(idate,'19970505')=0;
```

STRCMP() est une fonction de chaînes de caractères, il convertit donc idate en une chaîne et effectue une comparaison de chaînes. Il ne convertit pas '19970505' en date et n'effectue donc pas de comparaison de dates.

Notez que MySQL vérifie très peu l'intégrité des dates. Si vous stockez une date erronée, comme '1998-2-31' , la date erronée sera enregistrée.

Vu que MySQL compresse les dates pour les stocker, il ne peut stocker tout format donné car il risquerait de ne pas correspondre au tampon de résultat. Les règles d'acceptations de dates sont :

- Si MySQL peut enregistrer et récupérer une date donnée, la date erronée est acceptée pour les colonnes DATE et DATETIME
- Toutes les valeurs de ours comprises entre 0 et 31 sont acceptées. Cela est fort convenable pour les applications web où vous demandez l'année, mois et jour dans trois champs textes (ou liste déroulantes) différents.
- Le champ jour ou mois peut être un zéro. Cela est convenable si vous voulez enregistrer un anniversaire dans une colonne DATE et que vous ne connaissez qu'une partie de la date.

Si la date ne peut être convertie en une valeur raisonnable, un 0 est inséré dans le champ DATE , il sera récupéré en tant que 0000-00-00 . Cela est une solution rapide et convenue vu que nous considérons que la responsabilité de la base de données est de récupérer la même date que vous aviez stocké (même si la date n'est pas correcte). Nous pensons que c'est à l'application de vérifier les dates, et non au serveur de le faire.

24.5.3 Problèmes avec les valeurs NULL

Le concept de la valeur NULL est une source de confusions pour les débutants en SQL, qui pensent souvent que NULL est la même chose qu'une chaîne de caractères vide "" . Ce n'est pas le cas ! Par exemple, les deux requêtes suivantes sont complètement différentes :

```
mysql> INSERT INTO ma_table (telephone) VALUES (NULL);
mysql> INSERT INTO ma_table (telephone) VALUES ("");
```

Les deux requêtes insèrent des valeurs dans la colonne telephone , mais la première insère une valeur NULL et la seconde insère une chaîne vide. La signification de la première peut être ``le numéro de téléphone est inconnu'' et la seconde peut être considérée comme ``elle n'a pas de téléphone''.

En SQL, la valeur NULL est toujours false en comparaison à n'importe quelle autre valeur, même NULL . Une expression contenant NULL produit toujours un résultat NULL sauf si une indication contraire est présente dans la documentation des opérateurs et des fonctions impliquées dans l'expression. Toutes les colonnes de l'exemple suivant retournent NULL :

```
mysql> SELECT NULL,1+NULL,CONCAT('Invisible',NULL);
```

Si vous voulez trouver les colonnes dont la valeur est NULL , vous ne pouvez pas utiliser le test =NULL . La requête suivante ne retourne aucune ligne car expr = NULL est FALSE, pour n'importe quelle expression :

```
mysql> SELECT * FROM ma_table WHERE telephone = NULL;
```

Pour trouver les valeurs NULL , vous devez utiliser le test IS NULL . Ce qui suit montre comment trouver les numéros de téléphone NULL et les numéros vides :

```
mysql> SELECT * FROM ma_table WHERE telephone IS NULL;
mysql> SELECT * FROM ma_table WHERE telephone = "";
```

Notez que vous ne pouvez ajouter d'index qu'aux colonnes pouvant avoir la valeur NULL si vous utilisez la version 3.23.2 de MySQL ou plus récente avec des tables de type MyISAM ou InnoDB . Dans les versions précédentes et avec les autres types, vous devez déclarer de telles colonnes NOT NULL . Cela signifie aussi que vous ne pouvez pas insérer NULL dans les colonnes indexées.

Lors de la lecture de données avec LOAD DATA INFILE , les colonnes vides sont interprétées en tant que '' . Si vous voulez une valeur NULL dans une colonne, vous devez utiliser \N dans le fichier. Le mot littéral 'NULL' peut aussi être utilisé dans certaines circonstances. Syntaxe de LOAD DATA INFILE .

Lors de l'utilisation de ORDER BY , les valeurs NULL sont présentées en premier. Si vous triez dans l'ordre décroissant en utilisant DESC , les valeurs NULL sont présentées en dernier. Lors de l'utilisation de GROUP BY , toutes les valeurs NULL sont considérées comme égales.

Pour mieux gérer les valeurs NULL , vous pouvez utiliser les opérateurs IS NULL et IS NOT NULL et la fonction IFNULL() .

Lors de l'utilisation de GROUP BY , toutes les valeurs NULL sont considérées comme égales.

Les fonctions agrégeantes comme COUNT() , MIN() et SUM() ignorent les valeurs NULL . Exception faite de COUNT(*) , qui compte les lignes et non pas les valeurs de colonnes. Par exemple, la commande suivante va donner deux comptes différents. Le premier est le nombre de lignes de la table, et le second est le nombre de ligne non− NULL de la colonne age :

```
mysql> SELECT COUNT(*), COUNT(age) FROM person;
```

Pour certains types de colonnes, les valeurs NULL sont traitées spécialement. Si vous insérez NULL dans la première colonne TIMESTAMP d'une table, la date et le temps courants sont insérés. Si vous insérez NULL dans une colonne AUTO_INCREMENT , le nombre suivant de la séquence sera inséré.

24.5.4 Problèmes avec les alias

Vous pouvez utiliser un alias pour vous référer à une colonne dans une clause GROUP BY , ORDER BY , ou HAVING . Les alias peuvent aussi être utilisés pour donner de meilleurs noms aux colonnes :

```
SELECT SQRT(a*b) as rt FROM nom_de_table GROUP BY rt HAVING rt > 0;
SELECT id,COUNT(*) AS cnt FROM nom_de_table GROUP BY id HAVING cnt > 0;
SELECT id AS "Identité du client" FROM nom_de_table;
```

Notez que ANSI SQL ne vous permet pas de vous référer à un alias dans une clause WHERE . Il en est ainsi car lorsque le code de WHERE est exécuté, la valeur de la colonne ne peut pas encore être déterminée. Par exemple, la requête suivante est illégale :

```
SELECT id,COUNT(*) AS cnt FROM nom_de_table WHERE cnt > 0 GROUP BY id;
```

La clause WHERE est exécutée pour savoir quelles lignes devraient être inclues dans la partie GROUP BY tandis que HAVING est utilisé pour décider quelles lignes du jeu de résultats doivent être utilisées.

24.5.5 Erreur Some non−transactional changed tables couldn't be rolled back

Si vous obtenez le message d'erreur :

```
Warning: Some non-transactional changed tables couldn't be rolled back
```

en essayant de faire un ROLLBACK , cela signifie que certaines tables que vous avez utilisé dans la transaction ne supportent pas les transactions.

Ces tables non−transactionnelles ne seront pas affectées par la commande ROLLBACK .

Si vous ne mélangez pas délibérément des tables transactionnelles et non−transactionnelles dans une transaction, la source la plus probablement de cette erreur est qu'une table que vous croyiez transactionnelle ne l'est pas. Cela peut arriver lorsque vous créez une table en utilisant un moteur de table qui n'est pas supporté par le serveur mysqld (ou qui a été désactivée avec une option de démarrage). Si mysqld ne supporte pas le moteur de table, il va changer le type et utiliser MyISAM , qui n'est pas non−transactionnelle.

Vous pouvez vérifier le type de table en utilisant une de ces commandes :

```
SHOW TABLE STATUS LIKE 'tbl_name';
SHOW CREATE TABLE tbl_name;
```

Voyez Syntaxe de SHOW TABLE STATUS et Syntaxe de SHOW CREATE TABLE .

Vous pouvez vérifier les moteurs de tables disponibles avec le serveur mysqld , avec ces commandes :

```
SHOW ENGINES;
```

Avant MySQL 4.1.2, SHOW ENGINES est inaccessible. Utilisez la commande suivante et vérifiez la valeur de la variable qui est associée à au moteur de tables que vous voulez :

```
SHOW VARIABLES LIKE 'have_%';
```

Par exemple, pour déterminer si le moteur InnoDB est disponible, vérifiez la valeur de la variable have_innodb .Voyez Syntaxe de SHOW ENGINES et Syntaxe de SHOW VARIABLES .

24.5.6 Effacer des lignes de tables reliées

Comme MySQL ne supporte encore ni les sous−requêtes, ni l'utilisation de plusieurs tables dans une requête DELETE (avant la version 4.0), vous devez utiliser l'approche suivante pour effacer des lignes de deux tables reliées :

- Sélectionnez (SELECT) les lignes de la table principale en vous basant sur une condition WHERE .
- Effacez (DELETE) les lignes de la table principale en vous basant sur la même condition.
- DELETE FROM table_liée WHERE colonne_liée IN (lignes_séléctionnées) .

Si le nombre total des caractères dans la requête avec la colonne_liée est supérieur à 1,048,576 (la valeur par défaut est max_allowed_packet , vous devez la découper en parties plus petites et exécuter plusieurs DELETE . Vous obtiendrez probablement les suppressions les plus rapides en n'effaçant que 100−1000 colonne_liée par requête si colonne_liée est un index. Si ce n'est pas un index, la vitesse est indépendante du nombre d'arguments dans la clause IN .

24.5.7 Résoudre les problèmes des lignes non retournées

Si vous avez une requête complexe avec beaucoup de tables et qu'elle ne retourne aucun résultat, vous devez suivre la procédure suivante pour trouver ce qui cloche dans votre requête :

- Testez la requête avec EXPLAIN et vérifiez si vous trouvez quelque chose qui vous parait fausse. Syntaxe de EXPLAIN (Obtenir des informations SELECT) .
- Ne sélectionnez que les champs que vous utilisez dans la clause WHERE :
- Enlevez une table à la fois de la requête jusqu'à ce qu'elle retourne quelques lignes. Si les tables sont grandes, il est bon d'utiliser LIMIT 10 dans la requête.
- Exécutez un SELECT pour les colonnes qui auraient du trouver des lignes dans la dernière table supprimée de la requête.
- Si vous comparez des colonnes FLOAT ou DOUBLE avec des nombres à virgule, vous ne pouvez pas utiliser '=' . C'est un problème commun à la plupart des langages de programmation car les valeurs à virgules flottantes ne sont pas des valeurs exactes. Dans le plupart des cas, changer la colonnes FLOAT en DOUBLE corrigera cela. Problèmes lors de la comparaison de nombres à virgules flottantes .
- si vous ne pouvez toujours pas trouver ce qui ne va pas, créez un test minimal pouvant être exécuté avec mysql test < query.sql montrant votre problème. Vous pouvez créer un fichier de test avec mysqldump −−quick base tables > query.sql . Editez le fichier et supprimez quelques lignes d'insertions (s'il y en a trop), et ajoutez votre requête de sélection à la fin du fichier.

  Vérifiez que vous avez encore le problème en faisant :

  ```
  shell> mysqladmin create test2
  shell> mysql test2 < query.sql
  ```

  Envoyez le fichier de test, en utilisant mysqlbug , à sur les listes diffusion. Les listes de diffusion MySQL .

24.5.8 Problèmes de comparaisons avec nombres à virgule flottante

Les nombres à virgule flottante portent souvent à confusion, car ils ne sont pas enregistrés en tant que valeurs exactes dans l'architecture de l'ordinateur. Ce qu'on voit à l'écran n'est souvent pas la valeur exacte du nombre.

Ce discours s'applique aux champs de types FLOAT , DOUBLE et DECIMAL .

```
 CREATE TABLE t1 (i INT, d1 DECIMAL(9,2), d2 DECIMAL(9,2));
INSERT INTO t1 VALUES (1, 101.40, 21.40), (1, -80.00, 0.00),
(2, 0.00, 0.00), (2, -13.20, 0.00), (2, 59.60, 46.40),
(2, 30.40, 30.40), (3, 37.00, 7.40), (3, -29.60, 0.00),
(4, 60.00, 15.40), (4, -10.60, 0.00), (4, -34.00, 0.00),
(5, 33.00, 0.00), (5, -25.80, 0.00), (5, 0.00, 7.20),
(6, 0.00, 0.00), (6, -51.40, 0.00);

 mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | -51.40 |  0.00 |
+------+--------+-------+
```

Le résultat est correcte. Même si les cinq premiers enregistrements ne devraient pas passer le test à priori, ils le font sûrement car la différence entre les nombres se situe plus loin que les décimales, ou que cela dépend de l'architecture de l'ordinateur.Le problème ne peut être résolu en utilisant ROUND() (ou une fonction similaire), car le résultat est encore un nombre à virgule flottante. Exemple :

```
 mysql> SELECT i, ROUND(SUM(d1), 2) AS a, ROUND(SUM(d2), 2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | -51.40 |  0.00 |
+------+--------+-------+
```

Voilà ce à quoi les nombres dans le champ 'a' ressemblent :

```
 mysql> SELECT i, ROUND(SUM(d1), 2)*1.0000000000000000 AS a,
    -> ROUND(SUM(d2), 2) AS b FROM t1 GROUP BY i HAVING a <> b;
+------+----------------------+-------+
| i    | a                    | b     |
+------+----------------------+-------+
|    1 |  21.3999999999999986 | 21.40 |
|    2 |  76.7999999999999972 | 76.80 |
|    3 |   7.4000000000000004 |  7.40 |
|    4 |  15.4000000000000004 | 15.40 |
|    5 |   7.2000000000000002 |  7.20 |
|    6 | -51.3999999999999986 |  0.00 |
+------+----------------------+-------+
```

Selon l'architecture de votre ordinateur, vous pouvez obtenir ou non les mêmes résultats. Chaque CPU peut évaluer les nombres à virgule flottante d'une manière différente. Par exemple, sur des machines vous pouvez obtenir les résultats 'correctes' en multipliant les deux arguments par 1, l'exemple suivant illustre cela. **ATTENTION : NE FAITES JAMAIS CONFIANCE A CETTE METHODE DANS VOS APPLICATIONS, C'EST UN EXEMPLE DE MAUVAISE METHODES !!!**

```
 mysql> SELECT i, ROUND(SUM(d1), 2)*1 AS a, ROUND(SUM(d2), 2)*1 AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | -51.40 | 0.00 |
+------+--------+------+
```

La raison pour laquelle l'exemple précédent semble fonctionner est que sur la machine en particulier où le test a été effectué, l'arithmétique des nombres à virgule flottante du CPU arrondi les nombres à la même valeur, mais il n'y a aucune règle stipulant qu'un CPU doit faire cela, on ne peut donc pas s'y fier.La façon correcte d'effectuer des comparaisons de nombre à virgule flottante est d'en premier lieu décider du degré de tolérance voulu entre les nombres puis d'effectuer la comparaison selon le nombre de tolérance. Par exemple, si nous décidons que les nombres à virgule flottante sont considérés comme égaux si ils ont la même précision au dix-millième près (0.0001), la comparaison ressemblera à cela :

```
 mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) > 0.0001;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | -51.40 | 0.00 |
+------+--------+------+
1 row in set (0.00 sec)
```

Et vice versa, si vous voulez obtenir les lignes où les nombres sont les mêmes, le test sera :

```
 mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) < 0.0001;
+------+-------+-------+
| i    | a     | b     |
+------+-------+-------+
|    1 | 21.40 | 21.40 |
|    2 | 76.80 | 76.80 |
|    3 |  7.40 |  7.40 |
|    4 | 15.40 | 15.40 |
|    5 |  7.20 |  7.20 |
+------+-------+-------+
```

24.6 Problèmes liés à l'optimiseur

MySQL utilise un optimiseur de coûts pour trouver le meilleur moyen de résoudre une requête. Dans de nombreux cas, MySQL peut calculer la meilleure solution pour une requête, mais parfois, MySQL n'a pas les informations nécessaires, et il fait des évaluations.

Pour les cas où MySQL ne fait pas le "bon" choix, des outils sont disponibles, pour que vous aidiez MySQL :

- Utilisez la commande EXPLAIN pour savoir comment MySQL va traiter la requête. Pour l'utilisez, ajoutez simplement le mot clé EXPLAIN avant votre requête SELECT :
  ```
  mysql> EXPLAIN SELECT * FROM t1, t2 WHERE t1.i = t2.i;
  ```
  EXPLAIN est présenté en détails dans la section Syntaxe de EXPLAIN (Obtenir des informations SELECT) .
- Utilisez la commande ANALYZE TABLE tbl_name pour mettre à jour les distributions de clés dans la table traitée. Syntaxe de ANALYZE TABLE .
- Utilisez l'option FORCE INDEX pour la table scannée, pour dire à MySQL que les scans de tables sont très coûteux, comparés aux scans d'index. Syntaxe de SELECT .
  ```
  SELECT * FROM t1, t2 FORCE INDEX (index_for_column)
  WHERE t1.col_name=t2.col_name;
  ```
  USE INDEX et IGNORE INDEX peuvent aussi être utiles.
- STRAIGHT_JOIN de niveau global ou table. Syntaxe de SELECT .
- Vous pouvez paramétrer les variables globales ou spécifiques au thread. Par exemple, lancez mysqld avec l'option --max-seeks-for-key=1000 ou utilisez SET max_seeks_for_key=1000 pour dire à l'optimiseur qu'il doit supposer que les scans de clés ne représenteront pas plus de 1000 recherches. Voyez Variables sytème du serveur .

24.7 Questions relatives aux définitions de tables

24.7.1 Problèmes avec ALTER TABLE .

ALTER TABLE change une table avec le jeu de caractères courant. Si durant l'exécution d' ALTER TABLE vous obtenez une erreur de clef dupliquée, alors la cause est soit que le nouveau jeu de caractères interprète deux clefs à la même valeur ou que la table est corrompue, au quel cas vous devez exécuter REPAIR TABLE sur la table.

Si ALTER TABLE se termine avec une erreur de ce genre :

```
Error on rename of './database/name.frm' to './database/B-a.frm' (Errcode: 17)
```

le problème est peut-être que MySQL a planté lors d'un précédent appel à ALTER TABLE et qu'il y a une ancienne table nommée A-quelquechose ou B-quelquechose qui subsiste. Dans ce cas, placez-vous dans le dossier de données MySQL et effacez tout les fichiers dont les noms commencent par A- ou B- . (Vous voudrez peut-être les déplacer autre part plutôt que de les effacer.) ALTER TABLE fonctionne de la façon suivante :

- Crée une nouvelle table nommée A-xxx avec les changements voulus.
- Toutes les lignes de l'ancienne table sont copiées dans A-xxx .
- L'ancienne table est renommée B-xxx .
- A-xxx est renommée avec le nom de votre ancienne table.
- B-xxx est supprimée.

Si quelque chose se passe mal durant l'opération de changement de nom, MySQL essaye d'annuler les changements. Si quelque chose de grave se passe (cela ne devrait jamais arriver bien sûr), MySQL peut laisser l'ancienne table en tant que B-xxx , mais un simple renommage au niveau système devrait restaurer vos données.

24.7.2 Comment changer l'ordre des colonnes dans une table

Tout d'abord, réfléchissez bien à la raison qui vous pousse à changer l'ordre des colonnes. Le but de SQL est de séparer l'application du stockage des données. Vous devez toujours spécifier l'ordre dans lequel vous devez lire les données. La première commande retourne les valeurs dans l'ordre col_name1 , col_name2 , col_name3 , alors que la seconde utilise l'ordre col_name1 , col_name3 , col_name2 :

```
mysql> SELECT col_name1, col_name2, col_name3 FROM tbl_name;
mysql> SELECT col_name1, col_name3, col_name2 FROM tbl_name;
```

Si vous décidez de changer l'ordre des colonnes dans une table, vous pouvez le faire comme ceci :

- Créez une nouvelle table avec les colonnes dans le nouvel ordre.
- Exécutez cette commande :
  ```
  mysql> INSERT INTO new_table
      -> SELECT columns-in-new-order FROM old_table;
  ```
- Effacez ou renommez la table old_table .
- Renommez la nouvelle table avec l'ancien nom.

```
mysql> ALTER TABLE new_table RENAME old_table;
```

`SELECT *` est parfait pour tester les requêtes. Mais, dans une application, vous me devriez **jamais** utiliser `SELECT *` et lire les colonnes en vous basant sur l'ordre retourné par la table. L'ordre et la position des colonnes peut changer. Un simple modification de la structure de la table vous conduira à une erreur.

24.7.3 Problèmes avec les tables temporaires

Voici une liste des limitations avec `TEMPORARY TABLES` .

- Une table temporaire ne peut être que du type HEAP , ISAM , MyISAM ou InnoDB .
- Vous ne pouvez utiliser une table plus d'une fois dans une requête. Par exemple, ce qui suit ne marche pas :

```
mysql> SELECT * FROM temp_table, temp_table AS t2;
ERROR 1137: Can't reopen table: 'temp_table'
```

- La commande SHOW TABLES ne liste pas les tables TEMPORARY .
- Vous ne pouvez pas utiliser la commande RENAME pour renommer une table TEMPORARY . Cependant, vous pouvez utiliser la commande ALTER TABLE pour cela :

```
mysql> ALTER TABLE orig_name RENAME new_name;
```

25 Crédits

Cet annexe liste les développeurs, contributeurs, et supporters qui ont fait de MySQL est ce qu'il est aujourd'hui.

25.1 Développeurs chez MySQL AB

Voici les développeurs qui sont employés par MySQL AB pour travailler sur la base de données MySQL , grosso modo dans leur ordre d'embauche. Pour chaque développeur, vous trouverez une petite liste de leur tâches et de leur responsabilités, ainsi que leur réalisations. Tous les développeurs sont impliqués dans le support.

Michael (Monty) Widenius
: Chef d'équipe et auteur principal du serveur MySQL (mysqld).
: Nouvelles fonctions dans la bibliothèque de chaînes de caractères.
: La majorité de la bibliothèque mysys .
: Les bibliothèques ISAM et MyISAM (les fichiers d'index B-tree avec compression d'index et différentes formats d'enregistrement).
: La bibliothèque HEAP . Un système de gestion des tables en mémoire avec un hashage dynamique efficace. Utilisé depuis 1980 et publié en 1984.
: Le programme replace (jetez y un oeil c'est carrément **COOL** !).
: MyODBC , le pilote ODBC de Windows95.
: La correction de bogues avec les MIT-pthreads pour qu'ils fonctionnent avec MySQL. Et aussi l'Unireg, une application à base de curses avec de nombreuses utilisations.
: Le port d'outils mSQL comme msqlperl , DBD / DBI et DB2mysql .
: La majorité de crash-me et les fondations des tests de performances de MySQL.

David Axmark
: Auteur original du **manuel de référence** , incluent les améliorations de texi2html .
: La mise à jour automatique du site web depuis le manuel.
: Le support initial de Autoconf, Automake et Libtool.
: Les licences.
: Des parties de tous les fichiers textes (de nos jours, uniquement le README est encore présent. le reste est dans le manuel).
: Nombreux tests des nouvelles fonctionnalités.
: Notre expert légal des logiciels libres.
: Le responsable des listes de diffusion (qui n'a jamais le temps de le faire correctement...).
: Notre code initial pour le port (bientôt plus de 10 ans). De nos jours, seules des parties de mysys restent.
: La personne que Monty appelle au milieu de la nuit lorsqu'il a réussi a faire fonctionner la nouvelle fonctionnalité!
: Le chef "Open Source" (Relations avec la communauté MySQL).

Jani Tolonen
: mysqlimport
: Un grand nombre d'extension pour le client en ligne de commande.
: PROCEDURE ANALYSE()

Sinisa Milivojevic
: Compression du protocole client/serveur avec zlib .
: Hashing parfait pour la phase d'analyse lexicale.
: Insertions multi-lignes
: Option mysqldump -e
: LOAD DATA LOCAL INFILE
: Option SQL_CALC_FOUND_ROWS SELECT
: Option --max-user-connections=...
: net_read et net_write_timeout
: GRANT / REVOKE et SHOW GRANTS FOR
: Nouveau protocole client/serveur pour la version 4.0
: UNION en version 4.0
: Traitements multi-table de DELETE / UPDATE
: Tables dérivées en version 4.1
: Gestion des ressources utilisateurs.
: Développeur initial de l'API MySQL++ C++ API et du client MySQLGUI .

Tonu Samuel (ancien développeur)
: Interface VIO (la base pour le protocole client/serveur chiffré).
: Système de fichier MySQL (une méthode pour utiliser la base MySQL comme un système de fichiers).
: L'expression CASE .
: Les fonctions MD5() et COALESCE() .
: Le support RAID des tables MyISAM .

Sasha Pachev
: Implémentation initiale de la réplication (jusqu'en version 4.0).
: SHOW CREATE TABLE .
: mysql-bench

Matt Wagner
: Suite de tests MySQL.
: Webmestre (jusqu'en 2002).
: Coordinateur du développement.

Miguel Solorzano
: Développement Win32 et publications.
: Code du serveur sur Windows NT.
: WinMySQLAdmin

Timothy Smith (ancien développeur)
: Support des jeux de caractères dynamiques.
: le script configure, les RPM et d'autres parties du système de compilation.
: Développeur initial de libmysqld , le serveur intégré.

Sergei Golubchik
: Recherche en texte plein.
: bibliothèque de clés pour MERGE .

Jeremy Cole
: Relecture et édition de ce manuel.
: ALTER TABLE ... ORDER BY ... .

UPDATE ... ORDER BY ... .
DELETE ... ORDER BY ... .

Indrek Siitan
: Design et programmation de notre interface web.
Auteur de notre lettre d'actualité.

Jorge del Conde
: MySQLCC (MySQL Control Center)
Développement Win32
Implémentation initiale des portails du site web.

Venu Anuganti
: MyODBC 3.51
Nouveau protocole client/serveur pour la version 4.1 (pour les requêtes préparées).

Arjen Lentz
: Responsable du manuel de référence MySQL.
Préparation de la version imprimée chez O'Reilly.

Alexander (Bar) Barkov, Alexey (Holyfoot) Botchkov et Ramil Kalimullin
: Données spatiales (GIS) et R-Trees en version 4.1
Unicode et jeux de caractères pour la version 4.1

Oleksandr (Sanja) Byelkin
: Cache de requêtes en version 4.0
Implémentation des sous-requêtes en version 4.1.

Aleksey (Walrus) Kishkin et Alexey (Ranger) Stroganov
: Design des tests de performance et analyse.
Maintenance de la suite de test MySQL.

Zak Greant
: Porte parole Open Source, relation avec la communauté.

Carsten Pedersen
: Le programme de certification MySQL.

Lenz Grimmer
: Ingénierie de mise en production (compilation et publication).

Peter Zaitsev
: Fonction SHA1() , AES_ENCRYPT() et AES_DECRYPT() .
Débogage et nettoyage de diverses fonctionnalités.

Alexander (Salle) Keremidarski
: Support.
Débogage.

Per-Erik Martin
: Chef de projet pour les procédures stockées et les triggers.

Jim Winstead
: Chef développeur web.

Mark Matthews
: Connecteur/Pilote J (Java).

Peter Gulutzan
: Compatibilité avec les standards SQL-99, SQL-2003.
Documentation des codes/algorithmes MySQL existant.

Guilhem Bichot
: Réplication, depuis MySQL version 4.0.
Correction de la gestion des exposants pour les DECIMAL .
Auteur de mysql_tableinfo .

Antony T. Curtis
: Port de MySQL sur OS/2.

25.2 Contributeurs à MySQL

Bien que MySQL AB possède tous les droits du serveur MySQL et du manuel MySQL , nous voulons montrer notre reconnaissance à ceux qui ont apportés leur contribution d'une manière ou d'une autre à la distribution MySQL . Les contributeurs sont listés ici, dans un ordre aléatoire :

Gianmassimo Vigazzola qwerg@mbox.vol.it ou qwerg@tin.it
: Le portage initial sur Win32/NT.

Per Eric Olsson
: Pour des critiques plus ou moins constructives, et des tests sur les formats de lignes dynamiques.

Irena Pancirov irena@mail.yacc.it
: Portage sur Win32 avec le compilateur Borland. mysqlshutdown.exe et mysqlwatch.exe

David J. Hughes
: Pour ses efforts de constitution d'une base de données SQL partagée. Chez TcX, le prédécesseur de MySQL AB, nous avons commencé avec mSQL , mais nous nous sommes aperçus que cela ne satisfaisait pas nos besoins, et nous avons écrit une interface SQL avec notre application Unireg. Les clients mysqladmin et mysql ont été largement influencés par leur équivalent mSQL . Nous avons mis une grande partie de nos efforts à faire que la syntaxe de MySQL soit un sur ensemble de celle de mSQL . De nombreuses idées d'API ont été empruntées à mSQL pour rendre plus facile le portage des programmes depuis mSQL vers MySQL. Le logiciel MySQL ne contient aucun code extrait de mSQL . Les deux fichiers (client/insert_test.c et client/select_test.c) qui font partie de la distribution sont basés sur les fichiers correspondants, et sans droits de la distribution mSQL , mais sont modifiés, pour servir d'exemple aux modifications nécessaires à la conversion de code de mSQL vers MySQL. (mSQL est la propriété de David J. Hughes).

Patrick Lynch
: Pour nous avoir aidé à acquérir http://www.mysql.com/ .

Fred Lindberg
: Pour avoir configuré qmail pour qu'il gère les listes de diffusions MySQLm et pour l'aide incroyable que nous avons reçu pour gérer les listes de diffusions de MySQL.

Igor Romanenko igor@frog.kiev.ua
: mysqldump (précédemment appelé msqldump , mais porté et amélioré par Monty).

Yuri Dario
: Pour avoir suivi et amélioré la version MySQL pour OS/2.

Tim Bunce
: Auteur de mysqlhotcopy .

Zarko Mocnik zarko.mocnik@dem.si
: Tri pour le slovène.

"TAMITO" tommy@valley.ne.jp
: Les macros de jeu de caractères _MB m et les jeux de caractères ujis et sjis.

Joshua Chamas joshua@chamas.com
: La base des insertions concurrentes, la syntaxe de date améliorée, le débogage sous NT et les réponses sur la liste de diffusion.

Yves Carlier Yves.Carlier@rug.ac.be
: mysqlaccess , un programme qui affiche les droits des utilisateurs.

Rhys Jones rhys@wales.com (et GWE Technologies Limited)
: Pour JDBC, un module pour extraire des données de bases MySQL avec un client Java.

Dr Xiaokun Kelvin ZHU X.Zhu@brad.ac.uk
: Développement du pilote JDBC et d'autres outils MySQL liés à Java.

James Cooper pixel@organic.com
: Pour la configuration d'une archive indexée des listes de diffusion sur son site.

Rick Mehalick Rick_Mehalick@i-o.com
: Pour xmysql , un client graphique X pour MySQL.

Doug Sisk sisk@wix.com
: Pour les paquets RPM de MySQL pour RedHat Linux.

Diemand Alexander V. axeld@vial.ethz.ch
: Pour les paquets RPM de MySQL pour RedHat Linux-Alpha.

Antoni Pamies Olive toni@readysoft.es
: Pour les paquets RPM de MySQL pour Intel et SPARC.

Jay Bloodworth jay@pathways.sde.state.sc.us
: Pour les paquets RPM de MySQL pour MySQL Version 3.21.

David Sacerdote davids@secnet.com
: Idées pour la vérification sécuritaire des noms d'hôtes.

Wei-Jou Chen jou@nematic.ieo.nctu.edu.tw
: Le support des caractères chinois (BIG5).

Wei He hewei@mail.ied.ac.cn
: Un grand nombre de fonctionnalités pour le jeu de caractères Chinese(GBK).

Jan Pazdziora adelton@fi.muni.cz
: Ordre de tri tchèque.

Zeev Suraski bourbon@netvision.net.il
: Format d'heure FROM_UNIXTIME() , fonctions ENCRYPT() et conseiller bison . Membre actif des listes de diffusion.

Luuk de Boer luuk@wxs.nl
: Portage et amélioration de la suite de tests avec DBI / DBD . A été d'une grande aide avec le test crash-me et l'exécution des tests. Certaines améliorations de la fonction de date. Le script mysql_setpermissions .

Alexis Mikhailov root@medinf.chuvashia.su
: Fonctions utilisateurs (UDF); CREATE FUNCTION et DROP FUNCTION .

Andreas F. Bobak bobak@relog.ch
: L'extension AGGREGATE des fonctions UDF .

Ross Wakelin R.Wakelin@march.co.uk
: Aide avec InstallShield pour MySQL-Win32.

Jethro Wright III jetman@li.net
: La bibliothèque libmysql.dll .

James Pereria jpereira@iafrica.com
: Mysqlmanager, un outil d'administration Win32 graphique pour MySQL.

Curt Sampson cjs@portal.ca
: Portage des MIT-pthreads vers NetBSD/Alpha et NetBSD 1.3/i386.

Martin Ramsch m.ramsch@computer.org
: Exemples dans les tutoriels du manuel MySQL.

Steve Harvey
: Pour la sécurisation de mysqlaccess .

Konark IA-64 Centre of Persistent Systems Private Limited
: http://www.pspl.co.in/konark/ . Aide avec le port Win64 du serveur MySQL.

Albert Chin-A-Young.
: Modifications de la configuration pour Tru64, support des grands fichiers, et amélioration des gestionnaires TCP.

John Birrell
: Emulation de pthread_mutex() pour OS/2.

Benjamin Pflugmann
: Tables MERGE améliorée pour la gestion des INSERTS . Membre actif des listes de diffusion.

Jocelyn Fournier
: Travail excellent et report d'un grand nombre de bogues (surtout dans le code des requêtes imbriquées de MySQL 4.1).

Marc Liyanage
: Maintenance des paquets Mac OS X et informations de grande valeur en ce qui concerne la création des paquets pour Mac OS X.

D'autres contributeurs, chasseurs de bugs et testeurs : James H. Thompson, Maurizio Menghini, Wojciech Tryc, Luca Berra, Zarko Mocnik, Wim Bonis, Elmar Haneke, jehamby@lightside , psmith@BayNetworks.com , duane@connect.com.au , Ted Deppner ted@psyber.com , Mike Simons, Jaakko Hyvatti.Et de nombreux autres patches et rapports issus des mailing listes :

Un grand merci à ceux qui aident à répondre aux questions sur la liste de diffusion de MYSQL :

Daniel Koch dkoch@amcity.com
: Configuration Irix.

Luuk de Boer luuk@wxs.nl
: Questions performances.

Tim Sailer tps@users.buoy.com
: Questions sur DBD::mysql .

Boyd Lynn Gerber gerberb@zenez.com
: Questions sur SCO.

Richard Mehalick RM186061@shellus.com
: Questions sur xmysql et les questions d'installation de base.

Zeev Suraski bourbon@netvision.net.il
: Configuration du module Apache (log &identification), MySQL et PHP, syntaxe SQL et autres questions.

Francesc Guasch frankie@citel.upc.es
: Questions générales.

Jonathan J Smith jsmith@wtp.net
: Questions spécifiques aux OS comme Linux, syntaxe SQL et autres sujets intéressants.

David Sklar sklar@student.net
: Utilisation de MySQL avec PHP et Perl.

Alistair MacDonald A.MacDonald@uel.ac.uk
: Pas encore spécifié mais généraliste : Linux et un peu HP-UX. Pousse les utilisateurs à utiliser mysqlbug .

John Lyon jlyon@imag.net
: Questions sur l'installation de MySQL sur le système Linux, avec soit les fichiers .rpm ou en compilant depuis la source.

Lorvid Ltd. lorvid@WOLFENET.com
: Questions sur la facturation, les licences, le support et le copyright de MySQL.

Patrick Sherrill patrick@coconet.com
: Questions sur les interfaces ODBC et VisualC++.

Randy Harmon rjharmon@uptimecomputers.com
: DBD , Linux, et quelques questions SQL.

25.3 Documenteurs et traducteurs

La liste des personnes suivantes nous a aidé à écrire la documentation MySQL et à la traduire. Certains ont aussi travaillés sur les messages d'erreurs MySQL.

Paul DuBois
: Contribution continuelle pour rendre ce manuel présentable et compréhensible. Cela inclut la réécriture des courageux essais de Monty et David, dans un anglais que tout le monde comprend.

Kim Aldale
: Aide à la réécriture des premières versions de Monty et David.

Michael J. Miller Jr. mke@terrapin.turbolift.com
: Pour le premier manuel MySQL. Et beaucoup de corrections orthographiques et typographiques dans la FAQ, qui a été finalement transformée en une documentation complète.

Yan Cailin
: Premier traducteur du manuel de référence MySQL en chinois simplifié, au début de l'année 2000, en Big5 et HK. (http://mysql.hitstar.com/) Page personnelle linuxdb.yeah.net .

Jay Flaherty fty@mediapulse.com
: Grosses contributions dans la section Perl DBI / DBD du manuel.

Paul Southworth pauls@etext.org, Ray Loyzaga yar@cs.su.oz.au
: Relecture du manuel de référence.

Therrien Gilbert gilbert@ican.net, Jean-Marc Pouyot jmp@scalaire.fr
: Messages d'erreurs en français.

Petr Snajdr, snajdr@pvt.net
: Messages d'erreurs en tchèque.

Jaroslaw Lewandowski jotel@itnet.com.pl
: Messages d'erreurs en polonais.

Miguel Angel Fernandez Roiz
: Messages d'erreurs en espagnol.

Roy-Magne Mo rmo@www.hivolda.no
: Messages d'erreurs en norvégien et test de la version 3.21.#.

Timur I. Bakeyev root@timur.tatarstan.ru
: Messages d'erreurs en russe.

brenno@dewinter.com &Filippo Grassilli phil@hyppo.com
: Messages d'erreurs en italien.

Dirk Munzinger dirk@trinity.saar.de
: Messages d'erreurs en allemand.

Billik Stefan billik@sun.uniag.sk
: Messages d'erreurs en slovaque.

Stefan Saroiu tzoompy@cs.washington.edu
: Messages d'erreurs en roumain.

Peter Feher
: Messages d'erreurs en hongrois.

Roberto M. Serqueira
: Messages d'erreurs en portugais.

Carsten H. Pedersen
: Messages d'erreurs en danois.

Arjen G. Lentz
: Messages d'erreurs en hollandais, et fin de la traduction partielle.

25.4 Bibliothèques utilisées et incluses dans MySQL

Voici une liste des auteurs de bibliothèques que nous avons inclus dans le source du serveur MySQL, pour le rendre plus facile à compiler et à installer. Nous sommes reconnaissants envers ceux qui les ont créé : ils nous ont facilité la vie.

Fred Fish
: Pour son excellente bibliothèque de débogage et de trace. Monty a ajouté de nombreuses petites améliorations à la bibliothèque (en terme de vitesse et fonctionnalités).

Richard A. O'Keefe
: Pour la bibliothèque de gestion des chaînes de caractères, disponible dans le domaine public.

Henry Spencer
: Pour sa bibliothèque d'expressions régulières, utilisée dans WHERE column REGEXP regexp .

Chris Provenzano
: Pthreads utilisateurs. Dans le copyright : This product includes software developed by Chris Provenzano, the University of California, Berkeley, and contributors. Nous utilisons la version 1_60_beta6, modifiée par Monty (voir mit-pthreads/Changes-mysql).

Jean-loup Gailly et Mark Adler
: Pour la bibliothèque zlib (utilisée avec MySQL sous Windows).

Bjorn Benson
: Pour son paquet safe_malloc (contrôle de mémoire), qui est utilisé lorsque vous configurez MySQL avec --debug .

Free Software Foundation
: La bibliothèque readline (utilisée par le client mysql).

La fondation NetBSD
: Le paquet libedit (utilisé optionnellement par le client mysql).

25.5 Applications qui supportent MySQL

Voici une liste de responsables et auteurs d'interfaces, applications et paquets, que de nombreuses personnes utilisent avec MySQL.

Nous ne pouvons pas lister tous les paquets possibles car la liste serait trop difficile à entretenir. Pour les autres paquets, voyez le portail d'applications à http://www.mysql.com/portal/software .

Tim Bunce, Alligator Descartes
: Pour l'interface DBD (Perl).

Andreas Koenig a.koenig@mind.de
: Pour l'interface Perl avec MySQL.

Jochen Wiedmann wiedmann@neckar-alb.de
: Pour le module Perl DBD::mysql .

Eugene Chan eugene@acenet.com.sg
: Pour le port de PHP sur MySQL Server.

Georg Richter
: Tests de MySQL 4.1 et chasse aux bugs. Nouvelle extensions PHP 5.0 mysqli (API) pour MySQL 4.1 et plus récent.

Giovanni Maruzzelli maruzz@matrice.it
: Pour le port de iODBC (Unix ODBC).

Xavier Leroy Xavier.Leroy@inria.fr
: L'auteur de LinuxThreads (utilisé par MySQL Server sur Linux).

25.6 Outils utilisés pour créer MySQL

Voici une liste des outils que nous avons utilisés pour créer MySQL. Nous utilisons cette page pour remercier les auteurs qui les ont créé, et sans lesquels nous n'aurions pas pu faire de MySQL ce qu'il est aujourd'hui.

Free Software Foundation
: De qui nous avons reçu un excllent compilateur (gcc), un excellent débogueur (gdb et la bibliothèque libc (à laquelle nous avons emprunté le fichier strto.c pour faire fonctionner du code sur Linux).

Free Software Foundation &l'équipe de développement XEmacs
: Pour un éditeur texte et un environnement de programmation fantastique, utilisé par presque tout le monde chez MySQL AB.

Julian Seward
: L'auteur de valgrind , un excellent outil de vérification de mémoire, qui nous a aidé à trouver des bugs très difficiles à traquer.

Dorothea Lütkehaus et Andreas Zeller
: Pour DDD (Le Data Display Debugger) qui est une excellente interface graphique pour gdb).

25.7 Supporters de MySQL

Tandis que MySQL AB possède tous les droits du serveur MySQL et du manuel MySQL , nous voulons montrer notre reconnaissances aux compagnies suivantes, qui nous ont aidé à financer le développement du serveur MySQL , soit en nous payant pour développer de nouvelles fonctionnalités, soit en nous fournissant en matériel pour le développement du serveur MySQL .

VA Linux / Andover.net
: Financement de la réplication.

NuSphere
: Edition du manuel MySQL.

Stork Design studio
: Le site web MySQL utilisé entre 1998 et 2000.

Intel
: Contribution au développement sur les plates-formes Windows et Linux.

Compaq
: Contribution au développement sur Linux/Alpha.

SWSoft
: Développement de la version embarquée mysqld .

FutureQuest
: --skip-show-database

26 Historique des changements MySQL

Cet appendice liste les changements de version à version dans le code source de MySQL.

Nous travaillons maintenant activement sur MySQL 4.x et ne fournirons que les correctifs pour les bogues critiques de MySQL 3.23. Nous mettons à jour cette section lorsque nous ajoutons de nouvelles fonctionnalités pour que tout le monde puisse suivre le cours du développement.

Notre section TODO contient ce que nous planifions pour les versions 4.x. Liste de tâches MySQL .

Notez que nous essayons de mettre à jour le manuel en même temps que nous apportons des changements à MySQL. Si vous trouvez une version mentionnée ici que vous ne pouvez retrouver dans la page des téléchargements MySQL (http://www.mysql.com/downloads/), cela signifie que la version n'a pas encore été publiée !

26.1 Changements de la version 5.0.0 (Développement)

The following changelog shows what has already been done in the 5.0 tree:

- Basic support for stored procedures (SQL:2003 style). Procédures stockées et fonctions .
- Added SELECT INTO list_of_vars , which can be of mixed, that is, global and local type. Commande SELECT ... INTO .
- Removed the update log. It is fully replaced by the binary log. If the MySQL server is started with --log-update , it will be translated to --log-bin (or ignored if the server is explicitly started with --log-bin), and a warning message will be written to the error log. Setting SQL_LOG_UPDATE will silently set SQL_LOG_BIN instead (or do nothing if the server is explicitly started with --log-bin).
- User variable names are now case insensitive: if you do SET @a=10; then SELECT @A; will now return 10 . Case sensitivity of a variable's value depends on the collation of the value.

For a full list of changes, please refer to the changelog sections for each individual 5.0.x release.

26.1.1 Changements de la version 5.0.1 (pas encore publiée)

Functionality added or changed:

- For replication of MEMORY (HEAP) tables: Made the master automatically write a DELETE FROM statement to its binary log when a MEMORY table is opened for the first time since master's startup. This is for the case where the slave has replicated a non-empty MEMORY table, then the master is shut down and restarted: the table is now empty on master; the DELETE FROM empties it on slave too. Note that even with this fix, between the master's restart and the first use of the table on master, the slave still has out-of-date data in the table. But if you use the --init-file option to populate the MEMORY table on the master at startup, it ensures that the failing time interval is zero. (Bug #2477)
- When a session having open temporary tables terminates, the statement automatically written to the binary log is now DROP TEMPORARY TABLE IF EXISTS instead of DROP TEMPORARY TABLE , for more robustness.
- The MySQL server now returns an error if SET SQL_LOG_BIN is issued by a user without the SUPER privilege (in previous versions it just silently ignored the statement in this case).
- Changed that when the MySQL server has binary logging disabled (that is, no log-bin option was used) then no transaction binlog cache is allocated for connections (this should save binlog_cache_size bytes of memory (32 kilobytes by default) for every connection).
- Added option --replicate-same-server-id .

Bugs fixed:

- Strange results with index (x, y) ... WHERE x=val_1 AND y>=val_2 ORDER BY pk; (Bug #3155)
- Subquery and order by (Bug #3118)
- ALTER DATABASE caused the client to hang if the database did not exist. (Bug #2333)
- SLAVE START (which is a deprecated syntax, START SLAVE should be used instead) could crash the slave. (Bug #2516)
- Multiple-table DELETE statements were never replicated by the slave if there were any replicate-*-table options. (Bug #2527)
- The MySQL server did not report any error if the query (submitted through mysql_real_query() or mysql_prepare()) was terminated by garbage characters (which can happen if you pass a wrong length parameter to mysql_real_query() or mysql_prepare()); the result was that the garbage characters were written into the binary log. (Bug #2703)
- Replication: If a client connects to a slave server and issues an administrative statement for a table (for example, OPTIMIZE TABLE or REPAIR TABLE), this could sometimes stop the slave SQL thread. This does not lead to any corruption, but you must use START SLAVE to get replication going again. (Bug #1858)
- Made clearer the error message which one gets when an update is refused because of the read-only option. (Bug #2757)
- Fixed that replicate-wild-*-table rules apply to ALTER DATABASE when the table pattern is '%', like it is already the case for CREATE DATABASE and DROP DATABASE . (Bug #3000)
- Fixed that when a Rotate event is found by the slave SQL thread in the middle of a transaction, the value of Relay_Log_Pos in SHOW SLAVE STATUS remains correct. (Bug #3017)
- Corrected the master's binary log position that InnoDB reports when it is doing a crash recovery on a slave server. (Bug #3015)
- Changed the column Seconds_Behind_Master in SHOW SLAVE STATUS to never show a value of -1. (Bug #2826)
- Changed that when a DROP TEMPORARY TABLE statement is automatically written to the binlog when a session ends, the statement is recorded with an error code of value zero (this ensures that killing a SELECT on the master does not result in a superfluous error on the slave). (Bug #3063)
- Changed that when a thread handling INSERT DELAYED (also known as a delayed_insert thread) is killed, its statements are recorded with an error code of value zero (killing such a thread does not endanger replication, so we thus avoid a superfluous error on the slave). (Bug #3081)
- Fixed deadlock when two START SLAVE commands were run at the same time. (Bug #2921)
- Fixed that a statement never triggers a superfluous error on the slave, if it must be excluded given the replicate-* options. The bug was that if the statement had been killed on the master, the slave would stop. (Bug #2983)
- The --local-load option of mysqlbinlog now requires an argument.

- Fixed a segmentation fault when running LOAD DATA FROM MASTER after RESET SLAVE . (Bug #2922)
- mysqlbinlog --read-from-remote-server read all binary logs following the one that was requested. It now stops at the end of the requested file, the same was it does when reading a local binary log. (Bug #3204)
- Fixed mysqlbinlog --read-from-remote-server to print the exact positions of events in the "at #" lines. (Bug #3214)
- Fixed a rare error condition that caused the slave SQL thread spuriously to print the message Binlog has bad magic number and stop when it was not necessary to do so. (Bug #3401)
- Fixed mysqlbinlog not to forget to print a USE statement under rare circumstances where the binary log contained a LOAD DATA INFILE statement. (Bug #3415)
- Fixed a memory corruption when replicating a LOAD DATA INFILE when the master had version 3.23. (Bug #3422)
- Multiple-table DELETE statements were always replicated by the slave if there were some replicate-*-ignore-table options and no replicate-*-do-table options. (Bug #3461)
- Fixed a crash of the MySQL slave server when it was built with --with-debug and replicating itself. (BUG #3568)
- Fixed that in some replication error messages, a very long query caused the rest of the message to be invisible (truncated), by putting the query last in the message. (Bug #3357)

26.1.2 Changements de la version 5.0.0 (22 décembre 2003 : Alpha)

Functionality added or changed:

- The KILL statement now takes CONNECTION and QUERY variants. The first is the same as KILL with no modifier (it kills a given connection thread). The second kills only the statement currently being executed by the connection.
- Added TIMESTAMPADD() and TIMESTAMPDIFF() functions.
- Added WEEK and QUARTER values as INTERVAL arguments for DATE_ADD() and DATE_SUB() functions.
- New binary log format that enables replication of those session variables: sql_mode , SQL_AUTO_IS_NULL , FOREIGN_KEY_CHECKS (that one was already replicated since 4.0.14 but here it's done more efficiently: takes less space in the binary logs), UNIQUE_CHECKS . Other variables (like character sets, SQL_SELECT_LIMIT ...) will be replicated in next 5.0.x releases.
- Implemented Index Merge optimization for OR clauses. Comment MySQL optimise les clauses OR .
- Basic support for stored procedures (SQL:2003 style). Procédures stockées et fonctions .
- Added SELECT INTO list_of_vars , which can be of mixed, that is, global and local type. Commande SELECT ... INTO .
- Easier replication upgrade (5.0.0 masters can read older binary logs, 5.0.0 slaves can read older relay logs; see Compatibilité de la réplication entre les versions de MySQL for more details). The format of the binary log and relay log is changed compared to the one of MySQL 4.1 and older.
- **Important note:** If you upgrade to InnoDB-4.1.1 or higher, you cannot downgrade to a version lower than 4.1.1 any more! That is because earlier versions of InnoDB are not aware of multiple tablespaces.

Bugs fixed:

26.2 Changements de la version 4.1.x (Alpha)

Version 4.1 of the MySQL server includes many enhancements and new features. Binaries for this version are available for download at http://www.mysql.com/downloads/mysql−4.1.html .

- Subqueries and derived tables (unnamed views). Syntaxe des sous-requêtes .
- INSERT ... ON DUPLICATE KEY UPDATE ... syntax. This allows you to UPDATE an existing row if the insert would cause a duplicate value in a PRIMARY or UNIQUE key. (REPLACE allows you to overwrite an existing row, which is something entirely different.) Syntaxe de INSERT .
- A newly designed GROUP_CONCAT() aggregate function. Fonctions à utiliser dans les définitions GROUP BY .
- Extensive Unicode (UTF8) support.
- Character sets can be defined per column, table, and database.
- New key cache for MyISAM tables with many tunable parameters. You can have multiple key caches, preload index into caches for batches...
- BTREE index on HEAP tables.
- Support for OpenGIS spatial types (geographical data). Extensions spatiale de MySQL .
- SHOW WARNINGS shows warnings for the last command. Syntaxe de SHOW WARNINGS .
- Faster binary protocol with prepared statements and parameter binding. Commandes préparées en C .
- You can now issue multiple statements with a single C API call and then read the results in one go. Gestion des requêtes multiples en C .
- Create Table: CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table2 LIKE table1 .
- Server based HELP command that can be used in the mysql command line client (and other clients) to get help for SQL statements.

For a full list of changes, please refer to the changelog sections for each individual 4.1.x release.

26.2.1 Changements de la version 4.1.3 (pas encore publiée)

Functionality added or changed:

Bugs fixed:

26.2.2 Changements de la version 4.1.2 (pas encore publiée)

Functionality added or changed:

- Added explanation of hidden SELECT of UNION in output of EXPLAIN SELECT statement.
- mysql command-line client now supports multiple -e options. (Bug #591)
- New myisam_data_pointer_size system variable. Variables sytème du serveur .
- The --log-warnings server option now is enabled by default. Disable with --skip-log-warnings .
- The --defaults-file=file_name option now requires that the filename must exist (safety fix). (Bug #3413)
- mysqld_multi now creates the log in datadir (from [mysqld] section in my.cnf or compiled in), not in /tmp - vulnerability id CAN-2004-0388. Thanks to Christian Hammers from Debian Security Team for reporting this!
- **Warning: Incompatible change!** String comparison now works according to the SQL standard. Because we have that 'a' = 'a ' then from it must follow that 'a' > 'a\t' . (The latter was not the case before 4.1.2.) To implement it, we had to change how storage engines compare strings internally. As a side effect, if you have a table where a CHAR or VARCHAR column in some row has a value with the last character less than ASCII(32) , you will have to repair this table. CHECK TABLES will tell you if this problem exists. (Bug #3152)
- Added support for DEFAULT CURRENT_TIMESTAMP and for ON UPDATE CURRENT_TIMESTAMP specifications for TIMESTAMP columns. Now you can explicitly say that a TIMESTAMP column should be set automatically to the current timestamp for INSERT and/or UPDATE statements, or even prevent the column from updating automatically. Only one column with such an auto-set feature per table is supported. TIMESTAMP columns created with earlier versions of MySQL behave as before. Behavior of TIMESTAMP columns that were created without explicit specification of default/on as earlier depends on its position in table: If it is the first TIMESTAMP column, it will be treated as having been specified as TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP . In other cases, it would be treated as a TIMESTAMP DEFAULT 0 column. NOW is supported as an alias for CURRENT_TIMESTAMP . **Warning: Incompatible change!** Unlike in previous versions, explicit specification of default values for TIMESTAMP column is never ignored and turns off the auto-set feature (unless you have CURRENT_TIMESTAMP as the default).
- **Warning: Incompatible change!** Renamed prepared statements C API functions:

Old Name	**New Name**
mysql_bind_param()	mysql_stmt_bind_param()
mysql_bind_result()	mysql_stmt_bind_result()
mysql_prepare()	mysql_stmt_prepare()
mysql_execute()	mysql_stmt_execute()
mysql_fetch()	mysql_stmt_fetch()
mysql_fetch_column()	mysql_stmt_fetch_column()
mysql_param_count()	mysql_stmt_param_count()
mysql_param_result()	mysql_stmt_param_metadata()
mysql_get_metadata()	mysql_stmt_result_metadata()
mysql_send_long_data()	mysql_stmt_send_long_data()

  Now all functions that operate with a MYSQL_STMT structure begin with the prefix mysql_stmt_ .
- **Warning: Incompatible change!** The signature of the mysql_stmt_prepare() function was changed to int mysql_stmt_prepare(MYSQL_STMT *stmt, const char *query, unsigned long length) . To create a MYSQL_STMT handle, you should use the mysql_stmt_init() function.
- SHOW GRANTS with no FOR clause or with FOR CURRENT_USER() shows the privileges for the current session.
- The improved character set support introduced in MySQL 4.1.0 for the MyISAM and HEAP storage engines is now available for InnoDB as well.
- A name of ``Primary'' no longer can be specified as an index name. (That name is reserved for the PRIMARY KEY if the table has one.) (Bug #856)
- MySQL now issues a warning when a SET or ENUM column with duplicate values in the list is created. (Bug #1427)
- Now SQL_SELECT_LIMIT variable has no influence on subqueries. (Bug #2600)
- UNHEX() function implemented. Fonctions de chaînes de caractères .
- History in command line client does not store multiple copies of identical queries that are run consecutively.
- Multi-line queries in the command line client now are stored as a single line.
- UUID() function implemented. Note that it does not work with replication yet. Fonctions diverses .
- Prepared statements with all types of subqueries fixed.
- MySQL now supports up to 64 keys per table.
- MyISAM tables now support keys up to 1000 bytes long.
- MyISAM and InnoDB tables now support index prefix lengths up to 1000 bytes long.
- If you try to create a key with a key part that is too long, and it is safe to auto-truncate it to a smaller length, MySQL now does so. A warning is generated, rather than an error.
- The ft_boolean_syntax variable now can be changed while the server is running. Variables sytème du serveur .
- REVOKE ALL PRIVILEGES, GRANT FROM user_list is changed to a more consistent REVOKE ALL PRIVILEGES, GRANT OPTION FROM user_list . (Bug #2642)
- Internal string-to-number conversion now supports only SQL:2003 compatible syntax for numbers. In particular, '0x10'+0 will not work anymore. (Actually, it worked only on some systems before, such as Linux. It did not work on others, such as FreeBSD or Solaris. Making these queries OS-independent was the goal of this change). Use CONV() to convert hexadecimal numbers to decimal. E.g. CONV(MID('0x10',3),16,10)+0 .
- mysqlhotcopy now works on NetWare.
- ALTER TABLE DROP PRIMARY KEY no longer drops the first UNIQUE index if there is no primary index. (Bug #2361)
- Added latin1_spanish_ci (Modern Spanish) collation for the latin1 character set.
- Added the ENGINE table option as a synonym for the TYPE option for CREATE TABLE and ALTER TABLE .
- Added the --default-storage-engine server option as a synonym for --default-table-type .
- Added the storage_engine system variable as a synonym for table_type .
- Added init_connect and init_slave server variables. The values should be SQL statements to be executed when each client connects or each time a slave's SQL thread starts, respectively.
- C API enhancement: SERVER_QUERY_NO_INDEX_USED and SERVER_QUERY_NO_GOOD_INDEX_USED flags are now set in the server_status field of the MYSQL structure. It is these flags that make the query to be logged as slow if mysqld was started with --log-slow-queries --log-queries-not-using-indexes .
- For replication of MEMORY (HEAP) tables: Made the master automatically write a DELETE FROM statement to its binary log when a MEMORY table is opened for the first time since master's startup. This is for the case where the slave has replicated a non-empty MEMORY table, then the master is shut down and restarted: the table is now empty on master; the DELETE FROM empties it on slave too. Note that even with this fix, between the master's restart and the first use of the table on master, the slave still has out-of-date data in the table. But if you use the init-file option to populate the MEMORY table on the master at startup, it ensures that the failing time interval is zero. (Bug #2477)
- When a session having open temporary tables terminates, the statement automatically written to the binary log is now DROP TEMPORARY TABLE IF EXISTS instead of DROP TEMPORARY TABLE , for more robustness.
- The MySQL server now returns an error if SET SQL_LOG_BIN or SET SQL_LOG_UPDATE is issued by a user without the SUPER privilege (in previous versions it just silently ignored the statement in this case).
- Changed that when the MySQL server has binary logging disabled (that is, no log-bin option was used) then no transaction binlog cache is allocated for connections (this should save binlog_cache_size bytes of memory (32 kilobytes by default) for every connection).
- Added Binlog_cache_use and Binlog_cache_disk_use status variables that count the number of transactions that used transaction binary log and that had to flush this temporary binary log to disk instead of using only buffer in memory. They can be used for tuning the binlog_cache_size system variable.
- Added option --replicate-same-server-id .

- The Mac OS X Startup Item has been moved from the directory /Library/StartupItems/MySQL to /Library/StartupItems/MySQLCOM to avoid a file name collision with the MySQL Startup Item installed with Mac OS X Server. Notes relatives à Mac OS X .

Bugs fixed:

- Fixed check of EXPLAIN of UNION . (Bug #3639)
- Fixed a bug in a query that used DISTINCT and ORDER BY by column's real name, while the column had an alias, specified in SELECT clause. (Bug #3681)
- Fixed crash of group_concat on expression with ORDER BY and external ORDER BY in a query. (Bug #3752)
- Fixed a bug in ALL / SOME subqueries in case of optimisation (key field present in subquery). (Bug #3646)
- Fixed a bug in SHOW GRANTS and EXPLAIN SELECT character set conversion. (Bug #3403)
- Prepare statements parameter do not cause error message as fields used in select list but not included in ORDER BY list.
- UNION statements did not consult SQL_SELECT_LIMIT value when set. This is now fixed properly, which means that this limit is applied to the top level query, unless LIMIT for entire UNION is used.
- Fixed a bug in multiple-table UPDATE statements that resulted in an error when one of the tables was not updated but was used in the nested query, contained therein.
- Fixed mysql_stmt_send_long_data() behavior on second execution of prepared statement and in case when long data had zero length. (Bug #1664)
- Fixed crash on second execution of prepared statement with UNION . (Bug #3577)
- Fixed incorrect results of aggregate functions in subquery with empty result set. (Bug #3505)
- You can now call mysql_stmt_attr_set(..., STMT_ATTR_UPDATE_MAX_LENGTH) to tell the client library to update MYSQL_FIELD->max_length when doing mysql_stmt_store_result() . (Bug #1647).
- Added support for unsigned integer types to prepared statement API (Bug #3035).
- Fixed crash in prepared statements when subquery in the FROM clause with parameter used. (Bug #3020)
- Fixed unknown error when negative value bind to unsigned. (Bug #3223)
- Fixed aggregate function in prepared statements. (Bug #3360)
- Incorrect error message when wrong table used in multiple-table DELETE statement in prepared statements. (Bug #3411)
- Requiring UPDATE privilege for tables which will not be updated in multiple-table UPDATE statement in prepared statements.
- Fixed prepared statement support for INSERT , REPLACE , CREATE , DELETE , SELECT , DO , SET and SHOW . All other commands are prohibited via prepared statement interface. (Bug #3398, Bug #3406, Bug #2811)
- Fixed a lot of bugs in GROUP_CONCAT() . (Bug #2695, Bug #3381, Bug #3319)
- Added optimization that allows for prepared statements using a large number of tables or tables with a large number of columns to be re-executed significantly faster. (Bug #2050)
- Fixed bug that caused execution of prepared statements to fail then table that this statement were using left table cache. This bug showed up as if this prepared statement used random garbage as column names or as server crashes. (Bug #3307)
- Fixed a problem resulting from setting the character_set_results variable to NULL . (Bug #3296)
- Fixed query cache statistics.
- Fixed bug in ANALYZE TABLE on a BDB table inside a transaction that hangs server thread. (Bug #2342)
- Fixed a symlink vulnerability in mysqlbug script. (Bug #3284)
- Fixed a bug in parallel repair (myisamchk -p , myisam_repair_threads); sometimes the repair process failed to repair a table. (Bug #1334)
- A query that uses both UNION [DISTINCT] and UNION ALL now works correctly. (Bug #1428)
- Table default character set affects LONGBLOB columns. (Bug #2821)
- CONCAT_WS() makes the server die in case of illegal mix of collations. (Bug #3087)
- UTF8 charset breaks joins with mixed column/string constant. (Bug #2959)
- Fixed DROP DATABASE to report number of tables deleted.
- Fixed memory leak in the client library when statement handle was freed on closed connection (call to mysql_stmt_close after mysql_close). (Bug #3073)
- Fixed server segfaults when processing malformed prepared statements commands. (Bug #2795, Bug #2274)
- Fixed using subqueries with OR and AND functions. (Bug #2838)
- Fixed comparison of tables/database names with --lower_case_table_names option. (Bug #2880)
- Removed try to check NULL if index built on column where NULL is impossible in IN subquery optimization. (Bug #2393)
- Fixed incorrect parsing of subqueries in the FROM clause. (Bug #2421)
- Fixed processing of RAND() in subqueries with static tables. (bug #2645)
- Fixed bug with quoting of table names in mysqldump for various values of sql_mode of server. (Bug #2591)
- Fixed bug with storing values that are out of range for DOUBLE and FLOAT columns. (Bug #2082)
- Fixed bug with compiling --with-pstack with binutils 2.13.90. (Bug #1661)
- Fixed a bug in the GRANT system. When a password was assigned to an account at the global level and then privileges were granted at the database level (without specifying any password), the existing password was replaced temporarily in memory until the next FLUSH PRIVILEGES operation or the server was restarted. (Bug #2953)
- Fixed a bug in full-text search on multi-byte character set (such as UTF8) that appeared when a search word was shorter than a matching word from the index (for example, searching for ``Uppsala'' when table data contain ``Uppsåla''). (Bug #3011)
- Fixed a bug that made Max_used_connections to be less than the actual maximum number of connections in use simultaneously.
- Fixed calculation of Index_length in HEAP table status for BTREE indexes. (Bug #2719)
- Fixed mysql_stmt_affected_rows() call to always return number of rows affected by given statement. (Bug #2247)
- Fixed crash in MATCH ... AGAINST() on a phrase search operator with a missing closing double quote. (Bug #2708)
- Fixed output of mysqldump --tab . (Bug #2705)
- Fix for a bug in UNION operations that prevented proper handling of NULL columns. This happened only if a column in the first SELECT node was NOT NULL . (Bug #2508)
- Fix for a bug in UNION operations with InnoDB storage engine, when some columns from one table where used in one SELECT statement and some were used in another SELECT statement. (Bug #2552)
- Fixed a few years old bug in the range optimizer that caused a segmentation fault on some very rare queries. (Bug #2698)
- Fixed bug with SHOW CREATE TABLE ... which didn't properly double quotes. (Bug #2593)
- Queries with subqueries in FROM clause locks all tables at once for now. This also fixed bugs in EXPLAIN of subqueries in FROM output. (Bug #2120)
- Fixed bug with mysqldump not quoting ``tricky'' names correctly. (Bug #2592)
- Fix for a bug that prevented table / column privileges from being loaded on startup. (Bug #2546)
- Fixed bug in replication with CREATE TABLE ... LIKE ... that resulted in a statement not being written to the binary log. (Bug #2557)
- Fixed memory leak in INSERT ... ON DUPLICATE KEY UPDATE ... . (Bug #2438)
- Fixed bug in the parser, making the syntax CONVERT(expr,type) legal again.
- Fixed parsing of short-form IP addresses in INET_ATON() . (Bug #2310)
- Fixed a bug in CREATE ... SELECT that sometimes caused a string column with a multi-byte character set (such as utf8) to have insufficient length to hold the data.
- Fixed a rare table corruption on adding data (INSERT , REPLACE , UPDATE , etc. but not DELETE) to a FULLTEXT index. (Bug #2417)
- Compile the MySQL-client RPM package against libreadline instead of libedit . (Bug #2289)
- Fix for a crashing bug that was caused by not setting vio_timeout() virtual function for all protocols. This bug occurred on Windows. (Bug #2025)
- Fix for a bug that caused mysql client program to erroneously cache the value of the current database. (Bug #2025)
- Fix for a bug that caused client/server communication to be broken when mysql_set_server_option() or mysql_get_server_option() were invoked. (Bug #2207)
- Fix for a bug that caused wong results when CAST() was applied on NULL to signed or unsigned integer column. (Bug #2219)
- Fix for a crashing bug that occurred in the mysql client program when database name was longer then expected. (Bug #2221)
- Fixed a bug in CHECK TABLE that sometimes resulted in a spurious error Found key at page ... that points to record outside datafile for a table with a FULLTEXT index. (Bug #2190)
- Fixed bug in GRANT with table-level privilege handling. (Bug #2178)
- Fixed bug in ORDER BY on a small column. (Bug #2147)
- Fixed a bug with the INTERVAL() function when 8 or more comparison arguments are provided. (Bug #1561)

- Packaging: Fixed a bug in the Mac OS PKG postinstall script (mysql_install_db was called with an obsolete argument).
- Packaging: Added missing file mysql_create_system_tables to the server RPM package. This bug was fixed for the 4.1.1 RPMs by updating the MySQL-server RPM from MySQL-server-4.1.1-0 to MySQL-server-4.1.1-1 . The other RPMs were not affected by this change.
- Fixed a bug in myisamchk and CHECK TABLE that sometimes resulted in a spurious error Found key at page ... that points to record outside datafile for a table with a FULLTEXT index. (Bug #1977)
- Fixed a hang in full-text indexing of strings in multi-byte (all besides utf8) charsets. (Bug #2065)
- Fixed a crash in full-text indexing of UTF8 data. (Bug #2033)
- Replication: a rare race condition in the slave SQL thread that could lead to an incorrect complaint that the relay log is corrupted. (Bug #2011)
- Replication: If a client connects to a slave server and issues an administrative statement for a table (for example, OPTIMIZE TABLE or REPAIR TABLE), this could sometimes stop the slave SQL thread. This does not lead to any corruption, but you must use START SLAVE to get replication going again. (Bug #1858)
- Replication: in the slave SQL thread, a multiple-table UPDATE could produce an incorrect complaint that some record was not found in one table, if the UPDATE was preceded by a INSERT ... SELECT . (Bug #1701)
- Replication: sometimes the master gets a non-fatal error during the execution of a statement but finally the statements succeeds (for example, a write to a MyISAM table first receives "no space left on device" but is able to finally complete, Comment MySQL gère les disques saturés); the bug was that the master forgot to reset the error code to 0 after success, so the error code got into its binary log, thus making the slave giving false alarms like "did not get the same error as on master". (Bug #2083)
- Removed a misleading "check permissions on master.info" from a replication error message, because the cause of the problem could be different from permissions. (Bug #2121)
- Fixed a crash when the replication slave was unable to create the first relay log. (Bug #2145)
- ALTER DATABASE caused the client to hang if the database did not exist. (Bug #2333)
- Multiple-table DELETE statements were never replicated by the slave if there were any replicate-*-table options. (Bug #2527)
- Fixed bug in ALTER TABLE RENAME , when rename to the table with the same name in another database silently dropped destination table if it existed. (Bug #2628)
- The MySQL server did not report any error if the query (submitted through mysql_real_query() or mysql_prepare()) was terminated by garbage characters (which can happen if you pass a wrong length parameter to mysql_real_query() or mysql_prepare()); the result was that the garbage characters were written into the binary log. (Bug #2703)
- Fixed bug in client library which caused mysql_fetch and mysql_stmt_store_result() to hang if they were called without prior call of mysql_execute() . Now they give an error instead. (Bug #2248)
- Made clearer the error message which one gets when an update is refused because of the read-only option. (Bug #2757)
- Fixed that replicate-wild-*-table rules apply to ALTER DATABASE when the table pattern is '%', like it is already the case for CREATE DATABASE and DROP DATABASE . (Bug #3000)
- Fixed that when a Rotate event is found by the slave SQL thread in the middle of a transaction, the value of Relay_Log_Pos in SHOW SLAVE STATUS remains correct. (Bug #3017)
- Corrected the master's binary log position that InnoDB reports when it is doing a crash recovery on a slave server. (Bug #3015)
- Changed the column Seconds_Behind_Master in SHOW SLAVE STATUS to never show a value of -1. (Bug #2826)
- Changed that when a DROP TEMPORARY TABLE statement is automatically written to the binary log when a session ends, the statement is recorded with an error code of value zero (this ensures that killing a SELECT on the master does not result in a superfluous error on the slave). (Bug #3063)
- Changed that when a thread handling INSERT DELAYED (also known as a delayed_insert thread) is killed, its statements are recorded with an error code of value zero (killing such a thread does not endanger replication, so we thus avoid a superfluous error on the slave). (Bug #3081)
- Fixed deadlock when two START SLAVE commands were run at the same time. (Bug #2921)
- Fixed that a statement never triggers a superfluous error on the slave, if it must be excluded given the replicate-* options. The bug was that if the statement had been killed on the master, the slave would stop. (Bug #2983)
- The --local-load option of mysqlbinlog now requires an argument.
- Fixed a segmentation fault when running LOAD DATA FROM MASTER after RESET SLAVE . (Bug #2922)
- mysqlbinlog --read-from-remote-server read all binary logs following the one that was requested. It now stops at the end of the requested file, the same was it does when reading a local binary log. (Bug #3204)
- Fixed mysqlbinlog --read-from-remote-server to print the exact positions of events in the "at #" lines. (Bug #3214)
- Fixed a rare error condition that caused the slave SQL thread spuriously to print the message Binlog has bad magic number and stop when it was not necessary to do so. (Bug #3401)
- Fixed the Exec_master_log_pos column and its disk image in the relay-log.info file to be correct if the master had version 3.23. (The value was too big by six bytes.) This bug does not exist in MySQL 5.0. (Bug #3400)
- Fixed mysqlbinlog not to forget to print a USE statement under rare circumstances where the binary log contained a LOAD DATA INFILE statement. (Bug #3415)
- Fixed a memory corruption when replicating a LOAD DATA INFILE when the master had version 3.23. Some smaller problems remain in this setup, Fonctionnalités de la réplication et problèmes connus . (Bug #3422)
- Multiple-table DELETE statements were always replicated by the slave if there were some replicate-*-ignore-table options and no replicate-*-do-table options. (Bug #3461)
- Fixed a crash of the MySQL slave server when it was built with --with-debug and replicating itself. (BUG #3568)
- Fixed that in some replication error messages, a very long query caused the rest of the message to be invisible (truncated), by putting the query last in the message. (Bug #3357)

26.2.3 Changements de la version 4.1.1 (01 décembre 2003)

This release includes all fixes in MySQL 4.0.16 and most of the fixes in MySQL 4.0.17.Functionality added or changed:

- New CHECKSUM TABLE statement for reporting table checksum values.
- Added character_set_client , character_set_connection , character_set_database , character_set_results , character_set_server , character_set_system , collation_connection , collation_database , and collation_server system variables to provide information about character sets and collations.
- It is now possible to create multiple key caches, assign table indexes to particular caches, and to preload indexes into caches. Syntaxe de CACHE INDEX . Syntaxe de LOAD INDEX INTO CACHE . Structured system variables are introduced as a means of grouping related key cache parameters. Variables système structurées .
- New COERCIBILITY() function to return the collation coercibility of a string.
- The --quote-names option for mysqldump now is enabled by default.
- mysqldump now includes a statement in the dump output to set FOREIGN_KEY_CHECKS to 0 to avoid problems with tables having to be reloaded in a particular order when the dump is reloaded. The existing FOREIGN_KEY_CHECKS value is saved and restored.
- **Important note:** If you upgrade to InnoDB -4.1.1 or higher, you cannot downgrade to a version lower than 4.1.1 any more! That is because earlier versions of InnoDB are not aware of multiple tablespaces.
- One can revoke all privileges from a user with REVOKE ALL PRIVILEGES, GRANT FROM user_list .
- Added IGNORE option for DELETE statement.
- The MySQL source distribution now also includes the MySQL Internals Manual internals.texi .
- Added mysql_set_server_option() C API client function to allow multiple statement handling in the server to be enabled or disabled.
- The mysql_next_result() C API function now returns -1 if there are no more result sets.
- Renamed CLIENT_MULTI_QUERIES connect option flag to CLIENT_MULTI_STATEMENTS . To allow for a transition period, the old option will continue to be recognized for a while.

- Require DEFAULT before table and database default character set. This enables us to use ALTER TABLE tbl_name ... CHARACTER SET=... to change the character set for all CHAR , VARCHAR , and TEXT columns in a table.
- Added MATCH ... AGAINST(... WITH QUERY EXPANSION) and the ft_query_expansion_limit server variable.
- Removed unused ft_max_word_len_for_sort system variable.
- Removed unused ft_max_word_len_for_sort variable from myisamchk .
- Full-text search now supports multi-byte character sets and the Unicode utf8 character set. (The Unicode ucs2 character set is not yet supported.)
- Phrase search in MATCH ... AGAINST (... IN BOOLEAN MODE) no longer matches partial words.
- Added aggregate function BIT_XOR() for bitwise XOR operations.
- Replication over SSL now works.
- The START SLAVE statement now supports an UNTIL clause for specifying that the slave SQL thread should be started but run only until it reaches a given position in the master's binary logs or in the slave's relay logs.
- Produce warnings even for single-row INSERT statements, not just for multiple-row INSERT statements. Previously, it was necessary to set SQL_WARNINGS=1 to generate warnings for single-row statements.
- Added delimiter (\d) command to the mysql command-line client for changing the statement delimiter (terminator). The default delimiter is semicolon.
- CHAR , VARCHAR , and TEXT columns now have lengths measured in characters rather than in bytes. The character size depends on the column's character set. This means, for example, that a CHAR(n) column for a multi-byte character set will take more storage than before. Similarly, index values on such columns are measured in characters, not bytes.
- LIMIT no longer accepts negative arguments (they used to be treated as very big positive numbers before).
- The DATABASE() function now returns NULL rather than the empty string if there is no database selected.
- Added --sql-mode=NO_AUTO_VALUE_ON_ZERO option to suppress the usual behavior of generating the next sequence number when zero is stored in an AUTO_INCREMENT column. With this mode enabled, zero is stored as zero; only storing NULL generates a sequence number.
- **Warning: Incompatible change!** Client authentication now is based on 41-byte passwords in the user table, not 45-byte passwords as in 4.1.0. Any 45-byte passwords created for 4.1.0 must be reset after running the mysql_fix_privilege_tables script.
- Added secure_auth global server system variable and --secure-auth server option that disallow authentication for accounts that have old (pre-4.1.1) passwords.
- Added --secure-auth option to mysql command-line client. If this option is set, the client refuses to send passwords in old (pre-4.1.1) format.
- **Warning: Incompatible change!** Renamed the C API mysql_prepare_result() function to mysql_get_metadata() as the old name was confusing.
- Added DROP USER 'user_name'@'host_name' statement to drop an account that has no privileges.
- The interface to aggregated UDF functions has changed a bit. You must now declare a xxx_clear() function for each aggregate function XXX() .
- Added new ADDTIME() , DATE() , DATEDIFF() , LAST_DAY() , MAKEDATE() , MAKETIME() , MICROSECOND() , SUBTIME() , TIME() , TIMEDIFF() , TIMESTAMP() , UTC_DATE() , UTC_TIME() , UTC_TIMESTAMP() , and WEEKOFYEAR() functions.
- Added new syntax for ADDDATE() and SUBDATE() . The second argument now may be a number representing the number of days to be added to or subtracted from the first date argument.
- Added new type values DAY_MICROSECOND , HOUR_MICROSECOND , MINUTE_MICROSECOND , SECOND_MICROSECOND , and MICROSECOND for DATE_ADD() , DATE_SUB() , and EXTRACT() .
- Added new %f microseconds format specifier for DATE_FORMAT() and TIME_FORMAT() .
- All queries in which at least one SELECT does not use indexes properly now are written to the slow query log when long log format is used.
- It is now possible to create a MERGE table from MyISAM tables in different databases. Formerly, all the MyISAM tables had to be in the same database, and the MERGE table had to be created in that database as well.
- Added new COMPRESS() , UNCOMPRESS() , and UNCOMPRESSED_LENGTH() functions.
- When using SET sql_mode='mode' for a complex mode (like ANSI), we now update the sql_mode variable to include all the individual options implied by the complex mode.
- Added the OLAP (On-Line Analytical Processing) function ROLLUP , which provides summary rows for each GROUP BY level.
- Added SQLSTATE codes for all server errors.
- Added mysql_sqlstate() and mysql_stmt_sqlstate() C API client functions that return the SQLSTATE error code for the last error.
- TIME columns with hour values greater than 24 were returned incorrectly to the client.
- ANALYZE TABLE , OPTIMIZE TABLE , REPAIR TABLE , and FLUSH statements are now stored in the binary log and thus replicated to slaves. This logging does not occur if the optional NO_WRITE_TO_BINLOG keyword (or its alias LOCAL) is given. Exceptions are that FLUSH LOGS , FLUSH MASTER , FLUSH SLAVE , and FLUSH TABLES WITH READ LOCK are not logged in any case. For a syntax example, see Syntaxe de FLUSH .
- New global system variable relay_log_purge to enable or disable automatic relay log purging.
- LOAD DATA now produces warnings that can be fetched with SHOW WARNINGS .
- Added support for syntax CREATE TABLE table2 (LIKE table1) that creates an empty table table2 with a definition that is exactly the same as table1 , including any indexes.
- CREATE TABLE tbl_name (...) TYPE=storage_engine now generates a warning if the named storage engine is not available. The table is still created as a MyISAM table, as before.
- Most subqueries are now much faster than before.
- Added PURGE BINARY LOGS as an alias for PURGE MASTER LOGS .
- Disabled the PURGE LOGS statement that was added in version 4.1.0. The statement now should be issued as PURGE MASTER LOGS or PURGE BINARY LOGS .
- Added SHOW BDB LOGS as an alias for SHOW LOGS .
- Added SHOW MASTER LOGS (which had been deleted in version 4.1.0) as an alias for SHOW BINARY LOGS .
- Added Slave_IO_State and Seconds_Behind_Master columns to the output of SHOW SLAVE STATUS . Slave_IO_State indicates the state of the slave I/O thread, and Seconds_Behind_Master indicates the number of seconds by which the slave is late compared to the master.
- The --lower-case-table-names=1 server option now also makes aliases case insensitive. (Bug #534)
- Changed that the relay log is flushed to disk by the slave I/O thread every time it reads a relay log event. This reduces the risk of losing some part of the relay log in case of brutal crash.

Bugs fixed:

- Fixed mysql parser not to erroneously interpret ';' character within /* ... */ comment as statement terminator.
- Fixed merging types and length of result set columns for UNION operations. The types and lengths now are determined taking into account values for all SELECT statements in the UNION , not just the first SELECT .
- Fixed a bug in privilege handling that caused connections from certain IP addresses to be assigned incorrect database-level privileges. A connection could be assigned the database privileges of the previous successful authentication from one of those IP addresses, even if the IP address username and database name were different. (Bug #1636)
- Error-handling functions were not called properly when an error resulted from [CREATE | REPLACE| INSERT] ... SELECT statements.
- HASH , BTREE , RTREE , ERRORS , and WARNINGS no longer are reserved words. (Bug #724)
- Fix for bug in ROLLUP when all tables were const tables. (Bug #714)
- Fixed a bug in UNION that prohibited NULL values from being inserted into result set columns where the first SELECT of the UNION retrieved NOT NULL columns. The type and max_length of the result column is now defined based on all UNION parts.
- Fixed name resolution of columns of reduced subqueries in unions. (Bug #745)
- Fixed memory overrun in subqueries in select list with WHERE clause bigger than outer query WHERE clause. (Bug #726)
- Fixed a bug that caused MyISAM tables with FULLTEXT indexes created in 4.0.x to be unreadable in 4.1.x.
- Fixed a data loss bug in REPAIR TABLE ... USE_FRM when used with tables that contained TIMESTAMP columns and were created in 4.0.x.
- Fixed reduced subquery processing in ORDER BY / GROUP BY clauses. (Bug #442)
- Fixed name resolution of outer columns of subquery in INSERT / REPLACE statements. (Bug #446)

- Fixed bug in marking columns of reduced subqueries. (Bug #679)
- Fixed a bug that made CREATE FULLTEXT INDEX syntax illegal.
- Fixed a crash when a SELECT that required a temporary table (marked by Using temporary in EXPLAIN output) was used as a derived table in EXPLAIN command. (Bug #251)
- Fixed a rare table corruption bug in DELETE from a big table with a **new** (created by MySQL-4.1) full-text index.
- LAST_INSERT_ID() now returns 0 if the last INSERT statement didn't insert any rows.
- Fixed missing last character in function output. (Bug #447)
- Fixed a rare replication bug when a transaction spanned two or more relay logs, and the slave was stopped while executing the part of the transaction that was in the second or later relay log. Then replication would resume at the beginning of the second or later relay log, which was incorrect. (It should resume at BEGIN , in the first relay log.) (Bug #53)
- CONNECTION_ID() now is properly replicated. (Bug #177)
- The new PASSWORD() function in 4.1 is now properly replicated. (Bug #344)
- Fixed a bug with double freed memory.
- Fixed a crashing bug in UNION operations that involved temporary tables.
- Fixed a crashing bug in DERIVED TABLES when EXPLAIN is used on a DERIVED TABLES with a join.
- Fixed a crashing bug in DELETE with ORDER BY and LIMIT caused by an uninitialized array of reference pointers.
- Fixed a bug in the USER() function caused by an error in the size of the allocated string.
- Fixed a crashing bug when attempting to create a table containing a spatial (GIS) column with a storage engine that does not support spatial types.
- Fixed a crashing bug in UNION caused by the empty select list and a non-existent column being used in some of the individual SELECT statements.
- Fixed a replication bug with a 3.23 master and a 4.0 slave: The slave lost the replicated temporary tables if FLUSH LOGS was issued on the master. (Bug #254)
- Fixed a security bug: A server compiled without SSL support still allowed connections by users who had the REQUIRE SSL option specified for their accounts.
- When an undefined user variable was used in a updating query on the master (such as INSERT INTO t VALUES(@a) , where @a had never been set by this connection before), the slave could replicate the query incorrectly if a previous transaction on the master used a user variable of the same name. (Bug #1331)
- Fixed bug with prepared statements: Using the ? prepared statement parameter as the argument to certain functions or statement clauses caused a server crash when mysql_prepare() was invoked. (Bug #1500)
- Fixed bug with prepared statements: after call to mysql_prepare placeholders became allowed in all consequent statements, even if they are not prepared (Bug #1946)
- SLAVE START (which is a deprecated syntax, START SLAVE should be used instead) could crash the slave. (Bug #2516)
- Fixed bug in ALTER TABLE RENAME , when rename to the table with the same name in another database silently dropped destination table if it existed. (Bug #2628)

26.2.4 Changements de la version 4.1.0 (03 Avril 2003 : alpha)

Functionality added or changed:

- Added --compatible option to mysqldump for producing output that is compatible with other database systems or with older MySQL servers.
- The --opt option for mysqldump now is enabled by default, as are all the options implied by --opt .
- New CHARSET() and COLLATION() functions to return the character set and collation of a string.
- Allow index type to be specified explicitly for some storage engines via USING type_name syntax in index definition.
- New function IS_USED_LOCK() for determining the connection identifier of the client that holds a given advisory lock.
- New more secure client authentication based on 45-byte passwords in the user table.
- New CRC32() function to compute cyclic redundancy check value.
- On Windows, we are now using shared memory to communicate between server and client when they are running on the same machine and you are connecting to localhost .
- REPAIR TABLE of MyISAM tables now uses less temporary disk space when sorting char columns.
- DATE / DATETIME checking is now a bit stricter to support the ability to automatically distinguish between date, datetime, and time with microseconds. For example, dates of type YYYYMMDD HHMMDD are no longer supported; you must either have separators between each DATE / TIME part or not at all.
- Server side help for all MySQL functions. One can now type help week in the mysql client and get help for the week() function.
- Added new mysql_get_server_version() C API client function.
- Fixed bug in libmysqlclient that fetched column defaults.
- Fixed bug in mysql command-line client in interpreting quotes within comments. (Bug #539)
- Added record_in_range() method to MERGE tables to be able to choose the right index when there are many to choose from.
- Replication now works with RAND() and user variables @var .
- Allow one to change mode for ANSI_QUOTES on the fly.
- EXPLAIN SELECT now can be killed. Syntaxe de KILL .
- REPAIR TABLE now can be killed. Syntaxe de KILL .
- Allow empty index lists to be specified for USE INDEX , IGNORE INDEX , and FORCE INDEX .
- DROP TEMPORARY TABLE now drops only temporary tables and doesn't end transactions.
- Added support for UNION in derived tables.
- **Warning: Incompatible change!** TIMESTAMP is now returned as a string of type 'YYYY-MM-DD HH:MM:SS' and different timestamp lengths are not supported.This change was necessary for SQL standards compliance. In a future version, a further change will be made (backward compatible with this change), allowing the timestamp length to indicate the desired number of digits of fractions of a second.
- New faster client/server protocol that supports prepared statements, bound parameters, and bound result columns, binary transfer of data, warnings.
- Added database and real table name (in case of alias) to the MYSQL_FIELD structure.
- Multi-line queries: You can now issue several queries at once and then read the results in one go.
- In CREATE TABLE foo (a INT not null primary key) the PRIMARY word is now optional.
- In CREATE TABLE the attribute SERIAL is now an alias for BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE .
- SELECT ... FROM DUAL is an alias for SELECT ... . (To be compatible with some other databases).
- If one creates a too long CHAR / VARCHAR it's now automatically changed to TEXT or BLOB ; One will get a warning in this case.
- One can specify the different BLOB / TEXT types with the syntax BLOB(length) and TEXT(length) . MySQL will automatically change it to one of the internal BLOB / TEXT types.
- CHAR BYTE is an alias for CHAR BINARY .
- VARCHARACTER is an alias for VARCHAR .
- New operators integer MOD integer and integer DIV integer .
- SERIAL DEFAULT VALUE added as an alias for AUTO_INCREMENT .
- TRUE and FALSE added as alias for 1 and 0, respectively.
- Aliases are now forced in derived tables, as per standard SQL.
- Fixed SELECT .. LIMIT 0 to return proper row count for SQL_CALC_FOUND_ROWS .
- One can specify many temporary directories to be used in a round-robin fashion with: --tmpdir=dirname1:dirname2:dirname3 .
- Subqueries: SELECT * from t1 where t1.a=(SELECT t2.b FROM t2) .
- Derived tables:

```
SELECT a.col1, b.col2
      FROM (SELECT MAX(col1) AS col1 FROM root_table) a,
```

```
       other_table b
       WHERE a.col1=b.col1;
```

- Character sets to be defined per column, table and database.
- Unicode (UTF8) support.
- New CONVERT(... USING ...) syntax for converting string values between character sets.
- BTREE index on MEMORY (HEAP) tables.
- Faster embedded server (new internal communication protocol).
- One can add a comment per column in CREATE TABLE .
- SHOW FULL COLUMNS FROM tbl_name shows column comments.
- ALTER DATABASE .
- Support for GIS (Geometrical data). Extensions spatiale de MySQL .
- SHOW [COUNT(*)] WARNINGS shows warnings from the last command.
- One can specify a column type for a column in CREATE TABLE ... SELECT by defining the column in the CREATE part.

```
CREATE TABLE foo (a TINYINT NOT NULL) SELECT b+1 AS a FROM bar;
```

- expr SOUNDS LIKE expr same as SOUNDEX(expr)=SOUNDEX(expr) .
- Added new VARIANCE(expr) function returns the variance of expr
- One can create a table from the existing table using CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table (LIKE table) . The table can be either normal or temporary.
- New options --reconnect and --skip-reconnect for the mysql client, to reconnect automatically or not if the connection is lost.
- START SLAVE (STOP SLAVE) no longer returns an error if the slave is already started (stopped); it returns a warning instead.
- SLAVE START and SLAVE STOP are no longer accepted by the query parser; use START SLAVE and STOP SLAVE instead.

26.3 Changements de la version 4.0.x (Production)

Version 4.0 of the MySQL server includes many enhancements and new features:

- The InnoDB storage engine is now included in the standard binaries, adding transactions, row-level locking, and foreign keys. Le moteur de tables InnoDB .
- A query cache, offering vastly increased performance for many applications. By caching complete result sets, later identical queries can return instantly. Le cache de requêtes MySQL .
- Improved full-text indexing with boolean mode, truncation, and phrase searching. Fonctions de recherche en texte plein .
- Enhanced MERGE tables, now supporting INSERT statements and AUTO_INCREMENT . Tables MERGE .
- UNION syntax in SELECT . Syntaxe de UNION .
- Multiple-table DELETE statements. Syntaxe de DELETE .
- libmysqld , the embedded server library. libmysqld, la bibliothèque intégrée MySQL .
- Additional GRANT privilege options for even tighter control and security. Syntaxe de GRANT et REVOKE .
- Management of user resources in the GRANT system, particularly useful for ISPs and other hosting providers. Limiter la consommation de ressources .
- Dynamic server variables, allowing configuration changes to be made without having to stop and restart the server. Syntaxe de SET .
- Improved replication code and features. Réplication avec MySQL .
- Numerous new functions and options.
- Changes to existing code for enhanced performance and reliability.

For a full list of changes, please refer to the changelog sections for each individual 4.0.x release.

26.3.1 Changements de la version 4.0.21 (pas encore publiée)

Functionality added or changed:

Bugs fixed:

- Fixed a bug with truncation of big values (> 4294967295) of 64-bit system variables. (Bug #3754)

26.3.2 Changements de la version 4.0.20 (prochainement publiée)

Functionality added or changed:

- Phrase search in MATCH ... AGAINST (... IN BOOLEAN MODE) no longer matches partial words.

Bugs fixed:

- Fixed a bug in division / reporting incorrect metadata (number of digits after the decimal point). It can be seen, e.g. in CREATE TABLE t1 SELECT "0.01"/"3" . (Bug #3612)
- Fixed a problem with non-working DROP DATABASE on some configurations (in particular, Linux 2.6.5 with ext3 are known to expose this bug). (Bug #3594)
- Fixed that in some replication error messages, a very long query caused the rest of the message to be invisible (truncated), by putting the query last in the message. (Bug #3357)

26.3.3 Changements de la version 4.0.19 (04 mai 2004)

Note: The MySQL 4.0.19 binaries were uploaded to the download mirrors on May, 10th. However, a potential crashing bug was found just before the 4.0.19 release was publicly announced and published from the 4.0 download pages at http://dev.mysql.com/ .

See (Bug #3596) for details (it was reported against MySQL−4.1, but was confirmed to affect 4.0.19 as well).

A fix for this bug was pushed into the MySQL source tree shortly after it could be reproduced and will be included in the upcoming MySQL 4.0.20, to be released shortly. We recommend users to stick to MySQL 4.0.18 for the time being, until MySQL 4.0.20 has been released (this specific bug was introduced after MySQL 4.0.18 was released, so older versions were not affected by it). We apologize for the inconvenience!Functionality added or changed:

- If length of a timestamp field is defined as 19, the timestamp will be displayed as "YYYY-MM-DD HH:MM:SS . This is done to make it easier to use tables created in MySQL 4.1 to be used in MySQL 4.0.
- If you use RAID_CHUNKS with a value > 255 it will be set to 255. This was made to ensure that all raid directories are always 2 hex bytes. (Bug #3182)
- Changed that the optimizer will now consider the index specified in FORCE INDEX clause as a candidate to resolve ORDER BY as well.
- Non-standard behavior of UNION statements has changed to the standard ones. So far, a table name in the ORDER BY clause was tolerated. From now on a proper error message is issued (Bug #3064).
- Added max_insert_delayed_threads system variable as a synonym for max_delayed_threads .
- Added query_cache_wlock_invalidate system variable. It allow emulation of MyISAM table write-locking behavior, even for queries in the query cache. (Bug #2693)
- The keyword MASTER_SERVER_ID is not reserved anymore.
- The following is mainly relevant for Mac OS X users who use a case-insensitive filesystem. This is not relevant for Windows users as InnoDB in this case always stores file names in lower case:One can now force lower_case_table_names to 0 from the command line or a configuration file. This is useful with case-insensitive filesystems when you have previously not used lower_case_table_names=1 or lower_case_table_names=2 and your have already created InnoDB tables. With lower_case_table_names=0 , InnoDB tables were stored in mixed case while setting lower_case_table_names <> 0 will now force it to lower case (to make the table names case insensitive).

  Because it's possible to crash MyISAM tables by referring to them with different case on a case−insensitive filesystem, we recommend that you use lower_case_table_names or lower_case_table_names=2 on such filesystems.

  The easiest way to convert to use lower_case_table_names=2 is to dump all your InnoDB tables with mysqldump , drop them and then restore them.
- Non-standard behavior of UNION statements has changed to the standard ones. So far, a table name in the ORDER BY clause was tolerated. From now on a proper error message is issued (Bug #3064).
- Added max_insert_delayed_threads system variable as a synonym for max_delayed_threads .
- Added query_cache_wlock_invalidate system variable. It allow emulation of MyISAM table write-locking behavior, even for queries in the query cache. (Bug #2693)
- Changed that the relay log is flushed to disk by the slave I/O thread every time it reads a relay log event. This reduces the risk of losing some part of the relay log in case of brutal crash.
- When a session having open temporary tables terminates, the statement automatically written to the binary log is now DROP TEMPORARY TABLE IF EXISTS instead of DROP TEMPORARY TABLE , for more robustness.
- Added option --replicate-same-server-id .

Bugs fixed:

- Added missing full-text variable ft_stopword_file to myisamchk .
- Don't allow stray ',' at the end of field specifications. (Bug #3481)
- INTERVAL now can handle big values for seconds, minutes and hours. (Bug #3498)
- Blank hostname did not work as documented for table and column privileges. Now it's works the same way as '%' . (Bug #3473)
- Fixed a harmless buffer overflow in replace utility. (Bug# 3541)
- Fixed SOUNDEX() to ignore non-alphabetic characters also in the beginning of the string. (Bug #3556)
- Fixed a bug in MATCH ... AGAINST() searches when another thread was doing concurrent inserts into the MyISAM table in question. The first --- full-text search --- query could return incorrect results in this case (e.g. ``phantom'' rows or not all matching rows, even an empty result set). The easiest way to check whether you are affected is to start mysqld with --skip-concurrent-insert switch and see if it helps.
- Fixed bug when doing DROP DATABASE on a directory containing non- MySQL files. Now a proper error message is returned.
- Fixed bug in ANALYZE TABLE on a BDB table inside a transaction that hangs server thread. (Bug #2342)
- Fixed a symlink vulnerability in mysqlbug script. (Bug #3284)
- Fixed core dump bug in SELECT DISTINCT where all selected parts where constants and there were hidden columns in the created temporary table. (Bug #3203)
- Fixed core dump bug in COUNT(DISTINCT) when there was a lot of values and one had a big value for max_heap_table_size .
- Fixed problem with multi-table-update and BDB tables. (Bug: #3098)
- Fixed memory leak when dropping database with RAID tables. (Bug #2882)
- Fixed core dump crash in replication during relay-log switch when the relay log went over max_relay_log_size and the slave thread did a flush_io_cache() at the same time.
- Fixed hangup bug when issuing multiple SLAVE START from different threads at the same time. (Bug #2921)
- Fixed bug when using DROP DATABASE with lower_case_table_names=2 .
- Fixed wrong result in UNION when using lower_case_table_names=2 . (Bug #2858)
- One can now kill threads that is 'stuck' in the join optimizer (can happen when there is MANY tables in the join in which case the optimizer can take really long time). (Bug #2825)
- Rollback DELETE and UPDATE statements if thread is killed. (Bug #2422)
- Ensure that all rows in an INSERT DELAYED statement is written at once if binary logging is enabled. (Bug #2491).
- Fixed bug in query cache statistic, more accurate formula linked statistic variables mentioned in the manual.
- Fixed a bug in parallel repair (myisamchk -p , myisam_repair_threads) - sometimes repair process failed to repair a table. (Bug #1334)
- Fixed bugs with names of tables, databases and columns that end to space (Bug #2985)
- Fixed a bug in multiple-table UPDATE statements involving at least one constant table. Bug was exhibited in allowing non matching row to be updated. (Bug #2996).
- Fixed all bugs in scripts for creating/upgrading system database (Bug #2874) Added tests which guarantee against such bugs in the future.

- Fixed bug in mysql command-line client in interpreting quotes within comments. (Bug #539)
- --set-character-set and --character-sets-dir options in myisamchk now work.
- Fixed a bug in mysqlbinlog that caused one pointer to be free'd twice in some cases.
- Fixed a bug in boolean full-text search, that sometimes could lead to false matches in queries with several levels of subexpressions using + operator (for example, MATCH ... AGAINST('+(+(word1 word2)) +word3*' IN BOOLEAN MODE) .
- Fixed Windows-specific portability bugs in myisam_ftdump .
- Fixed a bug in multiple-table DELETE that was caused by foreign key constraints. If the order of the tables established by MySQL optimizer did not match parent-child order, no rows were deleted and no error message was provided. (Bug #2799)
- Fixed a few years old bug in the range optimizer that caused a segmentation fault on some very rare queries. (Bug #2698)
- Replication: If a client connects to a slave server and issues an administrative statement for a table (for example, OPTIMIZE TABLE or REPAIR TABLE), this could sometimes stop the slave SQL thread. This does not lead to any corruption, but you must use START SLAVE to get replication going again. (Bug #1858) The bug was accidentally not fixed in 4.0.17 as it was unfortunately earlier said.
- Fixed that when a Rotate event is found by the slave SQL thread in the middle of a transaction, the value of Relay_Log_Pos in SHOW SLAVE STATUS remains correct. (Bug #3017)
- Corrected the master's binary log position that InnoDB reports when it is doing a crash recovery on a slave server. (Bug #3015)
- Changed that when a DROP TEMPORARY TABLE statement is automatically written to the binary log when a session ends, the statement is recorded with an error code of value zero (this ensures that killing a SELECT on the master does not result in a superfluous error on the slave). (Bug #3063)
- Changed that when a thread handling INSERT DELAYED (also known as a delayed_insert thread) is killed, its statements are recorded with an error code of value zero (killing such a thread does not endanger replication, so we thus avoid a superfluous error on the slave). (Bug #3081)
- Fixed deadlock when two START SLAVE commands were run at the same time. (Bug #2921)
- Fixed that a statement never triggers a superfluous error on the slave, if it must be excluded given the replicate-* options. The bug was that if the statement had been killed on the master, the slave would stop. (Bug #2983)
- The --local-load option of mysqlbinlog now requires an argument.
- Fixed a segmentation fault when running LOAD DATA FROM MASTER after RESET SLAVE . (Bug #2922)
- Fixed a rare error condition that caused the slave SQL thread spuriously to print the message Binlog has bad magic number and stop when it was not necessary to do so. (Bug #3401)
- Fixed the column Exec_master_log_pos (and its disk image in the relay-log.info file) to be correct if the master had version 3.23 (it was too big by 6 bytes). This bug does not exist in the 5.0 version. (Bug #3400)
- Fixed that mysqlbinlog does not forget to print a USE command under rare circumstances where the binary log contained a LOAD DATA INFILE command. (Bug #3415)
- Fixed a memory corruption when replicating a LOAD DATA INFILE when the master had version 3.23. Some smaller problems remain in this setup, Fonctionnalités de la réplication et problèmes connus . (Bug #3422)
- Multiple-table DELETE statements were always replicated by the slave if there were some replicate-*-ignore-table options and no replicate-*-do-table options. (Bug #3461)
- Fixed a crash of the MySQL slave server when it was built with --with-debug and replicating itself. (BUG #3568)

26.3.4 Changements de la version 4.0.18 (pas encore publiée)

Functionality added or changed:

- Fixed processing of LOAD DATA by mysqlbinlog in remote mode. (Bug #1378)
- New utility program myisam_ftdump was added to binary distributions.
- ENGINE is now a synonym for the TYPE option for CREATE TABLE and ALTER TABLE .
- lower_case_table_names system variable now can take a value of 2 , to store table names in mixed case on case-insensitive filesystems. It's forced to 2 if the database directory is located on a case-insensitive filesystem.
- For replication of MEMORY (HEAP) tables: Made the master automatically write a DELETE FROM statement to its binary log when a MEMORY table is opened for the first time since master's startup. This is for the case where the slave has replicated a non-empty MEMORY table, then the master is shut down and restarted: the table is now empty on master; the DELETE FROM empties it on slave too. Note that even with this fix, between the master's restart and the first use of the table on master, the slave still has out-of-date data in the table. But if you use the init-file option to populate the MEMORY table on the master at startup, it ensures that the failing time interval is zero. (Bug #2477)
- Optimizer is now better tuned for the case where the first used key part (of many) is a constant. (Bug #1679)
- Removed old non-working --old-rpl-compat server option, which was a holdover from the very first 4.0.x versions. (Bug #2428)

Bugs fixed:

- mysqlhotcopy now works on NetWare.
- DROP DATABASE could not drop databases with RAID tables that had more than nine RAID_CHUNKS . (Bug #2627)
- Fixed bug in range optimizer when using overlapping ranges. (Bug #2448)
- Limit wait_timeout to 2147483 on Windows (OS limit). (Bug #2400)
- Fixed bug when --init-file crashes MySQL if it contains a large SELECT . (Bug #2526)
- SHOW KEYS now shows NULL in the Sub_part column for FULLTEXT indexes.
- The signal thread's stack size was increased to enable mysqld to run on Debian/IA-64 with a TLS-enabled glibc . (Bug #2599)
- Now only the SELECT privilege is needed for tables that are only read in multiple-table UPDATE statements. (Bug #2377)
- Give proper error message if one uses LOCK TABLES ... ; INSERT ... SELECT and one used the same table in the INSERT and SELECT part. (Bug #2296)
- SELECT INTO ... DUMPFILE now deletes the generated file on error.
- Fixed foreign key reference handling to allow references to column names that contain spaces. (Bug #1725)
- Fixed problem with index reads on character columns with BDB tables. The symptom was that data could be returned in the wrong lettercase. (Bug #2509)
- Fixed a spurious table corruption problem that could sometimes appear on tables with indexed TEXT columns if these columns happened to contain values having trailing spaces. This bug was introduced in 4.0.17.
- Fixed a problem where some queries could hang if a condition like indexed_TEXT_column = expr was present and the column contained values having trailing spaces. This bug was introduced in 4.0.17.
- Fixed a bug that could cause incorrect results from a query that involved range conditions on indexed TEXT columns that happened to contain values having trailing spaces. This bug was introduced in 4.0.17. (Bug #2295)
- Fixed incorrect path names in some of the manual pages. (Bug #2270)
- Fixed spurious ``table corrupted'' errors in parallel repair operations. Variables sytème du serveur .
- Fixed a crashing bug in parallel repair operations. Variables sytème du serveur .
- Fixed bug in updating MyISAM tables for BLOB values longer than 16MB. (Bug #2159)
- Fixed bug in mysqld_safe when running multiple instances of MySQL. (Bug #2114)
- Fixed a bug in using HANDLER statement with tables not from a current database. (Bug #2304)
- Fixed a crashing bug that occurred due to the fact that multiple-table UPDATE statements did not check that there was only one table to be updated. (Bug #2103)
- Fixed a crashing bug that occurred due to BLOB column type index size being calculated incorrectly in MIN() and MAX() optimizations. (Bug #2189)
- Fixed a bug with incorrect syntax for LOCK TABLES in mysqldump . (Bug #2242)
- Fixed a bug in mysqld_safe that caused mysqld to generate a warning about duplicate user=xxx options if this option was specified in the [mysqld] or [server] sections of my.cnf . (Bug #2163)

- INSERT DELAYED ... SELECT ... could cause table corruption because tables were not locked properly. This is now fixed by ignoring DELAYED in this context. (Bug #1983)
- Replication: Sometimes the master gets a non-fatal error during the execution of a statement that does not immediately succeed. (For example, a write to a MyISAM table may first receive ``no space left on device,'' but later complete when disk space becomes available. Comment MySQL gère les disques saturés .) The bug was that the master forgot to reset the error code to 0 after success, so the error code got into its binary log, thus causing the slave to issue false alarms such as ``did not get the same error as on master.'' (Bug #2083)
- Removed a misleading ``check permissions on master.info'' from a replication error message, because the cause of the problem could be something other than permissions. (Bug #2121)
- Fixed a crash when the replication slave was unable to create the first relay log. (Bug #2145)
- Replication of LOAD DATA INFILE for an empty file from a 3.23 master to a 4.0 slave caused the slave to print an error. (Bug #2452)
- When automatically forcing lower_case_table_names to 1 if the file system was case insensitive, mysqld could crash. This bug existed only in MySQL 4.0.17. (Bug #2481)
- Restored ability to specify default values for TIMESTAMP columns that was erroneously disabled in previous release. (Bug #2539) Fixed SHOW CREATE TABLE to reflect these values. (Bug #1885) Note that because of the auto-update feature for the first TIMESTAMP column in a table, it makes no sense to specify a default value for the column. Any such default will be silently ignored (unless another TIMESTAMP column is added before this one). Also fixed the meaning of the DEFAULT keyword when it is used to specify the value to be inserted into a TIMESTAMP column other than the first. (Bug #2464)
- Fixed bug for out-of-range arguments on QNX platform that caused UNIX_TIMESTAMP() to produce incorrect results or that caused non-zero values to be inserted into TIMESTAMP columns. (Bug #2523) Also, current time zone now is taken into account when checking if datetime values satisfy both range boundaries for TIMESTAMP columns. The range allowed for a TIMESTAMP column is time zone-dependent and equivalent to a range of 1970-01-01 00:00:01 UTC to 2037-12-31 23:59:59 UTC.
- Multiple-table DELETE statements were never replicated by the slave if there were any replicate-*-table options. (Bug #2527)
- Changes to session counterparts of variables query_prealloc_size , query_alloc_block_size , trans_prealloc_size , trans_alloc_block_size now have an effect. (Bug #1948)
- Fixed bug in ALTER TABLE RENAME , when rename to the table with the same name in another database silently dropped destination table if it existed. (Bug #2628)

26.3.5 Changements de la version 4.0.17 (14 décembre 2003)

Functionality added or changed:

- mysqldump no longer dumps data for MERGE tables. (Bug #1846)
- lower_case_table_names is now forced to 1 if the database directory is located on a case-insensitive file system. (Bug #1812)
- Symlink creation is now disabled on systems where realpath() doesn't work. (Before one could use CREATE TABLE .. DATA DIRECTORY=.. even if HAVE_BROKEN_REALPATH was defined. This is now disabled to avoid problems when running ALTER TABLE).
- Inserting a negative AUTO_INCREMENT value in a MyISAM table no longer updates the AUTO_INCREMENT counter to a big unsigned value. (Bug #1366)
- Added four new modes to WEEK(..., mode) function. Fonctions de date et heure . (Bug #1178)
- Allow UNION DISTINCT syntax.
- mysql_server_init() now returns 1 if it can't initialize the environment. (Previously mysql_server_init() called exit(1) if it could not create a key with pthread_key_create() . (Bug #2062)
- Allow spaces in Windows service names.
- Changed the default Windows service name for mysqld from MySql to MySQL . This should not affect usage, because service names are not case sensitive.
- When you install mysqld as a service on Windows systems, mysqld will read startup options in option files from the option group with the same name as the service name. (Except when the service name is MySQL).

Bugs fixed:

- One can now configure MySQL as a Windows service as a normal user. (Bug #1802). Thanks to Richard Hansen for fixing this.
- Database names are now compared in lowercase in ON clauses when lower_case_table_names is set. (Bug #1736)
- IGNORE ... LINES option to LOAD DATA INFILE didn't work when used with fixed length rows. (Bug #1704)
- Fixed problem with UNIX_TIMESTAMP() for timestamps close to 0. (Bug #1998)
- Fixed problem with character values greater than 128 in the QUOTE() function. (Bug #1868)
- Fixed searching of TEXT with end space. (Bug #1651)
- Fixed caching bug in multiple-table updates where same table was used twice. (Bug #1711)
- Fixed directory permissions for the MySQL-server RPM documentation directory. (Bug #1672)
- Fixed server crash when updating an ENUM column that is set to the empty string (for example, with REPLACE()). (Bug #2023)
- mysql client program now correctly prints connection identifier returned by mysql_thread_id() as unsigned integer rather than as signed integer. (Bug #1951)
- FOUND_ROWS() could return incorrect number of rows after a query with an impossible WHERE condition. (Bug #1468)
- SHOW DATABASES no longer shows .sym files (on Windows) that do not point to a valid directory. (Bug #1385)
- Fixed a possible memory leak on Mac OS X when using the shared libmysql.so library. (from pthread_key_create()). (Bug #2061)
- Fixed bug in UNION statement with alias * . (Bug #1249)
- Fixed a bug in DELETE ... ORDER BY ... LIMIT where the rows where not deleted in the proper order. (Bug #1024, Bug #1697).
- Fixed serious problem with multi-threaded programs on Windows that used the embedded MySQL libraries. (Locks of tables were not handled correctly between different threads).
- Code cleanup: Fixed a few code defects (potential memory leaks, null pointer dereferences, uninitialized variables). Thanks to Reasoning Inc. for informing us about these findings.
- Fixed a buffer overflow error that occurred with prepended '0' characters in some columns of type DECIMAL . (Bug #2128)
- Filesort was never shown in EXPLAIN if query contained an ORDER BY NULL clause. (Bug #1335)
- Fixed invalidation of whole query cache on DROP DATABASE . (Bug #1898)
- Fixed bug in range optimizer that caused wrong results for some unlikely AND / OR queries. (Bug #1828)
- Fixed a crash in ORDER BY when ordering by expression and identifier. (Bug #1945)
- Fixed a crash in an open HANDLER when an ALTER TABLE was executed in a different connection. (Bug #1826)
- Fixed a bug in trunc* operator of full-text search which sometimes caused MySQL not to find all matched rows.
- Fixed bug in prepending '0' characters to DECIMAL column values.
- Fixed optimizer bug, introduced in 4.0.16, when REF access plan was preferred to more efficient RANGE on another column.
- Fixed problem when installing a MySQL server as a Windows service using a command of the form mysqld --install mysql --defaults-file=path-to-file . (Bug #1643)
- Fixed an incorrect result from a query that uses only const tables (such as one-row tables) and non-constant expression (such as RAND()). (Bug #1271)
- Fixed bug when the optimizer did not take SQL_CALC_FOUND_ROWS into account if LIMIT clause was present. (Bug #1274)
- mysqlbinlog now asks for a password at the console when the -p or --password option is used with no argument. This is consistent with the way that other clients such mysqladmin and mysqldump already behave. **Note**: A consequence of this change is that it is no longer possible to invoke mysqlbinlog as mysqlbinlog -p pass_val (with a space between the -p option and the following password value). (Bug #1595)
- Fixed bug accidentally introduced in 4.0.16 where the slave SQL thread deleted its replicated temporary tables when STOP SLAVE was issued.
- In a ``chain'' replication setup A->B->C , if 2 sessions on A updated temporary tables of the same name at the same time, the binary log of B became incorrect, resulting in C becoming confused. (Bug #1686)

- In a ``chain'' replication setup A->B->C , if STOP SLAVE was issued on B while it was replicating a temporary table from A , then when START SLAVE was issued on B , the binary log of B became incorrect, resulting in C becoming confused. (Bug #1240)
- When MASTER_LOG_FILE and MASTER_LOG_POS were not specified, CHANGE MASTER used the coordinates of the slave I/O thread to set up replication, which broke replication if the slave SQL thread lagged behind the slave I/O thread. This caused the slave SQL thread to lose some events. The new behavior is to use the coordinates of the slave SQL thread instead. CHANGE MASTER TO . (Bug #1870)
- Now if integer is stored or converted to TIMESTAMP or DATETIME value checks of year, month, day, hour, minute and second ranges are performed and numbers representing illegal timestamps are converted to 0 value. This behavior is consistent with manual and with behavior of string to TIMESTAMP / DATETIME conversion. (Bug #1448)
- Fixed bug when BIT_AND() and BIT_OR() group functions returned incorrect value if SELECT used a temporary table and no rows were found. (Bug #1790).
- BIT_AND() is now unsigned in all contexts. This means that it will now return 18446744073709551615 (= 0xffffffffffffffff) instead of -1 if there were no rows in the result.
- Fixed bug with BIT_AND() still returning signed value for an empty set in some cases. (Bug #1972)
- Fixed bug with ^ (XOR) and >> (bit shift) still returning signed value in some cases. (Bug #1993)
- Replication: a rare race condition in the slave SQL thread, which could lead to a wrong complain that the relay log is corrupted. (Bug #2011)
- Replication: if an administrative command on a table (OPTIMIZE TABLE , REPAIR TABLE etc) was run on the slave, this could sometimes stop the slave SQL thread (this did not led to any corruption; one just had to type START SLAVE to get replication going again). (Bug #1858)
- Replication: in the slave SQL thread, a multi-table UPDATE could produce a wrong complain that some record was not found in one table, if the UPDATE was preceded by a INSERT ... SELECT . (Bug #1701)
- Fixed deficiency in MySQL code which is responsible for scanning directories. This deficiency caused SHOW TABLE STATUS to be very slow for big number of tables in database even if single particular table were specified. (Bug #1952)

26.3.6 Changements de la version 4.0.16 (17 octobre 2003)

Functionality added or changed:

- Option values in option files now may be quoted. This is useful for values that contain whitespace or comment characters.
- Write memory allocation information to error log when doing mysqladmin debug . This works only on systems that support the mallinfo() call (like newer Linux systems).
- Added the following new server variables to allow more precise memory allocation: range_alloc_block_size , query_alloc_block_size , query_prealloc_size , transaction_alloc_block_size , and transaction_prealloc_size .
- mysqlbinlog now reads option files. To make this work, you must now specify --read-from-remote-server when reading binary logs from a MySQL server. (Note that using a remote server is deprecated and may disappear in future mysqlbinlog versions).
- Block SIGPIPE signals also for non-threaded programs. The blocking is moved from mysql_init() to mysql_server_init() , which is automatically called on the first call to mysql_init() .
- Added --libs_r and --include options to mysql_config .
- New `> prompt for mysql . This prompt is similar to the '> and "> prompts, but indicates that an identifier quoted with backticks was begun on an earlier line and the closing backtick has not yet been seen.
- Updated mysql_install_db to be able to use the local machine's IP address instead of the hostname when building the initial grant tables if skip-name-resolve has been specified. This option can be helpful on FreeBSD to avoid thread-safety problems with the FreeBSD resolver libraries. (Thanks to Jeremy Zawodny for the patch.)
- A documentation change: Added a note that when backing up a slave, it is necessary also to back up the master.info and relay-log.info files, as well as any SQL_LOAD-* files located in the directory specified by the --slave-load-tmpdir option. All these files are needed when the slave resumes replication after you restore the slave's data.

Bugs fixed:

- Fixed a spurious error ERROR 14: Can't change size of file (Errcode: 2) on Windows in DELETE FROM tbl_name without a WHERE clause or TRUNCATE TABLE tbl_name , when tbl_name is a MyISAM table. (Bug #1397)
- Fixed a bug that resulted in thr_alarm queue is full warnings after increasing the max_connections variable with SET GLOBAL . (Bug #1435)
- Made LOCK TABLES to work when Lock_tables_priv is granted on the database level and Select_priv is granted on the table level.
- Fixed crash of FLUSH QUERY CACHE on queries that use same table several times (Bug #988).
- Fixed core dump bug when setting an enum system variable (such as SQL_WARNINGS) to NULL .
- Extended the default timeout value for Windows clients from 30 seconds to 1 year. (The timeout that was added in MySQL 4.0.15 was way too short). This fixes a bug that caused ERROR 2013: Lost connection to MySQL server during query for queries that lasted longer than 30 seconds, if the client didn't specify a limit with mysql_options() . Users of 4.0.15 on Windows should upgrade to avoid this problem.
- More ``out of memory'' checking in range optimizer.
- Fixed and documented a problem when setting and using a user variable within the same SELECT statement. (Bug #1194).
- Fixed bug in overrun check for BLOB values with compressed tables. This was a bug introduced in 4.0.14. It caused MySQL to regard some correct tables containing BLOB values as corrupted. (Bug #770, Bug #1304, and maybe Bug #1295)
- SHOW GRANTS showed USAGE instead of the real column-level privileges when no table-level privileges were given.
- When copying a database from the master, LOAD DATA FROM MASTER dropped the corresponding database on the slave, thus erroneously dropping tables that had no counterpart on the master and tables that may have been excluded from replication using replicate-*-table rules. Now LOAD DATA FROM MASTER no longer drops the database. Instead, it drops only the tables that have a counterpart on the master and that match the replicate-*-table rules. replicate-*-db rules can still be used to include or exclude a database as a whole from LOAD DATA FROM MASTER . A database will also be included or excluded as a whole if there are some rules like replicate-wild-do-table=db1.% or replicate-wild-ignore-table=db1.% , as is already the case for CREATE DATABASE and DROP DATABASE in replication. (Bug #1248)
- Fixed a bug where mysqlbinlog crashed with a segmentation fault when used with the -h or --host option. (Bug #1258)
- Fixed a bug where mysqlbinlog crashed with a segmentation fault when used on a binary log containing only final events for LOAD DATA . (Bug #1340)
- mysqlbinlog will not reuse temporary file names from previous runs. Previously mysqlbinlog failed if was used several times on the same binary log file that contained a LOAD DATA command.
- Fixed compilation problem when compiling with OpenSSL 0.9.7 with disabled old DES support (If OPENSSL_DISABLE_OLD_DES_SUPPORT option was enabled).
- Fixed a bug when two (or more) MySQL servers were running on the same machine, and they were both slaves, and at least one of them was replicating some LOAD DATA INFILE command from its master. The bug was that one slave MySQL server sometimes deleted the SQL_LOAD-* files (used for replication of LOAD DATA INFILE and located in the slave-load-tmpdir directory, which defaults to tmpdir) belonging to the other slave MySQL server of this machine, if these slaves had the same slave-load-tmpdir directory. When that happened, the other slave could not replicate LOAD DATA INFILE and complained about not being able to open some SQL_LOAD-* file. (Bug #1357)
- If LOAD DATA INFILE failed for a small file, the master forgot to write a marker (a Delete_file event) in its binary log, so the slave could not delete 2 files (SQL_LOAD-*.info and SQL_LOAD-*.data from its tmpdir . (Bug #1391)
- On Windows, the slave forgot to delete a SQL_LOAD-*.info file from tmpdir after successfully replicating a LOAD DATA INFILE command. (Bug #1392)
- When a connection terminates, MySQL writes DROP TEMPORARY TABLE statements to the binary log for all temporary tables which the connection had not explicitly dropped. MySQL forgot to use backticks to quote the database and table names in the statement. (Bug #1345)

- On some 64-bit machines (some HP-UX and Solaris machines), a slave installed with the 64-bit MySQL binary could not connect to its master (it connected to itself instead). (Bug #1256, Bug #1381)
- Code was introduced in MySQL 4.0.15 for the slave to detect that the master had died while writing a transaction to its binary log. This code reported an error in a legal situation: When the slave I/O thread was stopped while copying a transaction to the relay log, the slave SQL thread would later pretend that it found an unfinished transaction. (Bug #1475)

26.3.7 Changements de la version 4.0.15 (03 septembre 2003)

IMPORTANT :

If you are using this release on Windows, you should upgrade at least your clients (any program that uses `libmysql.lib`) to 4.0.16 or above. This is because the 4.0.15 release had a bug in the Windows client library that causes Windows clients using the library to die with a `Lost connection to MySQL server during query` error for queries that take more than 30 seconds. This problem is specific to Windows; clients on other platforms are unaffected.

Functionality added or changed:

- mysqldump now correctly quotes all identifiers when communicating with the server. This assures that during the dump process, mysqldump will never send queries to the server that result in a syntax error. This problem is **not** related to the mysqldump program's output, which was not changed. (Bug #1148)
- Change result set metadata information so that MIN() and MAX() report that they can return NULL (this is true because an empty set will return NULL). (Bug #324)
- Produce an error message on Windows if a second mysqld server is started on the same TCP/IP port as an already running mysqld server.
- The mysqld server variables wait_timeout , net_read_timeout , and net_write_timeout now work on Windows. One can now also set timeouts for read and writes in Windows clients with mysql_options() .
- Added option --sql-mode=NO_DIR_IN_CREATE to make it possible for slaves to ignore INDEX DIRECTORY and DATA DIRECTORY options given to CREATE TABLE . When this is mode is on, SHOW CREATE TABLE will not show the given directories.
- SHOW CREATE TABLE now shows the INDEX DIRECTORY and DATA DIRECTORY options, if they were specified when the table was created.
- The open_files_limit server variable now shows the real open files limit.
- MATCH ... AGAINST() in natural language mode now treats words that are present in more than 2,000,000 rows as stopwords.
- The Mac OS X installation disk images now include an additional MySQLStartupItem.pkg package that enables the automatic startup of MySQL on system bootup. Installer MySQL sous Mac OS X .
- Most of the documentation included in the binary tarball distributions (.tar.gz) has been moved into a subdirectory docs . Plans d'installation .
- The manual is now included as an additional info file in the binary distributions. (Bug #1019)
- The binary distributions now include the embedded server library (libmysqld.a) by default. Due to a linking problem with non-gcc compilers, it was not included in all packages of the initial 4.0.15 release. The affected packages were rebuilt and released as 4.0.15a. Bibliothèque MySQL embarquée .
- MySQL can now use range optimization for BETWEEN with non-constant limits. (Bug #991)
- Replication error messages now include the default database, so that users can check which database the failing query was run for.
- A documentation change: Added a paragraph about how the binlog-do-db and binlog-ignore-db options are tested against the database on the master (Le log binaire des mises à jour), and a paragraph about how replicate-do-db , replicate-do-table and analogous options are tested against the database and tables on the slave (Options de réplication).
- Now the slave does not replicate SET PASSWORD if it is configured to exclude the mysql database from replication (using for example replicate-wild-ignore-table=mysql.%). This was already the case for GRANT and REVOKE since version 4.0.13 (though there was Bug #980 in 4.0.13 &4.0.14, which has been fixed in 4.0.15).
- Rewrote the information shown in the State column of SHOW PROCESSLIST for replication threads and for MASTER_POS_WAIT() and added the most common states for these threads to the documentation, Détails d'implémentation de la réplication .
- Added a test in replication to detect the case where the master died in the middle of writing a transaction to the binlog; such unfinished transactions now trigger an error message on the slave.
- A GRANT command that creates an anonymous user (that is, an account with an empty username) no longer requires FLUSH PRIVILEGES for the account to be recognized by the server. (Bug #473)
- CHANGE MASTER now flushes relay-log.info . Previously this was deferred to the next run of START SLAVE , so if mysqld was shutdown on the slave after CHANGE MASTER without having run START SLAVE , the relay log's name and position were lost. At restart they were reloaded from relay-log.info , thus reverting to their old (incorrect) values from before CHANGE MASTER and leading to error messages (as the old relay log did not exist any more) and the slave threads refusing to start. (Bug #858)

Bugs fixed:

- Fixed buffer overflow in password handling which could potentially be exploited by MySQL users with ALTER privilege on the mysql.user table to execute random code or to gain shell access with the UID of the mysqld process (thanks to Jedi/Sector One for spotting and reporting this bug).
- Fixed server crash on FORCE INDEX in a query that contained "Range checked for each record" in the EXPLAIN output. (Bug #1172)
- Fixed table/column grant handling - proper sort order (from most specific to less specific, Contrôle d'accès, étape 2 : vérification des requêtes) was not honored. (Bug #928)
- Fixed rare bug in MYISAM introduced in 4.0.3 where the index file header was not updated directly after an UPDATE of split dynamic rows. The symptom was that the table had a corrupted delete-link if mysqld was shut down or the table was checked directly after the update.
- Fixed Can't unlock file error when running myisamchk --sort-index on Windows. (Bug #1119)
- Fixed possible deadlock when changing key_buffer_size while the key cache was actively used. (Bug #1088)
- Fixed overflow bug in MyISAM and ISAM when a row is updated in a table with a large number of columns and at least one BLOB/TEXT column.
- Fixed incorrect result when doing UNION and LIMIT #,# when one didn't use braces around the SELECT parts.
- Fixed incorrect result when doing UNION and ORDER BY .. LIMIT # when one didn't use braces around the SELECT parts.
- Fixed problem with SELECT SQL_CALC_FOUND_ROWS ... UNION ALL ... LIMIT # where FOUND_ROWS() returned incorrect number of rows.
- Fixed unlikely stack bug when having a BIG expression of type 1+1-1+1-1... in certain combinations. (Bug #871)
- Fixed the bug that sometimes prevented a table with a FULLTEXT index from being marked as "analyzed".
- Fixed MySQL so that the column length (in C API) for the second column in SHOW CREATE TABLE is always larger than the data length. The only known application that was affected by the old behavior was Borland dbExpress, which truncated the output from the command. (Bug #1064)
- Fixed crash in comparisons of strings using the tis620 character set. (Bug #1116)
- Fixed ISAM bug in MAX() optimization.
- myisamchk --sort-records=N no longer marks table as crashed if sorting failed because of an inappropriate key. (Bug #892)

- Fixed a minor bug in MyISAM compressed table handling that sometimes made it impossible to repair compressed table in "Repair by sort" mode. "Repair with keycache" (myisamchk --safe-recover) worked, though. (Bug #1015)
- Fixed bug in propagating the version number to the manual included in the distribution files. (Bug #1020)
- Fixed key sorting problem (a PRIMARY key declared for a column that is not explicitly marked NOT NULL was sorted after a UNIQUE key for a NOT NULL column).
- Fixed the result of INTERVAL when applied to a DATE value. (Bug #792)
- Fixed compiling of the embedded server library in the RPM spec file. (Bug #959)
- Added some missing files to the RPM spec file and fixed some RPM building errors that occurred on Red Hat Linux 9. (Bug #998)
- Fixed incorrect XOR evaluation in WHERE clause. (Bug #992)
- Fixed bug with processing in query cache merged tables constructed from more then 255 tables. (Bug #930)
- Fixed incorrect results from outer join query (e.g. LEFT JOIN) when ON condition is always false, and range search in used. (Bug #926)
- Fixed a bug causing incorrect results from MATCH ... AGAINST() in some joins. (Bug #942)
- MERGE tables do not ignore "Using index" (from EXPLAIN output) anymore.
- Fixed a bug that prevented an empty table from being marked as "analyzed". (Bug #937)
- Fixed myisamchk --sort-records crash when used on compressed table.
- Fixed slow (as compared to 3.23) ALTER TABLE and related commands such as CREATE INDEX . (Bug #712)
- Fixed segmentation fault resulting from LOAD DATA FROM MASTER when the master was running without the --log-bin option. (Bug #934)
- Fixed a security bug: A server compiled without SSL support still allowed connections by users that had the REQUIRE SSL option specified for their accounts.
- Fixed a random bug: Sometimes the slave would replicate GRANT or REVOKE queries even if it was configured to exclude the mysql database from replication (for example, using replicate-wild-ignore-table=mysql.%). (Bug #980)
- The Last_Errno and Last_Error fields in the output of SHOW SLAVE STATUS are now cleared by CHANGE MASTER and when the slave SQL thread starts. (Bug #986)
- A documentation mistake: It said that RESET SLAVE does not change connection information (master host, port, user, and password), whereas it does. The statement resets these to the startup options (master-host etc) if there were some. (Bug #985)
- SHOW SLAVE STATUS now shows correct information (master host, port, user, and password) after RESET SLAVE (that is, it shows the new values, which are copied from the startup options if there were some). (Bug #985)
- Disabled propagation of the original master's log position for events because this caused unexpected values for Exec_Master_Log_Pos and problems with MASTER_POS_WAIT() in A->B->C replication setup. (Bug #1086)
- Fixed a segfault in mysqlbinlog when --position=x was used with x being between a Create_file event and its fellow Append_block , Exec_load or Delete_file events. (Bug #1091)
- mysqlbinlog printed superfluous warnings when using --database , which caused syntax errors when piped to mysql . (Bug #1092)
- Made mysqlbinlog --database filter LOAD DATA INFILE too (previously, it filtered all queries except LOAD DATA INFILE). (Bug #1093)
- mysqlbinlog in some cases forgot to put a leading '#' in front of the original LOAD DATA INFILE (this command is displayed only for information, not to be run; it is later reworked to LOAD DATA LOCAL with a different filename, for execution by mysql). (Bug #1096)
- binlog-do-db and binlog-ignore-db incorrectly filtered LOAD DATA INFILE (it was half-written to the binary log). This resulted in a corrupted binary log, which could cause the slave to stop with an error. (Bug #1100)
- When, in a transaction, a transactional table (such as an InnoDB table) was updated, and later in the same transaction a non-transactional table (such as a MyISAM table) was updated using the updated content of the transactional table (with INSERT ... SELECT for example), the queries were written to the binary log in an incorrect order. (Bug #873)
- When, in a transaction, INSERT ... SELECT updated a non-transactional table, and ROLLBACK was issued, no error was returned to the client. Now the client is warned that some changes could not be rolled back, as this was already the case for normal INSERT . (Bug #1113)
- Fixed a potential bug: When STOP SLAVE was run while the slave SQL thread was in the middle of a transaction, and then CHANGE MASTER was used to point the slave to some non-transactional statement, the slave SQL thread could get confused (because it would still think, from the past, that it was in a transaction).

26.3.8 Changements de la version 4.0.14 (18 juillet 2003)

Functionality added or changed:

- Added default_week_format system variable. The value is used as the default mode for the WEEK() function.
- mysqld now reads an additional option file group having a name corresponding to the server's release series: [mysqld-4.0] for 4.0.x servers, [mysqld-4.1] for 4.1.x servers, and so forth. This allows options to be specified on a series-specific basis.
- The CONCAT_WS() function no longer skips empty strings. (Bug #586).
- InnoDB now supports indexing a prefix of a column. This means, in particular, that BLOB and TEXT columns can be indexed in InnoDB tables, which was not possible before.
- A documentation change: Function INTERVAL(NULL, ...) returns -1 .
- Enabled INSERT from SELECT when the table into which the records are inserted is also a table listed in the SELECT .
- Allow CREATE TABLE and INSERT from any UNION .
- The SQL_CALC_FOUND_ROWS option now always returns the total number of rows for any UNION .
- Removed --table option from mysqlbinlog to avoid repeating mysqldump functionality.
- Comment lines in option files can now start from the middle of a line, too (like basedir=c:\mysql # installation directory).
- Changed optimizer slightly to prefer index lookups over full table scans in some boundary cases.
- Added thread-specific max_seeks_for_key variable that can be used to force the optimizer to use keys instead of table scans even if the cardinality of the index is low.
- Added optimization that converts LEFT JOIN to normal join in some cases.
- A documentation change: added a paragraph about failover in replication (how to use a surviving slave as the new master, how to resume to the original setup). FAQ de la réplication .
- A documentation change: added warning notes about safe use of the CHANGE MASTER command. CHANGE MASTER TO .
- MySQL now issues a warning (not an error, as in 4.0.13) when it opens a table that was created with MySQL 4.1.
- Added --nice option to mysqld_safe to allow setting the niceness of the mysqld process. (Thanks to Christian Hammers for providing the initial patch.) (Bug #627)
- Added --read-only option to cause mysqld to allow no updates except from slave threads or from users with the SUPER privilege. (Original patch from Markus Benning).
- SHOW BINLOG EVENTS FROM x where x is less than 4 now silently converts x to 4 instead of printing an error. The same change was done for CHANGE MASTER TO MASTER_LOG_POS=x and CHANGE MASTER TO RELAY_LOG_POS=x .
- mysqld now only adds an interrupt handler for the SIGINT signal if you start it with the new --gdb option. This is because some MySQL users encountered strange problems when they accidentally sent SIGINT to mysqld threads.
- RESET SLAVE now clears the Last_Errno and Last_Error fields in the output of SHOW SLAVE STATUS .
- Added max_relay_log_size variable; the relay log will be rotated automatically when its size exceeds max_relay_log_size . But if max_relay_log_size is 0 (the default), max_binlog_size will be used (as in older versions). max_binlog_size still applies to binary logs in any case.
- FLUSH LOGS now rotates relay logs in addition to the other types of logs it already rotated.

Bugs fixed:

- Comparison/sorting for latin1_de character set was rewritten. The old algorithm could not handle cases like "sä" > "ßa" . Jeu de caractères allemand . In rare cases it resulted in table corruption.

- Fixed a problem with the password prompt on Windows. (Bug #683)
- ALTER TABLE ... UNION=(...) for MERGE table is now allowed even if some underlying MyISAM tables are read-only. (Bug #702)
- Fixed a problem with CREATE TABLE t1 SELECT x'41' . (Bug #801)
- Removed some incorrect lock warnings from the error log.
- Fixed memory overrun when doing REPAIR TABLE on a table with a multiple-part auto_increment key where one part was a packed CHAR .
- Fixed a probable race condition in the replication code that could potentially lead to INSERT statements not being replicated in the event of a FLUSH LOGS command or when the binary log exceeds max_binlog_size . (Bug #791)
- Fixed a crashing bug in INTERVAL and GROUP BY or DISTINCT . (Bug #807)
- Fixed bug in mysqlhotcopy so it actually aborts for unsuccessful table copying operations. Fixed another bug so that it succeeds when there are thousands of tables to copy. (Bug #812)
- Fixed problem with mysqlhotcopy failing to read options from option files. (Bug #808)
- Fixed bugs in optimizer that sometimes prevented MySQL from using FULLTEXT indexes even though it was possible (for example, in SELECT * FROM t1 WHERE MATCH a,b AGAINST("index") > 0).
- Fixed a bug with ``table is full'' in UNION operations.
- Fixed a security problem that enabled users with no privileges to obtain information on the list of existing databases by using SHOW TABLES and similar commands.
- Fixed a stack problem on UnixWare/OpenUnix.
- Fixed a configuration problem on UnixWare/OpenUNIX and OpenServer.
- Fixed a stack overflow problem in password verification.
- Fixed a problem with max_user_connections .
- HANDLER without an index now works properly when a table has deleted rows. (Bug #787)
- Fixed a bug with LOAD DATA in mysqlbinlog . (Bug #670)
- Fixed that SET CHARACTER SET DEFAULT works. (Bug #462)
- Fixed MERGE table behavior in ORDER BY ... DESC queries. (Bug #515)
- Fixed server crash on PURGE MASTER LOGS or SHOW MASTER LOGS when the binary log is off. (Bug #733)
- Fixed password-checking problem on Windows. (Bug #464)
- Fixed the bug in comparison of a DATETIME column and an integer constant. (Bug #504)
- Fixed remote mode of mysqlbinlog . (Bug #672)
- Fixed ERROR 1105: Unknown error that occurred for some SELECT queries, where a column that was declared as NOT NULL was compared with an expression that took NULL value.
- Changed timeout in mysql_real_connect() to use poll() instead of select() to work around problem with many open files in the client.
- Fixed incorrect results from MATCH ... AGAINST used with a LEFT JOIN query.
- Fixed a bug that limited the maximum value for mysqld variables to 4294967295 when they are specified on the command line.
- Fixed a bug that sometimes caused spurious ``Access denied'' errors in HANDLER ... READ statements, when a table is referenced via an alias.
- Fixed portability problem with safe_malloc , which caused MySQL to give "Freeing wrong aligned pointer" errors on SCO 3.2.
- ALTER TABLE ... ENABLE/DISABLE KEYS could cause a core dump when done after an INSERT DELAYED statement on the same table.
- Fixed problem with conversion of localtime to GMT where some times resulted in different (but correct) timestamps. Now MySQL should use the smallest possible timestamp value in this case. (Bug #316)
- Very small query cache sizes could crash mysqld . (Bug #549)
- Fixed a bug (accidentally introduced by us but present only in version 4.0.13) that made INSERT ... SELECT into an AUTO_INCREMENT column not replicate well. This bug is in the master, not in the slave. (Bug #490)
- Fixed a bug: When an INSERT ... SELECT statement inserted rows into a non-transactional table, but failed at some point (for example, due to a ``Duplicate key'' error), the query was not written to the binary log. Now it is written to the binary log, with its error code, as all other queries are. About the slave-skip-errors option for how to handle partially completed queries in the slave, see Options de réplication . (Bug #491)
- SET FOREIGN_KEY_CHECKS=0 was not replicated properly. The fix probably will not be backported to 3.23.
- On a slave, LOAD DATA INFILE which had no IGNORE or REPLACE clause on the master, was replicated with IGNORE . While this is not a problem if the master and slave data are identical (a LOAD that produces no duplicate conflicts on the master will produce none on the slave anyway), which is true in normal operation, it is better for debugging not to silently add the IGNORE . That way, you can get an error message on the slave and discover that for some reason, the data on master and slave are different and investigate why. (Bug #571)
- On a slave, LOAD DATA INFILE printed an incomplete ``Duplicate entry '%-.64s' for key %d''' message (the key name and value were not mentioned) in case of duplicate conflict (which does not happen in normal operation). (Bug #573)
- When using a slave compiled with --debug , CHANGE MASTER TO RELAY_LOG_POS could cause a debug assertion failure. (Bug #576)
- When doing a LOCK TABLES WRITE on an InnoDB table, commit could not happen, if the query was not written to the binary log (for example, if --log-bin was not used, or binlog-ignore-db was used). (Bug #578)
- If a 3.23 master had open temporary tables that had been replicated to a 4.0 slave, and the binary log got rotated, these temporary tables were immediately dropped by the slave (which caused problems if the master used them subsequently). This bug had been fixed in 4.0.13, but in a manner which caused an unlikely inconvenience: If the 3.23 master died brutally (power failure), without having enough time to automatically write DROP TABLE statements to its binary log, then the 4.0.13 slave would not notice the temporary tables have to be dropped, until the slave mysqld server is restarted. This minor inconvenience is fixed in 3.23.57 and 4.0.14 (meaning the master must be upgraded to 3.23.57 and the slave to 4.0.14 to remove the inconvenience). (Bug #254)
- If MASTER_POS_WAIT() was waiting, and the slave was idle, and the slave SQL thread terminated, MASTER_POS_WAIT() would wait forever. Now when the slave SQL thread terminates, MASTER_POS_WAIT() immediately returns NULL (``slave stopped''). (Bug #651)
- After RESET SLAVE; START SLAVE; , the Relay_Log_Space value displayed by SHOW SLAVE STATUS was too big by four bytes. (Bug #763)
- If a query was ignored on the slave (because of replicate-ignore-table and other similar rules), the slave still checked if the query got the same error code (0, no error) as on the master. So if the master had an error on the query (for example, ``Duplicate entry'' in a multiple-row insert), then the slave stopped and warned that the error codes didn't match. (Bug #797)

26.3.9 Changements de la version 4.0.13 (16 Mai 2003)

Functionality added or changed:

- PRIMARY KEY now implies NOT NULL . (Bug #390)
- The Windows binary packages are now compiled with --enable-local-infile to match the Unix build configuration.
- Removed timing of tests from mysql-test-run . time does not accept all required parameters on many platforms (for example, QNX) and timing the tests is not really required (it's not a benchmark anyway).
- SHOW MASTER STATUS and SHOW SLAVE STATUS required the SUPER privilege; now they accept REPLICATION CLIENT as well. (Bug #343)
- Added multi-threaded MyISAM repair optimization and myisam_repair_threads variable to enable it. Variables sytème du serveur .
- Added innodb_max_dirty_pages_pct variable which controls amount of dirty pages allowed in InnoDB buffer pool.
- CURRENT_USER() and Access denied error messages now report the hostname exactly as it was specified in the GRANT command.
- Removed benchmark results from the source and binary distributions. They are still available in the BK source tree, though.
- InnoDB tables now support ANALYZE TABLE .
- MySQL now issues an error when it opens a table that was created with MySQL 4.1.
- Option --new now changes binary items (0xFFDF) to be treated as binary strings instead of numbers by default. This fixes some problems with character sets where it's convenient to input the string as a binary item. After this change you have to convert the binary string to INTEGER with a CAST if you want to compare two binary items with each other and know which one is bigger than the other. SELECT CAST(0xfeff AS UNSIGNED) < CAST(0xff AS UNSIGNED) . This will be the default behavior in

MySQL 4.1. (Bug #152)
- Enabled delayed_insert_timeout on Linux (most modern glibc libraries have a fixed pthread_cond_timedwait()). (Bug #211)
- Don't create more insert delayed threads than given by max_delayed_threads . (Bug #211)
- Changed UPDATE ... LIMIT to apply the limit to rows that were matched, whether or not they actually were changed. Previously the limit was applied as a restriction on the number of rows changed.
- Tuned optimizer to favor clustered index over table scan.
- BIT_AND() and BIT_OR() now return an unsigned 64-bit value.
- Added warnings to error log of why a secure connection failed (when running with --log-warnings).
- Deprecated options --skip-symlink and --use-symbolic-links and replaced these with --symbolic-links .
- The default option for innodb_flush_log_at_trx_commit was changed from 0 to 1 to make InnoDB tables ACID by default. Options de démarrage InnoDB .
- Added a feature to SHOW KEYS to display keys that are disabled by ALTER TABLE DISABLE KEYS command.
- When using a non-existing table type with CREATE TABLE , first try if the default table type exists before falling back to MyISAM .
- Added MEMORY as an alias for HEAP .
- Renamed function rnd to my_rnd as the name was too generic and is an exported symbol in libmysqlclient (thanks to Dennis Haney for the initial patch).
- Portability fix: renamed include/dbug.h to include/my_dbug.h .
- mysqldump no longer silently deletes the binary logs when invoked with the --master-data or --first-slave option; while this behavior was convenient for some users, others may suffer from it. Now you must explicitly ask for binary logs to be deleted by using the new --delete-master-logs option.
- If the slave is configured (using for example replicate-wild-ignore-table=mysql.%) to exclude mysql.user , mysql.host , mysql.db , mysql.tables_priv and mysql.columns_priv from replication, then GRANT and REVOKE will not be replicated.

Bugs fixed:

- Logged Access denied error message had incorrect Using password value. (Bug #398)
- Fixed bug with NATURAL LEFT JOIN , NATURAL RIGHT JOIN and RIGHT JOIN when using many joined tables. The problem was that the JOIN method was not always associated with the tables surrounding the JOIN method. If you have a query that uses many RIGHT JOIN or NATURAL ... JOINS you should verify that they work as you expected after upgrading MySQL to this version. (Bug #291)
- Fixed mysql parser not to erroneously interpret ''' or '"' characters within /* ... */ comment as beginning a quoted string.
- mysql command line client no longer looks for * commands inside backtick-quoted strings.
- Fixed Unknown error when using UPDATE ... LIMIT . (Bug #373)
- Fixed problem with ANSI mode and GROUP BY with constants. (Bug #387)
- Fixed bug with UNION and OUTER JOIN . (Bug #386)
- Fixed bug if one used a multiple-table UPDATE and the query required a temporary table bigger than tmp_table_size . (Bug #286)
- Run mysql_install_db with the -IN-RPM option for the Mac OS X installation to not fail on systems with improperly configured hostname configurations.
- LOAD DATA INFILE will now read 000000 as a zero date instead as "2000-00-00" .
- Fixed bug that caused DELETE FROM table WHERE const_expression always to delete the whole table (even if expression result was false). (Bug #355)
- Fixed core dump bug when using FORMAT('nan',#) . (Bug #284)
- Fixed name resolution bug with HAVING ... COUNT(DISTINCT ...) .
- Fixed incorrect result from truncation operator (*) in MATCH ... AGAINST() in some complex joins.
- Fixed a crash in REPAIR ... USE_FRM command, when used on read-only, nonexisting table or a table with a crashed index file.
- Fixed a crashing bug in mysql monitor program. It occurred if program was started with --no-defaults , with a prompt that contained the hostname and a connection to a non-existent database was requested.
- Fixed problem when comparing a key for a multi-byte character set. (Bug #152)
- Fixed bug in LEFT , RIGHT and MID when used with multi-byte character sets and some GROUP BY queries. (Bug #314)
- Fix problem with ORDER BY being discarded for some DISTINCT queries. (Bug #275)
- Fixed that SET SQL_BIG_SELECTS=1 works as documented (This corrects a new bug introduced in 4.0)
- Fixed some serious bugs in UPDATE ... ORDER BY . (Bug #241)
- Fixed unlikely problem in optimizing WHERE clause with constant expression like in WHERE 1 AND (a=1 AND b=1) .
- Fixed that SET SQL_BIG_SELECTS=1 works again.
- Introduced proper backtick quoting for db.table in SHOW GRANTS .
- FULLTEXT index stopped working after ALTER TABLE that converts TEXT column to CHAR . (Bug #283)
- Fixed a security problem with SELECT and wildcarded select list, when user only had partial column SELECT privileges on the table.
- Mark a MyISAM table as "analyzed" only when all the keys are indeed analyzed.
- Only ignore world-writable my.cnf files that are regular files (and not, for example, named pipes or character devices).
- Fixed few smaller issues with SET PASSWORD .
- Fixed error message which contained deprecated text.
- Fixed a bug with two NATURAL JOIN s in the query.
- SUM() didn't return NULL when there was no rows in result or when all values was NULL .
- On Unix symbolic links handling was not enabled by default and there was no way to turn this on.
- Added missing dashes to parameter --open-files-limit in mysqld_safe . (Bug #264)
- Fixed incorrect hostname for TCP/IP connections displayed in SHOW PROCESSLIST .
- Fixed a bug with NAN in FORMAT(...) function ...
- Fixed a bug with improperly cached database privileges.
- Fixed a bug in ALTER TABLE ENABLE / DISABLE KEYS which failed to force a refresh of table data in the cache.
- Fixed bugs in replication of LOAD DATA INFILE for custom parameters (ENCLOSED , TERMINATED and so on) and temporary tables. (Bug #183, Bug #222)
- Fixed a replication bug when the master is 3.23 and the slave 4.0: the slave lost the replicated temporary tables if FLUSH LOGS was issued on the master. (Bug #254)
- Fixed a bug when doing LOAD DATA INFILE IGNORE : When reading the binary log, mysqlbinlog and the replication code read REPLACE instead of IGNORE . This could make the slave's table become different from the master's table. (Bug #218)
- Fixed a deadlock when relay_log_space_limit was set to a too small value. (Bug #79)
- Fixed a bug in HAVING clause when an alias is used from the **select list** .
- Fixed overflow bug in MyISAM when a row is inserted into a table with a large number of columns and at least one BLOB/TEXT column. Bug was caused by incorrect calculation of the needed buffer to pack data.
- Fixed a bug when SELECT **@nonexistent_variable** caused the error in client - server protocol due to net_printf() being sent to the client twice.
- Fixed a bug in setting SQL_BIG_SELECTS option.
- Fixed a bug in SHOW PROCESSLIST which only displayed a localhost in the "Host" column. This was caused by a glitch that used only current thread information instead of information from the linked list of threads.
- Removed unnecessary Mac OS X helper files from server RPM. (Bug #144)
- Allow optimization of multiple-table update for InnoDB tables as well.
- Fixed a bug in multiple-table updates that caused some rows to be updated several times.
- Fixed a bug in mysqldump when it was called with --master-data : the CHANGE MASTER TO commands appended to the SQL dump had incorrect coordinates. (Bug #159)
- Fixed a bug when an updating query using USER() was replicated on the slave; this caused segfault on the slave. (Bug #178). USER() is still badly replicated on the slave (it is replicated to "").

26.3.10 Changements de la version 4.0.12 (15 Mars 2003 : Production)

Functionality added or changed:

- mysqld no longer reads options from world-writeable config files.
- Integer values between 9223372036854775807 and 9999999999999999999 are now regarded as unsigned longlongs, not as floats. This makes these values work similar to values between 10000000000000000000 and 18446744073709551615.
- SHOW PROCESSLIST will now include the client TCP port after the hostname to make it easier to know from which client the request originated.

Bugs fixed:

- Fixed mysqld crash on extremely small values of sort_buffer variable.
- INSERT INTO u SELECT ... FROM t was written too late to the binary log if t was very frequently updated during the execution of this query. This could cause a problem with mysqlbinlog or replication. The master must be upgraded, not the slave. (Bug #136)
- Fixed checking of random part of WHERE clause. (Bug #142)
- Fixed a bug with multiple-table updates with InnoDB tables. This bug occurred as, in many cases, InnoDB tables cannot be updated ``on the fly,'' but offsets to the records have to be stored in a temporary table.
- Added missing file mysql_secure_installation to the server RPM subpackage. (Bug #141)
- Fixed MySQL (and myisamchk) crash on artificially corrupted .MYI files.
- Don't allow BACKUP TABLE to overwrite existing files.
- Fixed a bug with multiple-table UPDATE statements when user had all privileges on the database where tables are located and there were any entries in tables_priv table, that is, grant_option was true.
- Fixed a bug that allowed a user with table or column grants on some table, TRUNCATE any table in the same database.
- Fixed deadlock when doing LOCK TABLE followed by DROP TABLE in the same thread. In this case one could still kill the thread with KILL .
- LOAD DATA LOCAL INFILE was not properly written to the binary log (hence not properly replicated). (Bug #82)
- RAND() entries were not read correctly by mysqlbinlog from the binary log which caused problems when restoring a table that was inserted with RAND() . INSERT INTO t1 VALUES(RAND()) . In replication this worked ok.
- SET SQL_LOG_BIN=0 was ignored for INSERT DELAYED queries. (Bug #104)
- SHOW SLAVE STATUS reported too old positions (columns Relay_Master_Log_File and Exec_Master_Log_Pos) for the last executed statement from the master, if this statement was the COMMIT of a transaction. The master must be upgraded for that, not the slave. (Bug #52)
- LOAD DATA INFILE was not replicated by the slave if replicate_*_table was set on the slave. (Bug #86)
- After RESET SLAVE , the coordinates displayed by SHOW SLAVE STATUS looked un-reset (though they were, but only internally). (Bug #70)
- Fixed query cache invalidation on LOAD DATA .
- Fixed memory leak on ANALYZE procedure with error.
- Fixed a bug in handling CHAR(0) columns that could cause incorrect results from the query.
- Fixed rare bug with incorrect initialization of AUTO_INCREMENT column, as a secondary column in a multi-column key (Utiliser AUTO_INCREMENT), when data was inserted with INSERT ... SELECT or LOAD DATA into an empty table.
- On Windows, STOP SLAVE didn't stop the slave until the slave got one new command from the master (this bug has been fixed for MySQL 4.0.11 by releasing updated 4.0.11a Windows packages, which include this individual fix on top of the 4.0.11 sources). (Bug #69)
- Fixed a crash when no database was selected and LOAD DATA command was issued with full table name specified, including database prefix.
- Fixed a crash when shutting down replication on some platforms (for example, Mac OS X).
- Fixed a portability bug with pthread_attr_getstacksize on HP-UX 10.20 (Patch was also included in 4.0.11a sources).
- Fixed the bigint test to not fail on some platforms (for example, HP-UX and Tru64) due to different return values of the atof() function.
- Fixed the rpl_rotate_logs test to not fail on certain platforms (e.g. Mac OS X) due to a too long file name (changed slave-master-info.opt to .slave-mi).

26.3.11 Changements de la version 4.0.11 (20 Février 2003)

Functionality added or changed:

- NULL is now sorted **LAST** if you use ORDER BY ... DESC (as it was before MySQL 4.0.2). This change was required to comply with the SQL-99 standard. (The original change was made because we thought that SQL-99 required NULL to be always sorted at the same position, but this was incorrect).
- Added START TRANSACTION (SQL-99 syntax) as alias for BEGIN . This is recommended to use instead of BEGIN to start a transaction.
- Added OLD_PASSWORD() as a synonym for PASSWORD() .
- Allow keyword ALL in group functions.
- Added support for some new INNER JOIN and JOIN syntaxes. For example, SELECT * FROM t1 INNER JOIN t2 didn't work before.
- Novell NetWare 6.0 porting effort completed, Novell patches merged into the main source tree.

Bugs fixed:

- Fixed problem with multiple-table delete and InnoDB tables.
- Fixed a problem with BLOB NOT NULL columns used with IS NULL .
- Re-added missing pre- and post(un)install scripts to the Linux RPM packages (they were missing after the renaming of the server subpackage).
- Fixed that table locks are not released with multiple-table updates and deletes with InnoDB storage engine.
- Fixed bug in updating BLOB columns with long strings.
- Fixed integer-wraparound when giving big integer (>= 10 digits) to function that requires an unsigned argument, like CREATE TABLE (...) AUTO_INCREMENT=# .
- MIN(key_column) could in some cases return NULL on a column with NULL and other values.
- MIN(key_column) and MAX(key_column) could in some cases return incorrect values when used in OUTER JOIN .
- MIN(key_column) and MAX(key_column) could return incorrect values if one of the tables was empty.
- Fixed rare crash in compressed MyISAM tables with blobs.
- Fixed bug in using aggregate functions as argument for INTERVAL , CASE , FIELD , CONCAT_WS , ELT and MAKE_SET functions.
- When running with --lower-case-table-names (default on Windows) and you had tables or databases with mixed case on disk, then executing SHOW TABLE STATUS followed with DROP DATABASE or DROP TABLE could fail with Errcode 13 .

26.3.12 Changements de la version 4.0.10 (29 janvier 2003)

Functionality added or changed:

- Added option --log-error[=file_name] to mysqld_safe and mysqld . This option will force all error messages to be put in a log file if the option --console is not given. On Windows --log-error is enabled as default, with a default name of host_name.err if the name is not specified.
- Changed some things from Warning: to Note: in the log files.
- The mysqld server should now compile on NetWare.
- Added optimization that if one does GROUP BY ... ORDER BY NULL then result is not sorted.
- New --ft-stopword-file command-line option for mysqld to replace/disable the built-in stopword list that is used in full-text searches. Syntaxe de SHOW VARIABLES .
- Changed default stack size from 64K to 192K; This fixes a core dump problem on Red Hat 8.0 and other systems with a glibc that requires a stack size larger than 128K for gethostbyaddr() to resolve a hostname. You can fix this for earlier MySQL versions by starting mysqld with --thread-stack=192K .
- Added mysql_waitpid to the binary distribution and the MySQL-client RPM subpackage (required for mysql-test-run).
- Renamed the main MySQL RPM package to MySQL-server . When updating from an older version, MySQL-server.rpm will simply replace MySQL.rpm .
- If a slave is configured with replicate_wild_do_table=db.% or replicate_wild_ignore_table=db.% , these rules will be applied to CREATE/DROP DATABASE , too.
- Added timeout value for MASTER_POS_WAIT() .

Bugs fixed:

- Fixed initialization of the random seed for newly created threads to give a better rand() distribution from the first call.
- Fixed a bug that caused mysqld to hang when a table was opened with the HANDLER command and then dropped without being closed.
- Fixed bug in logging to binary log (which affects replication) a query that inserts a NULL in an AUTO_INCREMENT column and also uses LAST_INSERT_ID() .
- Fixed an unlikely bug that could cause a memory overrun when using ORDER BY constant_expression .
- Fixed a table corruption in myisamchk 's parallel repair mode.
- Fixed bug in query cache invalidation on simple table renaming.
- Fixed bug in mysqladmin --relative .
- On some 64-bit systems, show status reported a strange number for Open_files and Open_streams .
- Fixed incorrect number of columns in EXPLAIN on empty table.
- Fixed bug in LEFT JOIN that caused zero rows to be returned in the case the WHERE condition was evaluated as FALSE after reading const tables. (Unlikely condition).
- FLUSH PRIVILEGES didn't correctly flush table/column privileges when mysql.tables_priv is empty.
- Fixed bug in replication when using LOAD DATA INFILE one a file that updated an AUTO_INCREMENT column with NULL or 0 . This bug only affected MySQL 4.0 masters (not slaves or MySQL 3.23 masters). **Note** : If you have a slave that has replicated a file with generated AUTO_INCREMENT columns then the slave data is corrupted and you should reinitialize the affected tables from the master.
- Fixed possible memory overrun when sending a BLOB value larger than 16M to the client.
- Fixed incorrect error message when setting a NOT NULL column to an expression that returned NULL .
- Fixed core dump bug in str LIKE "%other_str%" where str or other_str contained characters >= 128.
- Fixed bug: When executing on master LOAD DATA and InnoDB failed with table full error the binary log was corrupted.

26.3.13 Changements de la version 4.0.9 (09 janvier 2003)

Functionality added or changed:

- OPTIMIZE TABLE will for MyISAM tables treat all NULL values as different when calculating cardinality. This helps in optimizing joins between tables where one of the tables has a lot of NULL values in a indexed column:
  ```
  SELECT * from t1,t2 where t1.a=t2.key_with_a_lot_of_null;
  ```
- Added join operator FORCE INDEX (key_list) . This acts likes USE INDEX (key_list) but with the addition that a table scan is assumed to be VERY expensive. One bad thing with this is that it makes FORCE a reserved word.
- Reset internal row buffer in MyISAM after each query. This will reduce memory in the case you have a lot of big blobs in a table.

Bugs fixed:

- A security patch in 4.0.8 causes the mysqld server to die if the remote hostname can't be resolved. This is now fixed.
- Fixed crash when replication big LOAD DATA INFILE statement that caused log rotation.

26.3.14 Changements de la version 4.0.8 (07 janvier 2003)

Functionality added or changed:

- Default max_packet_length for libmysqld.c is now 1024*1024*1024.
- One can now specify max_allowed_packet in a file ready by mysql_options(MYSQL_READ_DEFAULT_FILE) . for clients.
- When sending a too big packet to the server with the not compressed protocol, the client now gets an error message instead of a lost connection.
- We now send big queries/result rows in bigger hunks, which should give a small speed improvement.
- Fixed some bugs with the compressed protocol for rows > 16M.
- InnoDB tables now also support ON UPDATE CASCADE in FOREIGN KEY constraints. See the InnoDB section in the manual for the InnoDB changelog.

Bugs fixed:

- Fixed bug in ALTER TABLE with BDB tables.
- Fixed core dump bug in QUOTE() function.
- Fixed a bug in handling communication packets bigger than 16M. Unfortunately this required a protocol change; If you upgrade the server to 4.0.8 and above and have clients that uses packets >= 255*255*255 bytes (=16581375) you must also upgrade your

clients to at least 4.0.8. If you don't upgrade, the clients will hang when sending a big packet.
- Fixed bug when sending blobs longer than 16M to client.
- Fixed bug in GROUP BY when used on BLOB column with NULL values.
- Fixed a bug in handling NULL values in CASE ... WHEN ...

26.3.15 Changements de la version 4.0.7 (20 Décembre 2002)

Functionality added or changed:

- mysqlbug now also reports the compiler version used for building the binaries (if the compiler supports the option --version).

Bugs fixed:

- Fixed compilation problems on OpenUnix and HPUX 10.20.
- Fixed some optimization problems when compiling MySQL with -DBIG_TABLES on a 32-bit system.
- mysql_drop_db() didn't check permissions properly so anyone could drop another users database. DROP DATABASE is checked properly.

26.3.16 Changements de la version 4.0.6 (14 Décembre 2002 : Gamma)

Functionality added or changed:

- Added syntax support for CHARACTER SET xxx and CHARSET=xxx table options (to be able to read table dumps from 4.1).
- Fixed replication bug that caused the slave to loose its position in some cases when the replication log was rotated.
- Fixed that a slave will restart from the start of a transaction if it's killed in the middle of one.
- Moved the manual pages from man to man/man1 in the binary distributions.
- The default type returned by IFNULL(A,B) is now set to be the more 'general' of the types of A and B . (The order is STRING , REAL or INTEGER).
- Moved the mysql.server startup script in the RPM packages from /etc/rc.d/init.d/mysql to /etc/init.d/mysql (which almost all current Linux distributions support for LSB compliance).
- Added Qcache_lowmem_prunes status variable (number of queries that were deleted from cache because of low memory).
- Fixed mysqlcheck so it can deal with table names containing dashes.
- Bulk insert optimization (Syntaxe de SHOW VARIABLES) is no longer used when inserting small (less than 100) number of rows.
- Optimization added for queries like SELECT ... FROM merge_table WHERE indexed_column=constant_expr .
- Added functions LOCALTIME and LOCALTIMESTAMP as synonyms for NOW() .
- CEIL is now an alias for CEILING .
- The CURRENT_USER() function can be used to get a user@host value as it was matched in the GRANT system. Fonctions diverses .
- Fixed CHECK constraints to be compatible with SQL-99. This made CHECK a reserved word. (Checking of CHECK constraints is still not implemented).
- Added CAST(... as CHAR) .
- Added PostgreSQL compatible LIMIT syntax: SELECT ... LIMIT row_count OFFSET offset
- mysql_change_user() will now reset the connection to the state of a fresh connect (Ie, ROLLBACK any active transaction, close all temporary tables, reset all user variables etc..)
- CHANGE MASTER and RESET SLAVE now require that slave threads be both already stopped; these commands will return an error if at least one of these two threads is running.

Bugs fixed:

- Fixed number of found rows returned in multi table updates
- Make --lower-case-table-names default on Mac OS X as the default file system (HFS+) is case insensitive. Identifier Case Sensitivity .
- Transactions in AUTOCOMMIT=0 mode didn't rotate binary log.
- A fix for the bug in a SELECT with joined tables with ORDER BY and LIMIT clause when filesort had to be used. In that case LIMIT was applied to filesort of one of the tables, although it could not be. This fix also solved problems with LEFT JOIN .
- mysql_server_init() now makes a copy of all arguments. This fixes a problem when using the embedded server in C# program.
- Fixed buffer overrun in libmysqlclient library that allowed a malicious MySQL server to crash the client application.
- Fixed security-related bug in mysql_change_user() handling. All users are strongly recommended to upgrade to version 4.0.6.
- Fixed bug that prevented --chroot command-line option of mysqld from working.
- Fixed bug in phrase operator "..." in boolean full-text search.
- Fixed bug that caused OPTIMIZE TABLE to corrupt the table under some rare circumstances.
- Part rewrite of multiple-table-update to optimize it, make it safer and more bug free.
- LOCK TABLES now works together with multiple-table-update and multiple-table-delete.
- --replicate-do=xxx didn't work for UPDATE commands. (Bug introduced in 4.0.0)
- Fixed shutdown problem on Mac OS X.
- Major InnoDB bugs in REPLACE, AUTO_INCREMENT, INSERT INTO ... SELECT ... were fixed. See the InnoDB changelog in the InnoDB section of the manual.
- RESET SLAVE caused a crash if the slave threads were running.

26.3.17 Changements de la version 4.0.5 (13 novembre 2002)

Functionality added or changed:

- Port number was added to host name (if it is known) in SHOW PROCESSLIST command
- Changed handling of last argument in WEEK() so that one can get week number according to the ISO 8601 specification. (Old code should still work).
- Fixed that INSERT DELAYED threads don't hang on Waiting for INSERT when one sends a SIGHUP to mysqld .
- Change that AND works according to SQL-99 when it comes to NULL handling. In practice, this only affects queries where you do something like WHERE ... NOT (NULL AND 0) .
- mysqld will now resolve basedir to its full path (with realpath()). This enables one to use relative symlinks to the MySQL installation directory. This will however cause show variables to report different directories on systems where there is a symbolic link in the path.

- Fixed that MySQL will not use index scan on index disabled with IGNORE INDEX or USE INDEX . to be ignored.
- Added --use-frm option to mysqlcheck . When used with REPAIR , it gets the table structure from the .frm file, so the table can be repaired even if the .MYI header is corrupted.
- Fixed bug in MAX() optimization when used with JOIN and ON expressions.
- Added support for reading of MySQL 4.1 table definition files.
- BETWEEN behavior changed (Opérateurs de comparaison). Now datetime_col BETWEEN timestamp AND timestamp should work as expected.
- One can create TEMPORARY MERGE tables now.
- DELETE FROM myisam_table now shrinks not only the .MYD file but also the .MYI file.
- When one uses the --open-files-limit=# option to mysqld_safe it's now passed on to mysqld .
- Changed output from EXPLAIN from 'where used' to 'Using where' to make it more in line with other output.
- Removed variable safe_show_database as it was no longer used.
- Updated source tree to be built using automake 1.5 and libtool 1.4 .
- Fixed an inadvertently changed option (--ignore-space) back to the original --ignore-spaces in mysqlclient . (Both syntaxes will work).
- Don't require UPDATE privilege when using REPLACE .
- Added support for DROP TEMPORARY TABLE ... , to be used to make replication safer.
- When transactions are enabled, all commands that update temporary tables inside a BEGIN/COMMIT are now stored in the binary log on COMMIT and not stored if one does ROLLBACK . This fixes some problems with non-transactional temporary tables used inside transactions.
- Allow braces in joins in all positions. Formerly, things like SELECT * FROM (t2 LEFT JOIN t3 USING (a)), t1 worked, but not SELECT * FROM t1, (t2 LEFT JOIN t3 USING (a)) . Note that braces are simply removed, they do not change the way the join is executed.
- InnoDB now supports also isolation levels READ UNCOMMITTED and READ COMMITTED . For a detailed InnoDB changelog, see Changements de InnoDB in this manual.

Bugs fixed:

- Fixed bug in MAX() optimization when used with JOIN and ON expressions.
- Fixed that INSERT DELAY threads don't hang on Waiting for INSERT when one sends a SIGHUP to mysqld .
- Fixed that MySQL will not use an index scan on an index that has been disabled with IGNORE INDEX or USE INDEX .
- Corrected test for root user in mysqld_safe .
- Fixed error message issued when storage engine cannot do CHECK or REPAIR .
- Fixed rare core dump problem in complicated GROUP BY queries that didn't return any result.
- Fixed mysqlshow to work properly with wildcarded database names and with database names that contain underscores.
- Portability fixes to get MySQL to compile cleanly with Sun Forte 5.0.
- Fixed MyISAM crash when using dynamic-row tables with huge numbers of packed fields.
- Fixed query cache behavior with BDB transactions.
- Fixed possible floating point exception in MATCH relevance calculations.
- Fixed bug in full-text search IN BOOLEAN MODE that made MATCH to return incorrect relevance value in some complex joins.
- Fixed a bug that limited MyISAM key length to a value slightly less that 500. It is exactly 500 now.
- Fixed that GROUP BY on columns that may have a NULL value doesn't always use disk based temporary tables.
- The filename argument for the --des-key-file argument to mysqld is interpreted relative to the data directory if given as a relative pathname.
- Removed a condition that temp table with index on column that can be NULL has to be MyISAM . This was okay for 3.23, but not needed in 4.*. This resulted in slowdown in many queries since 4.0.2.
- Small code improvement in multiple-table updates.
- Fixed a newly introduced bug that caused ORDER BY ... LIMIT row_count to not return all rows.
- Fixed a bug in multiple-table deletes when outer join is used on an empty table, which gets first to be deleted.
- Fixed a bug in multiple-table updates when a single table is updated.
- Fixed bug that caused REPAIR TABLE and myisamchk to corrupt FULLTEXT indexes.
- Fixed bug with caching the mysql grant table database. Now queries in this database are not cached in the query cache.
- Small fix in mysqld_safe for some shells.
- Give error if a MyISAM MERGE table has more than 2 ^ 32 rows and MySQL was not compiled with -DBIG_TABLES .
- Fixed some ORDER BY ... DESC problems with InnoDB tables.

26.3.18 Changements de la version 4.0.4 (29 septembre 2002)

- Fixed bug where GRANT / REVOKE failed if hostname was given in non-matching case.
- Don't give warning in LOAD DATA INFILE when setting a timestamp to a string value of '0' .
- Fixed bug in myisamchk -R mode.
- Fixed bug that caused mysqld to crash on REVOKE .
- Fixed bug in ORDER BY when there is a constant in the SELECT statement.
- One didn't get an error message if mysqld couldn't open the privilege tables.
- SET PASSWORD FOR ... closed the connection in case of errors (bug from 4.0.3).
- Increased max possible max_allowed_packet in mysqld to 1 GB.
- Fixed bug when doing a multi-line INSERT on a table with an AUTO_INCREMENT key which was not in the first part of the key.
- Changed LOAD DATA INFILE to not recreate index if the table had rows from before.
- Fixed overrun bug when calling AES_DECRYPT() with incorrect arguments.
- --skip-ssl can now be used to disable SSL in the MySQL clients, even if one is using other SSL options in an option file or previously on the command line.
- Fixed bug in MATCH ... AGAINST(... IN BOOLEAN MODE) used with ORDER BY .
- Added LOCK TABLES and CREATE TEMPORARY TABLES privilege on the database level. One must run the mysql_fix_privilege_tables script on old installations to activate these.
- In SHOW TABLE ... STATUS , compressed tables sometimes showed up as dynamic .
- SELECT @@[global|session].var_name didn't report global | session in the result column name.
- Fixed problem in replication that FLUSH LOGS in a circular replication setup created an infinite number of binary log files. Now a rotate-binary-log command in the binary log will not cause slaves to rotate logs.
- Removed STOP EVENT from binary log when doing FLUSH LOGS .
- Disable the use of SHOW NEW MASTER FOR SLAVE as this needs to be completely reworked in a future release.
- Fixed a bug with constant expression (for example, field of a one-row table, or field from a table, referenced by a UNIQUE key) appeared in ORDER BY part of SELECT DISTINCT .
- --log-binary=a.b.c now properly strips off .b.c .
- FLUSH LOGS removed numerical extension for all future update logs.
- GRANT ... REQUIRE didn't store the SSL information in the mysql.user table if SSL was not enabled in the server.
- GRANT ... REQUIRE NONE can now be used to remove SSL information.
- AND is now optional between REQUIRE options.
- REQUIRE option was not properly saved, which could cause strange output in SHOW GRANTS .
- Fixed that mysqld --help reports correct values for --datadir and --bind-address .
- Fixed that one can drop UDFs that didn't exist when mysqld was started.
- Fixed core dump problem with SHOW VARIABLES on some 64-bit systems (like Solaris SPARC).
- Fixed a bug in my_getopt() ; --set-variable syntax didn't work for those options that didn't have a valid variable in the my_option struct. This affected at least the default-table-type option.
- Fixed a bug from 4.0.2 that caused REPAIR TABLE and myisamchk --recover to fail on tables with duplicates in a unique key.
- Fixed a bug from 4.0.3 in calculating the default datatype for some functions. This affected queries of type CREATE TABLE

table_name SELECT expression(),...
- Fixed bug in queries of type SELECT * FROM table-list GROUP BY ... and SELECT DISTINCT * FROM ... .
- Fixed bug with the --slow-log when logging an administrator command (like FLUSH TABLES).
- Fixed a bug that OPTIMIZE of locked and modified table, reported table corruption.
- Fixed a bug in my_getopt() in handling of special prefixes (--skip- , --enable-). --skip-external-locking didn't work and the bug may have affected other similar options.
- Fixed bug in checking for output file name of the tee option.
- Added some more optimization to use index for SELECT ... FROM many_tables .. ORDER BY key limit #
- Fixed problem in SHOW OPEN TABLES when a user didn't have access permissions to one of the opened tables.

26.3.19 Changements de la version 4.0.3 (26 Août 2002 : Beta)

- Fixed problem with types of user variables. (Bug #551)
- Fixed problem with configure ... --localstatedir=... .
- Cleaned up mysql.server script.
- Fixed a bug in mysqladmin shutdown when pid file was modified while mysqladmin was still waiting for the previous one to disappear. This could happen during a very quick restart and caused mysqladmin to hang until shutdown_timeout seconds had passed.
- Don't increment warnings when setting AUTO_INCREMENT columns to NULL in LOAD DATA INFILE .
- Fixed all boolean type variables/options to work with the old syntax, for example, all of these work: --lower-case-table-names , --lower-case-table-names=1 , -O lower-case-table-names=1 , --set-variable=lower-case-table-names=1
- Fixed shutdown problem (SIGTERM signal handling) on Solaris. (Bug from 4.0.2).
- SHOW MASTER STATUS now returns an empty set if binary log is not enabled.
- SHOW SLAVE STATUS now returns an empty set if slave is not initialized.
- Don't update MyISAM index file on update if not strictly necessary.
- Fixed bug in SELECT DISTINCT ... FROM many_tables ORDER BY not-used-column .
- Fixed a bug with BIGINT values and quoted strings.
- Added QUOTE() function that performs SQL quoting to produce values that can be used as data values in queries.
- Changed variable DELAY_KEY_WRITE to an enum to allow one set DELAY_KEY_WRITE for all tables without taking down the server.
- Changed behavior of IF(condition,column,NULL) so that it returns the value of the column type.
- Made safe_mysqld a symlink to mysqld_safe in binary distribution.
- Fixed security bug when having an empty database name in the user.db table.
- Fixed some problems with CREATE TABLE ... SELECT function() .
- mysqld now has the option --temp-pool enabled by default as this gives better performance with some operating systems.
- Fixed problem with too many allocated alarms on slave when connecting to master many times (normally not a very critical error).
- Fixed hang in CHANGE MASTER TO if the slave thread died very quickly.
- Big cleanup in replication code (less logging, better error messages, etc..)
- If the --code-file option is specified, the server calls setrlimit() to set the maximum allowed core file size to unlimited, so core files can be generated.
- Fixed bug in query cache after temporary table creation.
- Added --count=N (-c) option to mysqladmin , to make the program do only N iterations. To be used with --sleep (-i). Useful in scripts.
- Fixed bug in multiple-table UPDATE : when updating a table, do_select() became confused about reading records from a cache.
- Fixed bug in multiple-table UPDATE when several fields were referenced from a single table
- Fixed bug in truncating nonexisting table.
- Fixed bug in REVOKE that caused user resources to be randomly set.
- Fixed bug in GRANT for the new CREATE TEMPORARY TABLE privilege.
- Fixed bug in multiple-table DELETE when tables are re-ordered in the table initialization method and ref_lengths are of different sizes.
- Fixed two bugs in SELECT DISTINCT with large tables.
- Fixed bug in query cache initialization with very small query cache size.
- Allow DEFAULT with INSERT statement.
- The startup parameters myisam_max_sort_file_size and myisam_max_extra_sort_file_size are now given in bytes, not megabytes.
- External system locking of MyISAM / ISAM files is now turned off by default. One can turn this on with --external-locking . (For most users this is never needed).
- Fixed core dump bug with INSERT ... SET db_name.table_name.colname='' .
- Fixed client hangup bug when using some SQL commands with incorrect syntax.
- Fixed a timing bug in DROP DATABASE
- New SET [GLOBAL | SESSION] syntax to change thread-specific and global server variables at runtime.
- Added variable slave_compressed_protocol .
- Renamed variable query_cache_startup_type to query_cache_type , myisam_bulk_insert_tree_size to bulk_insert_buffer_size , record_buffer to read_buffer_size and record_rnd_buffer to read_rnd_buffer_size .
- Renamed some SQL variables, but old names will still work until 5.0. Mise à jour de Version 3.23 vers 4.0 .
- Renamed --skip-locking to --skip-external-locking .
- Removed unused variable query_buffer_size .
- Fixed a bug that made the pager option in the mysql client non-functional.
- Added full AUTO_INCREMENT support to MERGE tables.
- Extended LOG() function to accept an optional arbitrary base parameter. Fonctions mathématiques .
- Added LOG2() function (useful for finding out how many bits a number would require for storage).
- Added LN() natural logarithm function for compatibility with other databases. It is synonymous with LOG(X) .

26.3.20 Changements de la version 4.0.2 (01 Juillet 2002)

- Cleaned up NULL handling for default values in DESCRIBE table_name .
- Fixed truncate() to round up negative values to the nearest integer.
- Changed --chroot=path option to execute chroot() immediately after all options have been parsed.
- Don't allow database names that contain '\' .
- lower_case_table_names now also affects database names.
- Added XOR operator (logical and bitwise XOR) with ^ as a synonym for bitwise XOR .
- Added function IS_FREE_LOCK("lock_name") . Based on code contributed by Hartmut Holzgraefe hartmut@six.de .
- Removed mysql_ssl_clear() from C API, as it was not needed.
- DECIMAL and NUMERIC types can now read exponential numbers.
- Added SHA1() function to calculate 160 bit hash value as described in RFC 3174 (Secure Hash Algorithm). This function can be considered a cryptographically more secure equivalent of MD5() . Fonctions diverses .
- Added AES_ENCRYPT() and AES_DECRYPT() functions to perform encryption according to AES standard (Rijndael). Fonctions diverses .
- Added --single-transaction option to mysqldump , allowing a consistent dump of InnoDB tables. mysqldump , exporter les structures de tables et les données .
- Fixed bug in innodb_log_group_home_dir in SHOW VARIABLES .
- Fixed a bug in optimizer with merge tables when non-unique values are used in summing up (causing crashes).
- Fixed a bug in optimizer when a range specified makes index grouping impossible (causing crashes).

- Fixed a rare bug when FULLTEXT index is present and no tables are used.
- Added privileges CREATE TEMPORARY TABLES , EXECUTE , LOCK TABLES , REPLICATION CLIENT , REPLICATION SLAVE , SHOW DATABASES and SUPER . To use these, you must have run the mysql_fix_privilege_tables script after upgrading.
- Fixed query cache align data bug.
- Fixed mutex bug in replication when reading from master fails.
- Added missing mutex in TRUNCATE TABLE ; This fixes some core dump/hangup problems when using TRUNCATE TABLE .
- Fixed bug in multiple-table DELETE when optimizer uses only indexes.
- Fixed that ALTER TABLE table_name RENAME new_table_name is as fast as RENAME TABLE .
- Fixed bug in GROUP BY with two or more fields, where at least one field can contain NULL values.
- Use Turbo Boyer-Moore algorithm to speed up LIKE "%keyword%" searches.
- Fixed bug in DROP DATABASE with symlink.
- Fixed crash in REPAIR ... USE_FRM .
- Fixed bug in EXPLAIN with LIMIT offset != 0 .
- Fixed bug in phrase operator "..." in boolean full-text search.
- Fixed bug that caused duplicated rows when using truncation operator * in boolean full-text search.
- Fixed bug in truncation operator of boolean full-text search (incorrect results when there are only +word* s in the query).
- Fixed bug in boolean full-text search that caused a crash when an identical MATCH expression that did not use an index appeared twice.
- Query cache is now automatically disabled in mysqldump .
- Fixed problem on Windows 98 that made sending of results very slow.
- Boolean full-text search weighting scheme changed to something more reasonable.
- Fixed bug in boolean full-text search that caused MySQL to ignore queries of ft_min_word_len characters.
- Boolean full-text search now supports ``phrase searches''.
- New configure option --without-query-cache .
- Memory allocation strategy for ``root memory'' changed. Block size now grows with number of allocated blocks.
- INET_NTOA() now returns NULL if you give it an argument that is too large (greater than the value corresponding to 255.255.255.255).
- Fix SQL_CALC_FOUND_ROWS to work with UNION . It will work only if the first SELECT has this option and if there is global LIMIT for the entire statement. For the moment, this requires using parentheses for individual SELECT queries within the statement.
- Fixed bug in SQL_CALC_FOUND_ROWS and LIMIT .
- Don't give an error for CREATE TABLE ...(... VARCHAR(0)) .
- Fixed SIGINT and SIGQUIT problems in mysql.cc on Linux with some glibc versions.
- Fixed bug in convert.cc , which is caused by having an incorrect net_store_length() linked in the CONVERT::store() method.
- DOUBLE and FLOAT columns now honor the UNSIGNED flag on storage.
- InnoDB now retains foreign key constraints through ALTER TABLE and CREATE/DROP INDEX .
- InnoDB now allows foreign key constraints to be added through the ALTER TABLE syntax.
- InnoDB tables can now be set to automatically grow in size (autoextend).
- Added --ignore-lines=n option to mysqlimport . This has the same effect as the IGNORE n LINES clause for LOAD DATA .
- Fixed bug in UNION with last offset being transposed to total result set.
- REPAIR ... USE_FRM added.
- Fixed that DEFAULT_SELECT_LIMIT is always imposed on UNION result set.
- Fixed that some SELECT options can appear only in the first SELECT .
- Fixed bug with LIMIT with UNION , where last select is in the braces.
- Fixed that full-text works fine with UNION operations.
- Fixed bug with indexless boolean full-text search.
- Fixed bug that sometimes appeared when full-text search was used with const tables.
- Fixed incorrect error value when doing a SELECT with an empty HEAP table.
- Use ORDER BY column DESC now sorts NULL values first. (In other words, NULL values sort first in all cases, whether or not DESC is specified.) This is changed back in 4.0.10.
- Fixed bug in WHERE key_name='constant' ORDER BY key_name DESC .
- Fixed bug in SELECT DISTINCT ... ORDER BY DESC optimization.
- Fixed bug in ... HAVING 'GROUP_FUNCTION'(xxx) IS [NOT] NULL .
- Fixed bug in truncation operator for boolean full-text search.
- Allow value of --user=# option for mysqld to be specified as a numeric user ID.
- Fixed a bug where SQL_CALC_FOUND_ROWS returned an incorrect value when used with one table and ORDER BY and with InnoDB tables.
- Fixed that SELECT 0 LIMIT 0 doesn't hang thread.
- Fixed some problems with USE/IGNORE INDEX when using many keys with the same start column.
- Don't use table scan with BerkeleyDB and InnoDB tables when we can use an index that covers the whole row.
- Optimized InnoDB sort-buffer handling to take less memory.
- Fixed bug in multiple-table DELETE and InnoDB tables.
- Fixed problem with TRUNCATE and InnoDB tables that produced the error Can't execute the given command because you have active locked tables or an active transaction .
- Added NO_UNSIGNED_SUBTRACTION to the set of flags that may be specified with the --sql-mode option for mysqld . It disables unsigned arithmetic rules when it comes to subtraction. (This will make MySQL 4.0 behave more like 3.23 with UNSIGNED columns).
- The result returned for all bit functions (| , << , ...) is now of type unsigned integer .
- Added detection of nan values in MyISAM to make it possible to repair tables with nan in float or double columns.
- Fixed new bug in myisamchk where it didn't correctly update number of ``parts'' in the MyISAM index file.
- Changed to use autoconf 2.52 (from autoconf 2.13).
- Fixed optimization problem where the MySQL Server was in ``preparing'' state for a long time when selecting from an empty table which had contained a lot of rows.
- Fixed bug in complicated join with const tables. This fix also improves performance a bit when referring to another table from a const table.
- First pre-version of multiple-table UPDATE statement.
- Fixed bug in multiple-table DELETE .
- Fixed bug in SELECT CONCAT(argument_list) ... GROUP BY 1 .
- INSERT ... SELECT did a full rollback in case of an error. Fixed so that we only roll back the last statement in the current transaction.
- Fixed bug with empty expression for boolean full-text search.
- Fixed core dump bug in updating full-text key from/to NULL .
- ODBC compatibility: Added BIT_LENGTH() function.
- Fixed core dump bug in GROUP BY BINARY column .
- Added support for NULL keys in HEAP tables.
- Use index for ORDER BY in queries of type: SELECT * FROM t WHERE key_part1=1 ORDER BY key_part1 DESC,key_part2 DESC
- Fixed bug in FLUSH QUERY CACHE .
- Added CAST() and CONVERT() functions. The CAST and CONVERT functions are nearly identical and mainly useful when you want to create a column with a specific type in a CREATE ... SELECT statement. For more information, read Fonctions de transtypage .
- CREATE ... SELECT on DATE and TIME functions now create columns of the expected type.
- Changed order in which keys are created in tables.
- Added new columns Null and Index_type to SHOW INDEX output.
- Added --no-beep and --prompt options to mysql command-line client.
- New feature: management of user resources.

```
GRANT ... WITH MAX_QUERIES_PER_HOUR N1
               MAX_UPDATES_PER_HOUR N2
               MAX_CONNECTIONS_PER_HOUR N3;
```

  Limiter la consommation de ressources .
- Added mysql_secure_installation to the scripts/ directory.

26.3.21 Changements de la version 4.0.1 (23 décembre 2001)

- Added system command to mysql .
- Fixed bug when HANDLER was used with some unsupported table type.
- mysqldump now puts ALTER TABLE tbl_name DISABLE KEYS and ALTER TABLE tbl_name ENABLE KEYS in the sql dump.
- Added mysql_fix_extensions script.
- Fixed stack overrun problem with LOAD DATA FROM MASTER on OSF/1.
- Fixed shutdown problem on HP-UX.
- Added DES_ENCRYPT() and DES_DECRYPT() functions.
- Added FLUSH DES_KEY_FILE statement.
- Added --des-key-file option to mysqld .
- HEX(string) now returns the characters in string converted to hexadecimal.
- Fixed problem with GRANT when using lower_case_table_names=1 .
- Changed SELECT ... IN SHARE MODE to SELECT ... LOCK IN SHARE MODE (as in MySQL 3.23).
- A new query cache to cache results from identical SELECT queries.
- Fixed core dump bug on 64-bit machines when it got an incorrect communication packet.
- MATCH ... AGAINST(... IN BOOLEAN MODE) can now work without FULLTEXT index.
- Fixed slave to replicate from 3.23 master.
- Miscellaneous replication fixes/cleanup.
- Got shutdown to work on Mac OS X.
- Added myisam/ft_dump utility for low-level inspection of FULLTEXT indexes.
- Fixed bug in DELETE ... WHERE ... MATCH ... .
- Added support for MATCH ... AGAINST(... IN BOOLEAN MODE) . **Note: you must rebuild your tables with** ALTER TABLE tablename TYPE=MyISAM **to be able to use boolean full-text search .**
- LOCATE() and INSTR() are now case-sensitive if either argument is a binary string.
- Changed RAND() initialization so that RAND(N) and RAND(N+1) are more distinct.
- Fixed core dump bug in UPDATE ... ORDER BY .
- In 3.23, INSERT INTO ... SELECT always had IGNORE enabled. Now MySQL will stop (and possibly roll back) by default in case of an error unless you specify IGNORE .
- Ignore DATA DIRECTORY and INDEX DIRECTORY directives on Windows.
- Added boolean full-text search code. It should be considered early alpha.
- Extended MODIFY and CHANGE in ALTER TABLE to accept the FIRST and AFTER keywords.
- Indexes are now used with ORDER BY on a whole InnoDB table.

26.3.22 Changements de la version 4.0.0 (Octobre 2001 : alpha)

- Added --xml option to mysql for producing XML output.
- Added full-text variables ft_min_word_len , ft_max_word_len , and ft_max_word_len_for_sort system variables.
- Added full-text variables ft_min_word_len , ft_max_word_len , and ft_max_word_len_for_sort variables to myisamchk .
- Added documentation for libmysqld , the embedded MySQL server library. Also added example programs (a mysql client and mysqltest test program) which use libmysqld .
- Removed all Gemini hooks from MySQL server.
- Removed my_thread_init() and my_thread_end() from mysql_com.h , and added mysql_thread_init() and mysql_thread_end() to mysql.h .
- Support for communication packets > 16MB. In 4.0.1 we will extend MyISAM to be able to handle these.
- Secure connections (with SSL).
- Unsigned BIGINT constants now work. MIN() and MAX() now handle signed and unsigned BIGINT numbers correctly.
- New character set latin1_de which provides correct German sorting.
- STRCMP() now uses the current character set when doing comparisons, which means that the default comparison behavior now is case insensitive.
- TRUNCATE TABLE and DELETE FROM tbl_name are now separate functions. One bonus is that DELETE FROM tbl_name now returns the number of deleted rows, rather than zero.
- DROP DATABASE now executes a DROP TABLE on all tables in the database, which fixes a problem with InnoDB tables.
- Added support for UNION .
- Added support for multiple-table DELETE operations.
- A new HANDLER interface to MyISAM tables.
- Added support for INSERT on MERGE tables. Patch from Benjamin Pflugmann.
- Changed WEEK(date,0) to match the calendar in the USA.
- COUNT(DISTINCT) is about 30% faster.
- Speed up all internal list handling.
- Speed up IS NULL , ISNULL() and some other internal primitives.
- Full-text index creation now is much faster.
- Tree-like cache to speed up bulk inserts and myisam_bulk_insert_tree_size variable.
- Searching on packed (CHAR / VARCHAR) keys is now much faster.
- Optimized queries of type: SELECT DISTINCT * from tbl_name ORDER by key_part1 LIMIT row_count .
- SHOW CREATE TABLE now shows all table attributes.
- ORDER BY ... DESC can now use keys.
- LOAD DATA FROM MASTER ``automatically'' sets up a slave.
- Renamed safe_mysqld to mysqld_safe to make this name more in line with other MySQL scripts/commands.
- Added support for symbolic links to MyISAM tables. Symlink handling is now enabled by default for Windows.
- Added SQL_CALC_FOUND_ROWS and FOUND_ROWS() . This makes it possible to know how many rows a query would have returned without a LIMIT clause.
- Changed output format of SHOW OPEN TABLES .
- Allow SELECT expression LIMIT ... .
- Added ORDER BY syntax to UPDATE and DELETE .
- SHOW INDEXES is now a synonym for SHOW INDEX .
- Added ALTER TABLE tbl_name DISABLE KEYS and ALTER TABLE tbl_name ENABLE KEYS commands.
- Allow use of IN as a synonym for FROM in SHOW commands.
- Implemented ``repair by sort'' for FULLTEXT indexes. REPAIR TABLE , ALTER TABLE , and OPTIMIZE TABLE for tables with FULLTEXT indexes are now up to 100 times faster.
- Allow standard SQL syntax X'hexadecimal-number' .
- Cleaned up global lock handling for FLUSH TABLES WITH READ LOCK .
- Fixed problem with DATETIME = constant in WHERE optimization.
- Added --master-data and --no-autocommit options to mysqldump . (Thanks to Brian Aker for this.)
- Added script mysql_explain_log.sh to distribution. (Thanks to mobile.de).

26.4 Changements de la version 3.23.x (Recent; still supported)

Please note that since release 4.0 is now production level, only critical fixes are done in the 3.23 release series. You are recommended to upgrade when possible, to take advantage of all speed and feature improvements in 4.0. Mise à jour de Version 3.23 vers 4.0 .

The 3.23 release has several major features that are not present in previous versions. We have added three new table types:

MyISAM
: A new ISAM library which is tuned for SQL and supports large files.

InnoDB
: A transaction−safe storage engine that supports row level locking, and many Oracle−like features.

BerkeleyDB *or* BDB
: Uses the Berkeley DB library from Sleepycat Software to implement transaction−safe tables.

Note that only MyISAM is available in the standard binary distribution.The 3.23 release also includes support for database replication between a master and many slaves, full−text indexing, and much more.

All new features are being developed in the 4.x version. Only bug fixes and minor enhancements to existing features will be added to 3.23.

The replication code and BerkeleyDB code is still not as tested and as the rest of the code, so we will probably need to do a couple of future releases of 3.23 with small fixes for this part of the code. As long as you don't use these features, you should be quite safe with MySQL 3.23!

Note that the above doesn't mean that replication or Berkeley DB don't work. We have done a lot of testing of all code, including replication and BDB without finding any problems. It only means that not as many users use this code as the rest of the code and because of this we are not yet 100% confident in this code.

26.4.1 Changements de la version 3.23.59 (not released yet)

- Fixed a symlink vulnerability in mysqlbug script − vulnerability id CAN−2004−0381. (Bug #3284)
- Fixed bug in privilege checking of ALTER TABLE RENAME . (Bug #3270)
- Fixed bugs in ACOS() , ASIN() (Bug #2338) and in FLOOR() (Bug #3051). The cause of the problem is an overly strong optimization done by gcc in this case.
- Fixed bug in INSERT ... SELECT statements where, if a NOT NULL column is assigned a value of NULL , the following columns in the row might be assigned a value of zero. (Bug #2012)
- If a query was ignored on the slave (because of replicate−ignore−table and other similar rules), the slave still checked if the query got the same error code (0, no error) as on the master. So if the master had an error on the query (for example, ``Duplicate entry'' in a multiple−row insert), then the slave stopped and warned that the error codes didn't match. This is a backport of the fix for MySQL 4.0. (Bug #797)
- mysqlbinlog now asks for a password at console when the −p/−−password option is used with no argument. This is how the other clients (mysqladmin , mysqldump ..) already behave. Note that one now has to use mysqlbinlog −p<my_password> ; mysqlbinlog −p <my_password> will not work anymore (in other words, put no space after −p). (Bug #1595)
- On some 64−bit machines (some HP−UX and Solaris machines), a slave installed with the 64−bit MySQL binary could not connect to its master (it connected to itself instead). (Bug #1256, Bug #1381)
- Fixed a Windows−specific bug present since MySQL 3.23.57 and 3.23.58 that caused Windows slaves to crash when they started replication if a master.info file existed. (Bug #1720)
- Fixed bug in ALTER TABLE RENAME , when rename to the table with the same name in another database silently dropped destination table if it existed. (Bug #2628)

26.4.2 Changements de la version 3.23.58 (11 septembre 2003)

- Fixed buffer overflow in password handling which could potentially be exploited by MySQL users with ALTER privilege on the mysql.user table to execute random code or to gain shell access with the UID of the mysqld process (thanks to Jedi/Sector One for spotting and reporting this bug).
- mysqldump now correctly quotes all identifiers when communicating with the server. This assures that during the dump process, mysqldump will never send queries to the server that result in a syntax error. This problem is **not** related to the mysqldump program's output, which was not changed. (Bug #1148)
- Fixed table/column grant handling − proper sort order (from most specific to less specific, Contrôle d'accès, étape 2 : vérification des requêtes) was not honored. (Bug #928)
- Fixed overflow bug in MyISAM and ISAM when a row is updated in a table with a large number of columns and at least one BLOB/TEXT column.
- Fixed MySQL so that field length (in C API) for the second column in SHOW CREATE TABLE is always larger than the data length. The only known application that was affected by the old behavior was Borland dbExpress, which truncated the output from the command. (Bug #1064)
- Fixed ISAM bug in MAX() optimization.
- Fixed Unknown error when doing ORDER BY on reference table which was used with NULL value on NOT NULL column. (Bug #479)

26.4.3 Changements de la version 3.23.57 (06 juin 2003)

- Fixed problem in alarm handling that could cause problems when getting a packet that is too large.
- Fixed problem when installing MySQL as a service on Windows when one gave 2 arguments (option file group name and service name) to mysqld .
- Fixed kill pid-of-mysqld to work on Mac OS X.
- SHOW TABLE STATUS displayed incorrect Row_format value for tables that have been compressed with myisampack . (Bug #427)
- SHOW VARIABLES LIKE 'innodb_data_file_path' displayed only the name of the first datafile. (Bug #468)
- Fixed security problem where mysqld didn't allow one to UPDATE rows in a table even if one had a global UPDATE privilege and a database SELECT privilege.
- Fixed a security problem with SELECT and wildcarded select list, when user only had partial column SELECT privileges on the table.
- Fixed unlikely problem in optimizing WHERE clause with a constant expression such as in WHERE 1 AND (a=1 AND b=1) .
- Fixed problem on IA-64 with timestamps that caused mysqlbinlog to fail.
- The default option for innodb_flush_log_at_trx_commit was changed from 0 to 1 to make InnoDB tables ACID by default. Options de démarrage InnoDB .
- Fixed problem with too many allocated alarms on slave when connecting to master many times (normally not a very critical error).
- Fixed a bug in replication of temporary tables. (Bug #183)
- Fixed 64-bit bug that affected at least AMD hammer systems.
- Fixed a bug when doing LOAD DATA INFILE IGNORE : When reading the binary log, mysqlbinlog and the replication code read REPLACE instead of IGNORE . This could make the slave's table become different from the master's table. (Bug #218)
- Fixed overflow bug in MyISAM when a row is inserted into a table with a large number of columns and at least one BLOB/TEXT column. Bug was caused by incorrect calculation of the needed buffer to pack data.
- The binary log was not locked during TRUNCATE table_name or DELETE FROM table_name statements, which could cause an INSERT to table_name to be written to the log before the TRUNCATE or DELETE statements.
- Fixed rare bug in UPDATE of InnoDB tables where one row could be updated multiple times.
- Produce an error for empty table and column names.
- Changed PROCEDURE ANALYSE() to report DATE instead of NEWDATE .
- Changed PROCEDURE ANALYSE(#) to restrict the number of values in an ENUM column to # also for string values.
- mysqldump no longer silently deletes the binary logs when invoked with the --master-data or --first-slave option; while this behavior was convenient for some users, others may suffer from it. Now one has to explicitly ask for binary logs to be deleted by using the new --delete-master-logs option.
- Fixed a bug in mysqldump when it was invoked with the --master-data option: The CHANGE MASTER TO statements that were appended to the SQL dump had incorrect coordinates. (Bug #159)

26.4.4 Changements de la version 3.23.56 (13 mars 2003)

- Fixed mysqld crash on extremely small values of sort_buffer variable.
- Fixed a bug in privilege system for GRANT UPDATE on column level.
- Fixed a rare bug when using a date in HAVING with GROUP BY .
- Fixed checking of random part of WHERE clause. (Bug #142)
- Fixed MySQL (and myisamchk) crash on artificially corrupted .MYI files.
- Security enhancement: mysqld no longer reads options from world-writeable config files.
- Security enhancement: mysqld and safe_mysqld now only use the first --user option specified on the command line. (Normally this comes from /etc/my.cnf)
- Security enhancement: Don't allow BACKUP TABLE to overwrite existing files.
- Fixed unlikely deadlock bug when one thread did a LOCK TABLE and another thread did a DROP TABLE . In this case one could do a KILL on one of the threads to resolve the deadlock.
- LOAD DATA INFILE was not replicated by slave if replicate_*_table was set on the slave.
- Fixed a bug in handling CHAR(0) columns that could cause incorrect results from the query.
- Fixed a bug in SHOW VARIABLES on 64-bit platforms. The bug was caused by incorrect declaration of variable server_id .
- The Comment column in SHOW TABLE STATUS now reports that it can contain NULL values (which is the case for a crashed .frm file).
- Fixed the rpl_rotate_logs test to not fail on certain platforms (e.g. Mac OS X) due to a too long file name (changed slave-master-info.opt to .slave-mi).
- Fixed a problem with BLOB NOT NULL columns used with IS NULL .
- Fixed bug in MAX() optimization in MERGE tables.
- Better RAND() initialization for new connections.
- Fixed bug with connect timeout. This bug was manifested on OS's with poll() system call, which resulted in timeout the value specified as it was executed in both select() and poll() .
- Fixed bug in SELECT * FROM table WHERE datetime1 IS NULL OR datetime2 IS NULL .
- Fixed bug in using aggregate functions as argument for INTERVAL , CASE , FIELD , CONCAT_WS , ELT and MAKE_SET functions.
- When running with --lower-case-table-names (default on Windows) and you had tables or databases with mixed case on disk, then executing SHOW TABLE STATUS followed with DROP DATABASE or DROP TABLE could fail with Errcode 13 .
- Fixed bug in logging to binary log (which affects replication) a query that inserts a NULL in an auto_increment field and also uses LAST_INSERT_ID() .
- Fixed bug in mysqladmin --relative .
- On some 64-bit systems, show status reported a strange number for Open_files and Open_streams .

26.4.5 Changements de la version 3.23.55 (23 janvier 2003)

- Fixed double free 'd pointer bug in mysql_change_user() handling, that enabled a specially hacked version of MySQL client to crash mysqld . **Note** , that one needs to login to the server by using a valid user account to be able to exploit this bug.
- Fixed bug with the --slow-log when logging an administrator command (like FLUSH TABLES).
- Fixed bug in GROUP BY when used on BLOB column with NULL values.
- Fixed a bug in handling NULL values in CASE ... WHEN ... .
- Bugfix for --chroot (Changements de la version 3.23.54 (05 décembre 2002)) is reverted. Unfortunately, there is no way to make it to work, without introducing backward-incompatible changes in my.cnf . Those who need --chroot functionality, should upgrade to MySQL 4.0. (The fix in the 4.0 branch did not break backward-compatibility).
- Make --lower-case-table-names default on Mac OS X as the default file system (HFS+) is case insensitive.
- Fixed a bug in scripts/mysqld_safe.sh in NOHUP_NICENESS testing.
- Transactions in AUTOCOMMIT=0 mode didn't rotate binary log.
- Fixed a bug in scripts/make_binary_distribution that resulted in a remaining @HOSTNAME@ variable instead of replacing it with the correct path to the hostname binary.
- Fixed a very unlikely bug that could cause SHOW PROCESSLIST to core dump in pthread_mutex_unlock() if a new thread was connecting.
- Forbid SLAVE STOP if the thread executing the query has locked tables. This removes a possible deadlock situation.

26.4.6 Changements de la version 3.23.54 (05 décembre 2002)

- Fixed a bug, that allowed to crash mysqld with a specially crafted packet.
- Fixed a rare crash (double free 'd pointer) when altering a temporary table.
- Fixed buffer overrun in libmysqlclient library that allowed malicious MySQL server to crash the client application.
- Fixed security-related bug in mysql_change_user() handling. All users are strongly recommended to upgrade to the version 3.23.54.
- Fixed bug that prevented --chroot command-line option of mysqld from working.
- Fixed bug that made OPTIMIZE TABLE to corrupt the table under some rare circumstances.
- Fixed mysqlcheck so it can deal with table names containing dashes.
- Fixed shutdown problem on Mac OS X.
- Fixed bug with comparing an indexed NULL field with <=> NULL .
- Fixed bug that caused IGNORE INDEX and USE INDEX sometimes to be ignored.
- Fixed rare core dump problem in complicated GROUP BY queries that didn't return any result.
- Fixed a bug where MATCH ... AGAINST () >=0 was treated as if it was > .
- Fixed core dump in SHOW PROCESSLIST when running with an active slave (unlikely timing bug).
- Make it possible to use multiple MySQL servers on Windows (code backported from 4.0.2).
- One can create TEMPORARY MERGE tables now.
- Fixed that --core-file works on Linux (at least on kernel 2.4.18).
- Fixed a problem with BDB and ALTER TABLE .
- Fixed reference to freed memory when doing complicated GROUP BY ... ORDER BY queries. Symptom was that mysqld died in function send_fields .
- Allocate heap rows in smaller blocks to get better memory usage.
- Fixed memory allocation bug when storing BLOB values in internal temporary tables used for some (unlikely) GROUP BY queries.
- Fixed a bug in key optimizing handling where the expression WHERE column_name = key_column_name was calculated as true for NULL values.
- Fixed core dump bug when doing LEFT JOIN ... WHERE key_column=NULL .
- Fixed MyISAM crash when using dynamic-row tables with huge numbers of packed fields.
- Updated source tree to be built using automake 1.5 and libtool 1.4 .

26.4.7 Changements de la version 3.23.53 (09 octobre 2002)

- Fixed crash when SHOW INNODB STATUS was used and skip-innodb was defined.
- Fixed possible memory corruption bug in binary log file handling when slave rotated the logs (only affected 3.23, not 4.0).
- Fixed problem in LOCK TABLES on Windows when one connects to a database that contains upper case letters.
- Fixed that --skip-show-databases doesn't reset the --port option.
- Small fix in safe_mysqld for some shells.
- Fixed that FLUSH STATUS doesn't reset delayed_insert_threads .
- Fixed core dump bug when using the BINARY cast on a NULL value.
- Fixed race condition when someone did a GRANT at the same time a new user logged in or did a USE database .
- Fixed bug in ALTER TABLE and RENAME TABLE when running with -O lower_case_table_names=1 (typically on Windows) when giving the table name in uppercase.
- Fixed that -O lower_case_table_names=1 also converts database names to lower case.
- Fixed unlikely core dump with SELECT ... ORDER BY ... LIMIT .
- Changed AND/OR to report that they can return NULL. This fixes a bug in GROUP BY on AND/OR expressions that return NULL .
- Fixed a bug that OPTIMIZE of locked and modified MyISAM table, reported table corruption.
- Fixed a BDB -related ALTER TABLE bug with dropping a column and shutting down immediately thereafter.
- Fixed problem with configure ... --localstatedir=... .
- Fixed problem with UNSIGNED BIGINT on AIX (again).
- Fixed bug in pthread_mutex_trylock() on HPUX 11.0.
- Multi-threaded stress tests for InnoDB .

26.4.8 Changements de la version 3.23.52 (14 août 2002)

- Wrap BEGIN / COMMIT around transaction in the binary log. This makes replication honor transactions.
- Fixed security bug when having an empty database name in the user.db table.
- Changed initialization of RND() to make it less predicatable.
- Fixed problem with GROUP BY on result with expression that created a BLOB field.
- Fixed problem with GROUP BY on columns that have NULL values. To solve this we now create an MyISAM temporary table when doing a GROUP BY on a possible NULL item. From MySQL 4.0.5 we can use in memory HEAP tables for this case.
- Fixed problem with privilege tables when downgrading from 4.0.2 to 3.23.
- Fixed thread bug in SLAVE START , SLAVE STOP and automatic repair of MyISAM tables that could cause table cache to be corrupted.
- Fixed possible thread related key-cache-corruption problem with OPTIMIZE TABLE and REPAIR TABLE .
- Added name of 'administrator command' logs.
- Fixed bug with creating an auto-increment value on second part of a UNIQUE() key where first part could contain NULL values.
- Don't write slave-timeout reconnects to the error log.
- Fixed bug with slave net read timeouting
- Fixed a core-dump bug with MERGE tables and MAX() function.
- Fixed bug in ALTER TABLE with BDB tables.
- Fixed bug when logging LOAD DATA INFILE to binary log with no active database.
- Fixed a bug in range optimizer (causing crashes).
- Fixed possible problem in replication when doing DROP DATABASE on a database with InnoDB tables.
- Fixed mysql_info() to return 0 for Duplicates value when using INSERT DELAYED IGNORE .
- Added -DHAVE_BROKEN_REALPATH to the Mac OS X (darwin) compile options in configure.in to fix a failure under high load.

26.4.9 Changements de la version 3.23.51 (31 mai 2002)

- Fix bug with closing tags missing slash for mysqldump XML output.
- Remove end space from ENUM values. (This fixed a problem with SHOW CREATE TABLE .)
- Fixed bug in CONCAT_WS() that cut the result.
- Changed name of server variables Com_show_master_stat to Com_show_master_status and Com_show_slave_stat to Com_show_slave_status .
- Changed handling of gethostbyname() to make the client library thread-safe even if gethostbyname_r doesn't exist.
- Fixed core-dump problem when giving a wrong password string to GRANT .
- Fixed bug in DROP DATABASE with symlinked directory.
- Fixed optimization problem with DATETIME and value outside DATETIME range.

- Removed Sleepycat's BDB doc files from the source tree, as they're not needed (MySQL covers BDB in its own documentation).
- Fixed MIT-pthreads to compile with glibc 2.2 (needed for make dist).
- Fixed the FLOAT(X+1,X) is not converted to FLOAT(X+2,X) . (This also affected DECIMAL , DOUBLE and REAL types)
- Fixed the result from IF() is case in-sensitive if the second and third arguments are case sensitive.
- Fixed core dump problem on OSF/1 in gethostbyname_r .
- Fixed that underflowed decimal fields are not zero filled.
- If we get an overflow when inserting '+11111' for DECIMAL(5,0) UNSIGNED columns, we will just drop the sign.
- Fixed optimization bug with ISNULL(expression_which_cannot_be_null) and ISNULL(constant_expression) .
- Fixed host lookup bug in the glibc library that we used with the 3.23.50 Linux-x86 binaries.

26.4.10 Changements de la version 3.23.50 (21 avril 2002)

- Fixed buffer overflow problem if someone specified a too long datadir parameter to mysqld
- Add missing <row> tags for mysqldump XML output.
- Fixed problem with crash-me and gcc 3.0.4.
- Fixed that @@unknown_variable doesn't hang server.
- Added @@VERSION as a synonym for VERSION() .
- SHOW VARIABLES LIKE 'xxx' is now case-insensitive.
- Fixed timeout for GET_LOCK() on HP-UX with DCE threads.
- Fixed memory allocation bug in the glibc library used to build Linux binaries, which caused mysqld to die in 'free()'.
- Fixed SIGINT and SIGQUIT problems in mysql .
- Fixed bug in character table converts when used with big (> 64K) strings.
- InnoDB now retains foreign key constraints through ALTER TABLE and CREATE/DROP INDEX .
- InnoDB now allows foreign key constraints to be added through the ALTER TABLE syntax.
- InnoDB tables can now be set to automatically grow in size (autoextend).
- Our Linux RPMS and binaries are now compiled with gcc 3.0.4, which should make them a bit faster.
- Fixed some buffer overflow problems when reading startup parameters.
- Because of problems on shutdown we have now disabled named pipes on Windows by default. One can enable named pipes by starting mysqld with --enable-named-pipe .
- Fixed bug when using WHERE key_column = 'J' or key_column='j' .
- Fixed core-dump bug when using --log-bin with LOAD DATA INFILE without an active database.
- Fixed bug in RENAME TABLE when used with lower_case_table_names=1 (default on Windows).
- Fixed unlikely core-dump bug when using DROP TABLE on a table that was in use by a thread that also used queries on only temporary tables.
- Fixed problem with SHOW CREATE TABLE and PRIMARY KEY when using 32 indexes.
- Fixed that one can use SET PASSWORD for the anonymous user.
- Fixed core dump bug when reading client groups from option files using mysql_options() .
- Memory leak (16 bytes per every **corrupted** table) closed.
- Fixed binary builds to use --enable-local-infile .
- Update source to work with new version of bison .
- Updated shell scripts to now agree with new POSIX standard.
- Fixed bug where DATE_FORMAT() returned empty string when used with GROUP BY .

26.4.11 Changements de la version 3.23.49

- For a MERGE table, DELETE FROM merge_table used without a WHERE clause no longer clears the mapping for the table by emptying the .MRG file. Instead, it deletes records from the mapped tables.
- Don't give warning for a statement that is only a comment; this is needed for mysqldump --disable-keys to work.
- Fixed unlikely caching bug when doing a join without keys. In this case, the last used field for a table always returned NULL .
- Added options to make LOAD DATA LOCAL INFILE more secure.
- MySQL binary release 3.23.48 for Linux contained a new glibc library, which has serious problems under high load and Red Hat 7.2. The 3.23.49 binary release doesn't have this problem.
- Fixed shutdown problem on NT.

26.4.12 Changements de la version 3.23.48 (07 février 2002)

- Added --xml option to mysqldump for producing XML output.
- Changed to use autoconf 2.52 (from autoconf 2.13)
- Fixed bug in complicated join with const tables.
- Added internal safety checks for InnoDB .
- Some InnoDB variables were always shown in SHOW VARIABLES as OFF on high-byte-first systems (like SPARC).
- Fixed problem with one thread using an InnoDB table and another thread doing an ALTER TABLE on the same table. Before that, mysqld could crash with an assertion failure in row0row.c , line 474.
- Tuned the InnoDB SQL optimizer to favor index searches more often over table scans.
- Fixed a performance problem with InnoDB tables when several large SELECT queries are run concurrently on a multiprocessor Linux computer. Large CPU-bound SELECT queries will now also generally run faster on all platforms.
- If MySQL binlogging is used, InnoDB now prints after crash recovery the latest MySQL binlog name and the offset InnoDB was able to recover to. This is useful, for example, when resynchronizing a master and a slave database in replication.
- Added better error messages to help in installation problems of InnoDB tables.
- It is now possible to recover MySQL temporary tables that have become orphaned inside the InnoDB tablespace.
- InnoDB now prevents a FOREIGN KEY declaration where the signedness is not the same in the referencing and referenced integer columns.
- Calling SHOW CREATE TABLE or SHOW TABLE STATUS could cause memory corruption and make mysqld crash. Especially at risk was mysqldump , because it frequently calls SHOW CREATE TABLE .
- If inserts to several tables containing an AUTO_INCREMENT column were wrapped inside one LOCK TABLES , InnoDB asserted in lock0lock.c .
- In 3.23.47 we allowed several NULL values in a UNIQUE secondary index for an InnoDB table. But CHECK TABLE was not relaxed: it reports the table as corrupt. CHECK TABLE no longer complains in this situation.
- SHOW GRANTS now shows REFERENCES instead of REFERENCE .

26.4.13 Changements de la version 3.23.47 (27 décembre 2001)

- Fixed bug when using the following construct: SELECT ... WHERE key=@var_name OR key=@var_name2
- Restrict InnoDB keys to 500 bytes.
- InnoDB now supports NULL in keys.
- Fixed shutdown problem on HP−UX. (Introduced in 3.23.46)
- Fixed core dump bug in replication when using SELECT RELEASE_LOCK() .
- Added new command: DO expression,[expression]
- Added slave−skip−errors option.
- Added statistics variables for all MySQL commands. (SHOW STATUS is now much longer.)
- Fixed default values for InnoDB tables.
- Fixed that GROUP BY expr DESC works.
- Fixed bug when using t1 LEFT JOIN t2 ON t2.key=constant .
- mysql_config now also works with binary (relocated) distributions.

26.4.14 Changements de la version 3.23.46 (29 novembre 2001)

- Fixed problem with aliased temporary table replication.
- InnoDB and BDB tables will now use index when doing an ORDER BY on the whole table.
- Fixed bug where one got an empty set instead of a DEADLOCK error when using BDB tables.
- One can now kill ANALYZE , REPAIR , and OPTIMIZE TABLE when the thread is waiting to get a lock on the table.
- Fixed race condition in ANALYZE TABLE .
- Fixed bug when joining with caching (unlikely to happen).
- Fixed race condition when using the binary log and INSERT DELAYED which could cause the binary log to have rows that were not yet written to MyISAM tables.
- Changed caching of binary log to make replication slightly faster.
- Fixed bug in replication on Mac OS X.

26.4.15 Changements de la version 3.23.45 (22 novembre 2001)

- (UPDATE|DELETE) ...WHERE MATCH bugfix.
- shutdown should now work on Darwin (Mac OS X).
- Fixed core dump when repairing corrupted packed MyISAM files.
- −−core−file now works on Solaris.
- Fix a bug which could cause InnoDB to complain if it cannot find free blocks from the buffer cache during recovery.
- Fixed bug in InnoDB insert buffer B−tree handling that could cause crashes.
- Fixed bug in InnoDB lock timeout handling.
- Fixed core dump bug in ALTER TABLE on a TEMPORARY InnoDB table.
- Fixed bug in OPTIMIZE TABLE that reset index cardinality if it was up to date.
- Fixed problem with t1 LEFT_JOIN t2 ... WHERE t2.date_column IS NULL when date_column was declared as NOT NULL .
- Fixed bug with BDB tables and keys on BLOB columns.
- Fixed bug in MERGE tables on OS with 32−bit file pointers.
- Fixed bug in TIME_TO_SEC() when using negative values.

26.4.16 Changements de la version 3.23.44 (31 octobre 2001)

- Fixed Rows_examined count in slow query log.
- Fixed bug when using a reference to an AVG() column in HAVING .
- Fixed that date functions that require correct dates, like DAYOFYEAR(column) , will return NULL for 0000−00−00 dates.
- Fixed bug in const−propagation when comparing columns of different types. (SELECT * FROM date_col="2001−01−01" and date_col=time_col)
- Fixed bug that caused error message Can't write, because of unique constraint with some GROUP BY queries.
- Fixed problem with sjis character strings used within quoted table names.
- Fixed core dump when using CREATE ... FULLTEXT keys with other storage engines than MyISAM .
- Don't use signal() on Windows because this appears to not be 100% reliable.
- Fixed bug when doing WHERE col_name=NULL on an indexed column that had NULL values.
- Fixed bug when doing LEFT JOIN ... ON (col_name = constant) WHERE col_name = constant .
- When using replications, aborted queries that contained % could cause a core dump.
- TCP_NODELAY was not used on some systems. (Speed problem.)
- Applied portability fixes for OS/2. (Patch by Yuri Dario.)

The following changes are for InnoDB tables:

- Add missing InnoDB variables to SHOW VARIABLES .
- Foreign keys checking is now done for InnoDB tables.
- DROP DATABASE now works also for InnoDB tables.
- InnoDB now supports datafiles and raw disk partitions bigger than 4 GB on those operating systems that have big files.
- InnoDB calculates better table cardinality estimates for the MySQL optimizer.
- Accent characters in the default character set latin1 are ordered according to the MySQL ordering.Note: if you are using latin1 and have inserted characters whose code is greater than 127 into an indexed CHAR column, you should run CHECK TABLE on your table when you upgrade to 3.23.44, and drop and reimport the table if CHECK TABLE reports an error!
- A new my.cnf parameter, innodb_thread_concurrency , helps in performance tuning in heavily concurrent environments.
- A new my.cnf parameter, innodb_fast_shutdown , speeds up server shutdown.
- A new my.cnf parameter, innodb_force_recovery , helps to save your data in case the disk image of the database becomes corrupt.
- innodb_monitor has been improved and a new innodb_table_monitor added.
- Increased maximum key length from 500 to 7000 bytes.
- Fixed a bug in replication of AUTO_INCREMENT columns with multiple−line inserts.
- Fixed a bug when the case of letters changes in an update of an indexed secondary column.
- Fixed a hang when there are > 24 datafiles.
- Fixed a crash when MAX(col) is selected from an empty table, and col is not the first column in a multi−column index.
- Fixed a bug in purge which could cause crashes.

26.4.17 Changements de la version 3.23.43 (04 octobre 2001)

- Fixed a bug in INSERT DELAYED and FLUSH TABLES introduced in 3.23.42.
- Fixed unlikely bug, which returned non-matching rows, in SELECT with many tables and multi-column indexes and 'range' type.
- Fixed an unlikely core dump bug when doing EXPLAIN SELECT when using many tables and ORDER BY .
- Fixed bug in LOAD DATA FROM MASTER when using table with CHECKSUM=1 .
- Added unique error message when one gets a DEADLOCK during a transaction with BDB tables.
- Fixed problem with BDB tables and UNIQUE columns defined as NULL .
- Fixed problem with myisampack when using pre-space filled CHAR columns.
- Applied patch from Yuri Dario for OS/2.
- Fixed bug in --safe-user-create .

26.4.18 Changements de la version 3.23.42 (08 septembre 2001)

- Fixed problem when using LOCK TABLES and BDB tables.
- Fixed problem with REPAIR TABLE on MyISAM tables with row lengths in the range from 65517 to 65520 bytes.
- Fixed rare hang when doing mysqladmin shutdown when there was a lot of activity in other threads.
- Fixed problem with INSERT DELAYED where delayed thread could be hanging on upgrading locks for no apparent reason.
- Fixed problem with myisampack and BLOB .
- Fixed problem when one edited .MRG tables by hand. (Patch from Benjamin Pflugmann).
- Enforce that all tables in a MERGE table come from the same database.
- Fixed bug with LOAD DATA INFILE and transactional tables.
- Fix bug when using INSERT DELAYED with wrong column definition.
- Fixed core dump during REPAIR of some particularly broken tables.
- Fixed bug in InnoDB and AUTO_INCREMENT columns.
- Fixed bug in InnoDB and RENAME TABLE columns.
- Fixed critical bug in InnoDB and BLOB columns. If you have used BLOB columns larger than 8000 bytes in an InnoDB table, it is necessary to dump the table with mysqldump , drop it and restore it from the dump.
- Applied large patch for OS/2 from Yuri Dario.
- Fixed problem with InnoDB when one could get the error Can't execute the given command... even when no transaction was active.
- Applied some minor fixes that concern Gemini.
- Use real arithmetic operations even in integer context if not all arguments are integers. (Fixes uncommon bug in some integer contexts).
- Don't force everything to lowercase on Windows. (To fix problem with Windows and ALTER TABLE). Now --lower_case_names also works on Unix.
- Fixed that automatic rollback is done when thread end doesn't lock other threads.

26.4.19 Changements de la version 3.23.41 (11 août 2001)

- Added --sql-mode=value[,value[,value]] option to mysqld . Utiliser les options de ligne de commande .
- Fixed possible problem with shutdown on Solaris where the .pid file wasn't deleted.
- InnoDB now supports < 4 GB rows. The former limit was 8000 bytes.
- The doublewrite file flush method is used in InnoDB . It reduces the need for Unix fsync() calls to a fraction and improves performance on most Unix flavors.
- You can now use the InnoDB Monitor to print a lot of InnoDB state information, including locks, to the standard output. This is useful in performance tuning.
- Several bugs which could cause hangs in InnoDB have been fixed.
- Split record_buffer to record_buffer and record_rnd_buffer . To make things compatible to previous MySQL versions, if record_rnd_buffer is not set, then it takes the value of record_buffer .
- Fixed optimizing bug in ORDER BY where some ORDER BY parts where wrongly removed.
- Fixed overflow bug with ALTER TABLE and MERGE tables.
- Added prototypes for my_thread_init() and my_thread_end() to mysql_com.h
- Added --safe-user-create option to mysqld .
- Fixed bug in SELECT DISTINCT ... HAVING that caused error message Can't find record in #...

26.4.20 Changements de la version 3.23.40

- Fixed problem with --low-priority-updates and INSERT statements.
- Fixed bug in slave thread when under some rare circumstances it could get 22 bytes ahead on the offset in the master.
- Added slave_net_timeout for replication.
- Fixed problem with UPDATE and BDB tables.
- Fixed hard bug in BDB tables when using key parts.
- Fixed problem when using GRANT FILE ON database.* ... ; previously we added the DROP privilege for the database.
- Fixed DELETE FROM tbl_name ... LIMIT 0 and UPDATE FROM tbl_name ... LIMIT 0 , which acted as though the LIMIT clause was not present (they deleted or updated all selected rows).
- CHECK TABLE now checks if an AUTO_INCREMENT column contains the value 0.
- Sending a SIGHUP to mysqld will now only flush the logs, not reset the replication.
- Fixed parser to allow floats of type 1.0e1 (no sign after e).
- Option --force to myisamchk now also updates states.
- Added option --warnings to mysqld . Now mysqld prints the error Aborted connection only if this option is used.
- Fixed problem with SHOW CREATE TABLE when you didn't have a PRIMARY KEY .
- Properly fixed the rename of innodb_unix_file_flush_method variable to innodb_flush_method .
- Fixed bug when converting BIGINT UNSIGNED to DOUBLE . This caused a problem when doing comparisons with BIGINT values outside of the signed range.
- Fixed bug in BDB tables when querying empty tables.
- Fixed a bug when using COUNT(DISTINCT) with LEFT JOIN and there weren't any matching rows.
- Removed all documentation referring to the GEMINI table type. GEMINI is not released under an Open Source license.

26.4.21 Changements de la version 3.23.39 (12 juin 2001)

- The AUTO_INCREMENT sequence wasn't reset when dropping and adding an AUTO_INCREMENT column.
- CREATE ... SELECT now creates non-unique indexes delayed.
- Fixed problem where LOCK TABLES tbl_name READ followed by FLUSH TABLES put an exclusive lock on the table.
- REAL @variable values were represented with only 2 digits when converted to strings.
- Fixed problem that client ``hung'' when LOAD TABLE FROM MASTER failed.
- myisamchk --fast --force will no longer repair tables that only had the open count wrong.
- Added functions to handle symbolic links to make life easier in 4.0.
- We are now using the -lcma thread library on HP-UX 10.20 so that MySQL will be more stable on HP-UX.
- Fixed problem with IF() and number of decimals in the result.
- Fixed date-part extraction functions to work with dates where day and/or month is 0.
- Extended argument length in option files from 256 to 512 chars.
- Fixed problem with shutdown when INSERT DELAYED was waiting for a LOCK TABLE .
- Fixed core dump bug in InnoDB when tablespace was full.
- Fixed problem with MERGE tables and big tables (> 4G) when using ORDER BY .

26.4.22 Changements de la version 3.23.38 (09 mai 2001)

- Fixed a bug when SELECT from MERGE table sometimes results in incorrectly ordered rows.
- Fixed a bug in REPLACE() when using the ujis character set.
- Applied Sleepycat BDB patches 3.2.9.1 and 3.2.9.2.
- Added --skip-stack-trace option to mysqld .
- CREATE TEMPORARY now works with InnoDB tables.
- InnoDB now promotes sub keys to whole keys.
- Added option CONCURRENT to LOAD DATA .
- Better error message when slave max_allowed_packet is too low to read a very long log event from the master.
- Fixed bug when too many rows where removed when using SELECT DISTINCT ... HAVING .
- SHOW CREATE TABLE now returns TEMPORARY for temporary tables.
- Added Rows_examined to slow query log.
- Fixed problems with function returning empty string when used together with a group function and a WHERE that didn't match any rows.
- New program mysqlcheck .
- Added database name to output for administrative commands like CHECK , REPAIR , OPTIMIZE .
- Lots of portability fixes for InnoDB .
- Changed optimizer so that queries like SELECT * FROM tbl_name,tbl_name2 ... ORDER BY key_part1 LIMIT row_count will use index on key_part1 instead of filesort .
- Fixed bug when doing LOCK TABLE to_table WRITE,...; INSERT INTO to_table... SELECT ... when to_table was empty.
- Fixed bug with LOCK TABLE and BDB tables.

26.4.23 Changements de la version 3.23.37 (17 avril 2001)

- Fixed a bug when using MATCH() in HAVING clause.
- Fixed a bug when using HEAP tables with LIKE .
- Added --mysql-version option to safe_mysqld
- Changed INNOBASE to InnoDB (because the INNOBASE name was already used). All configure options and mysqld start options now use innodb instead of innobase . This means that before upgrading to this version, you have to change any configuration files where you have used innobase options!
- Fixed bug when using indexes on CHAR(255) NULL columns.
- Slave thread will now be started even if master-host is not set, as long as server-id is set and valid master.info is present.
- Partial updates (terminated with kill) are now logged with a special error code to the binary log. Slave will refuse to execute them if the error code indicates the update was terminated abnormally, and will have to be recovered with SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START after a manual sanity check/correction of data integrity.
- Fixed bug that erroneously logged a drop of internal temporary table on thread termination to the binary log --- this bug affected replication.
- Fixed a bug in REGEXP on 64-bit machines.
- UPDATE and DELETE with WHERE unique_key_part IS NULL didn't update/delete all rows.
- Disabled INSERT DELAYED for tables that support transactions.
- Fixed bug when using date functions on TEXT / BLOB column with wrong date format.
- UDFs now also work on Windows. (Patch by Ralph Mason.)
- Fixed bug in ALTER TABLE and LOAD DATA INFILE that disabled key-sorting. These commands should now be faster in most cases.
- Fixed performance bug where reopened tables (tables that had been waiting for FLUSH or REPAIR) would not use indexes for the next query.
- Fixed problem with ALTER TABLE to InnoDB tables on FreeBSD.
- Added mysqld variables myisam_max_sort_file_size and myisam_max_extra_sort_file_size .
- Initialize signals early to avoid problem with signals in InnoDB .
- Applied patch for the tis620 character set to make comparisons case-independent and to fix a bug in LIKE for this character set. **Note** : All tables that uses the tis620 character set must be fixed with myisamchk -r or REPAIR TABLE !
- Added --skip-safemalloc option to mysqld .

26.4.24 Changements de la version 3.23.36 (27 mars 2001)

- Fixed a bug that allowed use of database names containing a '.' character. This fixes a serious security issue when mysqld is run as root.
- Fixed bug when thread creation failed (could happen when doing a **lot** of connections in a short time).
- Fixed some problems with FLUSH TABLES and TEMPORARY tables. (Problem with freeing the key cache and error Can't reopen table... .)
- Fixed a problem in InnoDB with other character sets than latin1 and another problem when using many columns.
- Fixed bug that caused a core dump when using a very complex query involving DISTINCT and summary functions.
- Added SET TRANSACTION ISOLATION LEVEL ...
- Added SELECT ... FOR UPDATE .
- Fixed bug where the number of affected rows was not returned when MySQL was compiled without transaction support.
- Fixed a bug in UPDATE where keys weren't always used to find the rows to be updated.
- Fixed a bug in CONCAT_WS() where it returned incorrect results.
- Changed CREATE ... SELECT and INSERT ... SELECT to not allow concurrent inserts as this could make the binary log hard to repeat. (Concurrent inserts are enabled if you are not using the binary or update log.)

- Changed some macros to be able to use fast mutex with glibc 2.2.

26.4.25 Changements de la version 3.23.35 (15 mars 2001)

- Fixed newly introduced bug in ORDER BY .
- Fixed wrong define CLIENT_TRANSACTIONS .
- Fixed bug in SHOW VARIABLES when using INNOBASE tables.
- Setting and using user variables in SELECT DISTINCT didn't work.
- Tuned SHOW ANALYZE for small tables.
- Fixed handling of arguments in the benchmark script run−all−tests .

26.4.26 Changements de la version 3.23.34a

- Added extra files to the distribution to allow INNOBASE support to be compiled.

26.4.27 Changements de la version 3.23.34 (10 mars 2001)

- Added the INNOBASE storage engine and the BDB storage engine to the MySQL source distribution.
- Updated the documentation about GEMINI tables.
- Fixed a bug in INSERT DELAYED that caused threads to hang when inserting NULL into an AUTO_INCREMENT column.
- Fixed a bug in CHECK TABLE / REPAIR TABLE that could cause a thread to hang.
- REPLACE will not replace a row that conflicts with an AUTO_INCREMENT generated key.
- mysqld now only sets CLIENT_TRANSACTIONS in mysql−>server_capabilities if the server supports a transaction−safe storage engine.
- Fixed LOAD DATA INFILE to allow numeric values to be read into ENUM and SET columns.
- Improved error diagnostic for slave thread exit.
- Fixed bug in ALTER TABLE ... ORDER BY .
- Added max_user_connections variable to mysqld .
- Limit query length for replication by max_allowed_packet , not the arbitrary limit of 4 MB.
- Allow space around = in argument to −−set−variable .
- Fixed problem in automatic repair that could leave some threads in state Waiting for table .
- SHOW CREATE TABLE now displays the UNION=() for MERGE tables.
- ALTER TABLE now remembers the old UNION=() definition.
- Fixed bug when replicating timestamps.
- Fixed bug in bidirectional replication.
- Fixed bug in the BDB storage engine that occurred when using an index on multi−part key where a key part may be NULL .
- Fixed MAX() optimization on sub−key for BDB tables.
- Fixed problem where garbage results were returned when using BDB tables and BLOB or TEXT fields when joining many tables.
- Fixed a problem with BDB tables and TEXT columns.
- Fixed bug when using a BLOB key where a const row wasn't found.
- Fixed that mysqlbinlog writes the timestamp value for each query. This ensures that one gets same values for date functions like NOW() when using mysqlbinlog to pipe the queries to another server.
- Allow −−skip−gemini , −−skip−bdb , and −−skip−innodb options to be specified when invoking mysqld , even if these storage engines are not compiled in to mysqld .
- One can now do GROUP BY ... DESC .
- Fixed a deadlock in the SET code, when one ran SET @foo=bar , where bar is a column reference, an error was not properly generated.

26.4.28 Changements de la version 3.23.33 (09 février 2001)

- Fixed DNS lookups not to use the same mutex as the hostname cache. This will enable known hosts to be quickly resolved even if a DNS lookup takes a long time.
- Added −−character−sets−dir option to myisampack .
- Removed warnings when running REPAIR TABLE ... EXTENDED .
- Fixed a bug that caused a core dump when using GROUP BY on an alias, where the alias was the same as an existing column name.
- Added SEQUENCE() as an example UDF function.
- Changed mysql_install_db to use BINARY for CHAR columns in the privilege tables.
- Changed TRUNCATE tbl_name to TRUNCATE TABLE tbl_name to use the same syntax as Oracle. Until 4.0 we will also allow TRUNCATE tbl_name to not crash old code.
- Fixed ``no found rows'' bug in MyISAM tables when a BLOB was first part of a multi−part key.
- Fixed bug where CASE didn't work with GROUP BY .
- Added −−sort−recover option to myisamchk .
- myisamchk −S and OPTIMIZE TABLE now work on Windows.
- Fixed bug when using DISTINCT on results from functions that referred to a group function, like:

```
 SELECT a, DISTINCT SEC_TO_TIME(SUM(a))
FROM tbl_name GROUP BY a, b;
```

- Fixed buffer overrun in libmysqlclient library. Fixed bug in handling STOP event after ROTATE event in replication.
- Fixed another buffer overrun in DROP DATABASE .
- Added Table_locks_immediate and Table_locks_waited status variables.
- Fixed bug in replication that broke slave server start with existing master.info . This fixes a bug introduced in 3.23.32.
- Added SET SQL_SLAVE_SKIP_COUNTER=n command to recover from replication glitches without a full database copy.
- Added max_binlog_size variable; the binary log will be rotated automatically when the size crosses the limit.
- Added Last_Error , Last_Errno , and Slave_skip_counter variables to SHOW SLAVE STATUS .
- Fixed bug in MASTER_POS_WAIT() function.
- Execute core dump handler on SIGILL , and SIGBUS in addition to SIGSEGV .
- On x86 Linux, print the current query and thread (connection) id, if available, in the core dump handler.
- Fixed several timing bugs in the test suite.
- Extended mysqltest to take care of the timing issues in the test suite.
- ALTER TABLE can now be used to change the definition for a MERGE table.
- Fixed creation of MERGE tables on Windows.
- Portability fixes for OpenBSD and OS/2.
- Added −−temp−pool option to mysqld . Using this option will cause most temporary files created to use a small set of names, rather than a unique name for each new file. This is to work around a problem in the Linux kernel dealing with creating a bunch of new files with different names. With the old behavior, Linux seems to "leak" memory, as it's being allocated to the

directory entry cache instead of the disk cache.

26.4.29 Changements de la version 3.23.32 (22 Jan 2001: Production)

- Changed code to get around compiler bug in Compaq C++ on OSF/1, that broke BACKUP , RESTORE , CHECK , REPAIR , and ANALYZE TABLE .
- Added option FULL to SHOW COLUMNS . Now we show the privilege list for the columns only if this option is given.
- Fixed bug in SHOW LOGS when there weren't any BDB logs.
- Fixed a timing problem in replication that could delay sending an update to the client until a new update was done.
- Don't convert field names when using mysql_list_fields() . This is to keep this code compatible with SHOW FIELDS .
- MERGE tables didn't work on Windows.
- Fixed problem with SET PASSWORD=... on Windows.
- Added missing my_config.h to RPM distribution.
- TRIM("foo" from "foo") didn't return an empty string.
- Added --with-version-suffix option to configure .
- Fixed core dump when client aborted connection without mysql_close() .
- Fixed a bug in RESTORE TABLE when trying to restore from a non-existent directory.
- Fixed a bug which caused a core dump on the slave when replicating SET PASSWORD .
- Added MASTER_POS_WAIT() .

26.4.30 Changements de la version 3.23.31 (17 janvier 2001)

- The test suite now tests all reachable BDB interface code. During testing we found and fixed many errors in the interface code.
- Using HAVING on an empty table could produce one result row when it shouldn't.
- Fixed the MySQL RPM so it no longer depends on Perl5.
- Fixed some problems with HEAP tables on Windows.
- SHOW TABLE STATUS didn't show correct average row length for tables larger than 4G.
- CHECK TABLE ... EXTENDED didn't check row links for fixed size tables.
- Added option MEDIUM to CHECK TABLE .
- Fixed problem when using DECIMAL() keys on negative numbers.
- HOUR() (and some other TIME functions) on a CHAR column always returned NULL .
- Fixed security bug in something (please upgrade if you are using an earlier MySQL 3.23 version).
- Fixed buffer overflow bug when writing a certain error message.
- Added usage of setrlimit() on Linux to get -O --open-files-limit=# to work on Linux.
- Added bdb_version variable to mysqld .
- Fixed bug when using expression of type:

  ```
  SELECT ... FROM t1 LEFT JOIN t2 ON (t1.a=t2.a) WHERE t1.a=t2.a
  ```

  In this case the test in the WHERE clause was wrongly optimized away.
- Fixed bug in MyISAM when deleting keys with possible NULL values, but the first key-column was not a prefix-compressed text column.
- Fixed mysql.server to read the [mysql.server] option file group rather than the [mysql_server] group.
- Fixed safe_mysqld and mysql.server to also read the server option section.
- Added Threads_created status variable to mysqld .

26.4.31 Changements de la version 3.23.30 (04 janvier 2001)

- Added SHOW OPEN TABLES command.
- Fixed that myisamdump works against old mysqld servers.
- Fixed myisamchk -k# so that it works again.
- Fixed a problem with replication when the binary log file went over 2G on 32-bit systems.
- LOCK TABLES will now automatically start a new transaction.
- Changed BDB tables to not use internal subtransactions and reuse open files to get more speed.
- Added --mysqld=# option to safe_mysqld .
- Allow hex constants in the --fields-*-by and --lines-terminated-by options to mysqldump and mysqlimport . By Paul DuBois.
- Added --safe-show-database option to mysqld .
- Added have_bdb , have_gemini , have_innobase , have_raid and have_openssl to SHOW VARIABLES to make it easy to test for supported extensions.
- Added --open-files-limit option to mysqld .
- Changed --open-files option to --open-files-limit in safe_mysqld .
- Fixed a bug where some rows were not found with HEAP tables that had many keys.
- Fixed that --bdb-no-sync works.
- Changed --bdb-recover to --bdb-no-recover as recover should be on by default.
- Changed the default number of BDB locks to 10000.
- Fixed a bug from 3.23.29 when allocating the shared structure needed for BDB tables.
- Changed mysqld_multi.sh to use configure variables. Patch by Christopher McCrory.
- Added fixing of include files for Solaris 2.8.
- Fixed bug with --skip-networking on Debian Linux.
- Fixed problem that some temporary files where reported as having the name UNOPENED in error messages.
- Fixed bug when running two simultaneous SHOW LOGS queries.

26.4.32 Changements de la version 3.23.29 (16 décembre 2000)

- Configure updates for Tru64, large file support, and better TCP wrapper support. By Albert Chin-A-Young.
- Fixed bug in <=> operator.
- Fixed bug in REPLACE with BDB tables.
- LPAD() and RPAD() will shorten the result string if it's longer than the length argument.
- Added SHOW LOGS command.
- Remove unused BDB logs on shutdown.
- When creating a table, put PRIMARY keys first, followed by UNIQUE keys.
- Fixed a bug in UPDATE involving multi-part keys where one specified all key parts both in the update and the WHERE part. In this case MySQL could try to update a record that didn't match the whole WHERE part.
- Changed drop table to first drop the tables and then the .frm file.
- Fixed a bug in the hostname cache which caused mysqld to report the hostname as '' in some error messages.

- Fixed a bug with HEAP type tables; the variable max_heap_table_size wasn't used. Now either MAX_ROWS or max_heap_table_size can be used to limit the size of a HEAP type table.
- Changed the default server-id to 1 for masters and 2 for slaves to make it easier to use the binary log.
- Renamed bdb_lock_max variable to bdb_max_lock .
- Added support for AUTO_INCREMENT on sub-fields for BDB tables.
- Added ANALYZE of BDB tables.
- In BDB tables, we now store the number of rows; this helps to optimize queries when we need an approximation of the number of rows.
- If we get an error in a multi-row statement, we now only roll back the last statement, not the entire transaction.
- If you do a ROLLBACK when you have updated a non-transactional table you will get an error as a warning.
- Added --bdb-shared-data option to mysqld .
- Added Slave_open_temp_tables status variable to mysqld
- Added binlog_cache_size and max_binlog_cache_size variables to mysqld .
- DROP TABLE , RENAME TABLE , CREATE INDEX and DROP INDEX are now transaction endpoints.
- If you do a DROP DATABASE on a symbolically linked database, both the link and the original database is deleted.
- Fixed DROP DATABASE to work on OS/2.
- Fixed bug when doing a SELECT DISTINCT ... table1 LEFT JOIN table2 ... when table2 was empty.
- Added --abort-slave-event-count and --disconnect-slave-event-count options to mysqld for debugging and testing of replication.
- Fixed replication of temporary tables. Handles everything except slave server restart.
- SHOW KEYS now shows whether key is FULLTEXT .
- New script mysqld_multi . mysqld_multi , un programme pour gérer plusieurs serveurs MySQL .
- Added new script, mysql-multi.server.sh . Thanks to Tim Bunce Tim.Bunce@ig.co.uk for modifying mysql.server to easily handle hosts running many mysqld processes.
- safe_mysqld , mysql.server , and mysql_install_db have been modified to use mysql_print_defaults instead of various hacks to read the my.cnf files. In addition, the handling of various paths has been made more consistent with how mysqld handles them by default.
- Automatically remove Berkeley DB transaction logs that no longer are in use.
- Fixed bug with several FULLTEXT indexes in one table.
- Added a warning if number of rows changes on REPAIR / OPTIMIZE .
- Applied patches for OS/2 by Yuri Dario .
- FLUSH TABLES tbl_name didn't always flush the index tree to disk properly.
- --bootstrap is now run in a separate thread. This fixes a problem that caused mysql_install_db to core dump on some Linux machines.
- Changed mi_create() to use less stack space.
- Fixed bug with optimizer trying to over-optimize MATCH() when used with UNIQUE key.
- Changed crash-me and the MySQL benchmarks to also work with FrontBase.
- Allow RESTRICT and CASCADE after DROP TABLE to make porting easier.
- Reset status variable which could cause problem if one used --slow-log .
- Added connect_timeout variable to mysql and mysqladmin .
- Added connect-timeout as an alias for timeout for option files read by mysql_options() .

26.4.33 Changements de la version 3.23.28 (22 Nov 2000: Gamma)

- Added new options --pager[=...] , --no-pager , --tee=... and --no-tee to the mysql client. The new corresponding interactive commands are pager , nopager , tee and notee . L'outil de ligne de commande , mysql --help and the interactive help for more information.
- Fixed crash when automatic repair of MyISAM table failed.
- Fixed a major performance bug in the table locking code when one constantly had a lot of SELECT , UPDATE and INSERT statements running. The symptom was that the UPDATE and INSERT queries were locked for a long time while new SELECT statements were executed before the updates.
- When reading options_files with mysql_options() the return-found-rows option was ignored.
- One can now specify interactive-timeout in the option file that is read by mysql_options() . This makes it possible to force programs that run for a long time (like mysqlhotcopy) to use the interactive_timeout time instead of the wait_timeout time.
- Added to the slow query log the time and the user name for each logged query. If you are using --log-long-format then also queries that do not use an index are logged, even if the query takes less than long_query_time seconds.
- Fixed a problem in LEFT JOIN which caused all columns in a reference table to be NULL .
- Fixed a problem when using NATURAL JOIN without keys.
- Fixed a bug when using a multi-part keys where the first part was of type TEXT or BLOB .
- DROP of temporary tables wasn't stored in the update/binary log.
- Fixed a bug where SELECT DISTINCT * ... LIMIT row_count only returned one row.
- Fixed a bug in the assembler code in strstr() for SPARC and cleaned up the global.h header file to avoid a problem with bad aliasing with the compiler submitted with Red Hat 7.0. (Reported by Trond Eivind Glomsrod)
- The --skip-networking option now works properly on NT.
- Fixed a long outstanding bug in the ISAM tables when a row with a length of more than 65K was shortened by a single byte.
- Fixed a bug in MyISAM when running multiple updating processes on the same table.
- Allow one to use FLUSH TABLE tbl_name .
- Added --replicate-ignore-table , --replicate-do-table , --replicate-wild-ignore-table , and --replicate-wild-do-table options to mysqld .
- Changed all log files to use our own IO_CACHE mechanism instead of FILE to avoid OS problems when there are many files open.
- Added --open-files and --timezone options to safe_mysqld .
- Fixed a fatal bug in CREATE TEMPORARY TABLE ... SELECT ... .
- Fixed a problem with CREATE TABLE ... SELECT NULL .
- Added variables large_file_support , net_read_timeout , net_write_timeout and query_buffer_size to SHOW VARIABLES .
- Added status variables created_tmp_files and sort_merge_passes to SHOW STATUS .
- Fixed a bug where we didn't allow an index name after the FOREIGN KEY definition.
- Added TRUNCATE table_name as a synonym for DELETE FROM table_name .
- Fixed a bug in a BDB key compare function when comparing part keys.
- Added bdb_lock_max variable to mysqld .
- Added more tests to the benchmark suite.
- Fixed an overflow bug in the client code when using overly long database names.
- mysql_connect() now aborts on Linux if the server doesn't answer in timeout seconds.
- SLAVE START did not work if you started with --skip-slave-start and had not explicitly run CHANGE MASTER TO .
- Fixed the output of SHOW MASTER STATUS to be consistent with SHOW SLAVE STATUS . (It now has no directory in the log name.)
- Added PURGE MASTER LOGS TO .
- Added SHOW MASTER LOGS .
- Added --safemalloc-mem-limit option to mysqld to simulate memory shortage when compiled with the --with-debug=full option.
- Fixed several core dumps in out-of-memory conditions.
- SHOW SLAVE STATUS was using an uninitialized mutex if the slave had not been started yet.
- Fixed bug in ELT() and MAKE_SET() when the query used a temporary table.
- CHANGE MASTER TO without specifying MASTER_LOG_POS would set it to 0 instead of 4 and hit the magic number in the master binlog.
- ALTER TABLE ... ORDER BY ... syntax added. This will create the new table with the rows in a specific order.

26.4.34 Changements de la version 3.23.27 (24 octobre 2000)

- Fixed a bug where the automatic repair of MyISAM tables sometimes failed when the datafile was corrupt.
- Fixed a bug in SHOW CREATE when using AUTO_INCREMENT columns.
- Changed BDB tables to use new compare function in Berkeley DB 3.2.3.
- You can now use Unix sockets with MIT-pthreads.
- Added the latin5 (turkish) character set.
- Small portability fixes.

26.4.35 Changements de la version 3.23.26 (18 octobre 2000)

- Renamed FLUSH MASTER and FLUSH SLAVE to RESET MASTER and RESET SLAVE .
- Fixed <> to work properly with NULL .
- Fixed a problem with SUBSTRING_INDEX() and REPLACE() . (Patch by Alexander Igonitchev)
- Fix CREATE TEMPORARY TABLE IF NOT EXISTS not to produce an error if the table exists.
- If you don't create a PRIMARY KEY in a BDB table, a hidden PRIMARY KEY will be created.
- Added read-only-key optimization to BDB tables.
- LEFT JOIN in some cases preferred a full table scan when there was no WHERE clause.
- When using --log-slow-queries , don't count the time waiting for a lock.
- Fixed bug in lock code on Windows which could cause the key cache to report that the key file was crashed even if it was okay.
- Automatic repair of MyISAM tables if you start mysqld with --myisam-recover .
- Removed the TYPE= keyword from CHECK and REPAIR . Allow CHECK options to be combined. (You can still use TYPE= , but this usage is deprecated.)
- Fixed mutex bug in the binary replication log --- long update queries could be read only in part by the slave if it did it at the wrong time, which was not fatal, but resulted in a performance-degrading reconnect and a scary message in the error log.
- Changed the format of the binary log --- added magic number, server version, binlog version. Added the server ID and query error code for each query event.
- Replication thread from the slave now will kill all the stale threads from the same server.
- Long replication user names were not being handled properly.
- Added --replicate-rewrite-db option to mysqld .
- Added --skip-slave-start option to mysqld .
- Updates that generated an error code (such as INSERT INTO foo(some_key) values (1),(1)) erroneously terminated the slave thread.
- Added optimization of queries where DISTINCT is only used on columns from some of the tables.
- Allow floating-point numbers where there is no sign after the exponent (like 1e1).
- SHOW GRANTS didn't always show all column grants.
- Added --default-extra-file=# option to all MySQL clients.
- Columns referenced in INSERT statements now are initialized properly.
- UPDATE didn't always work when used with a range on a timestamp that was part of the key that was used to find rows.
- Fixed a bug in FULLTEXT index when inserting a NULL column.
- Changed to use mkstemp() instead of tempnam() . Based on a patch from John Jones.

26.4.36 Changements de la version 3.23.25 (29 septembre 2000)

- Fixed that databasename works as second argument to mysqlhotcopy .
- The values for the UMASK and UMASK_DIR environment variables now can be specified in octal by beginning the value with a zero.
- Added RIGHT JOIN . This makes RIGHT a reserved word.
- Added @@IDENTITY as a synonym for LAST_INSERT_ID() . (This is for MSSQL compatibility.)
- Fixed a bug in myisamchk and REPAIR when using FULLTEXT index.
- LOAD DATA INFILE now works with FIFOs. (Patch by Toni L. Harbaugh-Blackford.)
- FLUSH LOGS broke replication if you specified a log name with an explicit extension as the value of the log-bin option.
- Fixed a bug in MyISAM with packed multi-part keys.
- Fixed crash when using CHECK TABLE on Windows.
- Fixed a bug where FULLTEXT index always used the koi8_ukr character set.
- Fixed privilege checking for CHECK TABLE .
- The MyISAM repair/reindex code didn't use the --tmpdir option for its temporary files.
- Added BACKUP TABLE and RESTORE TABLE .
- Fixed core dump on CHANGE MASTER TO when the slave did not have the master to start with.
- Fixed incorrect Time in the processlist for Connect of the slave thread.
- The slave now logs when it connects to the master.
- Fixed a core dump bug when doing FLUSH MASTER if you didn't specify a filename argument to --log-bin .
- Added missing ha_berkeley.x files to the MySQL Windows distribution.
- Fixed some mutex bugs in the log code that could cause thread blocks if new log files couldn't be created.
- Added lock time and number of selected processed rows to slow query log.
- Added --memlock option to mysqld to lock mysqld in memory on systems with the mlockall() call (as in Solaris).
- HEAP tables didn't use keys properly. (Bug from 3.23.23.)
- Added better support for MERGE tables (keys, mapping, creation, documentation...). Tables MERGE .
- Fixed bug in mysqldump from 3.23 which caused some CHAR columns not to be quoted.
- Merged analyze , check , optimize and repair code.
- OPTIMIZE TABLE is now mapped to REPAIR with statistics and sorting of the index tree. This means that for the moment it only works on MyISAM tables.
- Added a pre-alloced block to root_malloc to get fewer mallocs.
- Added a lot of new statistics variables.
- Fixed ORDER BY bug with BDB tables.
- Removed warning that mysqld couldn't remove the .pid file under Windows.
- Changed --log-isam to log MyISAM tables instead of isam tables.
- Fixed CHECK TABLE to work on Windows.
- Added file mutexes to make pwrite() safe on Windows.

26.4.37 Changements de la version 3.23.24 (08 septembre 2000)

- Added created_tmp_disk_tables variable to mysqld .
- To make it possible to reliably dump and restore tables with TIMESTAMP(X) columns, MySQL now reports columns with X other than 14 or 8 to be strings.

* Changed sort order for latin1 as it was before MySQL Version 3.23.23. Any table that was created or modified with 3.23.22 must be repaired if it has CHAR columns that may contain characters with ASCII values greater than 128!
* Fixed small memory leak introduced from 3.23.22 when creating a temporary table.
* Fixed problem with BDB tables and reading on a unique (not primary) key.
* Restored the win1251 character set (it's now only marked deprecated).

26.4.38 Changements de la version 3.23.23 (01 septembre 2000)

* Changed sort order for 'German'; all tables created with 'German' sortorder must be repaired with REPAIR TABLE or myisamchk before use!
* Added --core-file option to mysqld to get a core file on Linux if mysqld dies on the SIGSEGV signal.
* MySQL client mysql now starts with option --no-named-commands (-g) by default. This option can be disabled with --enable-named-commands (-G). This may cause incompatibility problems in some cases, for example, in SQL scripts that use named commands without a semicolon, etc.! Long format commands still work from the first line.
* Fixed a problem when using many pending DROP TABLE statements at the same time.
* Optimizer didn't use keys properly when using LEFT JOIN on an empty table.
* Added shorter help text when invoking mysqld with incorrect options.
* Fixed non-fatal free() bug in mysqlimport .
* Fixed bug in MyISAM index handling of DECIMAL / NUMERIC keys.
* Fixed a bug in concurrent insert in MyISAM tables. In some contexts, usage of MIN(key_part) or MAX(key_part) returned an empty set.
* Updated mysqlhotcopy to use the new FLUSH TABLES table_list syntax. Only tables which are being backed up are flushed now.
* Changed behavior of --enable-thread-safe-client so that both non-threaded (-lmysqlclient) and threaded (-lmysqlclient_r) libraries are built. Users who linked against a threaded -lmysqlclient will need to link against -lmysqlclient_r now.
* Added atomic RENAME TABLE command.
* Don't count NULL values in COUNT(DISTINCT ...) .
* Changed ALTER TABLE , LOAD DATA INFILE on empty tables and INSERT ... SELECT ... on empty tables to create non-unique indexes in a separate batch with sorting. This will make the above calls much faster when you have many indexes.
* ALTER TABLE now logs the first used insert_id correctly.
* Fixed crash when adding a default value to a BLOB column.
* Fixed a bug with DATE_ADD/DATE_SUB where it returned a datetime instead of a date.
* Fixed a problem with the thread cache which made some threads show up as ***DEAD*** in SHOW PROCESSLIST .
* Fixed a lock in our thr_rwlock code, which could make selects that run at the same time as concurrent inserts crash. This only affects systems that don't have the pthread_rwlock_rdlock code.
* When deleting rows with a non-unique key in a HEAP table, all rows weren't always deleted.
* Fixed bug in range optimizer for HEAP tables for searches on a part index.
* Fixed SELECT on part keys to work with BDB tables.
* Fixed INSERT INTO bdb_table ... SELECT to work with BDB tables.
* CHECK TABLE now updates key statistics for the table.
* ANALYZE TABLE will now only update tables that have been changed since the last ANALYZE . Note that this is a new feature and tables will not be marked to be analysed until they are updated in any way with 3.23.23 or newer. For older tables, you have to do CHECK TABLE to update the key distribution.
* Fixed some minor privilege problems with CHECK , ANALYZE , REPAIR and SHOW CREATE commands.
* Added CHANGE MASTER TO statement.
* Added FAST , QUICK EXTENDED check types to CHECK TABLES .
* Changed myisamchk so that --fast and --check-only-changed are also honored with --sort-index and --analyze .
* Fixed fatal bug in LOAD TABLE FROM MASTER that did not lock the table during index re-build.
* LOAD DATA INFILE broke replication if the database was excluded from replication.
* More variables in SHOW SLAVE STATUS and SHOW MASTER STATUS .
* SLAVE STOP now will not return until the slave thread actually exits.
* Full-text search via the MATCH() function and FULLTEXT index type (for MyISAM files). This makes FULLTEXT a reserved word.

26.4.39 Changements de la version 3.23.22 (31 juillet 2000)

* Fixed that lex_hash.h is created properly for each MySQL distribution.
* Fixed that MASTER and COLLECTION are not reserved words.
* The log generated by --slow-query-log didn't contain the whole queries.
* Fixed that open transactions in BDB tables are rolled back if the connection is closed unexpectedly.
* Added workaround for a bug in gcc 2.96 (intel) and gcc 2.9 (IA-64) in gen_lex_hash.c .
* Fixed memory leak in the client library when using host= in the my.cnf file.
* Optimized functions that manipulate the hours/minutes/seconds.
* Fixed bug when comparing the result of DATE_ADD() / DATE_SUB() against a number.
* Changed the meaning of -F, --fast for myisamchk . Added -C, --check-only-changed option to myisamchk .
* Added ANALYZE tbl_name to update key statistics for tables.
* Changed binary items 0x... to be regarded as integers by default.
* Fix for SCO and SHOW PROCESSLIST .
* Added auto-rehash on reconnect for the mysql client.
* Fixed a newly introduced bug in MyISAM , where the index file couldn't get bigger than 64M.
* Added SHOW MASTER STATUS and SHOW SLAVE STATUS .

26.4.40 Changements de la version 3.23.21

* Added mysql_character_set_name() function to the MySQL C API.
* Made the update log ASCII 0 safe.
* Added the mysql_config script.
* Fixed problem when using < or > with a char column that was only partly indexed.
* One would get a core dump if the log file was not readable by the MySQL user.
* Changed mysqladmin to use CREATE DATABASE and DROP DATABASE statements instead of the old deprecated API calls.
* Fixed chown warning in safe_mysqld .
* Fixed a bug in ORDER BY that was introduced in 3.23.19.
* Only optimize the DELETE FROM tbl_name to do a drop+create of the table if we are in AUTOCOMMIT mode (needed for BDB tables).
* Added extra checks to avoid index corruption when the ISAM / MyISAM index files get full during an INSERT / UPDATE .
* myisamchk didn't correctly update row checksum when used with -ro (this only gave a warning in subsequent runs).
* Fixed bug in REPAIR TABLE so that it works with tables without indexes.
* Fixed buffer overrun in DROP DATABASE .
* LOAD TABLE FROM MASTER is sufficiently bug-free to announce it as a feature.
* MATCH and AGAINST are now reserved words.

26.4.41 Changements de la version 3.23.20

- Fixed bug in 3.23.19; DELETE FROM tbl_name removed the .frm file.
- Added SHOW CREATE TABLE .

26.4.42 Changements de la version 3.23.19

- Changed copyright for all files to GPL for the server code and utilities and to LGPL for the client libraries. See http://www.fsf.org/licenses/ .
- Fixed bug where all rows matching weren't updated on a MyISAM table when doing update based on key on a table with many keys and some key changed values.
- The Linux MySQL RPMs and binaries are now statically linked with a linuxthread version that has faster mutex handling when used with MySQL.
- ORDER BY can now use REF keys to find subsets of the rows that need to be sorted.
- Changed name of print_defaults program to my_print_defaults to avoid name confusion.
- Fixed NULLIF() to work as required by standard SQL.
- Added net_read_timeout and net_write_timeout as startup parameters to mysqld .
- Fixed bug that destroyed index when doing myisamchk --sort-records on a table with prefix compressed index.
- Added pack_isam and myisampack to the standard MySQL distribution.
- Added the syntax BEGIN WORK (the same as BEGIN).
- Fixed core dump bug when using ORDER BY on a CONV() expression.
- Added LOAD TABLE FROM MASTER .
- Added FLUSH MASTER and FLUSH SLAVE .
- Fixed big/little endian problem in the replication.

26.4.43 Changements de la version 3.23.18

- Fixed a problem from 3.23.17 when choosing character set on the client side.
- Added FLUSH TABLES WITH READ LOCK to make a global lock suitable for making a copy of MySQL datafiles.
- CREATE TABLE ... SELECT ... PROCEDURE now works.
- Internal temporary tables will now use compressed index when using GROUP BY on VARCHAR/CHAR columns.
- Fixed a problem when locking the same table with both a READ and a WRITE lock.
- Fixed problem with myisamchk and RAID tables.

26.4.44 Changements de la version 3.23.17

- Fixed a bug in FIND_IN_SET() when the first argument was NULL .
- Added table locks to Berkeley DB.
- Fixed a bug with LEFT JOIN and ORDER BY where the first table had only one matching row.
- Added 4 sample my.cnf example files in the support-files directory.
- Fixed duplicated key problem when doing big GROUP BY operations. (This bug was probably introduced in 3.23.15.)
- Changed syntax for INNER JOIN to match SQL-99.
- Added NATURAL JOIN syntax.
- A lot of fixes in the BDB interface.
- Added handling of --no-defaults and --defaults-file to safe_mysqld.sh and mysql_install_db.sh .
- Fixed bug in reading compressed tables with many threads.
- Fixed that USE INDEX works with PRIMARY keys.
- Added BEGIN statement to start a transaction in AUTOCOMMIT mode.
- Added support for symbolic links for Windows.
- Changed protocol to let client know if the server is in AUTOCOMMIT mode and if there is a pending transaction. If there is a pending transaction, the client library will give an error before reconnecting to the server to let the client know that the server did a rollback. The protocol is still backward-compatible with old clients.
- KILL now works on a thread that is locked on a 'write' to a dead client.
- Fixed memory leak in the replication slave thread.
- Added new log-slave-updates option to mysqld , to allow daisy-chaining the slaves.
- Fixed compile error on FreeBSD and other systems where pthread_t is not the same as int .
- Fixed master shutdown aborting the slave thread.
- Fixed a race condition in INSERT DELAYED code when doing ALTER TABLE .
- Added deadlock detection sanity checks to INSERT DELAYED .

26.4.45 Changements de la version 3.23.16

- Added SLAVE START and SLAVE STOP statements.
- Added TYPE=QUICK option to CHECK and to REPAIR .
- Fixed bug in REPAIR TABLE when the table was in use by other threads.
- Added a thread cache to make it possible to debug MySQL with gdb when one does a lot of reconnects. This will also improve systems where you can't use persistent connections.
- Lots of fixes in the Berkeley DB interface.
- UPDATE IGNORE will not abort if an update results in a DUPLICATE_KEY error.
- Put CREATE TEMPORARY TABLE commands in the update log.
- Fixed bug in handling of masked IP numbers in the privilege tables.
- Fixed bug with delay_key_write tables and CHECK TABLE .
- Added replicate-do-db and replicate-ignore-db options to mysqld , to restrict which databases get replicated.
- Added SQL_LOG_BIN option.

26.4.46 Changements de la version 3.23.15 (May 2000: Beta)

- To start mysqld as root , you must now use the --user=root option.
- Added interface to Berkeley DB. (This is not yet functional; play with it at your own risk!)
- Replication between master and slaves.
- Fixed bug that other threads could steal a lock when a thread had a lock on a table and did a FLUSH TABLES command.
- Added the slow_launch_time variable and the Slow_launch_threads status variable to mysqld . These can be examined with mysqladmin variables and mysqladmin extended-status .
- Added functions INET_NTOA() and INET_ATON() .
- The default type of IF() now depends on the second and third arguments and not only on the second argument.
- Fixed case when myisamchk could go into a loop when trying to repair a crashed table.
- Don't write INSERT DELAYED to update log if SQL_LOG_UPDATE=0 .
- Fixed problem with REPLACE on HEAP tables.
- Added possible character sets and time zone to SHOW VARIABLES output.
- Fixed bug in locking code that could result in locking problems with concurrent inserts under high load.
- Fixed a problem with DELETE of many rows on a table with compressed keys where MySQL scanned the index to find the rows.
- Fixed problem with CHECK on table with deleted keyblocks.
- Fixed a bug in reconnect (at the client side) where it didn't free memory properly in some contexts.
- Fixed problems in update log when using LAST_INSERT_ID() to update a table with an AUTO_INCREMENT key.
- Added NULLIF() function.
- Fixed bug when using LOAD DATA INFILE on a table with BLOB/TEXT columns.
- Optimized MyISAM to be faster when inserting keys in sorted order.
- EXPLAIN SELECT ... now also prints out whether MySQL needs to create a temporary table or use file sorting when resolving the SELECT .
- Added optimization to skip ORDER BY parts where the part is a constant expression in the WHERE part. Indexes can now be used even if the ORDER BY doesn't match the index exactly, as long as all the unused index parts and all the extra ORDER BY columns are constants in the WHERE clause. Comment MySQL utilise les index .
- UPDATE and DELETE on a whole unique key in the WHERE part are now faster than before.
- Changed RAID_CHUNKSIZE to be in 1024-byte increments.
- Fixed core dump in LOAD_FILE(NULL) .

26.4.47 Changements de la version 3.23.14

- Added mysqlbinlog program for displaying binary log files in text format.
- Added mysql_real_escape_string() function to the MySQL C API.
- Fixed a bug in CONCAT() where one of the arguments was a function that returned a modified argument.
- Fixed a critical bug in myisamchk , where it updated the header in the index file when one only checked the table. This confused the mysqld daemon if it updated the same table at the same time. Now the status in the index file is only updated if one uses --update-state . With older myisamchk versions you should use --read-only when only checking tables, if there is the slightest chance that the mysqld server is working on the table at the same time!
- Fixed that DROP TABLE is logged in the update log.
- Fixed problem when searching on DECIMAL() key field where the column data contained leading zeros.
- Fix bug in myisamchk when the AUTO_INCREMENT column isn't the first key.
- Allow DATETIME in ISO8601 format: 2000-03-12T12:00:00
- Dynamic character sets. A mysqld binary can now handle many different character sets (you can choose which when starting mysqld).
- Added command REPAIR TABLE .
- Added mysql_thread_safe() function to the MySQL C API.
- Added the UMASK_DIR environment variable.
- Added CONNECTION_ID() function to return the client connection thread ID.
- When using = on BLOB or VARCHAR BINARY keys, where only a part of the column was indexed, the whole column of the result row wasn't compared.
- Fix for sjis character set and ORDER BY .
- When running in ANSI mode, don't allow columns to be used that aren't in the GROUP BY part.

26.4.48 Changements de la version 3.23.13

- Fixed problem when doing locks on the same table more than 2 times in the same LOCK TABLE command; this fixed the problem one got when running the test-ATIS test with --fast or --check-only-changed .
- Added SQL_BUFFER_RESULT option to SELECT .
- Removed end space from double/float numbers in results from temporary tables.
- Added CHECK TABLE command.
- Added changes for MyISAM in 3.23.12 that didn't get into the source distribution because of CVS problems.
- Fixed bug so that mysqladmin shutdown will wait for the local server to close down.
- Fixed a possible endless loop when calculating timestamp.
- Added print_defaults program to the .rpm files. Removed mysqlbug from the client .rpm file.

26.4.49 Changements de la version 3.23.12 (07 mars 2000)

- Fixed bug in MyISAM involving REPLACE ... SELECT ... which could give a corrupted table.
- Fixed bug in myisamchk where it incorrectly reset the AUTO_INCREMENT value.
- LOTS of patches for Linux Alpha. MySQL now appears to be relatively stable on Alpha.
- Changed DISTINCT on HEAP temporary tables to use hashed keys to quickly find duplicated rows. This mostly concerns queries of type SELECT DISTINCT ... GROUP BY ... . This fixes a problem where not all duplicates were removed in queries of the above type. In addition, the new code is MUCH faster.
- Added patches to make MySQL compile on Mac OS X.
- Added IF NOT EXISTS clause to CREATE DATABASE .
- Added --all-databases and --databases options to mysqldump to allow dumping of many databases at the same time.
- Fixed bug in compressed DECIMAL() index in MyISAM tables.
- Fixed bug when storing 0 into a timestamp.
- When doing mysqladmin shutdown on a local connection, mysqladmin now waits until the PID file is gone before terminating.
- Fixed core dump with some COUNT(DISTINCT ...) queries.
- Fixed that myisamchk works properly with RAID tables.
- Fixed problem with LEFT JOIN and key_field IS NULL .
- Fixed bug in net_clear() which could give the error Aborted connection in the MySQL clients.
- Added options USE INDEX (key_list) and IGNORE INDEX (key_list) as parameters in SELECT .
- DELETE and RENAME should now work on RAID tables.

26.4.50 Changements de la version 3.23.11

- Allow the ALTER TABLE tbl_name ADD (field_list) syntax.
- Fixed problem with optimizer that could sometimes use incorrect keys.
- Fixed that GRANT/REVOKE ALL PRIVILEGES doesn't affect GRANT OPTION .
- Removed extra ')' from the output of SHOW GRANTS .
- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Allow the syntax UNIQUE INDEX in CREATE statements.
- mysqlhotcopy − fast online hot−backup utility for local MySQL databases. By Tim Bunce.
- New more secure mysqlaccess . Thanks to Steve Harvey for this.
- Added --i-am-a-dummy and --safe-updates options to mysql .
- Added select_limit and max_join_size variables to mysql .
- Added SQL_MAX_JOIN_SIZE and SQL_SAFE_UPDATES options.
- Added READ LOCAL lock that doesn't lock the table for concurrent inserts. (This is used by mysqldump .)
- Changed that LOCK TABLES ... READ no longer allows concurrent inserts.
- Added --skip-delay-key-write option to mysqld .
- Fixed security problem in the protocol regarding password checking.
- _rowid can now be used as an alias for an integer type unique indexed column.
- Added back blocking of SIGPIPE when compiling with --thread-safe-clients to make things safe for old clients.

26.4.51 Changements de la version 3.23.10

- Fixed bug in 3.23.9 where memory wasn't properly freed when using LOCK TABLES .

26.4.52 Changements de la version 3.23.9

- Fixed problem that affected queries that did arithmetic on group functions.
- Fixed problem with timestamps and INSERT DELAYED .
- Fixed that date_col BETWEEN const_date AND const_date works.
- Fixed problem when only changing a 0 to NULL in a table with BLOB/TEXT columns.
- Fixed bug in range optimizer when using many key parts and or on the middle key parts: WHERE K1=1 and K3=2 and (K2=2 and K4=4 or K2=3 and K4=5)
- Added source command to mysql to allow reading of batch files inside the mysql client. Original patch by Matthew Vanecek.
- Fixed critical problem with the WITH GRANT OPTION option.
- Don't give an unnecessary GRANT error when using tables from many databases in the same query.
- Added VIO wrapper (needed for SSL support; by Andrei Errapart and Tõnu Samuel).
- Fixed optimizer problem on SELECT when using many overlapping indexes. MySQL should now be able to choose keys even better when there are many keys to choose from.
- Changed optimizer to prefer a range key instead of a ref key when the range key can uses more columns than the ref key (which only can use columns with =). For example, the following type of queries should now be faster: SELECT * from key_part_1=const and key_part_2 > const2
- Fixed bug that a change of all VARCHAR columns to CHAR columns didn't change row type from dynamic to fixed.
- Disabled floating−point exceptions for FreeBSD to fix core dump when doing SELECT FLOOR(POW(2,63)) .
- Renamed mysqld startup option from --delay-key-write to --delay-key-write-for-all-tables .
- Added read−next−on−key to HEAP tables. This should fix all problems with HEAP tables when using non− UNIQUE keys.
- Added option to print default arguments to all clients.
- Added --log-slow-queries option to mysqld to log all queries that take a long time to a separate log file with a time indicating how long the query took.
- Fixed core dump when doing WHERE key_col=RAND(...) .
- Fixed optimization bug in SELECT ... LEFT JOIN ... key_col IS NULL , when key_col could contain NULL values.
- Fixed problem with 8−bit characters as separators in LOAD DATA INFILE .

26.4.53 Changements de la version 3.23.8 (02 janvier 2000)

- Fixed problem when handling indexfiles larger than 8G.
- Added latest patches to MIT−pthreads for NetBSD.
- Fixed problem with timezones that are < GMT − 11.
- Fixed a bug when deleting packed keys in NISAM .
- Fixed problem with ISAM when doing some ORDER BY ... DESC queries.
- Fixed bug when doing a join on a text key which didn't cover the whole key.
- Option --delay-key-write didn't enable delayed key writing.
- Fixed update of TEXT column which involved only case changes.
- Fixed that INSERT DELAYED doesn't update timestamps that are given.
- Added function YEARWEEK() and options x , X , v and V to DATE_FORMAT() .
- Fixed problem with MAX(indexed_column) and HEAP tables.
- Fixed problem with BLOB NULL keys and LIKE "prefix%" .
- Fixed problem with MyISAM and fixed−length rows < 5 bytes.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated GROUP BY queries.
- Fixed core dump if you got a crashed table where an ENUM field value was too big.

26.4.54 Changements de la version 3.23.7 (10 décembre 1999)

- Fixed workaround under Linux to avoid problems with pthread_mutex_timedwait() , which is used with INSERT DELAYED . Notes relatives à Linux (toutes les versions de Linux) .
- Fixed that one will get a 'disk full' error message if one gets disk full when doing sorting (instead of waiting until we got more disk space).
- Fixed a bug in MyISAM with keys > 250 characters.
- In MyISAM one can now do an INSERT at the same time as other threads are reading from the table.
- Added max_write_lock_count variable to mysqld to force a READ lock after a certain number of WRITE locks.
- Inverted flag delay_key_write on show variables .
- Renamed concurrency variable to thread_concurrency .

- The following functions are now multi-byte-safe: LOCATE(substr,str) , POSITION(substr IN str) , LOCATE(substr,str,pos) , INSTR(str,substr) , LEFT(str,len) , RIGHT(str,len) , SUBSTRING(str,pos,len) , SUBSTRING(str FROM pos FOR len) , MID(str,pos,len) , SUBSTRING(str,pos) , SUBSTRING(str FROM pos) , SUBSTRING_INDEX(str,delim,count) , RTRIM(str) , TRIM([[BOTH | TRAILING] [remstr] FROM] str) , REPLACE(str,from_str,to_str) , REVERSE(str) , INSERT(str,pos,len,newstr) , LCASE(str) , LOWER(str) , UCASE(str) and UPPER(str) ; patch by Wei He.
- Fix core dump when releasing a lock from a non-existent table.
- Remove locks on tables before starting to remove duplicates.
- Added option FULL to SHOW PROCESSLIST .
- Added option --verbose to mysqladmin .
- Fixed problem when automatically converting HEAP to MyISAM .
- Fixed bug in HEAP tables when doing insert + delete + insert + scan the table.
- Fixed bugs on Alpha with REPLACE() and LOAD DATA INFILE .
- Added interactive_timeout variable to mysqld .
- Changed the argument to mysql_data_seek() from ulong to ulonglong .

26.4.55 Changements de la version 3.23.6

- Added -O lower_case_table_names={0|1} option to mysqld to allow users to force table names to lowercase.
- Added SELECT ... INTO DUMPFILE .
- Added --ansi option to mysqld to make some functions SQL-99 compatible.
- Temporary table names now start with #sql .
- Added quoting of identifiers with ` (" in --ansi mode).
- Changed to use snprintf() when printing floats to avoid some buffer overflows on FreeBSD.
- Made FLOOR() overflow safe on FreeBSD.
- Added --quote-names option to mysqldump .
- Fixed bug that one could make a part of a PRIMARY KEY NOT NULL .
- Fixed encrypt() to be thread-safe and not reuse buffer.
- Added mysql_odbc_escape_string() function to support big5 characters in MyODBC.
- Rewrote the storage engine to use classes. This introduces a lot of new code, but will make table handling faster and better.
- Added patch by Sasha for user-defined variables.
- Changed that FLOAT and DOUBLE (without any length modifiers) no longer are fixed decimal point numbers.
- Changed the meaning of FLOAT(X) : Now this is the same as FLOAT if X <= 24 and a DOUBLE if 24 < X <= 53.
- DECIMAL(X) is now an alias for DECIMAL(X,0) and DECIMAL is now an alias for DECIMAL(10,0) . The same goes for NUMERIC .
- Added option ROW_FORMAT={DEFAULT | DYNAMIC | FIXED | COMPRESSED} to CREATE_TABLE .
- DELETE FROM table_name didn't work on temporary tables.
- Changed function CHAR_LENGTH() to be multi-byte character safe.
- Added function ORD(string) .

26.4.56 Changements de la version 3.23.5 (20 octobre 1999)

- Fixed some Y2K problems in the new date handling in 3.23.
- Fixed problem with SELECT DISTINCT ... ORDER BY RAND() .
- Added patches by Sergei A. Golubchik for text searching on the MyISAM level.
- Fixed cache overflow problem when using full joins without keys.
- Fixed some configure issues.
- Some small changes to make parsing faster.
- Adding a column after the last field with ALTER TABLE didn't work.
- Fixed problem when using an AUTO_INCREMENT column in two keys
- With MyISAM , you now can have an AUTO_INCREMENT column as a key sub part: CREATE TABLE foo (a INT NOT NULL AUTO_INCREMENT, b CHAR(5), PRIMARY KEY (b,a))
- Fixed bug in MyISAM with packed char keys that could be NULL .
- AS on field name with CREATE TABLE table_name SELECT ... didn't work.
- Allow use of NATIONAL and NCHAR when defining character columns. This is the same as not using BINARY .
- Don't allow NULL columns in a PRIMARY KEY (only in UNIQUE keys).
- Clear LAST_INSERT_ID() if one uses this in ODBC: WHERE auto_increment_column IS NULL . This seems to fix some problems with Access.
- SET SQL_AUTO_IS_NULL=0|1 now turns on/off the handling of searching after the last inserted row with WHERE auto_increment_column IS NULL .
- Added new variable concurrency to mysqld for Solaris.
- Added --relative option to mysqladmin to make extended-status more useful to monitor changes.
- Fixed bug when using COUNT(DISTINCT ...) on an empty table.
- Added support for the Chinese character set GBK.
- Fixed problem with LOAD DATA INFILE and BLOB columns.
- Added bit operator ~ (negation).
- Fixed problem with UDF functions.

26.4.57 Changements de la version 3.23.4 (28 septembre 1999)

- Inserting a DATETIME into a TIME column no longer will try to store 'days' in it.
- Fixed problem with storage of float/double on little endian machines. (This affected SUM() .)
- Added connect timeout on TCP/IP connections.
- Fixed problem with LIKE "%" on an index that may have NULL values.
- REVOKE ALL PRIVILEGES didn't revoke all privileges.
- Allow creation of temporary tables with same name as the original table.
- When granting a user a GRANT option for a database, he couldn't grant privileges to other users.
- New command: SHOW GRANTS FOR user (by Sinisa).
- New date_add syntax: date/datetime + INTERVAL # interval_type . By Joshua Chamas.
- Fixed privilege check for LOAD DATA REPLACE .
- Automatic fixing of broken include files on Solaris 2.7
- Some configure issues to fix problems with big filesystem detection.
- REGEXP is now case-insensitive if you use non-binary strings.

26.4.58 Changements de la version 3.23.3

- Added patches for MIT-pthreads on NetBSD.
- Fixed range bug in MyISAM .
- ASC is now the default again for ORDER BY .
- Added LIMIT to UPDATE .
- Added mysql_change_user() function to the MySQL C API.
- Added character set to SHOW VARIABLES .
- Added support of --[whitespace] comments.
- Allow INSERT into tbl_name VALUES () , that is, you may now specify an empty value list to insert a row in which each column is set to its default value.
- Changed SUBSTRING(text FROM pos) to conform to SQL-99. (Before this construct returned the rightmost pos characters.)
- SUM() with GROUP BY returned 0 on some systems.
- Changed output for SHOW TABLE STATUS .
- Added DELAY_KEY_WRITE option to CREATE TABLE .
- Allow AUTO_INCREMENT on any key part.
- Fixed problem with YEAR(NOW()) and YEAR(CURDATE()) .
- Added CASE construct.
- New function COALESCE() .

26.4.59 Changements de la version 3.23.2 (09 août 1999)

- Fixed range optimizer bug: SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const) . The bug was that some rows could be duplicated in the result.
- Running myisamchk without -a updated the index distribution incorrectly.
- SET SQL_LOW_PRIORITY_UPDATES=1 was causing a parse error.
- You can now update index columns that are used in the WHERE clause. UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100
- Date handling should now be a bit faster.
- Added handling of fuzzy dates (dates where day or month is 0), such as '1999-01-00' .
- Fixed optimization of SELECT ... WHERE key_part1=const1 AND key_part_2=const2 AND key_part1=const4 AND key_part2=const4 ; indextype should be range instead of ref .
- Fixed egcs 1.1.2 optimizer bug (when using BLOB values) on Linux Alpha.
- Fixed problem with LOCK TABLES combined with DELETE FROM table .
- MyISAM tables now allow keys on NULL and BLOB/TEXT columns.
- The following join is now much faster: SELECT ... FROM t1 LEFT JOIN t2 ON ... WHERE t2.not_null_column IS NULL .
- ORDER BY and GROUP BY can be done on functions.
- Changed handling of 'const_item' to allow handling of ORDER BY RAND() .
- Indexes are now used for WHERE key_column = function .
- Indexes are now used for WHERE key_column = col_name even if the columns are not identically packed.
- Indexes are now used for WHERE col_name IS NULL .
- Changed heap tables to be stored in low_byte_first order (to make it easy to convert to MyISAM tables)
- Automatic change of HEAP temporary tables to MyISAM tables in case of ``table is full'' errors.
- Added --init-file=file_name option to mysqld .
- Added COUNT(DISTINCT value, [value, ...]) .
- CREATE TEMPORARY TABLE now creates a temporary table, in its own namespace, that is automatically deleted if connection is dropped.
- New reserved words (required for CASE): CASE, THEN, WHEN, ELSE and END .
- New functions EXPORT_SET() and MD5() .
- Support for the GB2312 Chinese character set.

26.4.60 Changements de la version 3.23.1

- Fixed some compilation problems.

26.4.61 Changements de la version 3.23.0 (05 Aug 1999: Alpha)

- A new storage engine library (MyISAM) with a lot of new features. Tables MyISAM .
- You can create in-memory HEAP tables which are extremely fast for lookups.
- Support for big files (63-bit) on OSs that support big files.
- New function LOAD_FILE(filename) to get the contents of a file as a string value.
- New <=> operator that acts as = but returns TRUE if both arguments are NULL . This is useful for comparing changes between tables.
- Added the ODBC 3.0 EXTRACT(interval FROM datetime) function.
- Columns defined as FLOAT(X) are not rounded on storage and may be in scientific notation (1.0 E+10) when retrieved.
- REPLACE is now faster than before.
- Changed LIKE character comparison to behave as = ; This means that 'e' LIKE ''e' is now true. (If the line doesn't display correctly, the latter 'e' is a French 'e' with an acute accent above.)
- SHOW TABLE STATUS returns a lot of information about the tables.
- Added LIKE to the SHOW STATUS command.
- Added Privileges column to SHOW COLUMNS .
- Added Packed and Comment columns to SHOW INDEX .
- Added comments to tables (with CREATE TABLE ... COMMENT "xxx").
- Added UNIQUE , as in CREATE TABLE tbl_name (col INT not null UNIQUE)
- New create syntax: CREATE TABLE tbl_name SELECT ...
- New create syntax: CREATE TABLE IF NOT EXISTS ...
- Allow creation of CHAR(0) columns.
- DATE_FORMAT() now requires '%' before any format character.
- DELAYED is now a reserved word (sorry about that :( ).
- An example procedure is added: analyse , file: sql_analyse.c . This will describe the data in your query. Try the following:

  ```
   SELECT ... FROM ...
  WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
  ```

  This procedure is extremely useful when you want to check the data in your table!
- BINARY cast to force a string to be compared in case-sensitive fashion.
- Added --skip-show-database option to mysqld .
- Check whether a row has changed in an UPDATE now also works with BLOB / TEXT columns.
- Added the INNER join syntax. **Note:** This made INNER a reserved word!

* Added support for netmasks to the hostname in the MySQL grant tables. You can specify a netmask using the IP/NETMASK syntax.
* If you compare a NOT NULL DATE/DATETIME column with IS NULL , this is changed to a compare against 0 to satisfy some ODBC applications. (By shreeve@uci.edu .)
* NULL IN (...) now returns NULL instead of 0 . This will ensure that null_column NOT IN (...) doesn't match NULL values.
* Fix storage of floating-point values in TIME columns.
* Changed parsing of TIME strings to be more strict. Now the fractional second part is detected (and currently skipped). The following formats are supported:
    * [[DAYS] [H]H:]MM:]SS[.fraction]
    * [[[[[H]H]H]H]MM]SS[.fraction]
* Detect (and ignore) fractional second part from DATETIME .
* Added the LOW_PRIORITY attribute to LOAD DATA INFILE .
* The default index name now uses the same case as the column name on which the index name is based.
* Changed default number of connections to 100.
* Use bigger buffers when using LOAD DATA INFILE .
* DECIMAL(x,y) now works according to standard SQL.
* Added aggregate UDF functions. Thanks to Andreas F. Bobak (bobak@relog.ch) for this!
* LAST_INSERT_ID() is now updated for INSERT INTO ... SELECT .
* Some small changes to the join table optimizer to make some joins faster.
* SELECT DISTINCT is much faster; it uses the new UNIQUE functionality in MyISAM . One difference compared to MySQL 3.22 is that the output of DISTINCT is no longer sorted.
* All C client API macros are now functions to make shared libraries more reliable. Because of this, you can no longer call mysql_num_fields() on a MYSQL object, you must use mysql_field_count() instead.
* Added use of LIBWRAP ; patch by Henning P. Schmiedehausen.
* Don't allow AUTO_INCREMENT for other than numerical columns.
* Using AUTO_INCREMENT will now automatically make the column NOT NULL .
* Show NULL as the default value for AUTO_INCREMENT columns.
* Added SQL_BIG_RESULT ; SQL_SMALL_RESULT is now default.
* Added a shared library RPM. This enhancement was contributed by David Fox (dsfox@cogsci.ucsd.edu).
* Added --enable-large-files and --disable-large-files options to configure . See configure.in for some systems where this is automatically turned off because of broken implementations.
* Upgraded readline to 4.0.
* New CREATE TABLE options: PACK_KEYS and CHECKSUM .
* Added --default-table-type option to mysqld .

26.5 Changements de la version 3.22.x (Old; discontinued)

The 3.22 version has faster and safer connect code than version 3.21, as well as a lot of new nice enhancements. As there aren't really any major changes, upgrading from 3.21 to 3.22 should be very easy and painless. Mise à jour de Version 3.21 vers 3.22 .

26.5.1 Changements de la version 3.22.35

* Fixed problem with STD() .
* Merged changes from the newest ISAM library from 3.23.
* Fixed problem with INSERT DELAYED .
* Fixed a bug core dump when using a LEFT JOIN / STRAIGHT_JOIN on a table with only one row.

26.5.2 Changements de la version 3.22.34

* Fixed problem with GROUP BY on TINYBLOB columns; this caused bugzilla to not show rows in some queries.
* Had to do total recompile of the Windows binary version as VC++ didn't compile all relevant files for 3.22.33 :(

26.5.3 Changements de la version 3.22.33

* Fixed problems in Windows when locking tables with LOCK TABLE .
* Quicker kill of SELECT DISTINCT queries.

26.5.4 Changements de la version 3.22.32 (14 février 2000)

* Fixed problem when storing numbers in timestamps.
* Fix problem with timezones that have half hour offsets.
* Added mysqlhotcopy , a fast online hot-backup utility for local MySQL databases. By Tim Bunce.
* New more secure mysqlaccess . Thanks to Steve Harvey for this.
* Fixed security problem in the protocol regarding password checking.
* Fixed problem that affected queries that did arithmetic on GROUP functions.
* Fixed a bug in the ISAM code when deleting rows on tables with packed indexes.

26.5.5 Changements de la version 3.22.31

* A few small fixes for the Windows version.

26.5.6 Changements de la version 3.22.30

- Fixed optimizer problem on SELECT when using many overlapping indexes.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing SELECT FLOOR(POW(2,63)) .
- Added print of default arguments options to all clients.
- Fixed critical problem with the WITH GRANT OPTION option.
- Fixed non-critical Y2K problem when writing short date to log files.

26.5.7 Changements de la version 3.22.29 (02 janvier 2000)

- Upgraded the configure and include files to match the latest 3.23 version. This should increase portability and make it easier to build shared libraries.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT -11.
- Fixed a bug when deleting packed keys in NISAM.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated GROUP BY queries.
- Fixed core dump if you got a crashed table where an ENUM field value was too big.
- Added mysqlshutdown.exe and mysqlwatch.exe to the Windows distribution.
- Fixed problem when doing ORDER BY on a reference key.
- Fixed that INSERT DELAYED doesn't update timestamps that are given.

26.5.8 Changements de la version 3.22.28 (20 octobre 1999)

- Fixed problem with LEFT JOIN and COUNT() on a column which was declared NULL + and it had a DEFAULT value.
- Fixed core dump problem when using CONCAT() in a WHERE clause.
- Fixed problem with AVG() and STD() with NULL values.

26.5.9 Changements de la version 3.22.27

- Fixed prototype in my_ctype.h when using other character sets.
- Some configure issues to fix problems with big filesystem detection.
- Fixed problem when sorting on big BLOB columns.
- ROUND() will now work on Windows.

26.5.10 Changements de la version 3.22.26 (16 septembre 1999)

- Fixed core dump with empty BLOB/TEXT column argument to REVERSE() .
- Extended /*! */ with version numbers.
- Changed SUBSTRING(text FROM pos) to conform to SQL-99. (Before this construct returned the rightmost 'pos' characters.)
- Fixed problem with LOCK TABLES combined with DELETE FROM table
- Fixed problem that INSERT ... SELECT didn't use BIG_TABLES .
- SET SQL_LOW_PRIORITY_UPDATES=# didn't work.
- Password wasn't updated correctly if privileges didn't change on: GRANT ... IDENTIFIED BY
- Fixed range optimizer bug in SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const) .
- Fixed bug in compression key handling in ISAM .

26.5.11 Changements de la version 3.22.25

- Fixed some small problems with the installation.

26.5.12 Changements de la version 3.22.24 (05 juillet 1999)

- DATA is no longer a reserved word.
- Fixed optimizer bug with tables with only one row.
- Fixed bug when using LOCK TABLES table_name READ; FLUSH TABLES;
- Applied some patches for HP-UX.
- isamchk should now work on Windows.
- Changed configure to not use big file handling on Linux as this crashes some Red Hat 6.0 systems

26.5.13 Changements de la version 3.22.23 (08 juin 1999)

- Upgraded to use Autoconf 2.13, Automake 1.4 and libtool 1.3.2.
- Better support for SCO in configure .
- Added option --defaults-file=file_name to option file handling to force use of only one specific option file.
- Extended CREATE syntax to ignore MySQL Version 3.23 keywords.
- Fixed deadlock problem when using INSERT DELAYED on a table locked with LOCK TABLES .
- Fixed deadlock problem when using DROP TABLE on a table that was locked by another thread.
- Add logging of GRANT/REVOKE commands in the update log.
- Fixed isamchk to detect a new error condition.
- Fixed bug in NATURAL LEFT JOIN .

26.5.14 Changements de la version 3.22.22 (30 avril 1999)

- Fixed problem in the C API when you called mysql_close() directly after mysql_init() .
- Better client error message when you can't open socket.
- Fixed delayed_insert_thread counting when you couldn't create a new delayed_insert thread.
- Fixed bug in CONCAT() with many arguments.
- Added patches for DEC 3.2 and SCO.
- Fixed path−bug when installing MySQL as a service on NT.
- MySQL on Windows is now compiled with VC++ 6.0 instead of with VC++ 5.0.
- New installation setup for MySQL on Windows.

26.5.15 Changements de la version 3.22.21

- Fixed problem with DELETE FROM TABLE when table was locked by another thread.
- Fixed bug in LEFT JOIN involving empty tables.
- Changed the mysql.db column from CHAR(32) to CHAR(60) .
- MODIFY and DELAYED are no longer reserved words.
- Fixed a bug when storing days in a TIME column.
- Fixed a problem with Host '...' is not allowed to connect to this MySQL server after one had inserted a new MySQL user with a GRANT command.
- Changed to use TCP_NODELAY also on Linux (should give faster TCP/IP connections).

26.5.16 Changements de la version 3.22.20 (18 mars 1999)

- Fixed STD() for big tables when result should be 0.
- The update log didn't have newlines on some operating systems.
- INSERT DELAYED had some garbage at end in the update log.

26.5.17 Changements de la version 3.22.19 (Mar 1999: Production)

- Fixed bug in mysql_install_db (from 3.22.17).
- Changed default key cache size to 8M.
- Fixed problem with queries that needed temporary tables with BLOB columns.

26.5.18 Changements de la version 3.22.18

- Fixes a fatal problem in 3.22.17 on Linux; after shutdown not all threads died properly.
- Added option −O flush_time=# to mysqld . This is mostly useful on Windows and tells how often MySQL should close all unused tables and flush all updated tables to disk.
- Fixed problem that a VARCHAR column compared with CHAR column didn't use keys efficiently.

26.5.19 Changements de la version 3.22.17

- Fixed a core dump problem when using −−log−update and connecting without a default database.
- Fixed some configure and portability problems.
- Using LEFT JOIN on tables that had circular dependencies caused mysqld to hang forever.

26.5.20 Changements de la version 3.22.16 (Feb 1999: Gamma)

- mysqladmin processlist could kill the server if a new user logged in.
- DELETE FROM tbl_name WHERE key_column=col_name didn't find any matching rows. Fixed.
- DATE_ADD(column, ...) didn't work.
- INSERT DELAYED could deadlock with status upgrading lock .
- Extended ENCRYPT() to take longer salt strings than 2 characters.
- longlong2str is now much faster than before. For Intel x86 platforms, this function is written in optimized assembler.
- Added the MODIFY keyword to ALTER TABLE .

26.5.21 Changements de la version 3.22.15

- GRANT used with IDENTIFIED BY didn't take effect until privileges were flushed.
- Name change of some variables in SHOW STATUS .
- Fixed problem with ORDER BY with 'only index' optimization when there were multiple key definitions for a used column.
- DATE and DATETIME columns are now up to 5 times faster than before.
- INSERT DELAYED can be used to let the client do other things while the server inserts rows into a table.
- LEFT JOIN USING (col1,col2) didn't work if one used it with tables from 2 different databases.
- LOAD DATA LOCAL INFILE didn't work in the Unix version because of a missing file.
- Fixed problems with VARCHAR / BLOB on very short rows (< 4 bytes); error 127 could occur when deleting rows.
- Updating BLOB/TEXT through formulas didn't work for short (< 256 char) strings.
- When you did a GRANT on a new host, mysqld could die on the first connect from this host.
- Fixed bug when one used ORDER BY on column name that was the same name as an alias.
- Added BENCHMARK(loop_count,expression) function to time expressions.

26.5.22 Changements de la version 3.22.14

- Allow empty arguments to mysqld to make it easier to start from shell scripts.
- Setting a TIMESTAMP column to NULL didn't record the timestamp value in the update log.
- Fixed lock handler bug when one did INSERT INTO TABLE ... SELECT ... GROUP BY .
- Added a patch for localtime_r() on Windows so that it will no lonher crash if your date is > 2039, but instead will return a time of all zero.
- Names for user−defined functions are no longer case−sensitive.
- Added escape of ^Z (ASCII 26) to \Z as ^Z doesn't work with pipes on Windows.
- mysql_fix_privileges adds a new column to the mysql.func to support aggregate UDF functions in future MySQL releases.

26.5.23 Changements de la version 3.22.13

- Saving NOW() , CURDATE() or CURTIME() directly in a column didn't work.
- SELECT COUNT(*) ... LEFT JOIN ... didn't work with no WHERE part.
- Updated config.guess to allow MySQL to configure on UnixWare 7.1.x.
- Changed the implementation of pthread_cond() on the Windows version. get_lock() now correctly times out on Windows!

26.5.24 Changements de la version 3.22.12

- Fixed problem when using DATE_ADD() and DATE_SUB() in a WHERE clause.
- You can now set the password for a user with the GRANT ... TO user IDENTIFIED BY 'password' syntax.
- Fixed bug in GRANT checking with SELECT on many tables.
- Added missing file mysql_fix_privilege_tables to the RPM distribution. This is not run by default because it relies on the client package.
- Added option SQL_SMALL_RESULT to SELECT to force use of fast temporary tables when you know that the result set will be small.
- Allow use of negative real numbers without a decimal point.
- Day number is now adjusted to maximum days in month if the resulting month after DATE_ADD / DATE_SUB() doesn't have enough days.
- Fix that GRANT compares columns in case−insensitive fashion.
- Fixed a bug in sql_list.h that made ALTER TABLE dump core in some contexts.
- The hostname in user@hostname can now include '.' and '−' without quotes in the context of the GRANT , REVOKE and SET PASSWORD FOR ... statements.
- Fix for isamchk for tables which need big temporary files.

26.5.25 Changements de la version 3.22.11

- **Important** : You must run the mysql_fix_privilege_tables script when you upgrade to this version! This is needed because of the new GRANT system. If you don't do this, you will get Access denied when you try to use ALTER TABLE , CREATE INDEX , or DROP INDEX .
- GRANT to allow/deny users table and column access.
- Changed USER() to return a value in user@host format. Formerly it returned only user .
- Changed the syntax for how to set PASSWORD for another user.
- New command FLUSH STATUS that resets most status variables to zero.
- New status variables: aborted_threads , aborted_connects .
- New option variable: connection_timeout .
- Added support for Thai sorting (by Pruet Boonma pruet@ds90.intanon.nectec.or.th).
- Slovak and Japanese error messages.
- Configuration and portability fixes.
- Added option SET SQL_WARNINGS=1 to get a warning count also for simple (single−row) inserts.
- MySQL now uses SIGTERM instead of SIGQUIT with shutdown to work better on FreeBSD.
- Added option \G (print vertically) to mysql .
- SELECT HIGH_PRIORITY ... killed mysqld .
- IS NULL on a AUTO_INCREMENT column in a LEFT JOIN didn't work as expected.
- New function MAKE_SET() .

26.5.26 Changements de la version 3.22.10

- mysql_install_db no longer starts the MySQL server! You should start mysqld with safe_mysqld after installing it! The MySQL RPM will, however, start the server as before.
- Added −−bootstrap option to mysqld and recoded mysql_install_db to use it. This will make it easier to install MySQL with RPM s.
- Changed + , − (sign and minus), * , / , % , ABS() and MOD() to be BIGINT aware (64−bit safe).
- Fixed a bug in ALTER TABLE that caused mysqld to crash.
- MySQL now always reports the conflicting key values when a duplicate key entry occurs. (Before this was only reported for INSERT .)
- New syntax: INSERT INTO tbl_name SET col_name=value, col_name=value, ...
- Most errors in the .err log are now prefixed with a time stamp.
- Added option MYSQL_INIT_COMMAND to mysql_options() to make a query on connect or reconnect.
- Added option MYSQL_READ_DEFAULT_FILE and MYSQL_READ_DEFAULT_GROUP to mysql_options() to read the following parameters from the MySQL option files: port , socket , compress , password , pipe , timeout , user , init−command , host and database .
- Added maybe_null to the UDF structure.
- Added option IGNORE to INSERT statements with many rows.
- Fixed some problems with sorting of the koi8 character sets; users of koi8 **must** run isamchk −rq on each table that has an index on a CHAR or VARCHAR column.
- New script mysql_setpermission , by Luuk de Boer. It allows easy creation of new users with permissions for specific databases.
- Allow use of hexadecimal strings (0x...) when specifying a constant string (like in the column separators with LOAD DATA INFILE).
- Ported to OS/2 (thanks to Antony T. Curtis antony.curtis@olcs.net).
- Added more variables to SHOW STATUS and changed format of output to be like SHOW VARIABLES .
- Added extended−status command to mysqladmin which will show the new status variables.

26.5.27 Changements de la version 3.22.9

- SET SQL_LOG_UPDATE=0 caused a lockup of the server.
- New SQL command: FLUSH [TABLES | HOSTS | LOGS | PRIVILEGES] [, ...]
- New SQL command: KILL thread_id .
- Added casts and changed include files to make MySQL easier to compile on AIX and DEC OSF/1 4.x
- Fixed conversion problem when using ALTER TABLE from a INT to a short CHAR() column.
- Added SELECT HIGH_PRIORITY ; this will get a lock for the SELECT even if there is a thread waiting for another SELECT to get a WRITE LOCK .
- Moved wild_compare() to string class to be able to use LIKE on BLOB / TEXT columns with \0 .
- Added ESCAPE option to LIKE .
- Added a lot more output to mysqladmin debug .
- You can now start mysqld on Windows with the --flush option. This will flush all tables to disk after each update. This makes things much safer on the Windows platforms but also **much** slower.

26.5.28 Changements de la version 3.22.8

- Czech character sets should now work much better.
- DATE_ADD() and DATE_SUB() didn't work with group functions.
- mysql will now also try to reconnect on USE database commands.
- Fix problem with ORDER BY and LEFT JOIN and const tables.
- Fixed problem with ORDER BY if the first ORDER BY column was a key and the rest of the ORDER BY columns wasn't part of the key.
- Fixed a big problem with OPTIMIZE TABLE .
- MySQL clients on NT will now by default first try to connect with named pipes and after this with TCP/IP.
- Fixed a problem with DROP TABLE and mysqladmin shutdown on Windows (a fatal bug from 3.22.6).
- Fixed problems with TIME columns and negative strings.
- Added an extra thread signal loop on shutdown to avoid some error messages from the client.
- MySQL now uses the next available number as extension for the update log file.
- Added patches for UNIXWARE 7.

26.5.29 Changements de la version 3.22.7 (Sep 1998: Beta)

- Added LIMIT clause for the DELETE statement.
- You can now use the /*! ... */ syntax to hide MySQL-specific keywords when you write portable code. MySQL will parse the code inside the comments as if the surrounding /*! and */ comment characters didn't exist.
- OPTIMIZE TABLE tbl_name can now be used to reclaim disk space after many deletes. Currently, this uses ALTER TABLE to regenerate the table, but in the future it will use an integrated isamchk for more speed.
- Upgraded libtool to get the configure more portable.
- Fixed slow UPDATE and DELETE operations when using DATETIME or DATE keys.
- Changed optimizer to make it better at deciding when to do a full join and when using keys.
- You can now use mysqladmin proc to display information about your own threads. Only users with the PROCESS privilege can get information about all threads. (In 4.0.2 one needs the SUPER privilege for this.)
- Added handling of formats YYMMDD , YYYYMMDD , YYMMDDHHMMSS for numbers when using DATETIME and TIMESTAMP types. (Formerly these formats only worked with strings.)
- Added connect option CLIENT_IGNORE_SPACE to allow use of spaces after function names and before '(' (Powerbuilder requires this). This will make all function names reserved words.
- Added the --log-long-format option to mysqld to enable timestamps and INSERT_IDs in the update log.
- Added --where option to mysqldump (patch by Jim Faucette).
- The lexical analyzer now uses ``perfect hashing'' for faster parsing of SQL statements.

26.5.30 Changements de la version 3.22.6

- Faster mysqldump .
- For the LOAD DATA INFILE statement, you can now use the new LOCAL keyword to read the file from the client. mysqlimport will automatically use LOCAL when importing with the TCP/IP protocol.
- Fixed small optimize problem when updating keys.
- Changed makefiles to support shared libraries.
- MySQL-NT can now use named pipes, which means that you can now use MySQL-NT without having to install TCP/IP.

26.5.31 Changements de la version 3.22.5

- All table lock handing is changed to avoid some very subtle deadlocks when using DROP TABLE , ALTER TABLE , DELETE FROM TABLE and mysqladmin flush-tables under heavy usage. Changed locking code to get better handling of locks of different types.
- Updated DBI to 1.00 and DBD to 1.2.0.
- Added a check that the error message file contains error messages suitable for the current version of mysqld . (To avoid errors if you accidentally try to use an old error message file.)
- All count structures in the client (affected_rows() , insert_id() , ...) are now of type BIGINT to allow 64-bit values to be used. This required a minor change in the MySQL protocol which should affect only old clients when using tables with AUTO_INCREMENT values > 16M.
- The return type of mysql_fetch_lengths() has changed from uint * to ulong * . This may give a warning for old clients but should work on most machines.
- Change mysys and dbug libraries to allocate all thread variables in one struct. This makes it easier to make a threaded libmysql.dll library.
- Use the result from gethostname() (instead of uname()) when constructing .pid file names.
- New better compressed server/client protocol.
- COUNT() , STD() and AVG() are extended to handle more than 4G rows.
- You can now store values in the range -838:59:59 <= x <= 838:59:59 in a TIME column.
- **Warning: incompatible change!!** If you set a TIME column to too short a value, MySQL now assumes the value is given as: [[[D]HH:]MM:]SS instead of HH[:MM[:SS]] .
- TIME_TO_SEC() and SEC_TO_TIME() can now handle negative times and hours up to 32767.
- Added new option SET SQL_LOG_UPDATE={0|1} to allow users with the PROCESS privilege to bypass the update log. (Modified patch from Sergey A Mukhin violet@rosnet.net .)

- Fixed fatal bug in LPAD() .
- Initialize line buffer in mysql.cc to make BLOB reading from pipes safer.
- Added -O max_connect_errors=# option to mysqld . Connect errors are now reset for each correct connection.
- Increased the default value of max_allowed_packet to 1M in mysqld .
- Added --low-priority-updates option to mysqld , to give table-modifying operations (INSERT , REPLACE , UPDATE , DELETE) lower priority than retrievals. You can now use {INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ... You can also use SET SQL_LOW_PRIORITY_UPDATES={0|1} to change the priority for one thread. One side effect is that LOW_PRIORITY is now a reserved word. :(
- Add support for INSERT INTO table ... VALUES(...),(...),(...) , to allow inserting multiple rows with a single statement.
- INSERT INTO tbl_name is now also cached when used with LOCK TABLES . (Previously only INSERT ... SELECT and LOAD DATA INFILE were cached.)
- Allow GROUP BY functions with HAVING :

```
mysql> SELECT col FROM table GROUP BY col HAVING COUNT(*)>0;
```

- mysqld will now ignore trailing ';' characters in queries. This is to make it easier to migrate from some other SQL servers that require the trailing ';' .
- Fix for corrupted fixed-format output generated by SELECT INTO OUTFILE .
- **Warning: incompatible change!** Added Oracle GREATEST() and LEAST() functions. You must now use these instead of the MAX() and MIN() functions to get the largest/smallest value from a list of values. These can now handle REAL , BIGINT and string (CHAR or VARCHAR) values.
- **Warning: incompatible change!** DAYOFWEEK() had offset 0 for Sunday. Changed the offset to 1.
- Give an error for queries that mix GROUP BY columns and fields when there is no GROUP BY specification.
- Added --vertical option to mysql , for printing results in vertical mode.
- Index-only optimization; some queries are now resolved using only indexes. Until MySQL 4.0, this works only for numeric columns. Comment MySQL utilise les index .
- Lots of new benchmarks.
- A new C API chapter and lots of other improvements in the manual.

26.5.32 Changements de la version 3.22.4

- Added --tmpdir option to mysqld , for specifying the location of the temporary file directory.
- MySQL now automatically changes a query from an ODBC client:

```
SELECT ... FROM table WHERE auto_increment_column IS NULL
```

  to:

```
SELECT ... FROM table WHERE auto_increment_column == LAST_INSERT_ID()
```

  This allows some ODBC programs (Delphi, Access) to retrieve the newly inserted row to fetch the AUTO_INCREMENT id.
- DROP TABLE now waits for all users to free a table before deleting it.
- Fixed small memory leak in the new connect protocol.
- New functions BIN() , OCT() , HEX() and CONV() for converting between different number bases.
- Added function SUBSTRING() with 2 arguments.
- If you created a table with a record length smaller than 5, you couldn't delete rows from the table.
- Added optimization to remove const reference tables from ORDER BY and GROUP BY .
- mysqld now automatically disables system locking on Linux and Windows, and for systems that use MIT-pthreads. You can force the use of locking with the --enable-external-locking option.
- Added --console option to mysqld , to force a console window (for error messages) when using Windows.
- Fixed table locks for Windows.
- Allow '$' in identifiers.
- Changed name of user-specific configuration file from my.cnf to .my.cnf (Unix only).
- Added DATE_ADD() and DATE_SUB() functions.

26.5.33 Changements de la version 3.22.3

- Fixed a lock problem (bug in MySQL Version 3.22.1) when closing temporary tables.
- Added missing mysql_ping() to the client library.
- Added --compress option to all MySQL clients.
- Changed byte to char in mysql.h and mysql_com.h .

26.5.34 Changements de la version 3.22.2

- Searching on multiple constant keys that matched more than 30% of the rows didn't always use the best possible key.
- New functions << , >> , RPAD() and LPAD() .
- You can now save default options (like passwords) in a configuration file (my.cnf).
- Lots of small changes to get ORDER BY to work when no records are found when using fields that are not in GROUP BY (MySQL extension).
- Added --chroot option to mysqld , to start mysqld in a chroot environment (by Nikki Chumakov nikkic@cityline.ru).
- Trailing spaces are now ignored when comparing case-sensitive strings; this should fix some problems with ODBC and flag 512!
- Fixed a core dump bug in the range optimizer.
- Added --one-thread option to mysqld , for debugging with LinuxThreads (or glibc). (This replaces the -T32 flag)
- Added DROP TABLE IF EXISTS to prevent an error from occurring if the table doesn't exist.
- IF and EXISTS are now reserved words (they would have to be sooner or later).
- Added lots of new options to mysqldump .
- Server error messages are now in mysqld_error.h .
- The server/client protocol now supports compression.
- All bug fixes from MySQL Version 3.21.32.

26.5.35 Changements de la version 3.22.1 (Jun 1998: Alpha)

- Added new C API function mysql_ping() .
- Added new API functions mysql_init() and mysql_options() . You now MUST call mysql_init() before you call mysql_real_connect() . You don't have to call mysql_init() if you only use mysql_connect() .
- Added mysql_options(...,MYSQL_OPT_CONNECT_TIMEOUT,...) so you can set a timeout for connecting to a server.
- Added --timeout option to mysqladmin , as a test of mysql_options() .
- Added AFTER column and FIRST options to ALTER TABLE ... ADD columns . This makes it possible to add a new column at some specific location within a row in an existing table.

- WEEK() now takes an optional argument to allow handling of weeks when the week starts on Monday (some European countries). By default, WEEK() assumes the week starts on Sunday.
- TIME columns weren't stored properly (bug in MySQL Version 3.22.0).
- UPDATE now returns information about how many rows were matched and updated, and how many ``warnings'' occurred when doing the update.
- Fixed incorrect result from FORMAT(-100,2) .
- ENUM and SET columns were compared in binary (case-sensitive) fashion; changed to be case-insensitive.

26.5.36 Changements de la version 3.22.0

- New (backward-compatible) connect protocol that allows you to specify the database to use when connecting, to get much faster connections to a specific database.

  The mysql_real_connect() call is changed to:

```
mysql_real_connect(MYSQL *mysql, const char *host, const char *user,
                   const char *passwd, const char *db, uint port,
                   const char *unix_socket, uint client_flag)
```

- Each connection is handled by its own thread, rather than by the master accept() thread. This fixes permanently the telnet bug that was a topic on the mail list some time ago.
- All TCP/IP connections are now checked with backward-resolution of the hostname to get better security. mysqld now has a local hostname resolver cache so connections should actually be faster than before, even with this feature.
- A site automatically will be blocked from future connections if someone repeatedly connects with an ``improper header'' (like when one uses telnet).
- You can now refer to tables in different databases with references of the form tbl_name@db_name or db_name.tbl_name . This makes it possible to give a user read access to some tables and write access to others simply by keeping them in different databases!
- Added --user option to mysqld , to allow it to run as another Unix user (if it is started as the Unix root user).
- Added caching of users and access rights (for faster access rights checking)
- Normal users (not anonymous ones) can change their password with mysqladmin password "new_password" . This uses encrypted passwords that are not logged in the normal MySQL log!
- All important string functions are now coded in assembler for x86 Linux machines. This gives a speedup of 10% in many cases.
- For tables that have many columns, the column names are now hashed for much faster column name lookup (this will speed up some benchmark tests a lot!)
- Some benchmarks are changed to get better individual timing. (Some loops were so short that a specific test took < 2 seconds. The loops have been changed to take about 20 seconds to make it easier to compare different databases. A test that took 1-2 seconds before now takes 11-24 seconds, which is much better)
- Re-arranged SELECT code to handle some very specific queries involving group functions (like COUNT(*)) without a GROUP BY but with HAVING . The following now works:

```
mysql> SELECT COUNT(*) as C FROM table HAVING C > 1;
```

- Changed the protocol for field functions to be faster and avoid some calls to malloc() .
- Added -T32 option to mysqld , for running all queries under the main thread. This makes it possible to debug mysqld under Linux with gdb !
- Added optimization of not_null_column IS NULL (needed for some Access queries).
- Allow STRAIGHT_JOIN to be used between two tables to force the optimizer to join them in a specific order.
- String functions now return VARCHAR rather than CHAR and the column type is now VARCHAR for fields saved as VARCHAR . This should make the MyODBC driver better, but may break some old MySQL clients that don't handle FIELD_TYPE_VARCHAR the same way as FIELD_TYPE_CHAR .
- CREATE INDEX and DROP INDEX are now implemented through ALTER TABLE . CREATE TABLE is still the recommended (fast) way to create indexes.
- Added --set-variable option wait_timeout to mysqld .
- Added time column to mysqladmin processlist to show how long a query has taken or how long a thread has slept.
- Added lots of new variables to show variables and some new to show status .
- Added new type YEAR . YEAR is stored in 1 byte with allowable values of 0, and 1901 to 2155.
- Added new DATE type that is stored in 3 bytes rather than 4 bytes. All new tables are created with the new date type if you don't use the --old-protocol option to mysqld .
- Fixed bug in record caches; for some queries, you could get Error from table handler: # on some operating systems.
- Added --enable-assembler option to configure , for x86 machines (tested on Linux + gcc). This will enable assembler functions for the most important string functions for more speed!

26.6 Changements de la version 3.21.x

La version 3.21 est plutôt vieille, et doit être évitée si possible. Les informations de cette section sont conservées à des fins d'archives.

26.6.1 Changements de la version 3.21.33

- Fixed problem when sending SIGHUP to mysqld ; mysqld core dumped when starting from boot on some systems.
- Fixed problem with losing a little memory for some connections.
- DELETE FROM tbl_name without a WHERE condition is now done the long way when you use LOCK TABLES or if the table is in use, to avoid race conditions.
- INSERT INTO TABLE (timestamp_column) VALUES (NULL); didn't set timestamp.

26.6.2 Changements de la version 3.21.32

- Fixed some possible race conditions when doing many reopen/close on the same tables under heavy load! This can happen if you execute mysqladmin refresh often. This could in some very rare cases corrupt the header of the index file and cause error 126 or 138.
- Fixed fatal bug in refresh() when running with the --skip-external-locking option. There was a ``very small'' time gap after

a mysqladmin refresh when a table could be corrupted if one thread updated a table while another thread did mysqladmin refresh and another thread started a new update ont the same table before the first thread had finished. A refresh (or --flush-tables) will now not return until all used tables are closed!
- SELECT DISTINCT with a WHERE clause that didn't match any rows returned a row in some contexts (bug only in 3.21.31).
- GROUP BY + ORDER BY returned one empty row when no rows where found.
- Fixed a bug in the range optimizer that wrote Use_count: Wrong count for ... in the error log file.

26.6.3 Changements de la version 3.21.31

- Fixed a sign extension problem for the TINYINT type on Irix.
- Fixed problem with LEFT("constant_string",function) .
- Fixed problem with FIND_IN_SET() .
- LEFT JOIN core dumped if the second table is used with a constant WHERE/ON expression that uniquely identifies one record.
- Fixed problems with DATE_FORMAT() and incorrect dates. DATE_FORMAT() now ignores '%' to make it possible to extend it more easily in the future.

26.6.4 Changements de la version 3.21.30

- mysql now returns an exit code > 0 if the query returned an error.
- Saving of command-line history to file in mysql client. By Tommy Larsen tommy@mix.hive.no .
- Fixed problem with empty lines that were ignored in mysql.cc .
- Save the pid of the signal handler thread in the pid file instead of the pid of the main thread.
- Added patch by tommy@valley.ne.jp to support Japanese characters SJIS and UJIS.
- Changed safe_mysqld to redirect startup messages to 'hostname'.err instead of 'hostname'.log to reclaim file space on mysqladmin refresh .
- ENUM always had the first entry as default value.
- ALTER TABLE wrote two entries to the update log.
- sql_acc() now closes the mysql grant tables after a reload to save table space and memory.
- Changed LOAD DATA to use less memory with tables and BLOB columns.
- Sorting on a function which made a division / 0 produced a wrong set in some cases.
- Fixed SELECT problem with LEFT() when using the czech character set.
- Fixed problem in isamchk ; it couldn't repair a packed table in a very unusual case.
- SELECT statements with or | (bit functions) failed on columns with NULL values.
- When comparing a field = field, where one of the fields was a part key, only the length of the part key was compared.

26.6.5 Changements de la version 3.21.29

- LOCK TABLES + DELETE from tbl_name never removed locks properly.
- Fixed problem when grouping on an OR function.
- Fixed permission problem with umask() and creating new databases.
- Fixed permission problem on result file with SELECT ... INTO OUTFILE ...
- Fixed problem in range optimizer (core dump) for a very complex query.
- Fixed problem when using MIN(integer) or MAX(integer) in GROUP BY .
- Fixed bug on Alpha when using integer keys. (Other keys worked on Alpha.)
- Fixed bug in WEEK("XXXX-xx-01") .

26.6.6 Changements de la version 3.21.28

- Fixed socket permission (clients couldn't connect to Unix socket on Linux).
- Fixed bug in record caches; for some queries, you could get Error from table handler: # on some operating systems.

26.6.7 Changements de la version 3.21.27

- Added user level lock functions GET_LOCK(string,timeout) , RELEASE_LOCK(string) .
- Added Opened_tables to show status .
- Changed connect timeout to 3 seconds to make it somewhat harder for crackers to kill mysqld through telnet + TCP/IP.
- Fixed bug in range optimizer when using WHERE key_part_1 >= something AND key_part_2 <= something_else .
- Changed configure for detection of FreeBSD 3.0 9803xx and above
- WHERE with string_col_key = constant_string didn't always find all rows if the column had many values differing only with characters of the same sort value (like e and e with an accent).
- Strings keys looked up with 'ref' were not compared in case-sensitive fashion.
- Added umask() to make log files non-readable for normal users.
- Ignore users with old (8-byte) password on startup if not using --old-protocol option to mysqld .
- SELECT which matched all key fields returned the values in the case of the matched values, not of the found values. (Minor problem.)

26.6.8 Changements de la version 3.21.26

- FROM_DAYS(0) now returns "0000-00-00".
- In DATE_FORMAT() , PM and AM were swapped for hours 00 and 12.
- Extended the default maximum key size to 256.
- Fixed bug when using BLOB / TEXT in GROUP BY with many tables.
- An ENUM field that is not declared NOT NULL has NULL as the default value. (Previously, the default value was the first enumeration value.)
- Fixed bug in the join optimizer code when using many part keys on the same key: INDEX (Organization,Surname(35),Initials(35)) .

- Added some tests to the table order optimizer to get some cases with SELECT ... FROM many_tables much faster.
- Added a retry loop around accept() to possibly fix some problems on some Linux machines.

26.6.9 Changements de la version 3.21.25

- Changed typedef 'string' to typedef 'my_string' for better portability.
- You can now kill threads that are waiting on a disk-full condition.
- Fixed some problems with UDF functions.
- Added long options to isamchk . Try isamchk --help .
- Fixed a bug when using 8 bytes long (alpha); filesort() didn't work. Affects DISTINCT , ORDER BY and GROUP BY on 64-bit processors.

26.6.10 Changements de la version 3.21.24

- Dynamic loadable functions. Based on source from Alexis Mikhailov.
- You couldn't delete from a table if no one had done a SELECT on the table.
- Fixed problem with range optimizer with many OR operators on key parts inside each other.
- Recoded MIN() and MAX() to work properly with strings and HAVING .
- Changed default umask value for new files from 0664 to 0660 .
- Fixed problem with LEFT JOIN and constant expressions in the ON part.
- Added Italian error messages from brenno@dewinter.com .
- configure now works better on OSF/1 (tested on 4.0D).
- Added hooks to allow LIKE optimization with international character support.
- Upgraded DBI to 0.93.

26.6.11 Changements de la version 3.21.23

- The following symbols are now reserved words: TIME , DATE , TIMESTAMP , TEXT , BIT , ENUM , NO , ACTION , CHECK , YEAR , MONTH , DAY , HOUR , MINUTE , SECOND , STATUS , VARIABLES .
- Setting a TIMESTAMP to NULL in LOAD DATA INFILE ... didn't set the current time for the TIMESTAMP .
- Fix BETWEEN to recognize binary strings. Now BETWEEN is case-sensitive.
- Added --skip-thread-priority option to mysqld , for systems where mysqld 's thread scheduling doesn't work properly (BSDI 3.1).
- Added ODBC functions DAYNAME() and MONTHNAME() .
- Added function TIME_FORMAT() . This works like DATE_FORMAT() , but takes a time string ('HH:MM:SS') as argument.
- Fixed unlikely(?) key optimizer bug when using OR operators of key parts inside AND expressions.
- Added variables command to mysqladmin .
- A lot of small changes to the binary releases.
- Fixed a bug in the new protocol from MySQL Version 3.21.20.
- Changed ALTER TABLE to work with Windows (Windows can't rename open files). Also fixed a couple of small bugs in the Windows version.
- All standard MySQL clients are now ported to MySQL for Windows.
- MySQL can now be started as a service on NT.

26.6.12 Changements de la version 3.21.22

- Starting with this version, all MySQL distributions will be configured, compiled and tested with crash-me and the benchmarks on the following platforms: SunOS 5.6 sun4u, SunOS 5.5.1 sun4u, SunOS 4.14 sun4c, SunOS 5.6 i86pc, Irix 6.3 mips5k, HP-UX 10.20 hppa, AIX 4.2.1 ppc, OSF/1 V4.0 alpha, FreeBSD 2.2.2 i86pc and BSDI 3.1 i386.
- Fix COUNT(*) problems when the WHERE clause didn't match any records. (Bug from 3.21.17.)
- Removed that NULL = NULL is true. Now you must use IS NULL or IS NOT NULL to test whether a value is NULL . (This is according to SQL-99 but may break old applications that are ported from mSQL .) You can get the old behavior by compiling with -DmSQL_COMPLIANT .
- Fixed bug that core dumped when using many LEFT OUTER JOIN clauses.
- Fixed bug in ORDER BY on string formula with possible NULL values.
- Fixed problem in range optimizer when using <= on sub index.
- Added functions DAYOFYEAR() , DAYOFMONTH() , MONTH() , YEAR() , WEEK() , QUARTER() , HOUR() , MINUTE() , SECOND() and FIND_IN_SET() .
- Added SHOW VARIABLES command.
- Added support of ``long constant strings'' from SQL-99:

```
mysql> SELECT 'first ' 'second';       -> 'first second'
```

- Upgraded Msql-Mysql-modules to 1.1825.
- Upgraded mysqlaccess to 2.02.
- Fixed problem with Russian character set and LIKE .
- Ported to OpenBSD 2.1.
- New Dutch error messages.

26.6.13 Changements de la version 3.21.21a

- Configure changes for some operating systems.

26.6.14 Changements de la version 3.21.21

- Fixed optimizer bug when using WHERE data_field = date_field2 AND date_field2 = constant .
- Added SHOW STATUS command.
- Removed manual.ps from the source distribution to make it smaller.

26.6.15 Changements de la version 3.21.20

- Changed the maximum table name and column name lengths from 32 to 64.
- Aliases can now be of ``any'' length.
- Fixed mysqladmin stat to return the right number of queries.
- Changed protocol (downward compatible) to mark if a column has the AUTO_INCREMENT attribute or is a TIMESTAMP . This is needed for the new Java driver.
- Added Hebrew sorting order by Zeev Suraski.
- Solaris 2.6: Fixed configure bugs and increased maximum table size from 2G to 4G.

26.6.16 Changements de la version 3.21.19

- Upgraded DBD to 1.1823. This version implements mysql_use_result in DBD-Mysql .
- Benchmarks updated for empress (by Luuk).
- Fixed a case of slow range searching.
- Configure fixes (Docs directory).
- Added function REVERSE() (by Zeev Suraski).

26.6.17 Changements de la version 3.21.18

- Issue error message if client C functions are called in wrong order.
- Added automatic reconnect to the libmysql.c library. If a write command fails, an automatic reconnect is done.
- Small sort sets no longer use temporary files.
- Upgraded DBI to 0.91.
- Fixed a couple of problems with LEFT OUTER JOIN .
- Added CROSS JOIN syntax. CROSS is now a reserved word.
- Recoded yacc / bison stack allocation to be even safer and to allow MySQL to handle even bigger expressions.
- Fixed a couple of problems with the update log.
- ORDER BY was slow when used with key ranges.

26.6.18 Changements de la version 3.21.17

- Changed documentation string of --with-unix-socket-path to avoid confusion.
- Added ODBC and SQL-99 style LEFT OUTER JOIN .
- The following are new reserved words: LEFT , NATURAL , USING .
- The client library now uses the value of the environment variable MYSQL_HOST as the default host if it's defined.
- SELECT col_name, SUM(expr) now returns NULL for col_name when there are matching rows.
- Fixed problem with comparing binary strings and BLOB values with ASCII characters over 127.
- Fixed lock problem: when freeing a read lock on a table with multiple read locks, a thread waiting for a write lock would have been given the lock. This shouldn't affect data integrity, but could possibly make mysqld restart if one thread was reading data that another thread modified.
- LIMIT offset,count didn't work in INSERT ... SELECT .
- Optimized key block caching. This will be quicker than the old algorithm when using bigger key caches.

26.6.19 Changements de la version 3.21.16

- Added ODBC 2.0 &3.0 functions POWER() , SPACE() , COT() , DEGREES() , RADIANS() , ROUND(2 arg) and TRUNCATE() .
- **Warning: Incompatible change!** LOCATE() parameters were swapped according to ODBC standard. Fixed.
- Added function TIME_TO_SEC() .
- In some cases, default values were not used for NOT NULL fields.
- Timestamp wasn't always updated properly in UPDATE SET ... statements.
- Allow empty strings as default values for BLOB and TEXT , to be compatible with mysqldump .

26.6.20 Changements de la version 3.21.15

- **Warning: Incompatible change!** mysqlperl is now from Msql-Mysql-modules. This means that connect() now takes host , database , user , password arguments! The old version took host , database , password , user .
- Allow DATE '1997-01-01' , TIME '12:10:10' and TIMESTAMP '1997-01-01 12:10:10' formats required by SQL-99. **Warning: Incompatible change!** This has the unfortunate side-effect that you no longer can have columns named DATE , TIME or TIMESTAMP . :(Old columns can still be accessed through tablename.columnname !)
- Changed Makefiles to hopefully work better with BSD systems. Also, manual.dvi is now included in the distribution to avoid having stupid make programs trying to rebuild it.
- readline library upgraded to version 2.1.
- A new sortorder german-1. That is a normal ISO-Latin1 with a german sort order.
- Perl DBI / DBD is now included in the distribution. DBI is now the recommended way to connect to MySQL from Perl.
- New portable benchmark suite with DBD , with test results from mSQL 2.0.3, MySQL, PostgreSQL 6.2.1 and Solid server 2.2.
- crash-me is now included with the benchmarks; this is a Perl program designed to find as many limits as possible in an SQL server. Tested with mSQL , PostgreSQL, Solid and MySQL.
- Fixed bug in range-optimizer that crashed MySQL on some queries.

- Table and column name completion for mysql command-line tool, by Zeev Suraski and Andi Gutmans.
- Added new command REPLACE that works like INSERT but replaces conflicting records with the new record. REPLACE INTO TABLE ... SELECT ... works also.
- Added new commands CREATE DATABASE db_name and DROP DATABASE db_name .
- Added RENAME option to ALTER TABLE : ALTER TABLE name RENAME TO new_name .
- make_binary_distribution now includes libgcc.a in libmysqlclient.a . This should make linking work for people who don't have gcc .
- Changed net_write() to my_net_write() because of a name conflict with Sybase.
- New function DAYOFWEEK() compatible with ODBC.
- Stack checking and bison memory overrun checking to make MySQL safer with weird queries.

26.6.21 Changements de la version 3.21.14b

- Fixed a couple of small configure problems on some platforms.

26.6.22 Changements de la version 3.21.14a

- Ported to SCO Openserver 5.0.4 with FSU Pthreads.
- HP-UX 10.20 should work.
- Added new function DATE_FORMAT() .
- Added NOT IN .
- Added automatic removal of 'ODBC function conversions': {fn now() }
- Handle ODBC 2.50.3 option flags.
- Fixed comparison of DATE and TIME values with NULL .
- Changed language name from germany to german to be consistent with the other language names.
- Fixed sorting problem on functions returning a FLOAT . Previously, the values were converted to INT values before sorting.
- Fixed slow sorting when sorting on key field when using key_column=constant .
- Sorting on calculated DOUBLE values sorted on integer results instead.
- mysql no longer requires a database argument.
- Changed the place where HAVING should be. According to the SQL standards, it should be after GROUP BY but before ORDER BY . MySQL Version 3.20 incorrectly had it last.
- Added Sybase command USE database to start using another database.
- Added automatic adjusting of number of connections and table cache size if the maximum number of files that can be opened is less than needed. This should fix that mysqld doesn't crash even if you haven't done a ulimit -n 256 before starting mysqld .
- Added lots of limit checks to make it safer when running with too little memory or when doing weird queries.

26.6.23 Changements de la version 3.21.13

- Added retry of interrupted reads and clearing of errno . This makes Linux systems much safer!
- Fixed locking bug when using many aliases on the same table in the same SELECT .
- Fixed bug with LIKE on number key.
- New error message so you can check whether the connection was lost while the command was running or whether the connection was down from the start.
- Added --table option to mysql to print in table format. Moved time and row information after query result. Added automatic reconnect of lost connections.
- Added != as a synonym for <> .
- Added function VERSION() to make easier logs.
- New multi-user test tests/fork_test.pl to put some strain on the thread library.

26.6.24 Changements de la version 3.21.12

- Fixed ftruncate() call in MIT-pthreads. This made isamchk destroy the .ISM files on (Free)BSD 2.x systems.
- Fixed broken __P_ patch in MIT-pthreads.
- Many memory overrun checks. All string functions now return NULL if the returned string should be longer than max_allowed_packet bytes.
- Changed the name of the INTERVAL type to ENUM , because INTERVAL is used in SQL-99.
- In some cases, doing a JOIN + GROUP + INTO OUTFILE , the result wasn't grouped.
- LIKE with '_' as last character didn't work. Fixed.
- Added extended SQL-99 TRIM() function.
- Added CURTIME() .
- Added ENCRYPT() function by Zeev Suraski.
- Fixed better FOREIGN KEY syntax skipping. New reserved words: MATCH , FULL , PARTIAL .
- mysqld now allows IP number and hostname for the --bind-address option.
- Added SET CHARACTER SET cp1251_koi8 to enable conversions of data to and from the cp1251_koi8 character set.
- Lots of changes for Windows 95 port. In theory, this version should now be easily portable to Windows 95.
- Changed the CREATE COLUMN syntax of NOT NULL columns to be after the DEFAULT value, as specified in the SQL-99 standard. This will make mysqldump with NOT NULL and default values incompatible with MySQL Version 3.20.
- Added many function name aliases so the functions can be used with ODBC or SQL-92 syntax.
- Fixed syntax of ALTER TABLE tbl_name ALTER COLUMN col_name SET DEFAULT NULL .
- Added CHAR and BIT as synonyms for CHAR(1) .
- Fixed core dump when updating as a user who has only SELECT privilege.
- INSERT ... SELECT ... GROUP BY didn't work in some cases. An Invalid use of group function error occurred.
- When using LIMIT , SELECT now always uses keys instead of record scan. This will give better performance on SELECT and a WHERE that matches many rows.
- Added Russian error messages.

26.6.25 Changements de la version 3.21.11

- Configure changes.
- MySQL now works with the new thread library on BSD/OS 3.0.
- Added new group functions BIT_OR() and BIT_AND() .
- Added compatibility functions CHECK and REFERENCES . CHECK is now a reserved word.
- Added ALL option to GRANT for better compatibility. (GRANT is still a dummy function.)
- Added partly translated Dutch error messages.
- Fixed bug in ORDER BY and GROUP BY with NULL columns.
- Added function LAST_INSERT_ID() SQL function to retrieve last AUTO_INCREMENT value. This is intended for clients to ODBC that can't use the mysql_insert_id() API function, but can be used by any client.
- Added --flush-logs option to mysqladmin .
- Added command STATUS to mysql .
- Fixed problem with ORDER BY / GROUP BY because of bug in gcc .
- Fixed problem with INSERT ... SELECT ... GROUP BY .

26.6.26 Changements de la version 3.21.10

- New program mysqlaccess .
- CREATE now supports all ODBC types and the mSQL TEXT type. All ODBC 2.5 functions are also supported (added REPEAT). This provides better portability.
- Added text types TINYTEXT , TEXT , MEDIUMTEXT and LONGTEXT . These are actually BLOB types, but all searching is done in case-insensitive fashion.
- All old BLOB fields are now TEXT fields. This only changes that all searching on strings is done in case-sensitive fashion. You must do an ALTER TABLE and change the datatype to BLOB if you want to have tests done in case-sensitive fashion.
- Fixed some configure issues.
- Made the locking code a bit safer. Fixed very unlikely deadlock situation.
- Fixed a couple of bugs in the range optimizer. Now the new range benchmark test-select works.

26.6.27 Changements de la version 3.21.9

- Added --enable-unix-socket=pathname option to configure .
- Fixed a couple of portability problems with include files.
- Fixed bug in range calculation that could return empty set when searching on multiple key with only one entry (very rare).
- Most things ported to FSU Pthreads, which should allow MySQL to run on SCO. Notes sur SCO .

26.6.28 Changements de la version 3.21.8

- Works now in Solaris 2.6.
- Added handling of calculation of SUM() functions. For example, you can now use SUM(column)/COUNT(column) .
- Added handling of trigometric functions: PI() , ACOS() , ASIN() , ATAN() , COS() , SIN() and TAN() .
- New languages: Norwegian, Norwegian-ny and Portuguese.
- Fixed parameter bug in net_print() in procedure.cc .
- Fixed a couple of memory leaks.
- Now allow also the old SELECT ... INTO OUTFILE syntax.
- Fixed bug with GROUP BY and SELECT on key with many values.
- mysql_fetch_lengths() sometimes returned incorrect lengths when you used mysql_use_result() . This affected at least some cases of mysqldump --quick .
- Fixed bug in optimization of WHERE const op field .
- Fixed problem when sorting on NULL fields.
- Fixed a couple of 64-bit (Alpha) problems.
- Added --pid-file=# option to mysqld .
- Added date formatting to FROM_UNIXTIME() , originally by Zeev Suraski.
- Fixed bug in BETWEEN in range optimizer (did only test = of the first argument).
- Added machine-dependent files for MIT-pthreads i386-SCO. There is probably more to do to get this to work on SCO 3.5.

26.6.29 Changements de la version 3.21.7

- Changed Makefile.am to take advantage of Automake 1.2.
- Added the beginnings of a benchmark suite.
- Added more secure password handling.
- Added new client function mysql_errno() , to get the error number of the error message. This makes error checking in the client much easier. This makes the new server incompatible with the 3.20.x server when running without --old-protocol . The client code is backward-compatible. More information can be found in the README file!
- Fixed some problems when using very long, illegal names.

26.6.30 Changements de la version 3.21.6

- Fixed more portability issues (incorrect sigwait and sigset defines).
- configure should now be able to detect the last argument to accept() .

26.6.31 Changements de la version 3.21.5

- Should now work with FreeBSD 3.0 if used with FreeBSD-3.0-libc_r-1.0.diff , which can be found at http://www.mysql.com/downloads/os-freebsd.html .
- Added new -O tmp_table_size=# option to mysqld .
- New function FROM_UNIXTIME(timestamp) which returns a date string in 'YYYY-MM-DD HH:MM:SS' format.
- New function SEC_TO_TIME(seconds) which returns a string in 'HH:MM:SS' format.
- New function SUBSTRING_INDEX() , originally by Zeev Suraski.

26.6.32 Changements de la version 3.21.4

- Should now configure and compile on OSF/1 4.0 with the DEC compiler.
- Configuration and compilation on BSD/OS 3.0 works, but due to some bugs in BSD/OS 3.0, mysqld doesn't work on it yet.
- Configuration and compilation on FreeBSD 3.0 works, but I couldn't get pthread_create to work.

26.6.33 Changements de la version 3.21.3

- Added reverse check lookup of hostnames to get better security.
- Fixed some possible buffer overflows if filenames that are too long are used.
- mysqld doesn't accept hostnames that start with digits followed by a '.' , because the hostname may look like an IP number.
- Added --skip-networking option to mysqld , to allow only socket connections. (This will not work with MIT-pthreads!)
- Added check of too long table names for alias.
- Added check if database name is okay.
- Added check if too long table names.
- Removed incorrect free() that killed the server on CREATE DATABASE or DROP DATABASE .
- Changed some mysqld -O options to better names.
- Added -O join_cache_size=# option to mysqld .
- Added -O max_join_size=# option to mysqld , to be able to set a limit how big queries (in this case big = slow) one should be able to handle without specifying SET SQL_BIG_SELECTS=1 . A # = is about 10 examined records. The default is ``unlimited''.
- When comparing a TIME , DATE , DATETIME or TIMESTAMP column to a constant, the constant is converted to a time value before performing the comparison. This will make it easier to get ODBC (particularly Access97) to work with the above types. It should also make dates easier to use and the comparisons should be quicker than before.
- Applied patch from Jochen Wiedmann that allows query() in mysqlperl to take a query with \0 in it.
- Storing a timestamp with a 2-digit year (YYMMDD) didn't work.
- Fix that timestamp wasn't automatically updated if set in an UPDATE clause.
- Now the automatic timestamp field is the FIRST timestamp field.
- SELECT * INTO OUTFILE , which didn't correctly if the outfile already existed.
- mysql now shows the thread ID when starting or doing a reconnect.
- Changed the default sort buffer size from 2M to 1M.

26.6.34 Changements de la version 3.21.2

- The range optimizer is coded, but only 85% tested. It can be enabled with --new , but it crashes core a lot yet...
- More portable. Should compile on AIX and alpha-digital. At least the isam library should be relatively 64-bit clean.
- New isamchk which can detect and fix more problems.
- New options for isamlog .
- Using new version of Automake.
- Many small portability changes (from the AIX and alpha-digital port) Better checking of pthread(s) library.
- czech error messages by snajdr@pvt.net .
- Decreased size of some buffers to get fewer problems on systems with little memory. Also added more checks to handle ``out of memory'' problems.
- mysqladmin : you can now do mysqladmin kill 5,6,7,8 to kill multiple threads.
- When the maximum connection limit is reached, one extra connection by a user with the **process_acl** privilege is granted.
- Added -O backlog=# option to mysqld .
- Increased maximum packet size from 512K to 1024K for client.
- Almost all of the function code is now tested in the internal test suite.
- ALTER TABLE now returns warnings from field conversions.
- Port changed to 3306 (got it reserved from ISI).
- Added a fix for Visual FoxBase so that any schema name from a table specification is automatically removed.
- New function ASCII() .
- Removed function BETWEEN(a,b,c) . Use the standard SQL syntax instead: expr BETWEEN expr AND expr .
- MySQL no longer has to use an extra temporary table when sorting on functions or SUM() functions.
- Fixed bug that you couldn't use tbl_name.field_name in UPDATE .
- Fixed SELECT DISTINCT when using 'hidden group'. For example:

  ```
  mysql> SELECT DISTINCT MOD(some_field,10) FROM test
      ->        GROUP BY some_field;
  ```

  Note: some_field is normally in the SELECT part. Standard SQL should require it.

26.6.35 Changements de la version 3.21.0

- New reserved words used: INTERVAL , EXPLAIN , READ , WRITE , BINARY .
- Added ODBC function CHAR(num,...) .
- New operator IN . This uses a binary search to find a match.
- New command LOCK TABLES tbl_name [AS alias] {READ|WRITE} ...
- Added --log-update option to mysqld , to get a log suitable for incremental updates.
- New command EXPLAIN SELECT ... to get information about how the optimizer will do the join.
- For easier client code, the client should no longer use FIELD_TYPE_TINY_BLOB , FIELD_TYPE_MEDIUM_BLOB , FIELD_TYPE_LONG_BLOB or FIELD_TYPE_VAR_STRING (as previously returned by mysql_list_fields). You should instead only use FIELD_TYPE_BLOB or FIELD_TYPE_STRING . If you want exact types, you should use the command SHOW FIELDS .
- Added varbinary syntax: 0x@# which can be used as a string (default) or a number.
- FIELD_TYPE_CHAR is renamed to FIELD_TYPE_TINY .
- Changed all fields to C++ classes.

- Removed FORM struct.
- Fields with DEFAULT values no longer need to be NOT NULL .
- New field types:
- Now all function calculation is done with double or long long . This will provide the full 64-bit range with bit functions and fix some conversions that previously could result in precision losses. One should avoid using unsigned long long columns with full 64-bit range (numbers bigger than 9223372036854775807) because calculations are done with signed long long .
- ORDER BY will now put NULL field values first. GROUP BY will also work with NULL values.
- Full WHERE with expressions.
- New range optimizer that can resolve ranges when some keypart prefix is constant. Example:

```
mysql> SELECT * FROM tbl_name
   ->          WHERE key_part_1="customer"
   ->          AND key_part_2>=10 AND key_part_2<=10;
```

26.7 Changements de la version 3.20.x

La version 3.20 est plutôt vieille, et doit être évitée si possible. Les informations de cette section sont conservées à des fins d'archives.

Les changements des versions 3.20.18 à 3.20.32b ne sont pas documentés ici, car la version 3.21 a commencé ici. Les modifications importantes sont documentés dans l'histoiriqe de la version 3.21.

26.7.1 Changements de la version 3.20.18

- Added -p# (remove # directories from path) to isamlog . All files are written with a relative path from the database directory Now mysqld shouldn't crash on shutdown when using the --log-isam option.
- New mysqlperl version. It is now compatible with msqlperl-0.63 .
- New DBD module available.
- Added group function STD() (standard deviation).
- The mysqld server is now compiled by default without debugging information. This will make the daemon smaller and faster.
- Now one usually only has to specify the --basedir option to mysqld . All other paths are relative in a normal installation.
- BLOB columns sometimes contained garbage when used with a SELECT on more than one table and ORDER BY .
- Fixed that calculations that are not in GROUP BY work as expected (SQL-99 extension). Example:

```
mysql> SELECT id,id+1 FROM table GROUP BY id;
```

- The test of using MYSQL_PWD was reversed. Now MYSQL_PWD is enabled as default in the default release.
- Fixed conversion bug which caused mysqld to core dump with Arithmetic error on SPARC-386.
- Added --unbuffered option to mysql , for new mysqlaccess .
- When using overlapping (unnecessary) keys and join over many tables, the optimizer could get confused and return 0 records.

26.7.2 Changements de la version 3.20.17

- You can now use BLOB columns and the functions IS NULL and IS NOT NULL in the WHERE clause.
- All communication packets and row buffers are now allocated dynamically on demand. The default value of max_allowed_packet is now 64K for the server and 512K for the client. This is mainly used to catch incorrect packets that could trash all memory. The server limit may be changed when it is started.
- Changed stack usage to use less memory.
- Changed safe_mysqld to check for running daemon.
- The ELT() function is renamed to FIELD() . The new ELT() function returns a value based on an index: FIELD() is the inverse of ELT() Example: ELT(2,"A","B","C") returns "B" . FIELD("B","A","B","C") returns 2 .
- COUNT(field) , where field could have a NULL value, now works.
- A couple of bugs fixed in SELECT ... GROUP BY .
- Fixed memory overrun bug in WHERE with many unoptimizable brace levels.
- Fixed some small bugs in the grant code.
- If hostname isn't found by get_hostname , only the IP is checked. Previously, you got Access denied .
- Inserts of timestamps with values didn't always work.
- INSERT INTO ... SELECT ... WHERE could give the error Duplicated field .
- Added some tests to safe_mysqld to make it ``safer''.
- LIKE was case-sensitive in some places and case-insensitive in others. Now LIKE is always case-insensitive.
- mysql.cc : Allow '#' anywhere on the line.
- New command SET SQL_SELECT_LIMIT=# . See the FAQ for more details.
- New version of the mysqlaccess script.
- Change FROM_DAYS() and WEEKDAY() to also take a full TIMESTAMP or DATETIME as argument. Before they only took a number of type YYYYMMDD or YYMMDD .
- Added new function UNIX_TIMESTAMP(timestamp_column) .

26.7.3 Changements de la version 3.20.16

- More changes in MIT-pthreads to get them safer. Fixed also some link bugs at least in SunOS.
- Changed mysqld to work around a bug in MIT-pthreads. This makes multiple small SELECT operations 20 times faster. Now lock_test.pl should work.
- Added mysql_FetchHash(handle) to mysqlperl .
- The mysqlbug script is now distributed built to allow for reporting bugs that appear during the build with it.
- Changed libmysql.c to prefer getpwuid() instead of cuserid() .
- Fixed bug in SELECT optimizer when using many tables with the same column used as key to different tables.
- Added new latin2 and Russian KOI8 character tables.
- Added support for a dummy GRANT command to satisfy Powerbuilder.

26.7.4 Changements de la version 3.20.15

- Fixed fatal bug packets out of order when using MIT-pthreads.
- Removed possible loop when a thread waits for command from client and fcntl() fails. Thanks to Mike Bretz for finding this bug.
- Changed alarm loop in mysqld.cc because shutdown didn't always succeed in Linux.
- Removed use of termbits from mysql.cc . This conflicted with glibc 2.0.
- Fixed some syntax errors for at least BSD and Linux.
- Fixed bug when doing a SELECT as superuser without a database.
- Fixed bug when doing SELECT with group calculation to outfile.

26.7.5 Changements de la version 3.20.14

- If one gives -p or --password option to mysql without an argument, the user is solicited for the password from the tty.
- Added default password from MYSQL_PWD (by Elmar Haneke).
- Added command kill to mysqladmin to kill a specific MySQL thread.
- Sometimes when doing a reconnect on a down connection this succeeded first on second try.
- Fixed adding an AUTO_INCREMENT key with ALTER_TABLE .
- AVG() gave too small value on some SELECT statements with GROUP BY and ORDER BY .
- Added new DATETIME type (by Giovanni Maruzzelli maruzz@matrice.it).
- Fixed that defining DONT_USE_DEFAULT_FIELDS works.
- Changed to use a thread to handle alarms instead of signals on Solaris to avoid race conditions.
- Fixed default length of signed numbers. (George Harvey georgeh@pinacl.co.uk .)
- Allow anything for CREATE INDEX .
- Add prezeros when packing numbers to DATE , TIME and TIMESTAMP .
- Fixed a bug in OR of multiple tables (gave empty set).
- Added many patches to MIT-pthreads. This fixes at least one lookup bug.

26.7.6 Changements de la version 3.20.13

- Added standard SQL-92 DATE and TIME types.
- Fixed bug in SELECT with AND - OR levels.
- Added support for Slovenian characters. The Contrib directory contains source and instructions for adding other character sets.
- Fixed bug with LIMIT and ORDER BY .
- Allow ORDER BY and GROUP BY on items that aren't in the SELECT list. (Thanks to Wim Bonis bonis@kiss.de , for pointing this out.)
- Allow setting of timestamp values in INSERT .
- Fixed bug with SELECT ... WHERE ... = NULL .
- Added changes for glibc 2.0. To get glibc to work, you should add the gibc-2.0-sigwait-patch before compiling glibc .
- Fixed bug in ALTER TABLE when changing a NOT NULL field to allow NULL values.
- Added some SQL-92 synonyms as field types to CREATE TABLE . CREATE TABLE now allows FLOAT(4) and FLOAT(8) to mean FLOAT and DOUBLE .
- New utility program mysqlaccess by Yves.Carlier@rug.ac.be . This program shows the access rights for a specific user and the grant rows that determine this grant.
- Added WHERE const op field (by bonis@kiss.de).

26.7.7 Changements de la version 3.20.11

- When using SELECT ... INTO OUTFILE , all temporary tables are ISAM instead of HEAP to allow big dumps.
- Changed date functions to be string functions. This fixed some ``funny'' side effects when sorting on dates.
- Extended ALTER TABLE for SQL-92 compliance.
- Some minor compatibility changes.
- Added --port and --socket options to all utility programs and mysqld .
- Fixed MIT-pthreads readdir_r() . Now mysqladmin create database and mysqladmin drop database should work.
- Changed MIT-pthreads to use our tempnam() . This should fix the ``sort aborted'' bug.
- Added sync of records count in sql_update . This fixed slow updates on first connection. (Thanks to Vaclav Bittner for the test.)

26.7.8 Changements de la version 3.20.10

- New insert type: INSERT INTO ... SELECT ...
- MEDIUMBLOB fixed.
- Fixed bug in ALTER TABLE and BLOB values.
- SELECT ... INTO OUTFILE now creates the file in the current database directory.
- DROP TABLE now can take a list of tables.
- Oracle synonym DESCRIBE (DESC).
- Changes to make_binary_distribution .
- Added some comments to installation instructions about configure 's C++ link test.
- Added --without-perl option to configure .
- Lots of small portability changes.

26.7.9 Changements de la version 3.20.9

- ALTER TABLE didn't copy null bit. As a result, fields that were allowed to have NULL values were always NULL .
- CREATE didn't take numbers as DEFAULT .
- Some compatibility changes for SunOS.
- Removed config.cache from old distribution.

26.7.10 Changements de la version 3.20.8

- Fixed bug with ALTER TABLE and multi-part keys.

26.7.11 Changements de la version 3.20.7

- New commands: ALTER TABLE , SELECT ... INTO OUTFILE and LOAD DATA INFILE .
- New function: NOW() .
- Added new field File_priv to mysql/user table.
- New script add_file_priv which adds the new field File_priv to the user table. This script must be executed if you want to use the new SELECT ... INTO and LOAD DATA INFILE ... commands with a version of MySQL earlier than 3.20.7.
- Fixed bug in locking code, which made lock_test.pl test fail.
- New files NEW and BUGS .
- Changed select_test.c and insert_test.c to include config.h .
- Added status command to mysqladmin for short logging.
- Increased maximum number of keys to 16 and maximum number of key parts to 15.
- Use of sub keys. A key may now be a prefix of a string field.
- Added −k option to mysqlshow , to get key information for a table.
- Added long options to mysqldump .

26.7.12 Changements de la version 3.20.6

- Portable to more systems because of MIT−pthreads, which will be used automatically if configure cannot find a −lpthreads library.
- Added GNU−style long options to almost all programs. Test with program −−help .
- Some shared library support for Linux.
- The FAQ is now in .texi format and is available in .html , .txt and .ps formats.
- Added new SQL function RAND([init]) .
- Changed sql_lex to handle \0 unquoted, but the client can't send the query through the C API, because it takes a str pointer. You must use mysql_real_query() to send the query.
- Added API function mysql_get_client_info() .
- mysqld now uses the N_MAX_KEY_LENGTH from nisam.h as the maximum allowable key length.
- The following now works:

  ```
  mysql> SELECT filter_nr,filter_nr FROM filter ORDER BY filter_nr;
  ```

  Previously, this resulted in the error: Column: 'filter_nr' in order clause is ambiguous .
- mysql now outputs '\0' , '\t' , '\n' and '\\' when encountering ASCII 0, tab, newline or '\' while writing tab−separated output. This is to allow printing of binary data in a portable format. To get the old behavior, use −r (or −−raw).
- Added german error messages (60 of 80 error messages translated).
- Added new API function mysql_fetch_lengths(MYSQL_RES *) , which returns an array of column lengths (of type uint).
- Fixed bug with IS NULL in WHERE clause.
- Changed the optimizer a little to get better results when searching on a key part.
- Added SELECT option STRAIGHT_JOIN to tell the optimizer that it should join tables in the given order.
- Added support for comments starting with '−−' in mysql.cc (Postgres syntax).
- You can have SELECT expressions and table columns in a SELECT which are not used in the group part. This makes it efficient to implement lookups. The column that is used should be a constant for each group because the value is calculated only once for the first row that is found for a group.

  ```
  mysql> SELECT id,lookup.text,SUM(*) FROM test,lookup
      ->        WHERE test.id=lookup.id GROUP BY id;
  ```
- Fixed bug in SUM(function) (could cause a core dump).
- Changed AUTO_INCREMENT placement in the SQL query:

  ```
  INSERT INTO table (auto_field) VALUES (0);
  ```

  inserted 0, but it should insert an AUTO_INCREMENT value.
- mysqlshow.c : Added number of records in table. Had to change the client code a little to fix this.
- mysql now allows doubled '' or "" within strings for embedded ' or " .
- New math functions: EXP() , LOG() , SQRT() , ROUND() , CEILING() .

26.7.13 Changements de la version 3.20.3

- The configure source now compiles a thread−free client library −lmysqlclient . This is the only library that needs to be linked with client applications. When using the binary releases, you must link with −lmysql −lmysys −ldbug −lmystrings as before.
- New readline library from bash−2.0 .
- LOTS of small changes to configure and makefiles (and related source).
- It should now be possible to compile in another directory using VPATH . Tested with GNU Make 3.75.
- safe_mysqld and mysql.server changed to be more compatible between the source and the binary releases.
- LIMIT now takes one or two numeric arguments. If one argument is given, it indicates the maximum number of rows in a result. If two arguments are given, the first argument indicates the offset of the first row to return, the second is the maximum number of rows. With this it's easy to do a poor man's next page/previous page WWW application.
- Changed name of SQL function FIELDS() to ELT() . Changed SQL function INTERVALL() to INTERVAL() .
- Made SHOW COLUMNS a synonym for SHOW FIELDS . Added compatibility syntax FRIEND KEY to CREATE TABLE . In MySQL, this creates a non−unique key on the given columns.
- Added CREATE INDEX and DROP INDEX as compatibility functions. In MySQL, CREATE INDEX only checks if the index exists and issues an error if it doesn't exist. DROP INDEX always succeeds.
- mysqladmin.c : added client version to version information.
- Fixed core dump bug in sql_acl (core on new connection).
- Removed host , user and db tables from database test in the distribution.
- FIELD_TYPE_CHAR can now be signed (−128 to 127) or unsigned (0 to 255) Previously, it was always unsigned.
- Bug fixes in CONCAT() and WEEKDAY() .
- Changed a lot of source to get mysqld to be compiled with SunPro compiler.
- SQL functions must now have a '(' immediately after the function name (no intervening space). For example, 'USER(' is regarded as beginning a function call, and 'USER (' is regarded as an identifier USER followed by a '(' , not as a function call.

26.7.14 Changements de la version 3.20.0

- The source distribution is done with configure and Automake. It will make porting much easier. The readline library is included in the distribution.
- Separate client compilation: the client code should be very easy to compile on systems which don't have threads.
- The old Perl interface code is automatically compiled and installed. Automatic compiling of DBD will follow when the new DBD code is ported.
- Dynamic language support: mysqld can now be started with Swedish or English (default) error messages.
- New functions: INSERT() , RTRIM() , LTRIM() and FORMAT() .
- mysqldump now works correctly for all field types (even AUTO_INCREMENT). The format for SHOW FIELDS FROM tbl_name is changed so the Type column contains information suitable for CREATE TABLE . In previous releases, some CREATE TABLE information had to be patched when re−creating tables.
- Some parser bugs from 3.19.5 (BLOB and TIMESTAMP) are corrected. TIMESTAMP now returns different date information depending on its create length.
- Changed parser to allow a database, table or field name to start with a number or '_' .
- All old C code from Unireg changed to C++ and cleaned up. This makes the daemon a little smaller and easier to understand.
- A lot of small bug fixes done.
- New INSTALL files (not final version) and some information regarding porting.

26.8 Changements de la version 3.19.x

La version 3.19 est plutôt vieille, et doit être évitée si possible. Les informations de cette section sont conservées à des fins d'archives.

26.8.1 Changements de la version 3.19.5

- Some new functions, some more optimization on joins.
- Should now compile clean on Linux (2.0.x).
- Added functions DATABASE() , USER() , POW() , LOG10() (needed for ODBC).
- In a WHERE with an ORDER BY on fields from only one table, the table is now preferred as first table in a multi−join.
- HAVING and IS NULL or IS NOT NULL now works.
- A group on one column and a sort on a group function (SUM() , AVG() ...) didn't work together. Fixed.
- mysqldump : Didn't send password to server.

26.8.2 Changements de la version 3.19.4

- Fixed horrible locking bug when inserting in one thread and reading in another thread.
- Fixed one−off decimal bug. 1.00 was output as 1.0.
- Added attribute 'Locked' to process list as information if a query is locked by another query.
- Fixed full magic timestamp. Timestamp length may now be 14, 12, 10, 8, 6, 4 or 2 bytes.
- Sort on some numeric functions could sort incorrectly on last number.
- IF(arg,syntax_error,syntax_error) crashed.
- Added functions CEILING() , ROUND() , EXP() , LOG() and SQRT() .
- Enhanced BETWEEN to handle strings.

26.8.3 Changements de la version 3.19.3

- Fixed SELECT with grouping on BLOB columns not to return incorrect BLOB info. Grouping, sorting and distinct on BLOB columns will not yet work as expected (probably it will group/sort by the first 7 characters in the BLOB). Grouping on formulas with a fixed string size (use MID() on a BLOB) should work.
- When doing a full join (no direct keys) on multiple tables with BLOB fields, the BLOB was garbage on output.
- Fixed DISTINCT with calculated columns.

26.9 Evolutions de InnoDB

Note : cette section n'est pas traduite en français.

26.9.1 MySQL/InnoDB−4.0.21, pas publiée

Functionality added or changed:

Bugs fixed:

- If you configure innodb_additional_mem_pool_size so small that InnoDB memory allocation spills over from it, then every 4 billionth spill may cause memory corruption. A symptom is a printout like below in the .err log. The workaround is to make innodb_additional_mem_pool_size big enough to hold all memory allocation. Use SHOW INNODB STATUS to determine that there is plenty of free space available in the additional mem pool, and the total allocated memory stays rather constant.

```
 InnoDB: Error: Mem area size is 0. Possibly a memory overrun of the
InnoDB: previous allocated area!
InnoDB: Apparent memory corruption: mem dump  len 500; hex
```

26.9.2 MySQL/InnoDB−4.1.2, pas publiée

Functionality added or changed:

- Do not assert in log0log.c , line 856 if ib_logfiles are too small for innodb_thread_concurrency . Instead, print instructions how to adjust my.cnf and call exit(1) .
- If MySQL tries to SELECT from an InnoDB table without setting any table locks, print a descriptive error message and assert; some subquery bugs were of this type.
- Added a missing space to the output format of SHOW INNODB STATUS ; reported by Jocelyn Fournier.
- Support multiple character sets. Note that tables created in other collations than latin1_swedish_ci cannot be accessed in MySQL/InnoDB 4.0.
- Allow a key path length in InnoDB to be up to 3,500 bytes; this is needed so that one can create an index on a column with 255 UTF−8 characters.

Bugs fixed:

- Improved portability to 64−bit platforms, especially Win64.
- Fixed an assertion failure when a purge of a table was not possible because of missing .ibd file.
- Fixed a bug: do not retrieve all columns in a table if we only need the 'ref' of the row (usually, the PRIMARY KEY) to calculate an ORDER BY . (Bug #1942)
- On Unix−like systems, obtain an exclusive advisory lock on InnoDB files, to prevent corruption when multiple instances of MySQL are running on the same set of data files. The Windows version of InnoDB already took a mandatory lock on the files. (Bug #3608)
- All bugfixes from InnoDB−4.0.17, InnoDB−4.0.18, InnoDB−4.0.19 and InnoDB−4.0.20.

26.9.3 MySQL/InnoDB−4.0.20, 18 mai 2004

Bugs fixed:

- Make LOCK TABLE aware of InnoDB row−level locks. (Bug #3299)
- Fixed race conditions in SHOW INNODB STATUS. (Bug #3596)

26.9.4 MySQL/InnoDB−4.0.19, 4 mai 2004

Functionality added or changed:

- Better error message when the server has to crash because the buffer pool is exhausted by the lock table or the adaptive hash index.
- Print always the count of pending pread() and pwrite() calls if there is a long semaphore wait.
- Improve space utilization when rows of 1,500 to 8,000 bytes are inserted in the order of the primary key.
- Remove potential buffer overflow errors by sending diagnostic output to stderr or files instead of stdout or fixed−size memory buffers. As a side effect, the output of SHOW INNODB STATUS will be written to a file Array every 15 seconds.

Bugs fixed:

- Fixed a bug: DROP DATABASE did not work if FOREIGN KEY references were defined within the database. (Bug #3058)
- Remove unnecessary files, functions and variables. Many of these were needed in the standalone version of InnoDB. Remove debug functions and variables from non−debug build.
- Add diagnostic code to analyze an assertion failure in ha_innodb.cc on line 2020 reported by a user. (Bug #2903)
- Fixed a bug: in a FOREIGN KEY , ON UPDATE CASCADE was not triggered if the update changed a string to another value identical in alphabetical ordering, e.g., 'abc' −> 'aBc' .
- Protect the reading of the latest foreign key error explanation buffer with a mutex; in theory, a race condition could cause SHOW INNODB STATUS print garbage characters after the error info.
- Fixed a bug: The row count and key cardinality estimate was grossly too small if each clustered index page only contained one record.
- Parse CONSTRAINT FOREIGN KEY correctly. (Bug #3332)
- Fixed a memory corruption bug on Windows. The bug is present in all InnoDB versions in Windows, but it depends on how the linker places a static array in srv0srv.c , whether the bug shows itself. 4 bytes were overwritten with a pointer to a statically allocated string 'get windows aio return value' .
- Fix a glitch reported by Philippe Lewicki on the general mailing list: do not print a warning to the .err log if read_key fails with a lock wait timeout error 146.
- Allow quotes to be embedded in strings in the private SQL parser of InnoDB, so that ''' can be used in InnoDB table and column names. Display quotes within identifiers properly.
- Debugging: Allow UNIV_SYNC_DEBUG to be disabled while UNIV_DEBUG is enabled.
- Debugging: Handle magic numbers in a more consistent way.

26.9.5 MySQL/InnoDB−4.0.18, 13 février 2004

- Do not allow dropping a table referenced by a FOREIGN KEY constraint, unless the user does SET FOREIGN_KEY_CHECKS=0 . The error message here is somewhat misleading ``Cannot delete or update a parent row...'', and must be changed in a future version 4.1.x.
- Make InnoDB to remember the CONSTRAINT name given by a user for a FOREIGN KEY .

- Change the print format of FOREIGN KEY constraints spanning multiple databases to `db_name`.`tbl_name` . But when parsing them, we must also accept `db_name.tbl_name` , because that was the output format in < 4.0.18.
- An optimization in locking: If AUTOCOMMIT=1 , then we do not need to make a plain SELECT set shared locks even on the SERIALIZABLE isolation level, because we know that the transaction is read-only. A read-only transaction can always be performed on the REPEATABLE READ level, and that does not endanger the serializability.
- Implement an automatic downgrade from >= 4.1.1 -> 4.0.18 if the user has not created tables in .ibd files or used other 4.1.x features. **Consult** the manual section on **multiple tablespaces** carefully if you want to downgrade!
- Fixed a bug: MySQL should not allow REPLACE to internally perform an UPDATE if the table is referenced by a FOREIGN KEY . The MySQL manual states that REPLACE must resolve a duplicate-key error semantically with DELETE (s) + INSERT , and not by an UPDATE . In versions < 4.0.18 and < 4.1.2, MySQL could resolve a duplicate key conflict in REPLACE by doing an UPDATE on the existing row, and FOREIGN KEY checks could behave in a semantically wrong way. (Bug #2418)
- Fixed a bug: generate FOREIGN KEY constraint identifiers locally for each table, in the form db_name/tbl_name_ibfk_number . If the user gives the constraint name explicitly, then remember it. These changes should ensure that foreign key id's in a slave are the same as in the master, and DROP FOREIGN KEY does not break replication. (Bug #2167)
- Fixed a bug: allow quoting of identifiers in InnoDB's FOREIGN KEY definitions with a backtick (`) and a double quote ("). You can now use also spaces in table and column names, if you quote the identifiers. (Bug #1725, Bug #2424)
- Fixed a bug: FOREIGN KEY ... ON UPDATE/DELETE NO ACTION must check the foreign key constraint, not ignore it. Since we do not have deferred constraints in InnoDB , this bugfix makes InnoDB to check NO ACTION constraints immediately, like it checks RESTRICT constraints.
- Fixed a bug: InnoDB crashed in RENAME TABLE if 'db_name.tbl_name' is shorter than 5 characters. (Bug #2689)
- Fixed a bug: in SHOW TABLE STATUS , InnoDB row count and index cardinality estimates wrapped around at 512 million in 32-bit computers. Note that unless MySQL is compiled with the BIG_TABLES option, they will still wrap around at 4 billion.
- Fixed a bug: If there was a UNIQUE secondary index, and NULL values in that unique index, then with the IS NULL predicate, InnoDB returned only the first matching row, though there can be many. This bug was introduced in 4.0.16. (Bug #2483)

26.9.6 MySQL/InnoDB−5.0.0, 24 décembre 2003

- **IMPORTANT NOTE: if you upgrade to InnoDB-4.1.1 or higher, you cannot downgrade to a version lower than 4.1.1 any more! That is because earlier versions of InnoDB are not aware of multiple tablespaces.**
- InnoDB in 5.0.0 is essentially the same as InnoDB-4.1.1 with the bug fixes of InnoDB-4.0.17 included.

26.9.7 MySQL/InnoDB−4.0.17, 17 décembre 2003

- Fixed a bug: if you created a column prefix secondary index and updated it so that the last characters in the column prefix were spaces, InnoDB would assert in row0upd.c , line 713. The same assertion failed if you updated a column in an ordinary secondary index so that the new value was alphabetically equivalent, but had a different length. This could happen, for example, in the utf-8 character set if you updated a letter to its accented or umlaut form.
- Fixed a bug: InnoDB could think that a secondary index record was not locked though it had been updated to an alphabetically equivalent value, e.g., 'abc' -> 'aBc'.
- Fixed a bug: if you updated a secondary index column to an alphabetically equivalent value, and rolled back your update, InnoDB failed to restore the field in the secondary index to its original value.
- There are still several outstanding non-critical bugs reported in the MySQL bugs database. Their fixing has been delayed, because resources were allocated to the 4.1.1 release.

26.9.8 MySQL/InnoDB−4.1.1, 4 décembre 2003

- **IMPORTANT NOTE: if you upgrade to InnoDB-4.1.1 or higher, you cannot downgrade to a version lower than 4.1.1 any more! That is because earlier versions of InnoDB are not aware of multiple tablespaces.**
- Multiple tablespaces now available for InnoDB. You can store each InnoDB type table and its indexes into a separate .ibd file into a MySQL database directory, into the same directory where the .frm file is stored.
- The MySQL query cache now works for InnoDB tables also if AUTOCOMMIT=0 , or the statements are enclosed inside BEGIN ... COMMIT .
- Reduced InnoDB memory consumption by a few megabytes if one sets the buffer pool size < 8 MB.
- You can use raw disk partitions also in Windows.

26.9.9 MySQL/InnoDB−4.0.16, 22 octobre 2003

- Fixed a bug: in contrary to what was said in the manual, in a locking read InnoDB set two record locks if a unique exact match search condition was used on a multi-column unique key. For a single column unique key it worked right.
- Fixed a bug: if one used the rename trick #sql... -> rsql... to recover a temporary table, InnoDB asserted in row_mysql_lock_data_dictionary() .
- There are several outstanding non-critical bugs reported in the MySQL bugs database. Their fixing has been delayed, because resources are allocated to the upcoming 4.1.1 release.

26.9.10 MySQL/InnoDB−3.23.58, 15 septembre 2003

- Fixed a bug: InnoDB could make the index page directory corrupt in the first B-tree page splits after mysqld startup. A symptom would be an assertion failure in page0page.c , in function page_dir_find_slot() .
- Fixed a bug: InnoDB could in rare cases return an extraneous row if a rollback, purge, and a SELECT coincided.
- Fixed a possible hang over the btr0sea.c latch if SELECT was used inside LOCK TABLES .
- Fixed a bug: If a single DELETE statement first managed to delete some rows and then failed in a FOREIGN KEY error or a Table is full error, MySQL did not roll back the whole SQL statement as it should.

26.9.11 MySQL/InnoDB−4.0.15, 10 septembre 2003

- Fixed a bug: if you updated a row so that the 8000 byte maximum length (without BLOB and TEXT) was exceeded, InnoDB simply removed the record from the clustered index. In a similar insert, InnoDB would leak reserved file space extents, which would only be freed at the next mysqld startup.
- Fixed a bug: if you used big BLOB values, and your log files were relatively small, InnoDB could in a big BLOB operation temporarily write over the log produced after the latest checkpoint. If InnoDB would crash at that moment, then the crash recovery would fail, because InnoDB would not be able to scan the log even up to the latest checkpoint. Starting from this version, InnoDB tries to ensure the latest checkpoint is young enough. If that is not possible, InnoDB prints a warning to the .err log of MySQL and advises you to make the log files bigger.
- Fixed a bug: setting innodb_fast_shutdown=0 had no effect.
- Fixed a bug introduced in 4.0.13: if a CREATE TABLE ended in a comment, that could cause a memory overrun.
- Fixed a bug: If InnoDB printed Operating system error number .. in a file operation to the .err log in Windows, the error number explanation was wrong. Workaround: look at section 13.2 of http://www.innodb.com/ibman.php about Windows error numbers.
- Fixed a bug: If you created a column prefix PRIMARY KEY like in t(a CHAR(200), PRIMARY KEY (a(10))) on a fixed-length CHAR column, InnoDB would crash even in a simple SELECT . C CHECK TABLE would report the table as corrupt, also in the case where the created key was not PRIMARY .

26.9.12 MySQL/InnoDB−4.0.14, 22 juillet 2003

- InnoDB now supports the SAVEPOINT and ROLLBACK TO SAVEPOINT SQL statements. See http://www.innodb.com/ibman.php#Savepoints for the syntax.
- You can now create column prefix keys like in CREATE TABLE t (a BLOB, INDEX (a(10))) .
- You can also use O_DIRECT as the innodb_flush_method on the latest versions of Linux and FreeBSD. Beware of possible bugs in those operating systems, though.
- Fixed the checksum calculation of data pages. Previously most OS file system corruption went unnoticed. Note that if you downgrade from version >= 4.0.14 to an earlier version < 4.0.14 then in the first startup(s) InnoDB will print warnings:

  ```
   InnoDB: Warning: an inconsistent page in the doublewrite buffer
  InnoDB: space id 2552202359 page number 8245, 127'th page in dblwr buf.
  ```

  but that is not dangerous and can be ignored.
- Modified the buffer pool replacement algorithm so that it tries to flush modified pages if there are no replaceable pages in the last 10 % of the LRU list. This can reduce disk i/o if the workload is a mixture of reads and writes.
- The buffer pool checkpoint flush algorithm now tries to flush also close neighbors of the page at the end of the flush list. This can speed up database shutdown, and can also speed up disk writes if InnoDB log files are very small compared to the buffer pool size.
- In 4.0.13 we made SHOW INNODB STATUS to print detailed info on the latest UNIQUE KEY error, but storing that info could slow down REPLACE significantly. We no longer store or print the info.
- Fixed a bug: SET FOREIGN_KEY_CHECKS=0 was not replicated properly in the MySQL replication. The fix will not be backported to 3.23.
- Fixed a bug: the parameter innodb_max_dirty_pages_pct forgot to take into account the free pages in the buffer pool. This could lead to excessive flushing even though there were lots of free pages in the buffer pool. Workaround: SET GLOBAL innodb_max_dirty_pages_pct = 100 .
- Fixed a bug: if there were big index scans then a file read request could starve and InnoDB could assert because of a very long semaphore wait.
- Fixed a bug: if AUTOCOMMIT=1 then inside LOCK TABLES MySQL failed to do the commit after an updating SQL statement if binlogging was not on, and for SELECT statements did not commit regardless of binlogging state.
- Fixed a bug: InnoDB could make the index page directory corrupt in the first B-tree page splits after a mysqld startup. A symptom would be an assertion in page0page.c, in function page_dir_find_slot().
- Fixed a bug: if in a FOREIGN KEY with an UPDATE CASCADE clause the parent column was of a different internal storage length than the child column, then a cascaded update would make the column length wrong in the child table and corrupt the child table. Because of MySQL's 'silent column specification changes' a fixed-length CHAR column can change internally to a VARCHAR and cause this error.
- Fixed a bug: if a non- latin1 character set was used and if in a FOREIGN KEY the parent column was of a different internal storage length than the child column, then all inserts to the child table would fail in a foreign key error.
- Fixed a bug: InnoDB could complain that it cannot find the clustered index record, or in rare cases return an extraneous row if a rollback, purge, and a SELECT coincided.
- Fixed a possible hang over the btr0sea.c latch if SELECT was used inside LOCK TABLES .
- Fixed a bug: contrary to what the release note of 4.0.13 said, the group commit still did not work if the MySQL binlogging was on.
- Fixed a bug: os_event_wait() did not work properly in Unix, which might have caused starvation in various log operations.
- Fixed a bug: if a single DELETE statement first managed to delete some rows and then failed in a FOREIGN KEY error or a 'Table is full error', MySQL did not roll back the whole SQL statement as it should, and also wrote the failed statement to the binlog, reporting there a non-zero error_code.
- Fixed a bug: the maximum allowed number of columns in a table is 1000, but InnoDB did not check that limit in CREATE TABLE , and a subsequent INSERT or SELECT from that table could cause an assertion.

26.9.13 MySQL/InnoDB−3.23.57, 20 juin 2003

- Changed the default value of innodb_flush_log_at_trx_commit from 0 to 1. If you have not specified it explicitly in your my.cnf , and your application runs much slower with this new release, it is because the value 1 causes a log flush to disk at each transaction commit.
- Fixed a bug: InnoDB forgot to call pthread_mutex_destroy() when a table was dropped. That could cause memory leakage on FreeBSD and other non-Linux Unixes.
- Fixed a bug: MySQL could erroneously return 'Empty set' if InnoDB estimated an index range size to 0 records though the range was not empty; MySQL also failed to do the next-key locking in the case of an empty index range.
- Fixed a bug: GROUP BY and DISTINCT could treat NULL values inequal.

26.9.14 MySQL/InnoDB−4.0.13, 20 mai 2003

- InnoDB now supports ALTER TABLE DROP FOREIGN KEY . You have to use SHOW CREATE TABLE to find the internally generated foreign key ID when you want to drop a foreign key.
- SHOW INNODB STATUS now prints detailed information of the latest detected FOREIGN KEY and UNIQUE KEY errors. If you do not understand why InnoDB gives the error 150 from a CREATE TABLE , you can use this statement to study the reason.
- ANALYZE TABLE now works also for InnoDB type tables. It makes 10 random dives to each of the index trees and updates index cardinality estimates accordingly. Note that since it is only an estimate, repeated runs of ANALYZE TABLE may produce

different numbers. MySQL uses index cardinality estimates only in join optimization. If some join is not optimized in the right way, you may try using ANALYZE TABLE .
- InnoDB group commit capability now works also when MySQL binlogging is switched on. There have to be > 2 client threads for the group commit to become active.
- Changed the default value of innodb_flush_log_at_trx_commit from 0 to 1. If you have not specified it explicitly in your my.cnf , and your application runs much slower with this new release, it is because the value 1 causes a log flush to disk at each transaction commit.
- Added a new global settable MySQL system variable innodb_max_dirty_pages_pct . It is an integer in the range 0 − 100. The default is 90. The main thread in InnoDB tries to flush pages from the buffer pool so that at most this many percents are not yet flushed at any time.
- If innodb_force_recovery=6 , do not let InnoDB do repair of corrupt pages based on the doublewrite buffer.
- InnoDB start−up now happens faster because it does not set the memory in the buffer pool to zero.
- Fixed a bug: The InnoDB parser for FOREIGN KEY definitions was confused by the keywords 'foreign key' inside MySQL comments.
- Fixed a bug: If you dropped a table to which there was a FOREIGN KEY reference, and later created the same table with non−matching column types, InnoDB could assert in dict0load.c , in function dict_load_table() .
- Fixed a bug: GROUP BY and DISTINCT could treat NULL values as not equal. MySQL also failed to do the next−key locking in the case of an empty index range.
- Fixed a bug: Do not commit the current transaction when a MyISAM table is updated; this also makes CREATE TABLE not to commit an InnoDB transaction, even when binlogging is enabled.
- Fixed a bug: We did not allow ON DELETE SET NULL to modify the same table where the delete was made; we can allow it because that cannot produce infinite loops in cascaded operations.
- Fixed a bug: Allow HANDLER PREV and NEXT also after positioning the cursor with a unique search on the primary key.
- Fixed a bug: If MIN() or MAX() resulted in a deadlock or a lock wait timeout, MySQL did not return an error, but returned NULL as the function value.
- Fixed a bug: InnoDB forgot to call pthread_mutex_destroy() when a table was dropped. That could cause memory leakage on FreeBSD and other non−Linux Unix systems.

26.9.15 MySQL/InnoDB−4.1.0, 3 avril 2003

- InnoDB now supports up to 64 GB of buffer pool memory in a Windows 32−bit Intel computer. This is possible because InnoDB can use the AWE extension of Windows to address memory over the 4 GB limit of a 32−bit process. A new startup variable innodb_buffer_pool_awe_mem_mb enables AWE and sets the size of the buffer pool in megabytes.
- Reduced the size of buffer headers and the lock table. InnoDB uses 2 % less memory.

26.9.16 MySQL/InnoDB−3.23.56, 17 mars 2003

- Fixed a major bug in InnoDB query optimization: queries of type SELECT ... WHERE indexcolumn < x and SELECT ... WHERE indexcolumn > x could cause a table scan even if the selectivity would have been very good.
- Fixed a potential bug if MySQL calls store_lock with TL_IGNORE in the middle of a query.

26.9.17 MySQL/InnoDB−4.0.12, 18 mars 2003

- In crash recovery InnoDB now prints the progress in percents of a transaction rollback.
- Fixed a bug/feature: if your application program used mysql_use_result(), and used >= 2 connections to send SQL queries, it could deadlock on the adaptive hash S−latch in btr0sea.c. Now mysqld releases the S−latch whenever it passes data from a SELECT to the client.
- Fixed a bug: MySQL could erroneously return 'Empty set' if InnoDB estimated an index range size to 0 records though the range was not empty; MySQL also failed to do the next−key locking in the case of an empty index range.

26.9.18 MySQL/InnoDB−4.0.11, 25 février 2003

- Fixed a bug introduced in 4.0.10: SELECT ... FROM ... ORDER BY ... DESC could hang in an infinite loop.
- An outstanding bug: SET FOREIGN_KEY_CHECKS=0 is not replicated properly in the MySQL replication.

26.9.19 MySQL/InnoDB−4.0.10, 4 février 2003

- In INSERT INTO t1 SELECT ... FROM t2 WHERE ... MySQL previously set a table level read lock on t2. This lock is now removed.
- Increased SHOW INNODB STATUS max printed length to 200 KB.
- Fixed a major bug in InnoDB query optimization: queries of type SELECT ... WHERE indexcolumn < x and SELECT ... WHERE indexcolumn > x could cause a table scan even if the selectivity would have been very good.
- Fixed a bug: purge could cause a hang in a BLOB table where the primary key index tree was of height 1. Symptom: semaphore waits caused by an X−latch set in btr_free_externally_stored_field().
- Fixed a bug: using InnoDB HANDLER commands on a fresh handle crashed mysqld in ha_innobase::change_active_index().
- Fixed a bug: if MySQL estimated a query in the middle of a SELECT statement, InnoDB could hang on the adaptive hash index latch in btr0sea.c.
- Fixed a bug: InnoDB could report table corruption and assert in page_dir_find_owner_slot() if an adaptive hash index search coincided with purge or an insert.
- Fixed a bug: some file system snapshot tool in Windows 2000 could cause an InnoDB file write to fail with error 33 ERROR_LOCK_VIOLATION. In synchronous writes InnoDB now retries the write 100 times at 1 second intervals.
- Fixed a bug: REPLACE INTO t1 SELECT ... did not work if t1 has an auto−inc column.
- An outstanding bug: SET FOREIGN_KEY_CHECKS=0 is not replicated properly in the MySQL replication.

26.9.20 MySQL/InnoDB−3.23.55, 24 janvier 2003

- In INSERT INTO t1 SELECT ... FROM t2 WHERE ... MySQL previously set a table level read lock on t2. This lock is now removed.
- Fixed a bug: if the combined size of InnoDB log files was >= 2 GB in a 32−bit computer, InnoDB would write log in a wrong position. That could make crash recovery and InnoDB Hot Backup to fail in log scan.
- Fixed a bug: index cursor restoration could theoretically fail.
- Fixed a bug: an assertion in btr0sea.c, in function btr_search_info_update_slow could theoretically fail in a race of 3 threads.
- Fixed a bug: purge could cause a hang in a BLOB table where the primary key index tree was of height 1. Symptom: semaphore waits caused by an X−latch set in btr_free_externally_stored_field().
- Fixed a bug: if MySQL estimated a query in the middle of a SELECT statement, InnoDB could hang on the adaptive hash index latch in btr0sea.c.
- Fixed a bug: InnoDB could report table corruption and assert in page_dir_find_owner_slot() if an adaptive hash index search coincided with purge or an insert.
- Fixed a bug: some file system snapshot tool in Windows 2000 could cause an InnoDB file write to fail with error 33 ERROR_LOCK_VIOLATION. In synchronous writes InnoDB now retries the write 100 times at 1 second intervals.
- An outstanding bug: SET FOREIGN_KEY_CHECKS=0 is not replicated properly in the MySQL replication. The fix will appear in 4.0.11 and will probably not be backported to 3.23.
- Fixed bug in InnoDB page0cur.c file in function page_cur_search_with_match which caused InnoDB to remain on the same page forever. This bug is evident only in tables with more than one page.

26.9.21 MySQL/InnoDB−4.0.9, 14 janvier 2003

- Removed the warning message: 'InnoDB: Out of memory in additional memory pool.'
- Fixed a bug: if the combined size of InnoDB log files was >= 2 GB in a 32−bit computer, InnoDB would write log in a wrong position. That could make crash recovery and InnoDB Hot Backup to fail.
- Fixed a bug: index cursor restoration could theoretically fail.

26.9.22 MySQL/InnoDB−4.0.8, 7 janvier 2003

- InnoDB now supports also FOREIGN KEY (...) REFERENCES ...(...) [ON UPDATE CASCADE | ON UPDATE SET NULL | ON UPDATE RESTRICT | ON UPDATE NO ACTION].
- Tables and indexes now reserve 4 % less space in the tablespace. Also existing tables reserve less space. By upgrading to 4.0.8 you will see more free space in "InnoDB free" in SHOW TABLE STATUS.
- Fixed bugs: updating the PRIMARY KEY of a row would generate a foreign key error on all FOREIGN KEYs which referenced secondary keys of the row to be updated. Also, if a referencing FOREIGN KEY constraint only referenced the first columns in an index, and there were more columns in that index, updating the additional columns generated a foreign key error.
- Fixed a bug: if an index contains some column twice, and that column is updated, the table will become corrupt. From now on InnoDB prevents creation of such indexes.
- Fixed a bug: removed superfluous error 149 and 150 printouts from the .err log when a locking SELECT caused a deadlock or a lock wait timeout.
- Fixed a bug: an assertion in btr0sea.c, in function btr_search_info_update_slow could theoretically fail in a race of 3 threads.
- Fixed a bug: one could not switch a session transaction isolation level back to REPEATABLE READ after setting it to something else.

26.9.23 MySQL/InnoDB−4.0.7, 26 décembre 2002

- InnoDB in 4.0.7 is essentially the same as in 4.0.6.

26.9.24 MySQL/InnoDB−4.0.6, 19 décembre 2002

- Since innodb_log_arch_dir has no relevance under MySQL, there is no need to specify it any more in my.cnf.
- LOAD DATA INFILE in AUTOCOMMIT=1 mode no longer does implicit commits for each 1 MB of written binlog.
- Fixed a bug introduced in 4.0.4: LOCK TABLES ... READ LOCAL should not set row locks on the rows read. This caused deadlocks and lock wait timeouts in mysqldump.
- Fixed two bugs introduced in 4.0.4: in AUTO_INCREMENT, REPLACE could cause the counter to be left 1 too low. A deadlock or a lock wait timeout could cause the same problem.
- Fixed a bug: TRUNCATE on a TEMPORARY table crashed InnoDB.
- Fixed a bug introduced in 4.0.5: if binlogging was not switched on, INSERT INTO ... SELECT ... or CREATE TABLE ... SELECT ... could cause InnoDB to hang on a semaphore created in btr0sea.c, line 128. Workaround: switch binlogging on.
- Fixed a bug: in replication issuing SLAVE STOP in the middle of a multi−statement transaction could cause that SLAVE START would only perform a part of the transaction. A similar error could occur if the slave crashed and was restarted.

26.9.25 MySQL/InnoDB−3.23.54, 12 décembre 2003

- Fixed a bug: the InnoDB range estimator greatly exaggerated the size of a short index range if the paths to the endpoints of the range in the index tree happened to branch already in the root. This could cause unnecessary table scans in SQL queries.
- Fixed a bug: ORDER BY could fail if you had not created a primary key to a table, but had defined several indexes of which at least one was a UNIQUE index with all its columns declared as NOT NULL.
- Fixed a bug: a lock wait timeout in connection with ON DELETE CASCADE could cause corruption in indexes.
- Fixed a bug: if a SELECT was done with a unique key from a primary index, and the search matched to a delete−marked record, InnoDB could erroneously return the NEXT record.
- Fixed a bug introduced in 3.23.53: LOCK TABLES ... READ LOCAL should not set row locks on the rows read. This caused deadlocks and lock wait timeouts in mysqldump.
- Fixed a bug: if an index contains some column twice, and that column is updated, the table will become corrupt. From now on InnoDB prevents creation of such indexes.

26.9.26 MySQL/InnoDB−4.0.5, 18 novembre 2002

- InnoDB now supports also transaction isolation levels READ COMMITTED and READ UNCOMMITTED. READ COMMITTED more closely emulates Oracle and makes porting of applications from Oracle to MySQL easier.
- Deadlock resolution is now selective: we try to pick as victims transactions with less modified or inserted rows.
- FOREIGN KEY definitions are now aware of the lower_case_table_names setting in my.cnf.
- SHOW CREATE TABLE does not output the database name to a FOREIGN KEY definition if the referred table is in the same database as the table.
- InnoDB does a consistency check to most index pages before writing them to a datafile.
- If you set innodb_force_recovery > 0, InnoDB tries to jump over corrupt index records and pages when doing SELECT * FROM table. This helps in dumping.
- InnoDB now again uses asynchronous unbuffered I/O in Windows 2000 and XP; only unbuffered simulated async I/O in NT, 95/98/ME.
- Fixed a bug: the InnoDB range estimator greatly exaggerated the size of a short index range if the paths to the endpoints of the range in the index tree happened to branch already in the root. This could cause unnecessary table scans in SQL queries. The fix will also be backported to 3.23.54.
- Fixed a bug present in 3.23.52, 4.0.3, 4.0.4: InnoDB startup could take very long or even crash on some Windows 95/98/ME computers.
- Fixed a bug: the AUTO−INC lock was held to the end of the transaction if it was granted after a lock wait. This could cause unnecessary deadlocks.
- Fixed a bug: if SHOW INNODB STATUS, innodb_monitor, or innodb_lock_monitor had to print several hundred transactions in one report, and the output became truncated, InnoDB would hang, printing to the error log many waits for a mutex created at srv0srv.c, line 1621.
- Fixed a bug: SHOW INNODB STATUS on Unix always reported average file read size as 0 bytes.
- Fixed a potential bug in 4.0.4: InnoDB now does ORDER BY ... DESC like MyISAM.
- Fixed a bug: DROP TABLE could cause crash or a hang if there was a rollback concurrently running on the table. The fix will only be backported to 3.23 if this appears a real problem for users.
- Fixed a bug: ORDER BY could fail if you had not created a primary key to a table, but had defined several indexes of which at least one was a UNIQUE index with all its columns declared as NOT NULL.
- Fixed a bug: a lock wait timeout in connection with ON DELETE CASCADE could cause corruption in indexes.
- Fixed a bug: if a SELECT was done with a unique key from a primary index, and the search matched to a delete−marked record, InnoDB could return the NEXT record.
- Outstanding bugs: in 4.0.4 two bugs were introduced to AUTO_INCREMENT. REPLACE can cause the counter to be left 1 too low. A deadlock or a lock wait timeout can cause the same problem. These will be fixed in 4.0.6.

26.9.27 MySQL/InnoDB−3.23.53, 9 octobre 2002

- We again use unbuffered disk I/O to datafiles in Windows. Windows XP and Windows 2000 read performance seems to be very poor with normal I/O.
- Tuned range estimator so that index range scans are preferred over full index scans.
- Allow dropping and creating a table even if innodb_force_recovery is set. One can use this to drop a table which would cause a crash in rollback or purge, or if a failed table import causes a runaway rollback in recovery.
- Fixed a bug present in 3.23.52, 4.0.3, 4.0.4: InnoDB startup could take very long or even crash on some Windows 95/98/ME computers.
- Fixed a bug: fast shutdown (which is the default) sometimes was slowed down by purge and insert buffer merge.
- Fixed a bug: doing a big SELECT from a table where no rows were visible in a consistent read could cause a very long (> 600 seconds) semaphore wait in btr0cur.c line 310.
- Fixed a bug: the AUTO−INC lock was held to the end of the transaction if it was granted after a lock wait. This could cause unnecessary deadlocks.
- Fixed a bug: if you created a temporary table inside LOCK TABLES, and used that temporary table, that caused an assertion failure in ha_innobase.cc.
- Fixed a bug: if SHOW INNODB STATUS, innodb_monitor, or innodb_lock_monitor had to print several hundred transactions in one report, and the output became truncated, InnoDB would hang, printing to the error log many waits for a mutex created at srv0srv.c, line 1621.
- Fixed a bug: SHOW INNODB STATUS on Unix always reported average file read size as 0 bytes.

26.9.28 MySQL/InnoDB−4.0.4, 2 octobre 2002

- We again use unbuffered disk I/O in Windows. Windows XP and Windows 2000 read performance seems to be very poor with normal I/O.
- Increased the max key length of InnoDB tables from 500 to 1024 bytes.
- Increased the table comment field in SHOW TABLE STATUS so that up to 16000 characters of foreign key definitions can be printed there.
- The auto−increment counter is no longer incremented if an insert of a row immediately fails in an error.
- Allow dropping and creating a table even if innodb_force_recovery is set. One can use this to drop a table which would cause a crash in rollback or purge, or if a failed table import causes a runaway rollback in recovery.
- Fixed a bug: Using ORDER BY primarykey DESC in 4.0.3 causes an assertion failure in btr0pcur.c, line 203.
- Fixed a bug: fast shutdown (which is the default) sometimes was slowed down by purge and insert buffer merge.
- Fixed a bug: doing a big SELECT from a table where no rows were visible in a consistent read could cause a very long (> 600 seconds) semaphore wait in btr0cur.c line 310.
- Fixed a bug: if the MySQL query cache was used, it did not get invalidated by a modification done by ON DELETE CASCADE or ...SET NULL.
- Fixed a bug: if you created a temporary table inside LOCK TABLES, and used that temporary table, that caused an assertion failure in ha_innodb.cc.
- Fixed a bug: if you set innodb_flush_log_at_trx_commit to 1, SHOW VARIABLES would show its value as 16 million.

26.9.29 MySQL/InnoDB−4.0.3, 28 août 2002

- Removed unnecessary deadlocks when inserts have to wait for a locking read, update, or delete to release its next−key lock.
- The MySQL HANDLER SQL commands now work also for InnoDB type tables. InnoDB does the HANDLER reads always as consistent reads. HANDLER is a direct access path to read individual indexes of tables. In some cases HANDLER can be used as a substitute of server−side cursors.
- Fixed a bug in 4.0.2: even a simple insert could crash the AIX version.
- Fixed a bug: if you used in a table name characters whose code is > 127, in DROP TABLE InnoDB could assert on line 155 of pars0sym.c.
- Compilation from source now provides a working version both on HP−UX−11 and HP−UX−10.20. The source of 4.0.2 worked only on 11, and the source of 3.23.52 only on 10.20.

- Fixed a bug: if compiled on 64-bit Solaris, InnoDB produced a bus error at startup.

26.9.30 MySQL/InnoDB−3.23.52, 16 août 2002

- The feature set of 3.23 will be frozen from this version on. New features will go the 4.0 branch, and only bug fixes will be made to the 3.23 branch.
- Many CPU-bound join queries now run faster. On Windows also many other CPU-bound queries run faster.
- A new SQL command SHOW INNODB STATUS returns the output of the InnoDB Monitor to the client. The InnoDB Monitor now prints detailed information on the latest detected deadlock.
- InnoDB made the SQL query optimizer to avoid too much index-only range scans and choose full table scans instead. This is now fixed.
- BEGIN and COMMIT are now added in the binlog around transactions. The MySQL replication now respects transaction borders: a user will no longer see half transactions in replication slaves.
- A replication slave now prints in crash recovery the last master binlog position it was able to recover to.
- A new setting innodb_flush_log_at_trx_commit=2 makes InnoDB to write the log to the operating system file cache at each commit. This is almost as fast as the setting innodb_flush_log_at_trx_commit=0, and the setting 2 also has the nice feature that in a crash where the operating system does not crash, no committed transaction is lost. If the operating system crashes or there is a power outage, then the setting 2 is no safer than the setting 0.
- Added checksum fields to log blocks.
- SET FOREIGN_KEY_CHECKS=0 helps in importing tables in an arbitrary order which does not respect the foreign key rules.
- SET UNIQUE_CHECKS=0 speeds up table imports into InnoDB if you have UNIQUE constraints on secondary indexes. This flag should be used only if you are certain that the input records contain no UNIQUE constraint violations.
- SHOW TABLE STATUS now lists also possible ON DELETE CASCADE or ON DELETE SET NULL in the comment field of the table.
- When CHECK TABLE is run on any InnoDB type table, it now checks also the adaptive hash index for all tables.
- If you defined ON DELETE CASCADE or SET NULL and updated the referenced key in the parent row, InnoDB deleted or updated the child row. This is now changed to conform to SQL-92: you get the error 'Cannot delete parent row'.
- Improved the auto-increment algorithm: now the first insert or SHOW TABLE STATUS initializes the auto-increment counter for the table. This removes almost all surprising deadlocks caused by SHOW TABLE STATUS.
- Aligned some buffers used in reading and writing to datafiles. This allows using unbuffered raw devices as datafiles in Linux.
- Fixed a bug: If you updated the primary key of a table so that only the case of characters changed, that could cause assertion failures, mostly in page0page.ic line 515.
- Fixed a bug: If you delete or update a row referenced in a foreign key constraint and the foreign key check has to wait for a lock, then the check may report an erroneous result. This affects also the ON DELETE... operation.
- Fixed a bug: A deadlock or a lock wait timeout error in InnoDB causes InnoDB to roll back the whole transaction, but MySQL could still write the earlier SQL statements to the binlog, even though InnoDB rolled them back. This could, for example, cause replicated databases to get out-of-sync.
- Fixed a bug: If the database happened to crash in the middle of a commit, then the recovery might leak tablespace pages.
- Fixed a bug: If you specified a non-latin1 character set in my.cnf, then, in contrary to what is stated in the manual, in a foreign key constraint a string type column had to have the same length specification in the referencing table and the referenced table.
- Fixed a bug: DROP TABLE or DROP DATABASE could fail if there simultaneously was a CREATE TABLE running.
- Fixed a bug: If you configured the buffer pool bigger than 2 GB in a 32-bit computer, InnoDB would assert in buf0buf.ic line 214.
- Fixed a bug: on 64-bit computers updating rows which contained the SQL NULL in some column could cause the undo log and the ordinary log to become corrupt.
- Fixed a bug: innodb_log_monitor caused a hang if it suppressed lock prints for a page.
- Fixed a bug: in the HP-UX-10.20 version mutexes would leak and cause race conditions and crashes in any part of InnoDB code.
- Fixed a bug: if you ran in the AUTOCOMMIT mode, executed a SELECT, and immediately after that a RENAME TABLE, then RENAME would fail and MySQL would complain about error 1192.
- Fixed a bug: if compiled on 64-bit Solaris, InnoDB produced a bus error at startup.

26.9.31 MySQL/InnoDB−4.0.2, 10 juillet 2002

- InnoDB is essentially the same as InnoDB-3.23.51.
- If no innodb_data_file_path is specified, InnoDB at the database creation now creates a 10 MB auto-extending datafile ibdata1 to the datadir of MySQL. In 4.0.1 the file was 64 MB and not auto-extending.

26.9.32 MySQL/InnoDB−3.23.51, 12 juin 2002

- Fixed a bug: a join could result in a seg fault in copying of a BLOB or TEXT column if some of the BLOB or TEXT columns in the table contained SQL NULL values.
- Fixed a bug: if you added self-referential foreign key constraints with ON DELETE CASCADE to tables and a row deletion caused InnoDB to attempt the deletion of the same row twice because of a cascading delete, then you got an assertion failure.
- Fixed a bug: if you use MySQL 'user level locks' and close a connection, then InnoDB may assert in ha_innobase.cc, line 302.

26.9.33 MySQL/InnoDB−3.23.50, 23 avril 2002

- InnoDB now supports an auto-extending last datafile. You do not need to preallocate the whole datafile at the database startup.
- Made several changes to facilitate the use of the InnoDB Hot Backup tool. It is a separate non-free tool you can use to take online backups of your database without shutting down the server or setting any locks.
- If you want to run the InnoDB Hot Backup tool on an auto-extending datafile you have to upgrade it to version ibbackup-0.35.
- The log scan phase in crash recovery will now run much faster.
- Starting from this server version, the hot backup tool truncates unused ends in the backup InnoDB datafiles.
- To allow the hot backup tool to work, on Windows we no longer use unbuffered I/O or native async I/O; instead we use the same simulated async I/O as on Unix.
- You can now define the ON DELETE CASCADE or ON DELETE SET NULL clause on foreign keys.
- FOREIGN KEY constraints now survive ALTER TABLE and CREATE INDEX.
- We suppress the FOREIGN KEY check if any of the column values in the foreign key or referenced key to be checked is the SQL NULL. This is compatible with Oracle, for example.
- SHOW CREATE TABLE now lists also foreign key constraints. Also mysqldump no longer forgets about foreign keys in table definitions.
- You can now add a new foreign key constraint with ALTER TABLE ... ADD CONSTRAINT FOREIGN KEY (...) REFERENCES ... (...).

- FOREIGN KEY definitions now allow backquotes around table and column names.
- MySQL command SET TRANSACTION ISOLATION LEVEL ... has now the following effect on InnoDB tables: if a transaction is defined as SERIALIZABLE then InnoDB conceptually adds LOCK IN SHARE MODE to all consistent reads. If a transaction is defined to have any other isolation level, then InnoDB obeys its default locking strategy which is REPEATABLE READ.
- SHOW TABLE STATUS no longer sets an x-lock at the end of an auto-increment index if the auto-increment counter has already been initialized. This removes in almost all cases the surprising deadlocks caused by SHOW TABLE STATUS.
- Fixed a bug: in a CREATE TABLE statement the string 'foreign' followed by a non-space character confused the FOREIGN KEY parser and caused table creation to fail with errno 150.

26.9.34 MySQL/InnoDB−3.23.49, 17 février 2002

- Fixed a bug: if you called DROP DATABASE for a database on which there simultaneously were running queries, the MySQL server could crash or hang. Crashes fixed, but a full fix has to wait some changes in the MySQL layer of code.
- Fixed a bug: on Windows one had to put the database name in lower case for DROP DATABASE to work. Fixed in 3.23.49: case no longer matters on Windows. On Unix the database name remains case-sensitive.
- Fixed a bug: if one defined a non-latin1 character set as the default character set, then definition of foreign key constraints could fail in an assertion failure in dict0crea.c, reporting an internal error 17.

26.9.35 MySQL/InnoDB−3.23.48, 9 février 2002

- Tuned the SQL optimizer to favor more often index searches over table scans.
- Fixed a performance problem when several large SELECT queries are run concurrently on a multiprocessor Linux computer. Large CPU-bound SELECT queries will now also generally run faster on all platforms.
- If MySQL binlogging is used, InnoDB now prints after crash recovery the latest MySQL binlog file name and the position in that file (= byte offset) InnoDB was able to recover to. This is useful, for example, when resynchronizing a master and a slave database in replication.
- Added better error messages to help in installation problems.
- One can now recover also MySQL temporary tables which have become orphaned inside the InnoDB tablespace.
- InnoDB now prevents a FOREIGN KEY declaration where the signedness is not the same in the referencing and referenced integer columns.
- Fixed a bug: calling SHOW CREATE TABLE or SHOW TABLE STATUS could cause memory corruption and make mysqld to crash. Especially at risk was mysqldump, because it calls frequently SHOW CREATE TABLE.
- Fixed a bug: if on Unix you did an ALTER TABLE to an InnoDB table and simultaneously did queries to it, mysqld could crash with an assertion failure in row0row.c, line 474.
- Fixed a bug: if inserts to several tables containing an auto-inc column were wrapped inside one LOCK TABLES, InnoDB asserted in lock0lock.c.
- In 3.23.47 we allowed several NULLS in a UNIQUE secondary index. But CHECK TABLE was not relaxed: it reports the table as corrupt. CHECK TABLE no longer complains in this situation.
- Fixed a bug: on Sparc and other high-endian processors SHOW VARIABLES showed innodb_flush_log_at_trx_commit and other boolean-valued startup parameters always OFF even if they were switched on.
- Fixed a bug: if you ran mysqld-max-nt as a service on Windows NT/2000, the service shutdown did not always wait long enough for the InnoDB shutdown to finish.

26.9.36 MySQL/InnoDB−3.23.47, 28 décembre 2001

- Recovery happens now faster, especially in a lightly loaded system, because background checkpointing has been made more frequent.
- InnoDB allows now several similar key values in a UNIQUE secondary index if those values contain SQL NULLs. Thus the convention is now the same as in MyISAM tables.
- InnoDB gives a better row count estimate for a table which contains BLOBs.
- In a FOREIGN KEY constraint InnoDB is now case-insensitive to column names, and in Windows also to table names.
- InnoDB allows a FOREIGN KEY column of CHAR type to refer to a column of VARCHAR type, and vice versa. MySQL silently changes the type of some columns between CHAR and VARCHAR, and these silent changes do not hinder FOREIGN KEY declaration any more.
- Recovery has been made more resilient to corruption of log files.
- Unnecessary statistics calculation has been removed from queries which generate a temporary table. Some ORDER BY and DISTINCT queries will now run much faster.
- MySQL now knows that the table scan of an InnoDB table is done through the primary key. This will save a sort in some ORDER BY queries.
- The maximum key length of InnoDB tables is again restricted to 500 bytes. The MySQL interpreter is not able to handle longer keys.
- The default value of innodb_lock_wait_timeout was changed from infinite to 50 seconds, the default value of innodb_file_io_threads from 9 to 4.

26.9.37 MySQL/InnoDB−4.0.1, 3 décembre 2001

- InnoDB is the same as in 3.23.47.
- In 4.0.0 the MySQL interpreter did not know the syntax LOCK IN SHARE MODE. This has been fixed.
- In 4.0.0 multiple-table delete did not work for transactional tables. This has been fixed.

26.9.38 MySQL/InnoDB−3.23.46, 30 novembre 2001

- This is the same as 3.23.45.

26.9.39 MySQL/InnoDB−3.23.45, 23 novembre 2001

- This is a bugfix release.
- In versions 3.23.42−.44 when creating a table on Windows you have to use lower case letters in the database name to be able to access the table. Fixed in 3.23.45.
- InnoDB now flushes stdout and stderr every 10 seconds: if these are redirected to files, the file contents can be better viewed with an editor.
- Fixed an assertion failure in .44, in trx0trx.c, line 178 when you drop a table which has the .frm file but does not exist inside InnoDB.
- Fixed a bug in the insert buffer. The insert buffer tree could get into an inconsistent state, causing a crash, and also crashing the recovery. This bug could appear especially in large table imports or alterations.
- Fixed a bug in recovery: InnoDB could go into an infinite loop constantly printing a warning message that it cannot find free blocks from the buffer pool.
- Fixed a bug: when you created a temporary table of the InnoDB type, and then used ALTER TABLE to it, the MySQL server could crash.
- Prevented creation of MySQL system tables 'mysql.user', 'mysql.host', or 'mysql.db', in the InnoDB type.
- Fixed a bug which can cause an assertion failure in 3.23.44 in srv0srv.c, line 1728.

26.9.40 MySQL/InnoDB−3.23.44, 2 novembre 2001

- You can define foreign key constraints on InnoDB tables. An example: FOREIGN KEY (col1) REFERENCES table2(col2).
- You can create > 4 GB datafiles in those file systems that allow it.
- Improved InnoDB monitors, including a new innodb_table_monitor which allows you to print the contents of the InnoDB internal data dictionary.
- DROP DATABASE will now work also for InnoDB tables.
- Accent characters in the default character set latin1 will be ordered according to the MySQL ordering.
 NOTE: if you are using latin1 and have inserted characters whose code is > 127 to an indexed CHAR column, you should run CHECK TABLE on your table when you upgrade to 3.23.43, and drop and reimport the table if CHECK TABLE reports an error!
- InnoDB will calculate better table cardinality estimates.
- Change in deadlock resolution: in .43 a deadlock rolls back only the SQL statement, in .44 it will roll back the whole transaction.
- Deadlock, lock wait timeout, and foreign key constraint violations (no parent row, child rows exist) now return native MySQL error codes 1213, 1205, 1216, 1217, respectively.
- A new my.cnf parameter innodb_thread_concurrency helps in performance tuning in high concurrency environments.
- A new my.cnf option innodb_force_recovery will help you in dumping tables from a corrupted database.
- A new my.cnf option innodb_fast_shutdown will speed up shutdown. Normally InnoDB does a full purge and an insert buffer merge at shutdown.
- Raised maximum key length to 7000 bytes from a previous limit of 500 bytes.
- Fixed a bug in replication of auto-inc columns with multiline inserts.
- Fixed a bug when the case of letters changes in an update of an indexed secondary column.
- Fixed a hang when there are > 24 datafiles.
- Fixed a crash when MAX(col) is selected from an empty table, and col is a not the first column in a multi-column index.
- Fixed a bug in purge which could cause crashes.

26.9.41 MySQL/InnoDB−3.23.43, 4 octobre 2001

- This is essentially the same as InnoDB-3.23.42.

26.9.42 MySQL/InnoDB−3.23.42, 9 septembre 2001

- Fixed a bug which corrupted the table if the primary key of a > 8000-byte row was updated.
- There are now 3 types of InnoDB Monitors: innodb_monitor, innodb_lock_monitor, and innodb_tablespace_monitor. innodb_monitor now prints also buffer pool hit rate and the total number of rows inserted, updated, deleted, read.
- Fixed a bug in RENAME TABLE.
- Fixed a bug in replication with an auto-increment column.

26.9.43 MySQL/InnoDB−3.23.41, 13 août 2001

- Support for < 4 GB rows. The previous limit was 8000 bytes.
- Use the doublewrite file flush method.
- Raw disk partitions supported as datafiles.
- InnoDB Monitor.
- Several hang bugs fixed and an ORDER BY bug ('Sort aborted') fixed.

26.9.44 MySQL/InnoDB−3.23.40, 16 juillet 2001

- Only a few rare bugs fixed.

26.9.45 MySQL/InnoDB−3.23.39, 13 juin 2001

- CHECK TABLE now works for InnoDB tables.
- A new my.cnf parameter innodb_unix_file_flush_method introduced. It can be used to tune disk write performance.
- An auto-increment column now gets new values past the transaction mechanism. This saves CPU time and eliminates transaction deadlocks in new value assignment.
- Several bug fixes, most notably the rollback bug in 3.23.38.

26.9.46 MySQL/InnoDB−3.23.38, 12 mai 2001

- The new syntax SELECT ... LOCK IN SHARE MODE is introduced.
- InnoDB now calls fsync() after every disk write and calculates a checksum for every database page it writes or reads, which will reveal disk defects.
- Several bug fixes.

27 Port vers d'autres systèmes

Cet appendice vous aidera à porter MySQL vers un autre système d'exploitation. Vérifiez d'abord la liste des systèmes supportés avant toute chose. Plate−formes supportées par MySQL . Si vous avez crée un nouveau port de MySQL, merci de nous en avertir pour que nous puissions le lister ici et sur notre site web (http://www.mysql.com/), pour le recommander aux autres utilisateurs.

Note : Si vous créez un nouveau port de MySQL, vous êtes libre de le copier et le distribuer sous la licence GPL, mais cela ne signifie pas que vous êtes détenteur de droits sur MySQL.

Une bibliothèque de threads Posix qui fonctionne est requise pour le serveur. Pour Solaris 2.5 nous utilisons Sun PThreads (le support natif des threads de la version 2.4 et plus ancienne n'est pas assez bonne) et sur Linux nous utilisons `LinuxThreads` de Xavier Leroy, Xavier.Leroy@inria.fr .

La partie la plus difficile du port vers une nouvelle variante Unix ne bénéficiant pas d'un bon support natif des threads est probablement le port de MIT−pthreads . Voyez mit−pthreads/README et Programming POSIX Threads (http://www.humanfactor.com/pthreads/).

La distribution MySQL inclut une version patchée des Pthreads de Provenzano de MIT (voyez la page web des Pthreads MIT http://www.mit.edu/afs/sipb/project/pthreads/ et une introduction à la programmation sur http://www.mit.edu:8001/people/proven/IAP_2000/). Cela peut être utilisé pour certains systèmes d'exploitation à qui n'ont pas les threads POSIX. Remarques sur MIT−pthreads .

Il est aussi possible d'utiliser un autre paquet de threads au niveau utilisateur nommé FSU Pthreads (Voir http://moss.csc.ncsu.edu/~mueller/pthreads/). Cette implémentation est utilisée pour le port vers SCO.

Consultez les programmes `thr_lock.c` et `thr_alarm.c` dans le dossier `mysys` pour quelques tests/exemples de ces problèmes.

Le serveur et le client on besoin d'un compilateur C++ fonctionnel (nous utilisons gcc et avons essayé SPARCworks). Un autre compilateur connu maintenant pour fonctionner est Irix cc .

Pour ne compiler que le client, utilisez `./configure --without-server` .

Il n'y a actuellement aucun support pour ne compiler que le serveur, et il n'est pas prévu d'en ajouter un à moins que quelqu'un n'ait une bonne raison de le faire.

Si vous voulez ou avez besoin de changer un fichier `Makefile` ou le script de configuration vous aurez besoin d'avoir Automake et Autoconf. Installer à partir des sources de l'arbre de développement .

Toutes les étapes dont vous avez besoin pour reconstruire le tout à partir des fichiers de base.

```
 /bin/rm */.deps/*.P
/bin/rm −f config.cache
aclocal
autoheader
aclocal
automake
autoconf
./configure −−with−debug=full −−prefix='votre dossier installation'

 # les fichiers make générés plus haut ont besoin de GNU make 3.75 ou plus récent.
# (appelé gmake ci−dessous)
gmake clean all install init−db
```

Si vous rencontrez des problèmes avec un nouveau port, vois devrez faire du débogage de MySQL ! Déboguer un serveur MySQL . **Note** : avant de commencer à déboguer mysqld , faites d'abord fonctionner les programmes de tests mysys/thr_alarm et mysys/thr_lock . Cela assurera que votre installation des threads a une chance de fonctionner !

27.1 Déboguer un serveur MySQL

Si vous utilisez des fonctionnalités qui ont été ajoutées il y a peu de temps à MySQL, vous pouvez essayer de démarrer mysqld avec −−skip−new (qui désactivera toutes les fonctionnalités nouvelles, qui sont potentiellement non−stables) ou avec −−safe−mode qui désactive un tas d'optimisations qui pourraient poser problèmes. Que faire si MySQL crashe constamment .

Si `mysqld` ne veut pas démarrer, vous devez vérifier que vous n'avez pas de fichiers `my.cnf` qui interfèrent avec votre configuration ! Vous pouvez vérifier les arguments de votre `my.cnf` avec `mysqld`

`--print-defaults` et éviter de les utiliser en démarrant avec `mysqld --no-defaults ...` .

Si mysqld se met à trop consommer de mémoire ou de processu ou s'il se bloque, vous pouvez utiliser mysqladmin processlist status pour trouver si quelqu'un utilise une requête qui prend trop de temps à s'exécuter. C'est un bonne idée d'exécuter mysqladmin -i10 processlist status dans un terminal si vous avez des problèmes de performances ou des problèmes à la connexion de nouveaux clients.

La commande `mysqladmin debug` écrira des informations à propos des verrous en cours d'utilisation, de la mémoire utilisée et des requêtes dans le fichier de log de MySQL. Cela peut vous aider à résoudre certains problèmes. Cette commande fournit aussi des informations utiles même si vous n'avez pas compilé MySQL pour le débogage !

Si le problème vient du fait que certaines tables sont de plus en plus lentes vous devez essayer de les optimiser en utilisant OPTIMIZE TABLE ou myisamchk . Administration de la base . Vous devez aussi vérifier les requêtes qui prennent trop de temps avec la commande EXPLAIN .

Vous devriez aussi consulter les sections spécifiques aux systèmes d'exploitations dans ce manuel pour les problèmes pouvant être uniques à votre environnement. Remarques spécifiques aux systèmes d'exploitation .

27.1.1 Compiler MYSQL pour le débogage

Si vous avez un problème spécifique, vous pouvez toujours essayer de déboguer MySQL. Pour ce faire, vous devez configurer MySQL avec l'option `--with-debug` ou `--with-debug=full` . Vous pouvez vérifier si MySQL est déjà compilé avec le support du débogage en faisant ceci : `mysqld --help` . Si l'attribut `--debug` est listé avec les options, cela veut dire que le support du débogage est activé. Dans ce cas, `mysqladmin ver` liste aussi la version de `mysqld` en tant que `mysql ... --debug` .

Si vous utilisez gcc ou egcs, la ligne de configuration recommandée est :

```
CC=gcc CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors \
  -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
  --with-debug --with-extra-charsets=complex
```

Cela évitera les problèmes avec la bibliothèque libstdc++ et les exceptions C++ (plusieurs compilateurs ont des problèmes avec les exceptions C++ dans le code threadé) et compilera une version MySQL avec le support de tous les jeux de caractères.

Si vous suspectez un dépassement de la mémoire, vous pouvez configurer MySQL avec `--with-debug=full` , qui installera un vérificateur d'allocation mémoire (`SAFEMALLOC`). Fonctionner avec `SAFEMALLOC` est cependant un peu ralentissant, et donc, si vous rencontrez des problèmes de performances, vous devez démarrer `mysqld` avec l'option `--skip-safemalloc` . Cela désactivera les vérifications de dépassements de mémoire pour chaque appel à `malloc` ou `free` .

Si mysqld ne plante plus lorsque vous le compilez avec --with-debug , vous avez probablement trouvé un bogue du compilateur ou un bogue de temporisation dans MySQL. Dans ce cas, vous pouvez essayer d'ajouter -g aux variables CFLAGS et CXXFLAGS vues plus haut et ne pas utiliser --with-debug . Si mysqld plante maintenant, vous pouvez vous y attacher avec gdb ou utiliser gdb sur le fichier noyau pour trouver ce qui est arrivé.

Lorsque vous configurez MySQL pour le support du débogage, vous activez automatiquement un tas de fonctions de tests supplémentaires qui se chargent de surveiller le bon fonctionnement de `mysqld` . Si elles trouvent quelque chose d'inattendu (`` `unexpected` ''), une entrée sera écrite dans `stderr` , que `safe_mysqld` redirige vers le log d'erreurs ! Cela signifie aussi que si vous avez quelques problèmes inattendus avec MySQL et que vous utilisez une distribution des sources, la première chose à faire est de configurer MySQL avec le support du débogage ! (la seconde, bien sûr, étant d'envoyer un mail sur les listes de diffusion pour demander de l'aide.) Les listes de diffusion MySQL . Merci d'utiliser le script `mysqlbug` pour tous les rapport de bogues ou questions concernant la version de MySQL que vous utilisez !

Dans la distribution Windows de MySQL, mysqld.exe est par défaut compilé avec le support des fichiers de traçage.

27.1.2 Créer un fichier de traçage

Si le serveur `mysqld` ne démarre pas ou que vous pouvez le crasher facilement, vous pouvez essayer de créer un fichier de traçage pour trouver le problème.

Pour ce faire, vous devez avoir un mysqld qui est compilé pour le débogage. Vous pouvez le vérifier en exécutant mysqld -V . Si le numéro de version se termine par -debug , il est compilé avec le support des fichiers de traçage.

Démarrez le serveur `mysqld` avec un journal de suivi dans `/tmp/mysqld.trace` (ou `C:\mysqld.trace` sous Windows) :

```
shell> mysqld --debug
```

Sous Windows vous devez aussi utiliser l'option --standalone pour ne pas démarrer mysqld en tant que service :

Dans une console DOS entrez :

```
mysqld --debug --standalone
```

Après cela, vous pouvez utiliser l'outil en ligne de commande mysql.exe dans une seconde fenêtre pour reproduire le problème. Vous pouvez couper le serveur avec la commande mysqladmin shutdown .

Notez que le fichier de traçage deviendra **très gros** ! si vous voulez obtenir un fichier plus petit, utilisez ce qui suit par exemple :

```
mysqld --debug=d,info,error,query,general,where:O,/tmp/mysqld.trace
```

qui n'écrit que les informations les plus intéressantes.

Si vous créez un rapport de bogue, merci de n'envoyer que les lignes du fichier de traçage où le problème se concrétise à la liste de diffusion appropriée ! Si vous n'arrivez pas à trouver le bon endroit dans le fichier, vous pouvez envoyer la totalité du fichier ainsi que le rapport de bogue via FTP à ftp://support.mysql.com/pub/mysql/secret/ pour qu'un développeur MySQL y jette un coup d'oeil.

Le fichier de traçage est généré avec le paquet **DBUG** de Fred Fish. Le paquet DBUG .

27.1.3 Déboguer mysqld sous gdb

Sur la plupart des systèmes, vous pouvez démarrer `mysqld` à partir de `gdb` pour obtenir plus d'informations si `mysqld` plante.

Avec quelques anciennes versions de gdb sous Linux vous devez exécuter run --one-thread si vous voulez être capables de déboguer les threads de mysqld threads. Dans ce cas, vous ne pouvez n'avoir qu'un thread actif à la fois. Nous vous recommandons de mettre à jour gdb à la version 5.1 dès que possible vu que le débogage des threads fonctionne mieux avec cette version !

Lors de l'utilisation de `mysqld` sous `gdb` , vous devez désactiver le traçage de la pile avec `--skip-stack-trace` pour pouvoir trouver les erreurs de segmentations avec `gdb` .

Il est très difficile de déboguer MySQL sous gdb si vous effectuez plusieurs nouvelles connexions tout le temps vu que gdb ne libère pas la mémoire occupée par les anciens threads. Vous pouvez contourner ce problème en démarrant mysqld avec -O thread_cache_size= 'max_connections +1' . Dans la plupart des cas, le simple fait d'utiliser -O thread_cache_size=5' vous aidera beaucoup !

Si vous voulez obtenir un core dump sur Linux si `mysqld` se termine avec un signal `SIGSEGV` , vous pouvez démarrer `mysqld` avec l'option `--core-file` . Ce fichier noyau peut être utilisé pour effectuer des traçages qui peuvent vous aider à trouver pourquoi `mysqld` s'est terminée :

```
 shell> gdb mysqld core
gdb>   backtrace full
gdb>   exit
```

Que faire si MySQL crashe constamment .

Si vous utilisez `gdb` 4.17.x ou plus récent sous Linux, vous devez installer un fichier `.gdb` , avec les informations suivantes, dans votre répertoire courant :

```
 set print sevenbit off
handle SIGUSR1 nostop noprint
handle SIGUSR2 nostop noprint
handle SIGWAITING nostop noprint
handle SIGLWP nostop noprint
handle SIGPIPE nostop
handle SIGALRM nostop
handle SIGHUP nostop
handle SIGTERM nostop noprint
```

Si vous rencontrez des problèmes lors du débogage des threads avec gdb , vous devez obtenir la version 5.x de gdb et essayer cela à la place. La nouvelle version de gdb a une meilleur gestion des threads !

Voilà un exemple de comment déboguer `mysqld` :

```
 shell> gdb /usr/local/libexec/mysqld
gdb> run
...
backtrace full # A faire lorsque mysqld crashe
```

Incluez la sortie suivante dans un mail généré avec mysqlbug et envoyez le sur les listes de diffusion. Les listes de diffusion MySQL .

Si `mysqld` ne répond plus, vous pouvez utiliser des outils système tel que `strace` ou `/usr/proc/bin/pstack` pour savoir où `mysqld` s'est bloqué.

```
 strace /tmp/log libexec/mysqld
```

Si vous utilisez l'interface DBI de Perl, vous pouvez activer le débogage en utilisant la méthode trace ou en définissant la variable d'environnement DBI_TRACE .

27.1.4 Utilisation d'un traçage de pile mémoire

Sur quelques systèmes d'exploitation, le log d'erreurs contiendra un fichier de pile mémoire si `mysqld` se termine soudainement. Vous pouvez utiliser ceci pour trouver où (et peut−être pourquoi) `mysqld` s'est terminé. Le log d'erreurs . Pour obtenir un traçage de la pile, vous ne devez pas compiler `mysqld` avec l'option `-fomit-frame-pointer` de `gcc` . Compiler MYSQL pour le débogage .

Si le fichier d'erreurs contient quelque chose qui ressemble à ce qui suit :

```
 mysqld got signal 11;
The manual section 'Debugging a MySQL server' tells you how to use a
stack trace and/or the core file to produce a readable backtrace that may
help in finding out why mysqld died
Attemping backtrace. You can use the following information to find out
where mysqld died.  If you see no messages after this, something went
terribly wrong
stack range sanity check, ok, backtrace follows
0x40077552
0x81281a0
0x8128f47
0x8127be0
0x8127995
0x8104947
0x80ff28f
0x810131b
0x80ee4bc
0x80c3c91
0x80c6b43
0x80c1fd9
0x80c1686
```

vous pouvez trouver où s'est terminé mysqld en exécutant ce qui suit :

- Copiez les nombres précédents dans un fichier, mysqld.stack par exemple.
- créez un fichier symbolique pour le serveur mysqld :

  ```
   nm -n libexec/mysqld > /tmp/mysqld.sym
  ```

  Notez que beaucoup de distributions binaires MySQL fournissent le fichier précédent, nommé mysqld.sym.gz . Dans ce cas, décompressez le en faisant :

  ```
   gunzip < bin/mysqld.sym.gz > /tmp/mysqld.sym
  ```
- Exécutez resolve_stack_dump -s /tmp/mysqld.sym -n mysqld.stack .Cela affichera l'endroit où mysqld a planté. Si cela ne vous aide pas à trouver pourquoi mysqld a planté, vous devez créer un rapport de bogue et y inclure le résultat de la commande précédente.Notez toutefois que dans la plupart de cas le fait de n'avoir que le traçage de la pile ne nous aidera pas à

trouver d'où vient le problème. Pour être capable de trouver le bogue, ou fournir une parade, nous aurons besoin dans la plupart des cas, nous aurons besoin de connaître la requête qui a fait planter mysqld et une batterie de tests pour que nous puissions reproduire le problème ! Comment rapporter des bogues ou problèmes .

27.1.5 Utilisation des fichiers de log pour trouver d'où viennent les erreurs de mysqld

Notez qu'avant de démarrer mysqld avec --log vous devez vérifier toutes vos tables avec myisamchk . Administration de la base .

Si mysqld se termine ou se bloque, vous devez démarrer mysqld avec --log . Lorsque mysqld se termine à nouveau, vous pouvez examiner la fin de votre fichier de log pour trouver les requêtes qui ont terminé mysqld .

Si vous utilisez --log sans spécifier un nom de fichier, le log est enregistré dans le dossier des bases de données en tant que host_name.log . Dans la plupart des cas, c'est la dernière requête dans le fichier de log qui a terminé mysqld , mais si possible, vérifiez le en redémarrant mysqld et exécutant à nouveau la requête en question à partir du client en ligne de commande mysql . Si elle fonctionne, vous devez aussi tester les autres requêtes complexes qui n'ont pas abouties.

Vous pouvez aussi utiliser la commande EXPLAIN sur toutes vos requêtes SELECT qui prennent beaucoup de temps à s'exécuter pour être sûrs que mysqld utilise les index convenablement. Syntaxe de EXPLAIN (Obtenir des informations SELECT) .

Vous pouvez trouver les requêtes qui prennent trop de temps à s'exécuter en démarrant mysqld avec --log-slow-queries . Le log des requêtes lentes .

Si vous trouvez le texte mysqld restarted dans le log d'erreurs (normalement nommé hostname.err) vous avez probablement trouvé une requête qui fait planter mysqld . Si tel est le cas, vous devez vérifier toutes vos tables avec myisamchk (Administration de la base), et tester les requêtes dans les fichiers de log MySQL pour voir si elles ne fonctionnent toujours pas. si vous trouvez une requête de ce genre, essayez d'abord de mettre à jour votre version de MySQL en prenant la version la plus récente. Si cela ne vous aide pas et que vous ne pouvez trouver d'aide dans les archives des mails de mysql , vous devez reporter ce bogue à sur les listes de diffusion. Des liens vers les archives de mails sont disponibles en ligne à l'adresse suivante : http://lists.mysql.com/ .

Si vous avez démarré mysqld avec myisam-recover , MySQL vérifiera et essayera automatiquement de réparer les tables MyISAM si elles sont marquées comme " not closed properly " ou " crashed ". Si cela arrive, MySQL ajoutera une entrée dans le fichier hostname.err 'Warning: Checking table ...' qui sera suivie de Warning: Repairing table si la table devait être réparée. si vous obtenez beaucoup de ces erreurs, sans que mysqld n'ait planté juste avant, alors quelque chose ne va pas, et une enquête plus approfondie est nécessaire. Utiliser les options de ligne de commande .

Ce n'est bien sûr pas de bon augure si mysqld a crashé, mais dans ce cas, il ne faut pas s'attarder sur les messages Checking table... mais plutôt essayer de savoir pourquoi mysqld a crashé.

27.1.6 Faire une batterie de tests lorsque vous faites face à un problème de table corrompue

Si vos tables sont corrompues ou que mysqld échoue toujours avec quelques commandes de mises à jour, vous pouvez tester si le bogue est reproductible en effectuant ce qui suit :

- Coupez le démon MySQL (avec mysqladmin shutdown).
- Créez une copie de vos tables (pour prévoir le cas très improbable ou la réparation tournerait mal).
- Vérifiez toutes les tables avec myisamchk -s base/*.MYI . Réparez toute table corrompue avec myisamchk -r base/table.MYI .
- Créez une seconde copie des tables.
- Effacez (ou déplacez) tout les vieux fichiers de log du répertoire de données de MySQL si vous avez besoin de plus d'espace.
- Démarrez mysqld avec --log-bin . Le log binaire des mises à jour . Si vous voulez trouver une requête qui fait planter mysqld , vous devez utiliser --log --log-bin .
- Lorsque vous obtenez une table corrompue, stoppez le serveur mysqld .
- Restaurez les sauvegardes.
- Redémarrez le serveur mysqld **sans** --log-bin
- Re-exécutez les commandes avec mysqlbinlog update-log-file | mysql . Le log des mises à jour est est sauvegardé dans le dossier des données de MySQL avec le nom hostname-bin.# .
- Si les tables sont à nouveau corrompues ou que vous pouvez faire échouer mysqld avec la commande précédente, vous avez trouvé un bogue reproductible qui devrait être facile à corriger ! Envoyez les tables et le log binaire via FTP à ftp://support.mysql.com/pub/mysql/secret/ et envoyez un mail à bugs@lists.mysql.com ou (si vous êtes client du support) à

support@mysql.com à propos du problème et l'équipe MySQL le corrigera le plus vite possible.

Vous pouvez aussi utiliser le script mysql_find_rows pour n'exécuter que quelques requêtes de mises à jour si vous voulez mieux cerner le problème.

27.2 Débogage un client MySQL

Pour pouvoir déboguer un client MySQL avec le paquet de débogage intégré, vous devez configurer MySQL avec --with-debug ou --with-debug=full . Les options habituelles de configure .

Avant de mettre en marche un client, vous devez définir la variable d'environnement MYSQL_DEBUG :

```
 shell> MYSQL_DEBUG=d:t:O,/tmp/client.trace
shell> export MYSQL_DEBUG
```

Cela fait générer au client un fichier de traçage dans /tmp/client.trace .

Si vous avez un problème avec votre propre code client, vous devez essayer de vous connecter au serveur et exécuter vos requêtes en utilisant un client qui fonctionne. Faites le en utilisant mysql en mode débogage (en supposant que vous ayez compilé MySQL avec le support du débogage) :

```
 shell> mysql --debug=d:t:O,/tmp/client.trace
```

Il vous fournira des informations utiles si vous voulez envoyer un rapport de bogue. Comment rapporter des bogues ou problèmes .

Si votre client se plante au niveau d'un code qui vous parait "valide", vous devez vérifier que votre fichier mysql.h inclus correspond à votre bibliothèque MySQL. Une erreur très courante est d'utiliser un vieux fichier mysql.h d'une ancienne installation avec la nouvelle bibliothèque MySQL.

27.3 Le paquet DBUG

Le serveur MySQL et la plupart des clients MySQL sont compilés avec le paquet DBUG écrit, à l'origine, par Fred Fish. Lorsque MySQL est compilé avec le support du débogage, ce paquet permet d'obtenir des fichiers de traçage de ce que le programme débogue. Créer des fichiers de traçage .

Le paquet de débogage est utilisé en invoquant le programme avec l'option --debug="..." ou -#... .

La plupart des programmes MySQL ont une chaîne de débogage par défaut qui sera utilisée si vous ne spécifiez aucune option à --debug . Le fichier de traçage par défaut est usuellement /tmp/nomprogramme.trace sur Unix et \nomprogramme.trace sur Windows.

La chaîne de caractères de controle du débogage est une séquence de champs séparés par des deux-points (:) comme celle qui suit :

```
 <champ_1>:<champ_2>:...:<champ_N>
```

Chaque champ consiste d'un caractère attribut suivi d'une liste de modificateurs, commençant optionnellement par une virgule ',' , séparés par des virgules :

```
 flag[,modificateur,modificateur,...,modificateur]
```

Les caractères attributs actuellement reconnus sont :

Attribut	Description
d	Active les sorties des macros DBUG_<N> pour l'état courant. Peut être suivi d'une liste de mots clefs, ce qui sélectionne la sortie seuelemnt pour les macros DBUG contenant ces mots. Une liste de mots clefs vides implique les sorties de toutes les macros.
D	Attendre après chaque ligne résultante du débogueur. L'argument est le nombre de dixièmme de secondes à attendre, sujet aux capabilités de la machine. Et donc, -#D,20 est une attente de deux secondes.
f	Limiter le débogage et/ou le traçage aux fonctions citées. Notez qu'une liste nulle désactivera toutes les fonctions. Les attributs appropriés "d" ou "t" doivent quand même être donnés, cet attribut ne limite que leurs actions si ils sont activés.
F	Identifie le nom du fichier source pour chaque ligne de débogage ou de traçage affichée.
i	Identifie le processus avec son identifiant pour chaque ligne de débogage ou de traçage affichée.
g	Active le profiling. Crée un fichier nommé dbugmon.out contenant des informations qui peuvent être utilisées pour profiler le programme. Peut être suivi d'une liste de mots clefs qui sélectionnent le profiling uniquement pour les fonctions présentes dans ctte liste. Une liste nulle implique que toutes les fonctions sont considérées.

L	Identifie le numéro de ligne du fichier source pour chaque ligne renvoyée par le traçage ou le débogage.
n	Imprime le niveau de profondeur de la fonction en cours d'exécution pour la sortie du traçage ou du débogage.
N	Numérote chaque ligne de la sortie du débogage.
o	Redirige le flux de sortie du débogueur vers le fichier spécifié. Par défaut, c'est stderr.
O	Comme o mais le fichier est vraiement écrit entre chaque ajout de contenu. Lorsque le besoin en est, le fichier est fermé puis réouvert entre chaque écriture.
p	Limite les actions du débogueur aux processus spécifiés. Un processus peut être identifié avec la macro DBUG_PROCESS et correspondre à un processus dans la liste pour que le débogage ait lieu.
P	Affiche le nom du processus courant pour chaque ligne de sortie de débogage ou de traçage.
r	Lors du passage à un nouvel état, ne pas hériter le niveau de profondeur de l'état de la fonction précédente. Utile lorsque l'affichage commence à la marge gauche.
S	Exécute la fonction _sanity(_file_,_line_) sur chaque fonction déboguée jusqu'à ce que la valeur de retour de _sanity() diffère de 0. (La plupart du temps utilisée avec safemalloc pour trouver les pertes de mémoire)
t	Active le traçage des appels/sorties des fonctions. Peut être suivi d'une liste (ne contenant qu'un seul modificateur) donnant un maximum numérique du traçage, au delà duquel aucune sortie ne sera affichée pour les macros de débogage ou de traçage. Par défaut, c'est une option de temps de compilation.

Quelques exemples de chaînes de controle de débogage pouvant être utilisée en ligne de commande dans le shell ("−#" est typiquement utilisé pour introduire une chaîne de controle à un programme d'application) sont :

```
 -#d:t
-#d:f,main,subr1:F:L:t,20
-#d,input,output,files:n
-#d:t:i:O,\\mysqld.trace
```

En MySQL, les balises communes affichées (avec l'option d) sont : enter , exit , error , warning , info et loop .

27.4 Commentaires à propos des threads RTS

J'ai essayé d'utiliser le paquet de threads RTS avec MySQL mais je suis resté bloqué au niveau des problèmes suivants :

Ils utilisent une veille version avec beaucoup d'appels POSIX et il est vraiment difficile de créer une couche d'abstraction pour toutes les fonctions. Je pense qu'il serait plus facile de changer la bibliothèque des threads pour qu'elle suive les nouvelles spécifications POSIX.

Quelques couches d'abstractions sont déjà écrites. Voyez `mysys/my_pthread.c` pour plus d'informations.

Au minimum, ce qui suit devra être changé :

`pthread_get_specific` doit utiliser un seul argument. `sigwait` doit prendre deux arguments. Beaucoup de fonctions (du moins `pthread_cond_wait` , `pthread_cond_timedwait`) doivent retourner le code erreur lorsqu'elles en rencontrent. Elle retournent à présent −1 et définissent `errno` .

Un autre problème est que les threads au niveau utilisateurs utilisent les signaux ALRM et que ceux−ci fait échouer beaucoup de fonctions (read , write , open ...). MySQL devrait faire une autre tentative à chaque interruption mais cela n'est pas facile à vérifier.

Le plus gros problème non−résolu est le suivant :

Pour avoir des alertes au niveau des threads, j'ai changé mysys/thr_alarm.c pour avoir une attente entre les alarmes avec pthread_cond_timedwait() , mais cela échoue avec une erreur EINTR . J'ai essayé de déboguer la bibliothèque des threads pour voir d'où cela venait, mais je n'ai pu trouver aucune solution simple.

Si quelqu'un veut utiliser MySQL avec les threads RTS je suggère ce qui suit :

- Changez les fonctions que MySQL utilise à partir de la bibliothèque des threads en POSIX. Cela ne devrait pas vous prendre beaucoup de temps.
- Compilez toutes les bibliothèques avec -DHAVE_rts_threads .
- Compilez thr_alarm .
- S'il y a de petites différences dans l'implémentation, elles peuvent être corrigées en changeant les fichiers my_pthread.h et my_pthread.c .
- Exécutez thr_alarm . S'il tourne sans aucun message du type `` warning '', `` error '' ou `` aborted '', vous êtes sur le bon chemin. Voici une bonne exécution se déroulant sur Solaris :

```
 Main thread: 1
Thread 0 (5) started
Thread: 5  Waiting
process_alarm
Thread 1 (6) started
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 1 (1) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 2 (2) sec
```

```
Thread: 6  Simulation of no alarm needed
Thread: 6  Slept for 0 (3) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 4 (4) sec
Thread: 6  Waiting
process_alarm
thread_alarm
Thread: 5  Slept for 10 (10) sec
Thread: 5  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 5 (5) sec
Thread: 6  Waiting
process_alarm
process_alarm

 ...
thread_alarm
Thread: 5  Slept for 0 (1) sec
end
```

27.5 Différences entre les différents paquets de threads

MySQL est très dépendant du paquet de threads utilisé. Ce qui fait que lors du choix d'une bonne plate−forme pour MySQL, le paquet des threads est très important.

Il y a au moins trois types de paquets de threads :

- Threads utilisateurs dans un processus unique. Le changement de threads est géré avec des alarmes et la bibliothèque des threads gère les fonctions non-utilisables avec les threads par des verrouillages. Les opérations de lectures, d'écritures et de sélections sont usuellement gérées avec une sélection spécifique aux threads qui passe à un autre thread si celui en cours d'exécution attend des données. Si les paquets des threads utilisateurs sont intégrés dans les bibliothèques standards (threads FreeBSD et BSDI) le paquet de thread nécessite moins de ressources que les paquets de thread qui doivent mapper toutes les appels non−sûrs (MIT−pthreads , FSU Pthreads et RTS threads). Avec quelques environnements, (SCO par exemple), tous les appels système sont sûrs pour les threads, ce qui fait que la liaison peut se faire très facilement (Pthreads FSU sur SCO). Mauvais côté : Tous les appels mappés prennent un peu de temps, et il est assez difficile de pouvoir gérer toutes les situations. Il y a aussi souvent des appels système qui ne sont pas gérés par le paquet de threads (comme MIT−pthreads et les sockets). La gestion des threads n'est pas toujours optimale.
- Threads utilisateurs dans des processus séparés. Les changements de threads sont effectués par le noyau et toutes les données sont partagées entre les threads. Le paquet de thread gère les appels threads standards pour permettre le partage entre les threads. LinuxThreads utilise cette méthode. Mauvais côté : Beaucoup de processus. La création des threads est lente. Si un thread s'interrompt, les autres restent en suspens et vous devez tous les terminer avant de redémarrer. Le changement de thread est d'une certaine façon consommateur de ressources.
- Threads noyau. Le changement de threads est géré par la bibliothèque de threads ou le noyau est très rapide. Tout est fait en un seul processus, mais sur certains systèmes, ps peut montrer plusieurs threads. Si un thread échoue, tout le processus échoue. La plupart des appels système sont bons pour les threads et ne devraient avoir besoin que d'une petite perte de performances. Solaris, HP−UX, AIX et OSF/1 ont des threads noyau.

Avec quelques systèmes, les threads du noyau sont gérés en intégrant les threads niveau utilisateur dans les bibliothèques du système. Dans ces cas, le changement de thread ne peut être fait qu'avec la bibliothèque de threads et le noyau n'est pas vraiment ``attentif aux threads''.

28 Variables d'environnement

Voici une liste de toutes les variables d'environnement qui sont directement ou indirectement utilisées par MySQL. La plupart peuvent être retrouvées dans d'autres parties du manuel.

Notez que toute option en ligne de commande prendra la précédence par rapport aux valeurs spécifiées dans les fichiers de configurations et les variables d'environnement, et que les valeurs dans les fichiers de configurations prennent la précédence sur les valeurs des variables d'environnement.

Dans beaucoup de cas, il est préférable d'utiliser un fichier de configuration plutôt que les variables d'environnement pour modifier le comportement de MySQL. Utiliser les fichiers d'options .

Variable	**Description**
CCX	A passer au compilateur C++ lors de l'exécution de configure.
CC	A passer au compilateur C lors de l'exécution de configure.
CFLAGS	A passer au compilateur C lors de l'exécution de configure.
CXXFLAGS	A passer au compilateur C++ lors de l'exécution de configure.
DBI_USER	L'utilisateur par défaut pour Perl DBI .
DBI_TRACE	Utilisée lors du traçage de Perl DBI .
HOME	Le chemin par défaut vers les fichiers d'historique de mysql est $HOME/.mysql_history .
LD_RUN_PATH	Utilisé pour spécifier où se trouve libmysqlclient.so .
MYSQL_DEBUG	Options de débogage/traçage lors de la phase de débogage.
MYSQL_HISTFILE	Le chemin vers le fichier d'historique de mysql . Si cette variable existe, sa valeur remplace la valeur par défaut de $HOME/.mysql_history .
MYSQL_HOST	Nom d'hôte par défaut utilisé par le client en ligne de commande mysql .
MYSQL_PS1	Invite de commande à utiliser dans le client en ligne de commande mysql . L'outil de ligne de commande.
MYSQL_PWD	Le mot de passe par défaut lors de la connexion à mysqld . Notez que cette utilisation n'est pas sécurisée.
MYSQL_TCP_PORT	Le port TCP/IP par défaut.
MYSQL_UNIX_PORT	La socket par défaut; utilisée pour les connexion à localhost .
PATH	Utilisé par le Shell pour trouver les programmes MySQL.
TMPDIR	Le dossier où les tables et fichiers temporaires sont crées.
TZ	Cela devrait être configuré à votre fuseau horaire local. Les problèmes de fuseau horaire.
UMASK_DIR	Le masque de création des dossiers. Notez que cette valeur est soumise à un ET logique par rapport UMASK !
UMASK	Le masque de création des fichiers utilisateurs lors de la création de fichiers.
USER	L'utilisateur par défaut de Windows à utiliser lors de la connexion à mysqld .

29 Expressions régulières MySQL

Une expression régulière (regex) est la meilleure méthode pour spécifier une recherche complexe.

MySQL utilise l'implémentation de Henry Spencer des expressions régulières qui tend à être conforme à POSIX 1003.2. MySQL en utilise la version étendue. Crédits . MySQL la version améliorée pour supporter les expressions régulières effectuées avec `REGEXP` dans les commandes SQL. Correspondances avec masques .

Ceci est une référence simplifiée qui n'aborde pas les détails. Pour avoir plus d'informations, reportez-vous à la page de manuel regex(7) de Henry Spencer. Ce manuel est inclus dans la distribution MySQL, dans le fichier regex.7 du dossier regex .

Une expression régulière décrit un jeu de chaînes de caractères. La plus simple est celle qui ne comporte pas de caractères spéciaux. Par exemple, l'expression régulière `bonjour` trouvera `bonjour` et rien d'autre.

Les expression régulières non-triviales utilisent des constructions spéciales pour pouvoir trouver plus d'une chaîne. Par exemple, l'expression régulière bonjour|monde trouve la chaîne bonjour ou la chaîne monde .

Voici un exemple encore plus complexe : l'expression régulière `B[an]*s` trouve l'une des chaînes suivantes `Bananas` , `Baaaaas` , `Bs` , et n'importe quelle autre chaîne commençant par un `B` , se terminant par un `s` , et contenant n'importe quel nombre de `a` et de `n` au milieu.

Une expression régulière peut utiliser l'un des caractères spéciaux ou constructions suivants :

^

Correspond au début de la chaîne.

```
mysql> SELECT "fo\nfo" REGEXP "^fo$";           -> 0
mysql> SELECT "fofo" REGEXP "^fo";              -> 1
```

$

Correspond à la fin de la chaîne.

```
mysql> SELECT "fo\no" REGEXP "^fo\no$";         -> 1
mysql> SELECT "fo\no" REGEXP "^fo$";            -> 0
```

.

N'importe quel caractère (nouvelle ligne inclus).

```
mysql> SELECT "fofo" REGEXP "^f.*";             -> 1
mysql> SELECT "fo\nfo" REGEXP "^f.*";           -> 1
```

*a**

Correspond à toute séquence de zéro ou plus caractères a .

```
mysql> SELECT "Ban" REGEXP "^Ba*n";             -> 1
mysql> SELECT "Baaan" REGEXP "^Ba*n";           -> 1
mysql> SELECT "Bn" REGEXP "^Ba*n";              -> 1
```

a+

Correspond à toute séquence de un ou plus caractères a .

```
mysql> SELECT "Ban" REGEXP "^Ba+n";             -> 1
mysql> SELECT "Bn" REGEXP "^Ba+n";              -> 0
```

a?

Correspond à zéro ou un caractère a .

```
mysql> SELECT "Bn" REGEXP "^Ba?n";              -> 1
mysql> SELECT "Ban" REGEXP "^Ba?n";             -> 1
mysql> SELECT "Baan" REGEXP "^Ba?n";            -> 0
```

de|abc

Correspond aux séquences de de ou de abc .

```
mysql> SELECT "pi" REGEXP "pi|apa";             -> 1
mysql> SELECT "axe" REGEXP "pi|apa";            -> 0
mysql> SELECT "apa" REGEXP "pi|apa";            -> 1
mysql> SELECT "apa" REGEXP "^(pi|apa)$";        -> 1
mysql> SELECT "pi" REGEXP "^(pi|apa)$";         -> 1
mysql> SELECT "pix" REGEXP "^(pi|apa)$";        -> 0
```

*(abc)**

Correspond à zéro ou plus séquences de abc .

```
mysql> SELECT "pi" REGEXP "^(pi)*$";            -> 1
mysql> SELECT "pip" REGEXP "^(pi)*$";           -> 0
mysql> SELECT "pipi" REGEXP "^(pi)*$";          -> 1
```

{1}
{2,3}

Voici une façon plus générale d'écrire les expressions régulières qui correspondent à plusieurs occurrences du dernier atome. m et n sont des entiers.

*a**
Peut être écrit a{0,} .

a+
Peut être écrit a{1,} .

a?
Peut être écrit a{0,1} .

Pour être plus précis, un atome suivi d'une accolade contenant un entier i et pas de virgule trouve une séquence de exactement i atomes.

Un atome suivi d'une accolade contenant un entier i et une virgule trouve une séquence de i ou plus atomes.

Un atome suivi d'une accolade contenant deux entiers i et j séparés d'une virgule trouve les séquences de i à j (inclusif) atomes.

Les deux arguments doivent être compris entre 0 et RE_DUP_MAX (par défaut 255), inclusif. S'il y a deux arguments, le second doit être supérieur ou égal au premier.

```
mysql> SELECT 'abcde' REGEXP 'a[bcd]&#123;2&#125;e';              -> 0
mysql> SELECT 'abcde' REGEXP 'a[bcd]&#123;3&#125;e';              -> 1
mysql> SELECT 'abcde' REGEXP 'a[bcd]&#123;1,10&#125;e';           -> 1
```

[a-dX]
[^a-dX]

Trouve n'importe quel caractère qui est (ou n'est pas, si ^ est utilisé) a , b , c , d ou X . Pour inclure le caractère littéral] , il doit suivre immédiatement le crochet ouvrant [. Pour inclure le caractère littéral - , il doit être écrit en premier ou en dernier. Ce qui fait que [0-9] correspond à n'importe quel chiffre. Chaque caractère qui n'a pas de signification spéciale à l'intérieur une paire de [] ne joue pas de rôle spécial et ne correspond qu'à lui même.

```
mysql> SELECT 'aXbc' REGEXP '[a-dXYZ]';                 -> 1
mysql> SELECT 'aXbc' REGEXP '^[a-dXYZ]$';               -> 0
mysql> SELECT 'aXbc' REGEXP '^[a-dXYZ]+$';              -> 1
mysql> SELECT 'aXbc' REGEXP '^[^a-dXYZ]+$';             -> 0
mysql> SELECT 'gheis' REGEXP '^[^a-dXYZ]+$';            -> 1
mysql> SELECT 'gheisa' REGEXP '^[^a-dXYZ]+$';           -> 0
```

[.characters.]

La séquence de caractères de cet élément d'assemblage. La séquence est un élément de la liste contenue entre les crochets. Une telle expression contenant un élément d'assemblage multi-caractères peut ainsi trouver plus d'un caractère. Vous trouverez la liste complète des noms de caractères dans regexp/cname.h .

```
mysql> SELECT '~' REGEXP '[[.~.]]';                     -> 1
mysql> SELECT '~' REGEXP '[[.tilde.]]';                 -> 1
```

[=character_class=]

Une classe d'équivalence, remplaçant les séquences de caractères de tous les éléments de l'assemblage équivalents à celui-ci, lui même inclut.Par exemple, si o et (+) sont membres d'une classe d'équivalence, alors [[=o=]] , [[=(+)=]] , et [o(+)] sont tous des synonymes. Une classe d'équivalence ne doit pas être un point final d'intervalle.

[:character_class:]

Dans une expression entre crochets, le nom d'une classe de caractères entourée de [: et :] remplace la liste de tous les caractères appartenant à cette classe. Les noms des classes de caractères sont :

alnum	Caractères alpha-numériques
alpha	Caractères alphabétiques
blank	Caractères espace
cntrl	Caractères de contrôle
digit	Chiffres
graph	Caractères graphiques
lower	Minuscules
print	Caractères graphiques ou espaces
punct	Ponctuation
space	Espace, tabulation, nouvelle ligne et retour chariot
upper	Majuscules
xdigit	Chiffres hexadécimaux

Voilà les classes de caractères définies dans la page de manuel ctype(3) . Une locale peut en fournir d'autres. Une classe de caractère ne doit pas être utilisée en tant que point final d'intervalle.

```
mysql> SELECT "justalnums" REGEXP "[[:alnum:]]+";       -> 1
mysql> SELECT "!!" REGEXP "[[:alnum:]]+";               -> 0
```

[[:<:]]
[[:>:]]

Ceux là trouvent la chaîne nulle qui précède et suit chaque mot. Un mot est défini comme étant une séquence de caractères qui n'est ni suivi ni précédée d'un caractère de mot. Un caractère de mot est un caractère alnum (défini par ctype(3)) ou un tiret bas (_).

```
mysql> SELECT 'a word a' REGEXP '[[:<:]]word[[:>:]]';   -> 1
mysql> SELECT 'a xword a' REGEXP '[[:<:]]word[[:>:]]';  -> 0
```

Pour utiliser une instance littérale d'un caractère spécial dans une expression régulière, vous devez la faire précéder de deux caractères anti-slash. L'analyseur MySQL interpréte le premier anti-slash, et la bibliothèque d'expressions régulières utilisera le second. Par exemple, pour rechercher l'expression 1+2 qui contient le caractère spécial + , seule la dernière expression régulière sera correcte :

```
mysql> SELECT '1+2' REGEXP '1+2';                       -> 0
mysql> SELECT '1+2' REGEXP '1\+2';                      -> 0
mysql> SELECT '1+2' REGEXP '1\\+2';                     -> 1
```

30 Licence Publique Générale GNU

Notice d'accompagnement de la traduction non officielle à conserver dans toute reproduction de cette traduction

This is an unofficial translation of the GNU General Public License into french. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL--only the original English text of the GNU GPL does that. However, we hope that this translation will help french speakers understand the GNU GPL better.

Ceci est une traduction non officielle de la GNU General Public License en français. Elle n'a pas été publiée par la Free Software Foundation, et ne détermine pas les termes de distribution pour les logiciels qui utilisent la GNU GPL−−seul le texte anglais original de la GNU GPL en a le droit. Cependant, nous espérons que cette traduction aidera les francophones à mieux comprendre la GPL.

Cette traduction est sous Copyright 2001 APRIL (http://www.april.org). La version la plus à jour de ce document est disponible sur http://www.april.org/gnu/gpl_french.html

Il est permis à tout le monde de reproduire et distribuer des copies conformes de cette traduction, mais aucune modification ne doit y être apportée, et la présente notice doit être préservée.

Nous autorisons la FSF à apporter toute modification qu'elle jugera nécessaire pour rendre la traduction plus claire.

Version 2, June 1991

```
 Copyright &copy; 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

 Il est permis à tout le monde de reproduire et distribuer des copies conformes
de ce document de licence, mais aucune modification ne doit y être apportée.
```

Preamble Les licences relatives à la plupart des logiciels sont destinées à supprimer votre liberté de les partager et de les modifier. Par contraste, la licence publique générale GNU General Public License veut garantir votre liberté de partager et de modifier les logiciels libres, pour qu'ils soient vraiment libres pour tous leurs utilisateurs. La présente licence publique générale s'applique à la plupart des logiciels de la Free Software Foundation, ainsi qu'à tout autre programme dont les auteurs s'engagent à l'utiliser. (Certains autres logiciels sont couverts par la Licence Publique Générale pour Bibliothèques GNU à la place). Vous pouvez aussi l'appliquer à vos programmes.

Quand nous parlons de logiciels libres, nous parlons de liberté, non de gratuité. Nos licences publiques générales veulent vous garantir que vous avez toute liberté de distribuer des copies des logiciels libres (et de facturer ce service, si vous le souhaitez), que vous recevez les codes sources ou pouvez les obtenir si vous le souhaitez, que vous pouvez modifier les logiciels ou en utiliser des éléments dans de nouveaux programmes libres, et que vous savez que vous pouvez le faire. Pour protéger vos droits, nous devons apporter des restrictions, qui vont interdire à quiconque de vous dénier ces droits, ou de vous demander de vous en désister. Ces restrictions se traduisent par certaines responsabilités pour ce qui vous concerne, si vous distribuez des copies de logiciels, ou si vous les modifiez. Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou contre une rémunération, vous devez transférer aux destinataires tous les droits dont vous disposez. Vous devez vous garantir qu'eux−mêmes, par ailleurs, reçoivent ou peuvent recevoir le code source. Et vous devez leur montrer les présentes dispositions, de façon qu'ils connaissent leurs droits. Nous protégeons vos droits en deux étapes : (1) Nous assurons le droit d'auteur (copyright) du logiciel, et (2) Nous vous proposons cette licence, qui vous donne l'autorisation légale de dupliquer, distribuer et/ou modifier le logiciel.

De même, pour la protection de chacun des auteurs, et pour notre propre protection, nous souhaitons nous assurer que tout le monde comprenne qu'il n'y a aucune garantie portant sur ce logiciel libre. Si le logiciel est modifié par quelqu'un d'autre puis transmis à des tiers, nous souhaitons que les destinataires sachent que ce qu'ils possèdent n'est pas l'original, de façon que tous problèmes introduits par d'autres ne se traduisent pas par une répercussion négative sur la réputation de l'auteur original.

Enfin, tout programme libre est en permanence menacé par des brevets de logiciels. Nous souhaitons éviter le danger que des sous−distributeurs d'un programme libre obtiennent à titre individuel des licences de brevets, avec comme conséquence qu'ils ont un droit de propriété sur le programme. Pour éviter cette situation, nous avons fait tout ce qui est nécessaire pour que tous brevets doivent faire l'objet d'une concession de licence qui en permette l'utilisation libre par quiconque, ou bien qu'il ne soit pas concédé du tout.

Nous présentons ci−dessous les clauses et dispositions concernant la duplication, la distribution et la modification.

- Le présent contrat de licence s'applique à tout programme ou autre ouvrage contenant un avis, apposé par le détenteur du droit de propriété, disant qu'il peut être distribué au titre des dispositions de la présente Licence Publique Générale. Ci−après, le "Programme" désigne l'un quelconque de ces programmes ou ouvrages, et un "ouvrage fondé sur le programme" désigne soit le programme, soit un ouvrage qui en dérive au titre de la loi sur le droit d'auteur ; plus précisément, il s'agira d'un ouvrage contenant le programme ou une version de ce dernier, soit mot à mot, soit avec des modifications et/ou traduit en une autre langue (ci−après, le terme "modification" englobe, sans aucune limitation, les traductions qui en sont faites). Chaque titulaire de licence sera appelé "concessionnaire". Les activités autres que la duplication, la distribution et la modification ne sont pas couvertes par la présente licence ; elles n'entrent pas dans le cadre de cette dernière. L'exécution du programme n'est soumise à aucune restriction, et les résultats du programme ne sont couverts que si son contenu constitue un ouvrage fondé sur le programme (indépendamment du fait qu'il a été réalisé par exécution du programme). La véracité de ce qui précède dépend de ce que fait le programme.
- Le concessionnaire peut dupliquer et distribuer des copies mot à mot du code source du programme tel qu'il les reçoit, et ce sur un support quelconque, du moment qu'il appose, d'une manière parfaitement visible et appropriée, sur chaque exemplaire, un avis approprié de droits d'auteur (Copyright) et de renonciation à garantie ; qu'il maintient intacts tous les avis qui se rapportent à la présente licence et à l'absence de toute garantie ; et qu'il transmet à tout destinataire du programme un exemplaire de la présente licence en même temps que le programme.

  Le concessionnaire peut facturer l'acte physique de transfert d'un exemplaire, et il peut, à sa discrétion, proposer en échange d'une rémunération une protection en garantie.
- Le concessionnaire peut modifier son ou ses exemplaires du programme ou de toute portion de ce dernier, en formant ainsi un ouvrage fondé sur le programme, et dupliquer et distribuer ces modifications ou cet ouvrage selon les dispositions de la section 1 ci−dessus, du moment que le concessionnaire satisfait aussi à toutes ces conditions :
    - Le concessionnaire doit faire en sorte que les fichiers modifiés portent un avis, parfaitement visible, disant que le concessionnaire a modifié les fichiers, avec la date de tout changement.
    - Le concessionnaire doit faire en sorte que tout ouvrage qu'il distribue ou publie, et qui, en totalité ou en partie, contient le programme ou une partie quelconque de ce dernier ou en dérive, soit concédé en bloc, à titre gracieux, à tous tiers au titre des dispositions de la présente licence.
    - Si le programme modifié lit normalement des instructions interactives lors de son exécution, le concessionnaire doit, quand il commence l'exécution du programme pour une telle utilisation interactive de la manière la plus usuelle, faire en sorte que ce programme imprime ou affiche une annonce, comprenant un avis approprié de droits d'auteur, et un avis selon lequel il n'y a aucune garantie (ou autrement, que le concessionnaire fournit une garantie), et que les utilisateurs peuvent redistribuer le programme au titre de ces dispositions, et disant à l'utilisateur comment visualiser une copie de cette licence (exception : si le programme par lui−même est interactif mais n'imprime normalement pas une telle annonce, l'ouvrage du concessionnaire se fondant sur le programme n'a pas besoin d'imprimer une annonce).

  Les exigences ci−dessus s'appliquent à l'ouvrage modifié pris en bloc. Si des sections identifiables de cet ouvrage ne dérivent pas du programme et peuvent être considérées raisonnablement comme représentant des ouvrages indépendants et distincts par eux−mêmes, alors la présente licence, et ses dispositions, ne s'appliquent pas à ces sections quand le concessionnaire les distribue sous forme d'ouvrages distincts. Mais quand le concessionnaire distribue ces mêmes sections en tant qu'élément d'un tout qui représente un ouvrage se fondant sur le programme, la distribution de ce tout doit se faire conformément aux dispositions de la présente licence, dont les autorisations, portant sur d'autres concessionnaires, s'étendent à la totalité dont il est question, et ainsi à chacune de ces parties, indépendamment de celui qu'il a écrite.

  Ainsi, cette section n'a pas pour but de revendiquer des droits ou de contester vos droits sur un ouvrage entièrement écrit par le concessionnaire ; bien plus, l'intention est d'exercer le droit de surveiller la distribution d'ouvrages dérivée ou collective se fondant sur le programme.

  De plus, un simple assemblage d'un autre ouvrage ne se fondant pas sur le programme, avec le programme (ou avec un ouvrage se fondant sur le programme) sur un volume d'un support de stockage ou distribution, ne fait pas entrer l'autre ouvrage dans le cadre de la présente licence.
- Le concessionnaire peut dupliquer et distribuer le programme (ou un ouvrage se fondant sur ce dernier, au titre de la Section 2), en code objet ou sous une forme exécutable, au titre des dispositions des Sections 1 et 2 ci−dessus, du moment que le concessionnaire effectue aussi l'une des opérations suivantes :
    - Lui joindre le code source complet correspondant, exploitable par une machine, code qui doit être distribué au titre des Sections 1 et 2 ci−dessus sur un support couramment utilisé pour l'échange de logiciels ; ou bien
    - Lui joindre une offre écrite, dont la validité se prolonge pendant au moins 3 ans, de transmettre à un tiers quelconque, pour un montant non supérieur au coût pour le concessionnaire, de réalisation physique de la distribution de la source, un exemplaire complet, exploitable par une machine, du code source correspondant, qui devra être distribué au titre des dispositions des Sections 1 et 2 ci−dessus sur un support couramment utilisé pour l'échange des logiciels ; ou bien
    - Lui joindre les informations que le concessionnaire a reçues, pour proposer une distribution du code source correspondant (cette variante n'est autorisée que pour la distribution non commerciale, et seulement si le concessionnaire a reçu le programme sous forme exécutable ou sous forme d'un code objet, avec une telle offre, conformément à l'alinéa b) ci−dessus).

  Le code source d'un ouvrage représente la forme préférée de l'ouvrage pour y effectuer des modifications. Pour un ouvrage exécutable, le code source complet représente la totalité du code source pour tous les modules qu'il contient, plus tous fichiers de définitions d'interface associés, plus les informations en code machine pour commander la compilation et l'installation du programme exécutable. Cependant, à titre d'exceptions spéciales, le code source distribué n'a pas besoin de comprendre quoi que ce soit qui est normalement distribué (sous forme source ou sous forme binaire) avec les composants principaux (compilateur, noyau de système d'exploitation, etc.) du système d'exploitation sur lequel est exécuté le programme exécutable, à moins que le composant, par lui−même, soit joint au programme exécutable.

Si la distribution de l'exécutable ou du code objet est réalisée de telle sorte qu'elle offre d'accéder à une copie à partir d'un lieu désigné, alors le fait d'offrir un accès équivalent à la duplication du code source à partir de ce même lieu s'entend comme distribution du code source, même si des tiers ne sont pas contraints de dupliquer la source en même temps que le code objet.

- Le concessionnaire ne peut dupliquer, modifier, concéder en sous−licence ou distribuer le programme, sauf si cela est expressément prévu par les dispositions de la présente licence. Toute tentative pour autrement dupliquer, modifier, concéder en sous−licence ou distribuer le programme est répétée nulle, et met automatiquement fin aux droits du concessionnaire au titre de la présente licence. Cependant, les parties qui ont reçu des copies, ou des droits, de la part du concessionnaire au titre de la présente licence, ne verront pas expirer leur contrat de licence, tant que ces parties agissent d'une manière parfaitement conforme.
- Il n'est pas exigé du concessionnaire qu'il accepte la présente licence, car il ne l'a pas signée. Cependant, rien d'autre n'octroie au concessionnaire l'autorisation de modifier ou de distribuer le programme ou ses ouvrages dérivés. Ces actions sont interdites par la loi si le concessionnaire n'accepte pas la présente licence. En conséquence, par le fait de modifier ou de distribuer le programme (ou un ouvrage quelconque se fondant sur le programme), le concessionnaire indique qu'il accepte la présente licence, et qu'il a la volonté de se conformer à toutes les clauses et dispositions concernant la duplication, la distribution ou la modification du programme ou d'ouvrages se fondant sur ce dernier.
- Chaque fois que le concessionnaire redistribue le programme (ou tout ouvrage se fondant sur le programme), le destinataire reçoit automatiquement une licence de l'émetteur initial de la licence, pour dupliquer, distribuer ou modifier le programme, sous réserve des présentes clauses et dispositions. Le concessionnaire ne peut imposer aucune restriction plus poussée sur l'exercice, par le destinataire, des droits octroyés au titre des présentes. Le concessionnaire n'a pas pour responsabilité d'exiger que des tiers se conforment à la présente licence.
- Si, en conséquence une décision de justice ou une allégation d'infraction au droit des brevets, ou pour toute autre raison (qui n'est pas limitée à des problèmes de propriétés industrielles), des conditions sont imposées au concessionnaire (par autorité de justice, par convention ou autrement), qui entrent en contradiction avec les dispositions de la présente licence, elles n'exemptent pas le concessionnaire de respecter les dispositions de la présente licence. Si le concessionnaire ne peut procéder à la distribution de façon à satisfaire simultanément à ces obligations au titre de la présente licence et à toutes autres obligations pertinentes, alors, en conséquence de ce qui précède, le concessionnaire peut ne pas procéder du tout à la distribution du programme. Par exemple, si une licence de brevet ne permettait pas une redistribution du programme, sans redevances, par tous ceux qui reçoivent des copies directement ou indirectement par l'intermédiaire du concessionnaire, alors le seul moyen par lequel le concessionnaire pourrait satisfaire tant à cette licence de brevet qu'à la présente licence, consisterait à s'abstenir complètement de distribuer le programme.

Si une partie quelconque de cette section est considérée comme nulle ou non exécutoire dans certaines circonstances particulières, le reste de cette section est réputé s'appliquer, et la section dans son ensemble est considérée comme s'appliquant dans les autres circonstances.

La présente section n'a pas pour objet de pousser le concessionnaire à enfreindre tous brevets ou autres revendications à droit de propriété, ou encore à contester la validité de une ou plusieurs quelconques de ces revendications ; la présente section a pour objet unique de protéger l'intégrité du système de distribution des logiciels libres, système qui est mis en oeuvre par les pratiques liées aux licences publiques. De nombreuses personnes ont apporté une forte contribution à la gamme étendue des logiciels distribués par ce système, en comptant sur l'application systématique de ce système ; c'est à l'auteur/donateur de décider s'il a la volonté de distribuer le logiciel par un quelconque autre système, et un concessionnaire ne peut imposer ce choix.

La présente section veut rendre parfaitement claire ce que l'on pense être une conséquence du reste de la présente licence.

- Si la distribution et/ou l'utilisation du Programme est restreinte dans certains pays, sous l'effet de brevets ou d'interfaces présentant un droit d'auteur, le détenteur du droit d'auteur original, qui soumet le Programme aux dispositions de la présente licence, pourra ajouter une limitation expresse de distribution géographique excluant ces pays, de façon que la distribution ne soit autorisée que dans les pays ou parmi les pays qui ne sont pas ainsi exclus. Dans ce cas, la limitation fait partie intégrante de la présente licence, comme si elle était écrite dans le corps de la présente licence.
- La Free Software Foundation peut, de temps à autre, publier des versions révisées et/ou nouvelles du General Public License. Ces nouvelles versions seront analogues, du point de vue de leur esprit, à la présente version, mais pourront en différer dans le détail, pour résoudre de nouveaux problèmes ou de nouvelles situations.Chaque version reçoit un numéro de version qui lui est propre. Si le programme spécifie un numéro de version de la présente licence, qui s'applique à cette dernier et "à toute autre version ultérieure", le concessionnaire a le choix de respecter les clauses et dispositions de cette version, ou une quelconque version ultérieure publiée par la Free Software Foundation. Si le programme ne spécifie pas de numéro de version de la présente licence, le concessionnaire pourra choisir une version quelconque publiée à tout moment par la Free Software Foundation.
- Si le concessionnaire souhaite incorporer des parties du programme dans d'autres programmes libres dont les conditions de distribution sont différentes, il devrait écrire à l'auteur pour demander son autorisation. Pour un logiciel soumis à droit d'auteur par la Free Software Foundation, il devra écrire à la Free Software Foundation ; nous faisons quelquefois des exceptions à cette règle. Notre décision va être guidée par le double objectif de protéger le statut libre de tous les dérivés de nos logiciels libres, et de favoriser le partage et la réutilisation des logiciels en général.
- COMME LA LICENCE DU PROGRAMME EST CONCEDEE A TITRE GRATUIT, IL N'Y AUCUNE GARANTIE S'APPLIQUANT AU PROGRAMME, DANS LA MESURE AUTORISEE PAR LA LOI EN VIGUEUR. SAUF MENTION CONTRAIRE ECRITE, LES DETENTEURS DU DROIT D'AUTEUR ET/OU LES AUTRES PARTIES METTENT LE PROGRAMME A DISPOSITON "EN L'ETAT", SANS AUCUNE GARANTIE DE QUELQUE NATURE QUE CE SOIT, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS LIMITATION, LES GARANTIES IMPLICITES DE COMMERCIALISATION ET DE L'APTITUDE A UN OBJET PARTICULIER. C'EST LE CONCESSIONNAIRE QUI PREND LA TOTALITE DU RISQUE QUANT A LA QUALITE ET AUX PERFORMANCES DU PROGRAMME. SI LE PROGRAMME SE REVELAIT DEFECTUEUX, C'EST LE CONCESSIONNAIRE QUI PRENDRAIT A SA CHARGE LE COUT DE L'ENSEMBLE DES OPERATIONS NECESSAIRES D'ENTRETIEN, REPARATION OU CORRECTION.
- EN AUCUN CAS, SAUF SI LA LOI EN VIGUEUR L'EXIGE OU SI UNE CONVENTION ECRITE EXISTE A CE SUJET, AUCUN DETENTEUR DE DROITS D'AUTEUR, OU AUCUNE PARTIE AYANT LE POUVOIR DE MODIFIER ET/OU DE REDISTRIBUER LE PROGRAMME CONFORMEMENT AUX AUTORISATIONS CI−DESSUS, N'EST RESPONSABLE VIS−A−VIS DU CONCESSIONNAIRE POUR CE QUI EST DES DOMMAGES, Y COMPRIS TOUS DOMMAGES GENERAUX, SPECIAUX, ACCIDENTELS OU INDIRECTS, RESULTANT DE L'UTILISATION OU DU PROGRAMME OU DE L'IMPOSSIBILITE D'UTILISER LE PROGRAMME (Y COMPRIS, MAIS SANS LIMITATION, LA PERTE DE DONNEES, OU LE FAIT QUE DES DONNEES SONT RENDUES IMPRECISES, OU ENCORE LES PERTES EPROUVEES PAR LE CONCESSIONNAIRE OU PAR DES TIERS, OU ENCORE UN MANQUEMENT DU PROGRAMME A FONCTIONNER AVEC TOUS AUTRES PROGRAMMES), MEME SI CE DETENTEUR OU CETTE AUTRE PARTIE A ETE AVISE DE LA POSSIBILITE DE TELS DOMMAGES.

Comment appliquer ces dispositions a vos nouveaux programmes? Si le concessionnaire développe un nouveau programme, et s'il en souhaite l'utilisation la plus large possible dans le public, le meilleur moyen d'y arriver est d'en faire un logiciel libre, que tout le monde pourra redistribuer et modifier au titre des présentes dispositions. Dans ce but, il convient de rattacher au programme les avis suivants. Le moyen le plus sûr consiste à les rattacher au début de chaque fichier source, pour avertir le plus efficacement possible de l'exclusion de garantie ; et chaque fichier doit comporter au moins la ligne ``copyright'', et un pointeur indiquant où est localisée la totalité de l'avis.

```
 Une ligne pour donner le nom du programme et une idée de ce qu'il fait.
Copyright (C) yyyy  nom de l'auteur

 Ce programme est un logiciel libre ; vous pouvez le redistribuer et/ou le
modifier conformément aux dispositions de la Licence Publique Générale GNU,
telle que publiée par la Free Software Foundation ; version 2 de la licence, ou
encore (à votre choix) toute version ultérieure.

 Ce programme est distribué dans l'espoir qu'il sera utile, mais SANS AUCUNE
GARANTIE ; sans même la garantie implicite de COMMERCIALISATION ou D'ADAPTATION
A UN OBJET PARTICULIER. Pour plus de détail, voir la Licence Publique Générale
GNU .

 Vous devez avoir reçu un exemplaire de la Licence Publique Générale GNU en même
temps que ce programme ; si ce n'est pas le cas, écrivez à la Free Software
Foundation Inc., 675 Mass Ave, Cambridge, MA 02139, Etats−Unis.
```

```
Ajoutez aussi des informations sur le moyen permettant d'entrer en contact avec vous par courrier électronique (e−mail) et courrier
normal.
```

Si le programme est interactif, prévoyez en sortie un court avis, tel que celui qui est présenté ci−dessous, lors du démarrage en mode interactif.

```
 Gnomovision version 69, Copyright (C) yyyy nom de l'auteur

 Gnomovision est livré absolument SANS AUCUNE GARANTIE ; pour plus de détail,
tapez 'show w'. Il s'agit d'un logiciel libre, et vous avez le droit de le
redistribuer dans certaines conditions ; pour plus de détail, tapez 'show c'.
```

```
Les instructions hypothétiques 'show w' et 'show c' doivent présenter les parties appropriées de la Licence Publique Générale. Bien
évidemment, les instructions que vous utilisez peuvent porter d'autres noms que 'show w' et 'show c' ; elles peuvent même correspondre
à des clics de souris ou à des éléments d'un menu, selon ce qui convient à votre programme.
```

Si nécessaire, vous devrez aussi demander à votre employeur (si vous travaillez en tant que programmeur) ou à votre éventuelle école ou université, de signer une ``renonciation à droit d'auteur'' concernant le programme. En voici un échantillon (il suffit de modifier les noms) :

```
 Yoyodyne, Inc., par la présente, renonce à tout intérêt de droits d'auteur dans
le programme `Gnomovision' (qui fait des passages au niveau des compilateurs)
écrit par James Hacker.

 signature de Ty Coon, 1er avril 1989

 Ty Coon, President of Vice
```

```
La présente Licence Publique Générale n'autorise pas le concessionnaire à incorporer son programme dans des programmes propriétaires.
Si votre programme est une bibliothèque de sous−programmes, vous pouvez considérer comme plus intéressant d'autoriser une édition de
liens des applications propriétaires avec la bibliothèque. Si c'est ce que vous souhaitez, vous devrez utiliser non pas la présente
licence, mais la Licence Publique Générale pour Bibliothèques GNU.
```

Sauf mention contraire indiquée plus haut, le présent document est soumis aux conditions d'exploitation suivantes :

```
Copyright 2001 APRIL
```

Ce document peut être reproduit par n'importe quel moyen que ce soit, pourvu qu'aucune modification ne soit effectuée et que cette notice soit préservée.

31 Index des commandes, types et fonctions SQL

32 Index conceptuel

unnumbered