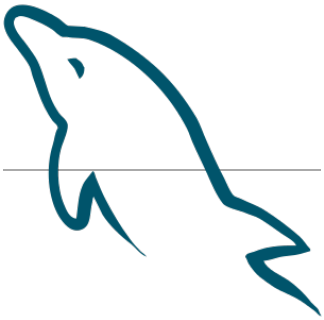


MySQL

Sections

Introduction

Pourquoi MySQL ?

- Gratuiciel en licence GPL version 2.
- Facile d'utilisation : intuitif et ergonomique.
- Vitesse performante[1].
- Supporte la plupart des commandes SQL ANSI.
- Support technique complet, avec des tutoriels en ligne, des forums, mailing list (lists.mysql.com),et des contrats payants possibles.
- Portabilité : importation et exportation faciles vers des fichiers Excel et autres bases de données.
- Échelonnable : pratique aussi bien pour des petites bases, que pour celles contenant des milliards d'enregistrements avec plusieurs téraoctets de données et des centaines de milliers de tables.
- Contrôle des permission des utilisateurs précis.

La licence MySQL

MySQL est disponible sous double licence :

- Licence publique générale GNU version 2 : copyleft, permettant d'utiliser MySQL à des fins commerciales ou pas, tant que l'application est à la même licence. Il y existe par ailleurs une exception Free/Libre Open Source Software (FLOSS) qui autorise des programmes non GPL mais gratuits à se connecter au serveur MySQL (comme par exemple des programmes en licence PHP).
- Une soi-disante "commerciale" (bien que GNU GPL puisse être aussi utilisée en commercial mais pas propriétaire), licence payante, conférant le droit d'intégrer MySQL avec une application non FLOSS, redistribuable en dehors de son organisation. Mais ces bibliothèque ne peuvent pas se connecter aux nouvelles versions de MySQL.

MySQL et ses forks

MySQL étant un freeware, il a donc engendré des fork officieux.

MariaDB

En 2008, Sun Microsystems acheta MySQL, puis fût acheté par Oracle en 2010. Après l'acquisition, le processus de développement changea. L'équipe commença à sortir de nouvelles versions de MySQL moins fréquemment, avec du nouveau code moins testé, par une communauté moins active

En 2009 Monty Widenius, fondateur de MySQL, quitta l'entreprise pour en créer une autre, appelée *The Monty Program* (http://www.askmonty.org/). Son fork fût appelé *MariaDB*, il permet de :

- importer le nouveau code qui sera ajouté à la branche MySQL, en le rendant plus stable ;
- nettoyer le code MySQL ;
- ajouter des contributions de la communauté (plugins et fonctionnalités) ;
- développer le moteur de stockage *Aria*, anciennement *Maria* ;
- augmenter les performances ;
- ajouter d'autres fonctionnalités au serveur.

Sa licence est GNU GPLv2, héritée de MySQL.

La plateforme primaire de MariaDB est GNU/Linux, mais il tourne aussi sur Windows[2]. Les moteurs de stockage suivants ont été ajoutés :

- Aria (utilisé pour les tables internes)
- PBXT
- XtraDB
- FederatedX
- SphinxSE
- OQGRAPH

Installation sur Ubuntu : `sudo apt-get install mariadb-server && apt-get install mariadb-client`

Drizzle

En 2008 Brian Aker, architecte en chef de MySQL, quitta le projet pour démarrer un nouveau fork appelé Drizzle (http://www.drizzle.org/). Initialement financé par Oracle, Drizzle l'est maintenant par Rackspace. Ses caractéristiques sont :

- seule une petite partie du code MySQL a été conservée : les fonctionnalités essentielles ;
- modularité : beaucoup de choses peuvent être implémentées sous forme de plugins ;
- optimisé multiprocesseur et multicore 64 bits ;
- seuls les systèmes GNU/Linux et UNIX sont supportés.

Il n'existe pas de version publique de ce fork. Il est en licence GNU GPLv2 (héritée de MySQL), mais une licence BSD peut être appliquée.

OurDelta

OurDelta (http://ourdelta.org/) est un autre fork, maintenu par *Open Query*. La première branche (5.0), est basée sur MySQL 5.0. La 5.1 est issue de MariaDB. OurDelta contient des patchs développés par la communauté ou des tiers. Il fournit des packages pour certaines distributions GNU/Linux : Debian, Ubuntu, Red Hat/CentOS. Il n'est pas disponible sur d'autres plateformes mais son code source est disponible gratuitement.

Percona Server

Percona Server set un fork maintenu par Percona. Il propose le moteur de stockage ExtraDB, fork d'InnoDB, et des patchs d'amélioration des performances.

Installation d'Apache / MySQL pour Windows

Tout-en-un

Des logiciels tout-en-un (serveur Web, base de donnée MySQL, et PHP) permettent de s'affranchir d'une installation fastidieuse et rédhibitoire pour le débutant :

1. EasyPHP téléchargement (http://www.easyphp.org) : n'a pas vocation à être installé pour de la production, mais pour le développement. Il stocke les bases de données dans C:\Program Files (x86)\EasyPHP\binaries\mysql\data.
2. WAMP téléchargement (http://www.wampserver.com) : est du même type qu'EasyPHP : ce logiciel installe facilement un serveur Web Apache, une base de données MySQL et PHP 4 et 5. Il a l'avantage de permettre de passer facilement de PHP 4 à PHP 5, sans avoir à refaire une installation ou une compilation. Tout comme EasyPHP, c'est un environnement de développement, et non un environnement de production. Attention : la résolution des noms d'hôtes se réalise séparément. Les installations WAMP servent à tester en local sur votre PC. Dans la plupart des cas, il suffit d'utiliser le fichier *Hosts* local, comme on le ferait sur une machine Linux, afin de lier des noms aux adresses IP. Dans Windows XP, Vista et 7, ce fichier se trouve dans le répertoire *systemroot\System32\Drivers\Etc*. Il peut se faire que le service ait déjà été configuré. Lorsque vous vous en doutez, contactez votre administrateur réseau. Remarque : vous trouverez une liste des possibilités de résolution de noms avec MS Windows sur Microsoft.com (http://www.microsoft.com/technet/prodtechnol/winxppro/reskit/c24621675.mspx).
3. XAMPP téléchargement (http://www.apachefriends.org/fr/xampp.html) : est du même type qu'EasyPHP ou WAMP, le deuxième P étant pour Perl. Son usage est recommandé avec PHPEclipse (http://www.phpeclipse.de/tiki-view_articles.php), et il fournit aussi un serveur Apache Tomcat par défaut.
4. The Uniform Server téléchargement (http://www.uniformserver.com) : en anglais seulement avec Apache2, Perl5, PHP5, MySQL5, phpMyAdmin.

> **Attention !**
>
> Sur Windows 10 pro, le serveur IIS est installé par défaut, et oblige Apache à changer de port (888 au lieu de 80) lors de l'installation. Pour résoudre cela il suffit de décocher *Internet Information Services* dans *Programmes et fonctionnalités*, *Activer ou désactiver des fonctionnalités Windows.*
>
> De même, le port MySQL est susceptible de passer de 3306 à 3388.

> **Attention !**
>
> Sur Windows 10, *EasyPHP development server* (alias *Devserver*, la version rouge) ne fonctionne pas (*il manque MSVCR110.dll*), mais *EasyPHP hosting server* (alias *Webserver*, la bleue) tourne normalement. Or, elle se lance automatiquement à chaque démarrage, ce qui le ralentit significativement. Pour éviter cela, exécuter *services.msc*, puis passer les trois services ci-dessous en démarrage manuel. Ensuite pour les lancer à souhait (en tant qu'administrateur), créer un script MySQL.cmd contenant les lignes suivantes :

```
net start ews-dbserver
net start ews-httpserver
net start ews-dashboard
pause
net stop ews-dashboard
net stop ews-httpserver
net stop ews-dbserver
```

Message d'erreur relatif à SSL

Pour l'instant, WAMP ne supporte pas encore le *Secure Socket Layer* (SSL). L'installation se finit par un message qui vous informe de ce fait. Afin de pouvoir travailler sans problèmes, éditez le fichier *c:\windows\php.ini*. Cherchez dans ce fichier la ligne qui commence avec *extension=php_openssl.dll*. Commentez cette ligne en la faisant précéder d'un point-virgule :

```
;extensions=php_openssl.dll
```

Si tout se passe bien, vous pouvez ouvrir la page de test dans votre navigateur.

Logo EasyPHP.

Installation manuelle

- Apache est disponible sur le site Web de Apache Software Foundation apache.org (http://www.apache.org).
- PHP est téléchargeable sur le site officiel de php (http://www.php.net). Choisissez le fichier au format ZIP.
- Enfin, vous trouverez MySQL sur mysql.com (http://www.mysql.com).

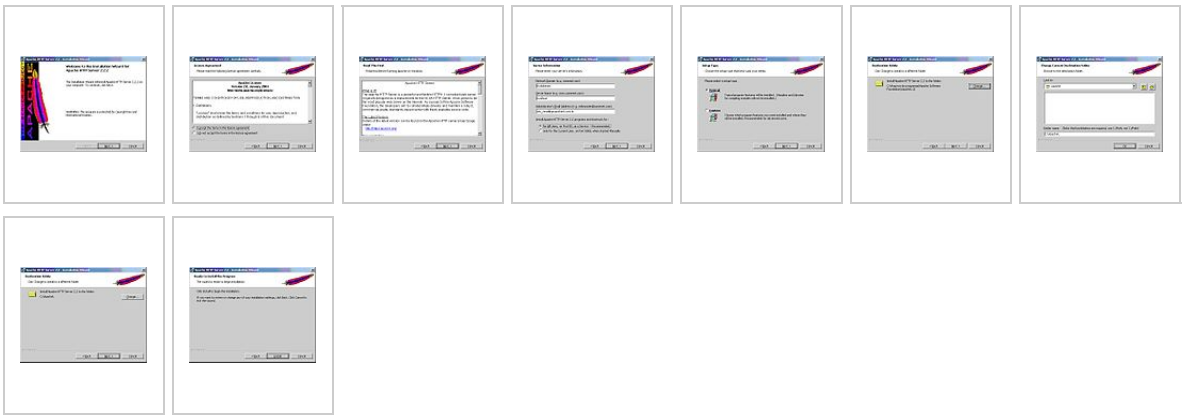
Installer Apache

Pour installer Apache, double-cliquez sur le fichier exécutable, et suivez les instructions d'installation automatique.

Si vous installez Apache sur un ordinateur de développement, renseignez le champ "nom de domaine" avec la valeur `localhost`.

Si vous installez un serveur de production et que vous disposez d'un nom de domaine, vous devriez disposer des informations nécessaires concernant votre nom de domaine, fournies par le *registrar*.

Une fois l'installation terminée, il faut encore indiquer à Apache qu'il doit fonctionner conjointement avec PHP, car il ne sait pas les traiter par défaut. Pour cela, il faut modifier les informations de configuration d'Apache, contenues dans le fichier `httpd.conf`, qui se trouve dans le dossier d'installation d'Apache, dans le sous-dossier `conf`.



Installer PHP

Une fois l'archive téléchargée, décompressez-la à la racine de votre disque dur et renommez le dossier en `'PHP'`. Dans le dossier PHP, vous trouverez deux fichiers: `php.ini-dist` et `php.ini-recommended`. Copiez `php.ini-recommended` dans votre dossier `C:\Windows` ou `C:\winnt` (le nom du dossier dépend de la version de votre système.
renommez-le en `php.ini`.

Ce fichier est le fichier de configuration qui contrôle les options dont vous disposerez. Par exemple :

PHP.ini	PHP	Rôle
`error_reporting E_ALL`	`error_reporting(E_ALL);`	Affiche tous les avertissements et erreurs directement sur le site. C'est utile pour la préproduction car cela évite de rechercher d'éventuels messages dans les logs, mais peut perturber la mise en page pour des avertissements bénins.
`error_reporting 0`	`error_reporting(0);`	N'affiche aucun message sur le site relatif à son exécution
`max_execution_time = 300`		Définit le "timeout", c'est-à-dire le temps maximum en secondes autorisé pour exécuter un script PHP.
`post_max_size = 80M`		Définit la taille maximum d'un fichier que l'on peut envoyer au serveur en HTTP.

MySQL

Télécharger et installer le .msi sur http://dev.mysql.com/downloads/gui-tools/5.0.html.

Pour arrêter, démarrer, démarrer automatiquement le serveur MySQL vous devez aller dans la gestion des services (Démarrer/Exécuter/services.msc).

Installation d'Apache / MySQL pour Linux

LAMP

Logiciel tout-en-un pour Linux (Apache + MySQL + PHP), comme WAMP pour Windows.

commande nécessitant les privilèges root

```
# apt-get install tasksel
# tasksel install lamp-server
```

Installation manuelle

Apache sur Debian / Ubuntu

commande nécessitant les privilèges root

```
# apt-get install apache2
```

Le service peut ne pas être lancé par défaut, mais même s'il l'est on peut quand-même essayer de l'activer avec :

commande nécessitant les privilèges root

```
# /etc/init.d/apache2 start
```

On peut ensuite tester le serveur, pour voir si une page s'affiche ou s'il refuse la connexion :

commande

```
$ lynx http://localhost/
```

Cette adresse est le rebouclage, elle peut aussi être rentrée directement dans tout navigateur web.

Si Apache était déjà installé vérifier le fichier pour indiquer le démarrage automatique d'Apache 2 **/etc/default/apache2** :

```
# vi /etc/default/apache2
...
NO_START=0
```

PHP

PHP peut-être installé avec toutes les déclinaisons de la distribution Debian (stable, testing, unstable). Il suffit pour cela d'insérer vos lignes préférées dans le fichier */etc/apt/sources.list* :

```
deb http://ftp.fr.debian.org/debian/ stable main non-free contrib
deb-src http://ftp.fr.debian.org/debian/ stable main non-free contrib
```

Ce qui suit suppose que le serveur Web a bien été installé : exécuter les commandes suivantes :

```
sudo apt-get update && apt-get install php7.0 && apt-get install libapache2-mod-php7.0
```

Une fois ces commandes exécutées, redémarrer le serveur Web. Dans le cas d'Apache cela s'effectue avec la commande suivante :

```
/etc/init.d/apache2 restart
```

Si tout s'est bien passé, vous disposez maintenant d'un serveur Web qui a la capacité d'exécuter des scripts PHP dans votre navigateur.

Testons :

commande

```
$ lynx http://localhost/test.php
```

Pour débugger :

commande

```
$ tail /var/log/apache2/error.log
```

Mise à jour

Pour la v7.2 :

```
sudo add-apt-repository ppa:ondrej/php
sudo apt update
sudo apt install php7.2 php7.2-common php7.2-cli php7.2-fpm
sudo a2enmod php7.2
sudo a2dismod php7.0
```

> **Attention !**
>
> Une fois les serveurs Web installés, ils se lancent automatiquement à chaque démarrage de la machine, ce qui est souhaitable pour un serveur, mais pas toujours pour un PC. Pour éviter cela, il suffit d'y désactiver les daemons :
>
> ```
> sudo update-rc.d apache2 disable
> sudo update-rc.d mysql disable
> ```

Apache sur Gentoo

Premièrement il faut installer Apache si ce n'est pas déjà fait :

```
emerge apache
```

Ensuite, il faut installer PHP :

```
emerge dev-lang/php
```

Puis il faut qu'apache utilise PHP dans sa configuration.

Code: Configuration de apache

```
# nano -w /etc/conf.d/apache2
APACHE2_OPTS="-D PHP5"
```

MySQL seul

MySQL est disponible sur http://dev.mysql.com/downloads/gui-tools/5.0.html au format :

1. .msi (Windows)
2. .dmg (Mac)
3. .rpm (Linux)
4. .tar

En l'absence de gestionnaire de paquets, utiliser le .tar ainsi :

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
```

APT

```
$ sudo apt-get install mysql-server mysql_secure_installation
```

Puis, modifier PHP pour qu'il supporte MySQL :

```
$ sudo apt-get install php4-mysql
```

Variante

La dénomination des paquets mentionnés peut varier légèrement selon la version. Dans un terminal, entrez :

```
$ sudo apt-get install mysql-server
```

et confirmez.

(Remarque : il semblerait qu'en installant le paquet "mysql-server-5.0", au lieu du paquet mentionné plus haut, certaines personnes rencontrent des problèmes. Il est donc préférable d'installer ce paquet, ou d'installer la dernière version 4 stable avec : $ sudo apt-get install mysql-server-4.1. Consultez le forum pour plus d'informations : [1] (http://forum.ubuntu-fr.org/viewtopic.php?id=15352))

Lancez ensuite la commande :

```
cd && sudo mysql_secure_installation
```

Appuyez sur Entrée lorsqu'il vous demande le mot de passe root MySQL : pour le moment il n'y en a pas.

***NB :** MySQL a ses propres utilisateurs, avec leurs propres privilèges. Le root MySQL n'est donc pas le root système. Il est conseillé de ne pas mettre les mêmes mots de passes pour les utilisateurs MySQL et les utilisateur du système.*

Le script vous demande alors si vous voulez mettre un mot de passe pour l'utilisateur root. Répondez Y, et entrez (2 fois le nouveau mot de passe du root MySQL). Il vous pose ensuite une série de questions. Si vous ne savez pas quoi répondre, acceptez les choix par défaut en appuyant simplement sur Enter.

Votre serveur MySQL est prêt. Par défaut il se lance à chaque démarrage du système, si vous ne le souhaitez pas, il vous suffit de lancer :

```
$ sudo dpkg-reconfigure mysql-server
```

et de répondre "Non" à la question du démarrage systématique de MySQL.

Sur Gentoo

```
emerge mysql
```

Installer PhpMyAdmin

Depuis un tout-en-un, il suffit de créer un chemin accessible depuis le serveur Web :

```
sudo ln -s /usr/share/phpmyadmin /var/www/phpmyadmin
```

Sinon :

```
sudo apt-get install phpmyadmin php5
```

Installer Apache et PHP avec PhpMyAdmin

Grâce aux dépendances des paquets, cette opération peut se faire en une seule fois : *Remarque : Vérifiez que la case "Traiter les paquets recommandés comme des dépendances" soit cochée dans Synaptic, configuration, préférences.*

```
$ sudo apt-get install phpmyadmin
```

Cela installera automatiquement apache2 + php + modules d'apache pour PHP et MySQL + PhpMyAdmin. Pour accéder à PhpMyAdmin, il faut se rendre à la page http://localhost/PhpMyAdmin.

Note : En cas de problème d'authentification (erreur 2002 notamment) installer le paquet mysql-server peut résoudre ce dernier.

Après l'installation, il vaut mieux modifier les droits d'accès de root, et ajouter un mot de passe pour un peu plus de sécurité. Pour cela, il faut se rendre à la page *privilèges* de PhpMyAdmin.

Remarque pour Ubuntu 5.04 (Hoary Hedgehog) : Afin que cette commande fonctionne il est nécessaire d'avoir effectué les modifications suivantes : dans /etc/apt/ éditer le fichier sources.list supprimer les # des lignes suivantes :

```
# deb http://fr.archive.ubuntu.com/ubuntu hoary universe
```

(cette ligne est dans certain cas '# deb http://archive.ubuntu.com/ubuntu/ hoary universe main restricted multiverse')

```
# deb-src http://fr.archive.ubuntu.com/ubuntu hoary universe
```

Pour la version d'Ubuntu 5.10 (Breezy), vous pouvez effectuer ces changements avec le gestionnaire de paquets synaptic (apt) : Système ---> Administration ---> Gestionnaire de paquets Synaptic

```
          Catégories ---> Dépôts ----> Ajouter et ensuite, sélectionner : maintenu par la communauté universe...
```

Lancer le chargement des nouvelles sources :

```
$ sudo apt-get update
```

Puis lancer l'installation de PhpMyAdmin comme décrit ci-dessus.

Extensions

Pour activer des modules complémentaires :

```
a2enmod Nom_du_module # passe dans /etc/apache2/mods-enabled/
```

Ex :

```
a2enmod rewrite
```

Pour les désactiver :

```
a2dismod Nom_du_module # passe dans /etc/apache2/mods-available/
```

Pour activer des sites :

```
a2ensite Nom_du_site # passe dans /etc/apache2/sites-enabled/
```

Pour les désactiver :

```
a2dissite Nom_du_site # passe dans /etc/apache2/sites-available/
```

Les extensions PHP nécessitent une autre commande. Ex :

```
phpenmod mbstring
```

Problème d'encodage d'Apache2

Si vous rencontrez un problème d'encodage des caractères de vos pages, par exemple les caractères accentués apparaissant sous la forme "<?>" (<?>), c'est probablement parce qu'Apache2 déclare dans les en-têtes HTTP qui accompagnent les pages visionnées un encodage par défaut en Unicode (UTF-8) :

```
Content-Type: text/html; charset=UTF-8
```

Tandis que les pages visionnées utilisent un autre encodage des caractères, comme par exemple Latin1 (ISO-8859-1). Même si vos documents indiquent le jeu de caractères utilisé, le paramètre donné par le serveur dans les en-têtes HTTP est prioritaire !

Pour corriger ce problème, il faudra éditer /etc/apache2/apache2.conf :

```
$ sudo gedit /etc/apache2/apache2.conf
```

Encodage par défaut en Latin1 (ISO-8859-1)

Cherchez la ligne suivante :

```
#AddDefaultCharset     ISO-8859-1
```

Décommentez-la en enlevant le # :

```
AddDefaultCharset      ISO-8859-1
```

Pour ceux qui ont la locale iso-8859-15 (sinon vous pouvez faire "sudo dpkg-reconfigure locales" pour l'ajouter) et qui désirent l'utiliser par défaut, ajoutez un 5 en fin de ligne :

```
AddDefaultCharset      ISO-8859-15
```

ainsi que la ligne suivante dans le paragraphe en-dessous :

```
AddCharset ISO-8859-15 .iso8859-15  .latin15 .fr
```

Il ne vous reste plus qu'à mettre "fr" en première position dans la ligne LanguagePriority (juste au-dessus), et à demander à apache de relire sa configuration :

```
$ sudo /etc/init.d/apache2 reload
```

Aucun encodage par défaut

Il est également possible de s'affranchir de tout encodage par défaut, de la manière suivante :

Cherchez la directive *AddDefaultCharset* :

```
AddDefaultCharset      ISO-8859-1
```

Remplacez l'attribut par la valeur *Off* :

```
AddDefaultCharset      Off
```

Là encore, on demandera à Apache de relire sa configuration :

```
$ sudo /etc/init.d/apache2 reload
```

Maintenant, les en-têtes HTTP ne contiendront plus d'indication d'encodage des caractères. Attention : il faudra alors que chaque page indique l'encodage utilisé, car s'en remettre à la détection automatique par les navigateurs peut s'avérer assez aléatoire !

Hello world

Pour entrer des commandes SQL on peut soit :

- Lancer le logiciel en shell.
  - Linux : `mysql -h localhost -u root MaBase`
  - Windows : `"C:\Program Files (x86)\EasyPHP\binaries\mysql\bin\mysql.exe" -h localhost -u root MaBase`
- Ouvrir une fenêtre SQL dans PhpMyAdmin (ex : http://localhost/modules/phpmyadmin/#PMAURL-1:server_sql.php?server=1).

```
select "hello world";
+-------------+
| hello world |
+-------------+
| hello world |
+-------------+
1 row in set (0.00 sec)
```

Références

1. http://www.mysql.com/why-mysql/benchmarks/
2. http://archive.mariadb.org/mariadb-5.5.28a/

PhpMyAdmin

PhpMyAdmin est un paquet qui permet, grâce à une interface web, d'éditer/créer/supprimer des bases MySQL, des tables et leur contenu.

Prérequis :

- Serveur Apache installé.
- PHP installé.

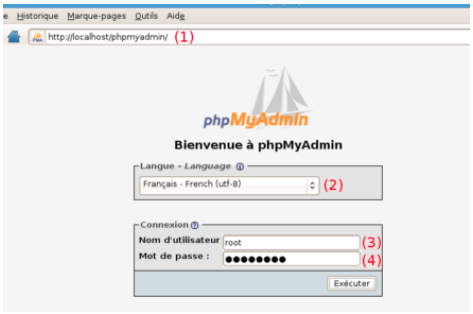
Installation de PhpMyAdmin

1. Sous Windows, il est fourni avec WAMP ou EasyPHP, mais peut aussi être installé indépendamment depuis http://www.phpmyadmin.net/home_page/downloads.php.
2. Sous Linux : Voir LAMP ou bien :

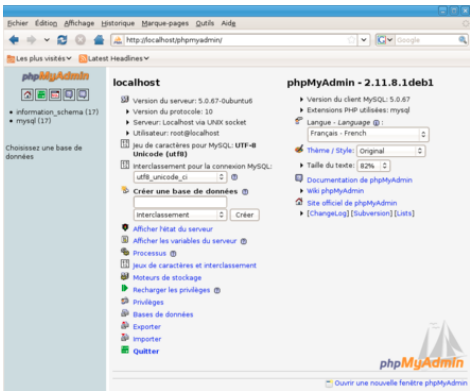
```
apt-get install phpmyadmin
```

Une fois les paquets téléchargés et installés, taper http://localhost/phpmyadmin/ dans la barre d'adresse d'un navigateur (1) et faire "Entrée". Une page s'affichera, invitant à choisir sa langue d'affichage (2), entrer son nom utilisateur (3, sinon il définit *phpmyadmin* par défaut) et son mot de passe (4). Valider avec le bouton "Exécuter"

S'il y a un message d'erreur persistant à la place, il faut décommenter et définir les identifiants dans le fichier `phpmyadmin\config.inc.php`.



Vous obtiendrez cet écran où toutes les bases sont paramétrables.



Configuration

PHP

La configuration par défaut ne permet que d'importer des bases de données de maximum 2 Mo. Au-delà l'erreur suivante survient :

- *Aucune données n'a été reçu en vue de l'importation. Aucun nom de fichier n'a été fourni, ou encore la taille du fichier a dépassé la limite permise par votre configuration de PHP.*

Pour étendre ce quota, modifier quatre lignes dans php.ini :

```
max_execution_time = 600
...
max_input_time = 600
...
upload_max_filesize = 100M
...
post_max_size = 100M
```

Puis relancer Apache.

De même, la durée de session par défaut étant de 1440 s, il convient de les étendre dans php.ini (paramètre *session.gc_maxlifetime*) avant de les étendre dans *config.inc.php*, ou par l'interface graphique.

MySQL

Sur la page d'accueil, sans sélectionner de base, il y a un onglet *Paramètres*, puis dedans un *Fonctionnalités* pour gérer l'interface PHPMyAdmin.

De plus, l'onglet *Variable* permet de lire et configurer les paramètres de MySQL.

Créer des utilisateurs MySQL

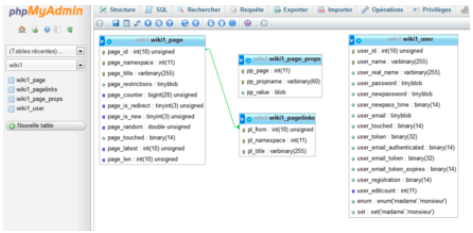
Lancez votre navigateur préféré sur l'adresse http://localhost/phpmyadmin/

- Connectez-vous en tant que 'root' avec le mot de passe du root MySQL que vous avez défini tout à l'heure
- Cliquez sur "Privilèges", puis sur "Ajouter un utilisateur"
- Entrez les informations de l'utilisateur
- Dans la table "Privilèges globaux", définissez les droits de l'utilisateur. Si vous ne savez pas quoi mettre, cochez toutes les cases des colons "Données" et "Structures".
- Validez en cliquant sur "Exécuter".

Vous pouvez maintenant vous déconnecter en cliquant sur "Quitter" et vous connecter avec le login et le mot de passe du nouvel utilisateur.

Conception

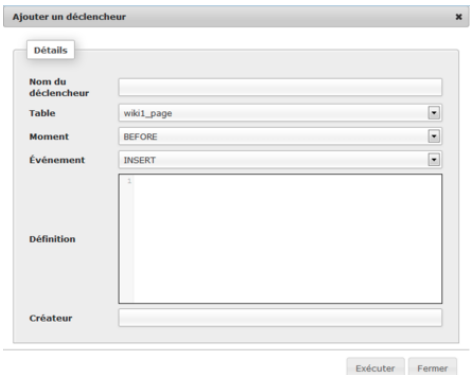
Il est possible de modéliser les tables et d'en convertir le code, grâce au concepteur accessible dans les onglets depuis une table :



Optimisation

Ce SGBD permet l'optimisation de requête, tel que décrit dans MySQL/Optimisation.

Par exemple, dans les évènements il est possible de lancer un `OPTIMIZE TABLE` toutes les nuits, d'obtenir une interface graphique avec version imprimable pour visionner le tableau d'un `EXPLAIN`, ou encore pour ajouter un trigger :



Parcourir les bases de données

mysql


Base *information_schema* dans phpMyAdmin.

`mysql` est une base de données système contenant des variables propres au serveur, telles que le fuseau horaire[1].

Il est possible d'y stocker l'historique des requêtes entrées sur le serveur en activant :

```
SET GLOBAL general_log = 'ON';
SET GLOBAL log_output = 'TABLE';
```

Elles apparaissent ainsi dans la table *general_log*, ce qui peut être pratique pour déboguer une application en boite noire.

```
SELECT * FROM mysql.general_log;
```

INFORMATION_SCHEMA

`information_schema` est une base de données virtuelle apparue dans MySQL 5, qui contient des métadonnées sur le serveur et ses bases. Elle n'est pas modifiable (ni la structure, ni les données), on peut donc juste la lire.

Beaucoup de ses informations sont récupérables aussi avec la commande `SHOW`, plus rapide. Toutefois `information_schema` est plus flexible.

La table de *INFORMATION_SCHEMA* sur les bases est *SCHEMATA*. Le programme `mysqlshow` (en ligne de commande DOS/Unix) peut aussi être utilisé à la place.

Cela ne peut fonctionner que si le serveur est démarré, et sans l'option `--skip-all-databases`.

En l'absence des privilèges `SHOW DATABASES`, seule les bases sur lesquelles le compte a des permissions seront visibles.

Lister les bases

Les commandes SQL suivante fournissent les informations relatives aux bases de données situées sur le serveur courant.

Toutes :

```
SHOW DATABASES;
```

le mot clé `SCHEMA` peut être utilisé en lieu et place de `DATABASES`. MySQL ne supporte pas les `SCHEMA` des standards SQL, donc il est synonyme de `DATABASES`. il a été ajouté pour la compatibilité avec d'autres SGBD.

Ajouter un filtre sur les noms des bases

```
SHOW DATABASES LIKE 'expression';
```

L'opérateur `LIKE` fonctionne selon le langage de manipulation de données standard. Donc il est faisable de lister toutes les bases commençant par 'wiki' ainsi :

```
SHOW DATABASES LIKE 'wiki%';
```

Filtres complexes

En utilisant la clause `WHERE` :

```
SHOW DATABASES WHERE conditions;
```

Elle autorise les expressions rationnelles, les opérateur de comparaison '=', '<' et '>', et les fonctions sur les chaines de caractères.

Lister les tables et les vues

Les tables `TABLES` et `VIEWS` de la base *INFORMATION_SCHEMA* fournissent des informations sur les tables et les vues de toutes les bases du serveur.

Les commandes SQL suivantes donnant relativement peu d'information sur les vues, il faudra recourir à la table `VIEWS` pour les métadonnées.

`mysqlshow` peut aussi être utilisé à la place.

Show all tables

```
USE `database`;
SHOW TABLES;

SHOW TABLES FROM `database`;
```

Les deux formes sont équivalentes.

Appliquer un filtre

la syntaxe est la même que pour les bases :

```
SHOW TABLES LIKE `expression`;
SHOW TABLES WHERE condition;
```

De plus, par défaut `SHOW TABLES` ne retourne que la colonne du nom des tables. Le mot `FULL` permet d'en ajouter une deuxième appelée `Table_type` :

```
SHOW FULL TABLES;
```

Elle peut contenir trois valeurs différentes : 'BASE TABLE' pour les tables, 'VIEW' pour les vues, et 'SYSTEM VIEW' pour les tables spéciales du serveur (généralement celles de la base *INFORMATION_SCHEMA*).

Donc pour lister les vues :

```
SHOW FULL TABLES WHERE `Table_type`='VIEW';
```

Filtrer les tables ouvertes

La liste des tables non temporaires (sans les vues) ouvertes dans le cache :

```
SHOW OPEN TABLES;
```

Lister les champs

Les commandes suivantes correspondent aux informations de la table *COLUMNS* de *INFORMATION_SCHEMA*.

`mysqlshow` le permet également.

DESCRIBE

```
USE `base`;
DESCRIBE `table`;
-- ou
DESCRIBE `base`.`table`;
```

Le résultat contient six colonnes :

Field	Type	Null	Key	Default	Extra
...	...	...	...	...	...

`DESC` est un alias de `DESCRIBE`.

```
USE `base`;
DESC `table` 'filtre';
```

'filtre' peut être un nom de colonne. S'il est spécifié, seule cette colonne sera affichée. Si 'filtre' contient '%' ou '_', il sera évalué comme une condition `LIKE`. Par exemple, pour obtenir tous les champs commençant par 'wiki' :

```
DESC `table` 'wiki%';
```

EXPLAIN

Synonyme de `DESC` :

```
EXPLAIN `table`;
```

SHOW FIELDS

Autre synonyme de `DESC` :

```
SHOW FIELDS FROM `table`;
```

Remarque : le mot FULL rajoute une colonne "Privileges" et une "Comment" :

```
SHOW FULL FIELDS FROM `table`;
```

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
...	...	...	...	...	...	...	...	...

La colonne des commentaires peut servir à expliquer la signification du nom d'un champ, en apparaissant sous ce dernier lors des sélection dans PhpMyAdmin, et peut même être interprétée par certains logiciels. Par exemple, l'ORM Doctrine comprend le commentaire "DC2Type:array" pour choisir comment désérialiser les tableaux JSON stockés en LONGTEXT.

SHOW COLUMNS

Autre synonyme de `DESC` :

```
SHOW COLUMNS FROM `table`;
```

En fait `FIELDS` et `COLUMNS` sont synonymes. EXPLAIN et DESC ne supportent pas toutes leurs clauses (filtre).

De plus, les syntaxes ci-dessous sot équivalentes :

```
SHOW COLUMNS FROM `table` FROM `base`;
-- ou
SHOW COLUMNS FROM `base`.`table`;
```

Lister les indexes

Les commande suivantes renseignent sur les indexes d'une table, ses clés. Elles sont aussi dans la table `COLUMNS` de *INFORMATION_SCHEMA*, et accessibles via `mysqlshow -k`.

```
SHOW INDEX FROM `TABLE`;
SHOW INDEX FROM `TABLE` FROM `bases`;
```

Exemple de résultat :

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
Table1	0	PRIMARY	1	id	A	19	NULL	NULL		BTREE		

Le mot `KEYS` est synonyme de `INDEX`. Aucune autre clause n'est possible avec.

> **Attention !**
>
> Avec phpMyAdmin il est facile de créer plusieurs fois le même index, ce qui ralentit ensuite toutes les requêtes.

Pour supprimer un index :

```
DROP INDEX `date_2` on `Table1`
```

Un alias existe aussi : `SHOW KEYS FROM `TABLE`;`.

Lister les clés étrangères

Pour le nom des clés étrangères d'une table :

```
SELECT column_name, constraint_name FROM `KEY_COLUMN_USAGE` where table_name = 'maTable'
```

Spécifier les noms

Afin de distinguer les variables des mots réservés, on place les identificateurs MySQL (noms des tables, champs, et bases) entre deux accents graves (`). Il s'agit du caractère ASCII 96, disponible sous Linux en pressant les deux touches ALT + '.

Généralement il est optionnel, mais il permet de meilleurs messages d'erreur, par exemple :

```
mysql> SELECT user_id, group_id FROM user,group LIMIT 1;
ERROR 1064 (42000): You have an error in your SQL syntax;
  check the manual that corresponds to your MySQL server version
  for the right syntax to use near 'group LIMIT 1' at line 1
```

vs :

```
mysql> SELECT user_id, group_id FROM `user`,`group` LIMIT 1;
ERROR 1146 (42S02): Table 'savannah.group' doesn't exist
```

Montre qu'il manquait un *s* à *group*:

```
mysql>  SELECT user_id, group_id FROM `user`,`groups` LIMIT 1;
+---------+----------+
| user_id | group_id |
+---------+----------+
|     100 |        2 |
+---------+----------+
1 row in set (0.02 sec)
```

Cette syntaxe autorise l'utilisateur à employer des mots réservés dans leurs noms d'objets. On peut même utiliser des accents graves en es tapant deux fois, à la manière des caractères d'échappement :

```
RENAME TABLE `user` TO ````
```

Par contre cette syntaxe n'est pas portable, car le standard SQL recommande le guillemet (").

Syntaxe

Principe

- Le LDD est composé de CREATE, ALTER et DROP. Il permet d'ajouter, modifier et supprimer les structures logiques qui contiennent les données, ou autorisent les utilisateurs à y accéder ou à les maintenir (bases, MaTables, vues, clés...). le LDD concerne les métadonnées.
- Le LMD est constitué de INSERT, UPDATE et DELETE. Pour ajouter, modifier et supprimer les données stockées dans les bases.
- Le LCD représente GRANT et REVOKE. Il s'agit de la sécurité de la base, des permissions des utilisateurs.

On peut aussi distinguer deux autres catégories :

- Le DQL (Data Query Language : langage de requête de données), comme SELECT, SHOW et HELP. Ils sont rattachés au LMD dans le modèle traditionnel.
- Le LCT (en anglais DTL ou Data Transaction Language : langage de transaction de données) avec START TRANSACTION, SAVEPOINT, COMMIT et ROLLBACK [TO SAVEPOINT]. Affiliable au LCD dans le modèle à trois catégories.

Visualisation

```
SELECT * FROM MaTable
SELECT * FROM MaTable1, MaTable2, ...
SELECT champ1, champ2, ... FROM MaTable1, MaTable2, ...
SELECT ... FROM ... WHERE condition
SELECT ... FROM ... WHERE condition GROUPBY champ
SELECT ... FROM ... WHERE condition GROUPBY champ HAVING condition2
SELECT ... FROM ... WHERE condition ORDER BY champ1, champ2
SELECT ... FROM ... WHERE condition ORDER BY champ1, champ2 DESC
SELECT ... FROM ... WHERE condition LIMIT 10
SELECT DISTINCT champ1 FROM ...
SELECT DISTINCT champ1, champ2 FROM ...
```

Jointures

```
SELECT ... FROM t1 JOIN t2 ON t1.id1 = t2.id2 WHERE condition
SELECT ... FROM t1, t2 WHERE t1.id1 = t2.id2 AND condition
SELECT ... FROM t1 INNER JOIN t2 ON (t1.id1 = t2.id2) WHERE condition
SELECT ... FROM t1 NATURAL JOIN t2 WHERE condition
SELECT ... FROM t1 LEFT JOIN t2 ON t1.id1 = t2.id2 WHERE condition
SELECT ... FROM t1 JOIN (t2 JOIN t3 ON ...) ON ...
```

Conditions

```
champ1 = valeur1
champ1 <> valeur1
champ1 LIKE 'valeur _ %'
champ1 IS NULL
champ1 IS NOT NULL
champ1 IS IN (valeur1, valeur2)
champ1 IS NOT IN (valeur1, valeur2)
champ1 BETWEEN valeur1 AND valeur2
condition1 AND condition2
condition1 OR condition2
```

Modification du contenu

```
INSERT INTO MaTable1 (champ1, champ2, ...) VALUES (valeur1, valeur2, ...)

DELETE FROM MaTable1 / TRUNCATE MaTable1
DELETE FROM MaTable1 WHERE condition
-- jointure :
DELETE FROM MaTable1, MaTable2 WHERE MaTable1.id1 = MaTable2.id2 AND condition

UPDATE MaTable1 SET champ1=nouvelle_valeur1 WHERE condition
-- jointure :
UPDATE MaTable1, MaTable2 SET champ1=nouvelle_valeur1, champ2=nouvelle_valeur2, ... WHERE MaTable1.id1 = MaTable2.id2 AND condition
```

Naviguer dans MySQL

```
SHOW DATABASES
SHOW TABLES
SHOW INDEX FROM MaTable
SHOW FIELDS FROM MaTable / DESCRIBE MaTable
SHOW CREATE TABLE MaTable
SHOW PROCESSLIST
KILL numero
USE ma_bdd
```

Créer / supprimer une base

```
CREATE DATABASE MaBase
CREATE DATABASE MaBase CHARACTER SET utf8
DROP DATABASE `MaBase`

ALTER DATABASE MaBase CHARACTER SET utf8
```

Créer/supprimer/modifier une table

```
CREATE TABLE MaTable (champ1 type1, champ2 type2, ...)
CREATE TABLE MaTable (champ1 type1, champ2 type2, ..., INDEX (champ))
```

```
CREATE TABLE MaTable (champ1 type1, champ2 type2, ..., PRIMARY KEY (champ1))
CREATE TABLE MaTable (champ1 type1, champ2 type2, ..., PRIMARY KEY (champ1, champ2))
CREATE TABLE MaTable1 (fk_champ1 type1, champ2 type2, ...,
  FOREIGN KEY (fk_champ1) REFERENCES MaTable2 (t2_champA))
    [ON UPDATE|ON DELETE] [CASCADE|SET NULL]
CREATE TABLE MaTable1 (fk_champ1 type1, fk_champ2 type2, ...,
  FOREIGN KEY (fk_champ1, fk_champ2) REFERENCES MaTable2 (t2_champA, t2_champB))
CREATE TABLE MaTable IF NOT EXISTS (...)
CREATE TABLE MaTable (champ1 type1, champ2 type2, ...) SELECT ...

CREATE TEMPORARY TABLE MaTable (...)

DROP TABLE MaTable
DROP TABLE IF EXISTS MaTable
DROP TABLE MaTable1, MaTable2, ...

ALTER TABLE MaTable ADD (champ1 type1, champ2 type2, ...)
ALTER TABLE MaTable MODIFY champ1 type1
ALTER TABLE MaTable MODIFY champ1 type1 NOT NULL ...
ALTER TABLE MaTable CHANGE ancien_nom_champ1 nouveau_nom_champ1 type1
ALTER TABLE MaTable CHANGE ancien_nom_champ1 nouveau_nom_champ1 type1 NOT NULL ...
ALTER TABLE MaTable ALTER champ1 SET DEFAULT ...
ALTER TABLE MaTable ALTER champ1 DROP DEFAULT
ALTER TABLE MaTable ADD INDEX (champ);
DROP INDEX champ ON MaTable;

ALTER TABLE ancien_nom RENAME nouveau_nom;
```

Clés primaires et étrangères

Lecture

Pour lister les clés d'une table :

```
SHOW CREATE TABLE MaTable
```

ou

```
SELECT *
FROM  `information_schema`.`TABLE_CONSTRAINTS`
WHERE `TABLE_NAME` = 'MaTable'
```

Création

```
CREATE TABLE MaTable (..., PRIMARY KEY (champ1, champ2))
CREATE TABLE MaTable (..., FOREIGN KEY (champ1, champ2) REFERENCES MaTable2 (t2_champ1, t2_champ2))
```

Pour ajouter une clé étrangère à une table existante :

```
ALTER TABLE MaTable ADD FOREIGN KEY (maTable2_id) REFERENCES maTable2(id);
```

Remarque : dans PhpMyAdmin, ceci peut être fait à la souris en parcourant des menus déroulant : menu "Structure" de la table, puis "Relation view".

Facultativement, la contrainte peut aussi être nommée : on la baptise après le mot `CONSTRAINT`.

```
ALTER TABLE MaTable ADD CONSTRAINT fk_maTable2_id FOREIGN KEY (maTable2_id) REFERENCES maTable2(id);
```

Suppression

Pour désactiver les contraintes le temps d'une session :

```
SET FOREIGN_KEY_CHECK = 0;
```

Pour le faire globalement :

```
SET GLOBAL FOREIGN_KEY_CHECKS = 0;
```

Pour faire supprimer définitivement une contrainte :

```
ALTER TABLE MaTable1
DROP FOREIGN KEY FK_MaTable1_MaTable2
```

Créer/supprimer une vue

```
 CREATE VIEW nomvue AS SELECT champ1, champ2 FROM MaTable1              -- ou
 CREATE VIEW nomvue (champ1, champ2...) AS SELECT champ1, champ2 FROM MaTable1

 ALTER VIEW nomvue (champ1, champ2...) AS SELECT champ2 FROM MaTable1;
 DROP VIEW nomvue;
```

Il est impossible d'ajouter un index à une vue[2].

Permissions

```
GRANT ALL PRIVILEGES ON base.* TO 'utilisateur'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT, DELETE ON base.* TO 'utilisateur'@'localhost' IDENTIFIED BY 'password';
REVOKE ALL PRIVILEGES ON base.* FROM 'utilisateur'@'hôte'; -- une seule permission
```

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'utilisateur'@'hôte'; -- toutes les permissions

SET PASSWORD = PASSWORD('nouveau_pass')
SET PASSWORD FOR 'utilisateur'@'hôte' = PASSWORD('nouveau_pass')
SET PASSWORD = OLD_PASSWORD('nouveau_pass')

DROP USER 'utilisateur'@'hôte'
```

Oubli de mot de passe oublié

```
$ service mysql stop
$ mysqld_safe --skip-grant-MaTables
> UPDATE mysql.user SET password=PASSWORD('nouveau') WHERE user='root';
## Tuer mysqld_safe, avec Control + \
$ service mysql start
```

Réparer les tables après un arrêt soudain

```
mysqlcheck --all-databases
mysqlcheck --all-databases --fast
```

Relancer la synchronisation de la base du serveur secondaire

```
$ mysql
mysql> slave start;
mysql> show slave status\G
```

Manipuler des variables

Les définitions sont effectuées à l'aide des mots clés "select" (suivi de ":=") ou "set" (avec "=") :

```
SELECT @test := 2;
SELECT @test + 1

SET @date1='date une', @date1='date deux'
```

Pour les afficher ensuite :

```
show variables like 'test';
show variables like 'date1';
show variables like 'date2';
```

Certaines variables globales représentent la configuration du système, et peuvent être changées provisoirement le temps d'une session, ou de façon permanente :

```
mysql> set @@global.max_connections = 1000;
mysql> show global variables like 'wait_timeout';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| wait_timeout | 60 |
+---------------+-------+
1 row in set (0.00 sec)
mysql> set @@session.wait_timeout=120;
```

Attention !

En cas de conversion de dates au format *Text* en *Datetime*, cela les efface toutes à *0000-00-00 00:00:00*

Références

1. https://dev.mysql.com/doc/refman/5.7/en/system-database.html
2. http://dev.mysql.com/doc/refman/5.7/en/view-restrictions.html

Variables utilisateurs

Variables locales

Les variables locales ne peuvent pas être lues en dehors de leur fonction ou procédure stockée[1].

Elles sont déclarées après DECLARE avec leur nom, leur type, et éventuellement leur valeur par défaut[2] :

```
DECLARE MaVariable1 INT DEFAULT 1;
```

Variables de session

Les variables obéissent à certaines règles :

- Leurs noms commencent par "@" (ex : @total).
- Elles sont déclarées avec le mot SET, ou bien SELECT accompagné de l'opérateur d'assignation :=.
- Une variable définie dans la liste de champ ne peut pas être utilisée comme une condition.
- Les variables de session durent le temps du thread.

```
select @test := 2;
select @test + 1;  -- renvoie 3

set @datedebut='date_de_debut', @datefin='date_de_fin';

SELECT @nbmembre:=count(*) FROM membres;

select @numzero := count(*) from table1 where field=0;
select @numdistint := count(distinct field) from table1 where field <> 0 ;
select @numzero @numdistinct;
```

Pour copier dans valeurs d'une sélection dans une ou plusieurs variables :

```
SET @id = 0, @nom = '';
SELECT id, nom INTO @id, @nom FROM table1 LIMIT 1;
SELECT @id, @nom;
```

Elles peuvent être utiles quand on doit agréger plusieurs valeurs sans jointures entre leurs tables. Ex :

```
SET @idCountry = (SELECT `id` FROM `country` WHERE `code` = "FR" LIMIT 1);
SET @idLanguage = (SELECT `id` FROM `language` WHERE `code` = "fr" LIMIT 1);

INSERT INTO `page` (`country`, `language`, `description`)
VALUES (@idCountry, @idLanguage, 'Text')
```

Variables globales

Une variable globale est visible pour tous les utilisateurs, elle est précédée de "@@".

Elles peuvent modifier les fichiers de configuration définitivement pendant la session. Donc en les changeant, il est nécessaire de préciser le critère définitif ou éphémère, en distinguant *set global* et *set session*.

Exemple :

```
mysql> set @@global.max_connections = 1000;
mysql> show global variables like 'wait_timeout';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| wait_timeout | 60 |
+---------------+-------+
1 row in set (0.00 sec)
mysql> set @@session.wait_timeout=120;
```

sql_mode

Un script peut avoir un comportant différent sur deux bases tournant sur la même version de MySQL. Par exemple il est possible d'imposer de préciser dans le GROUP BY toutes les variables sélectionnées à regrouper avec ONLY_FULL_GROUP_BY.

Ce paramétrage est visible avec sql_mode[3] :

```
SELECT @@sql_mode;
```

```
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

Références

1. http://stackoverflow.com/questions/1009954/mysql-variable-vs-variable-whats-the-difference
2. http://dev.mysql.com/doc/refman/5.7/en/declare-local-variable.html
3. https://dev.mysql.com/doc/refman/5.7/en/sql-mode.html

Alias

Une expression ou une colonne peut être baptisée avec `AS`. Cet alias est utilisé comme nom de colonne et peut donc être nommé dans les clauses des requêtes. Exemple :

```
SELECT
    CONCAT(nom,' ', prenom) AS nom_complet,
    pseudonyme AS pseudo
FROM
    table1
ORDER BY
    nom_complet;
```

Ces alias fonctionnent avec `ORDER BY`, `GROUP BY` et `HAVING`, mais pas `WHERE`.

Cela peut aussi servir à raccourcir les noms des tables employées comme préfixes.

```
SELECT
    COUNT(R.ID_reservation), U.Localisation
FROM
    Utilisateurs U
LEFT OUTER JOIN
    Reservations AS R
ON
    U.ID_Utilisateur    = R.ID_Utilisateur AND
    R.ID_Projet = '10'
GROUP BY
    U.Localisation;
```

De plus les alias peuvent jouer un rôle crucial pour les auto-jointures. Par exemples ci-dessous, la table *personne* est référencée par *p* et *c* :

```
SELECT
    p.nom AS parent,
    e.nom AS enfant,
    MIN((TO_DAYS(NOW())-TO_DAYS(e.date_naissance))/365) AS agemini
FROM
    personne AS p
LEFT JOIN
    personne AS e
ON
    p.nom=e.parent WHERE e.nom IS NOT NULL
GROUP BY
    parent HAVING agemini > 50 ORDER BY p.date_naissance;
```

Types de données

Types de données principaux

Voici les valeurs acceptées avant débordement[1] :

```
TINYINT (1o : -127+128)
SMALLINT (2o : +-65 000)
MEDIUMINT (3o : +-16 000 000)
INT (4o : +- 2 000 000 000)
BIGINT (8o : +- 9 trillions)
   Intervalle précis : -(2^(8*N-1)) -> (2^8*N)-1
   /!\ INT(2) = "2 chiffres affichés" -- ET NON PAS "nombre à 2 chiffres"

FLOAT(M,D) DOUBLE(M,D) FLOAT(D=0->53)
   /!\ 8,3 -> 12345,678 -- PAS 12345678,123!

TIME (HH:MM)
YEAR (AAAA)
DATE (AAAA-MM-JJ)
DATETIME (AAAA-MM-JJ HH:MM; années 1000->9999)
TIMESTAMP (comme date, mais 1970->2038, compatible Unix)

VARCHAR(ligne)
TEXT (multi-lignes; taille max=65535)
BLOB (binaire; taille max=65535)

Variantes :
TINY (max=255)
MEDIUM (max=~16000)
LONG (max=4Go)
   Ex : TINYTEXT, LONGBLOB, MEDIUMTEXT

ENUM ('valeur1', 'valeur2', ...) -- (default NULL, ou '' si NOT NULL)
```

Attention !

Il faut préférer DECIMAL(10,2) à FLOAT car ce dernier peut se révéler imprécis. Ex : 39.99 x 1 = 39.9900016784668.

VARCHAR

`VARCHAR` est l'abréviation de `CHARACTER VARYING` (*caractère variant* en anglais). 'n' représente la taille maximum de colonne (jusqu'à 65 535 caractères). Par exemple, une colonne de type `VARCHAR(10)` peut contenir 10 caractères maximum. La taille du stockage correspondant en fait à la taille du texte contenu (L), plus un ou deux octets (un si la taille est inférieure à 255).

Par exemple pour la chaine "abcd", L = 4 et le stockage = 5.

`CHAR(n)` est similaire à `VARCHAR(n)` sauf qu'il occupe une taille fixe, il ne tient pas compte de son contenu.

TEXT et BLOB

Les types `TEXT` et `BLOB` (binary large object) ont une taille maximum de 65 535 caractères. L'espace requis est la taille réelle des données stockées, plus un ou deux octets (un si < 255). Comme elles ne sont pas stockées dans le fichier de données, toutes les opérations (`INSERT / UPDATE / DELETE / SELECT`) les concernant sont plus lentes, mais cela a l'avantage de rendre celles qui ne les touchent pas plus rapides.

Toutefois le `BLOB` possède plusieurs déclinaisons[2] :

- `TINYBLOB` : 255 o.
- `MEDIUMBLOB` : 16 Mo.
- `LONGBLOB` : 4,29 Go.

INTEGER

Spécifier une valeur n n'a aucun effet. De toute façon, la taille maximum est des données stockées est de 429 fois 10^7.

Pour les nombres uniquement positifs, utiliser `UNSIGNED`, sinon `SIGNED`.

Attention !

En rentrant un nombre supérieur à la limite (ex : 1234567890123456789), le logiciel dira qu'une ligne a été affectée mais en fait il ne modifie pas le champ. Pour pallier cela, modifier le type en `BIGINT`.

Remarque : les booléens sont des `tinyint(1)`.

Le nombre entre parenthèses après les types entiers indique sur combien de chiffres l'entier stocké est prévu pour être affiché[3]. Toutefois s'il est plus long que cela, cela n'empêchera pas son stockage.

DECIMAL

Syntaxe : `DECIMAL(n,m)`.

Ex : `DECIMAL(4,2)` signifie des nombres jusqu'à 99,99 (quatre chiffres dont deux réservés aux décimales).

DATE

Il existe trois types pour stocker des dates : `DATETIME`, `DATE`, et `TIMESTAMP`.

MySQL récupère et affiche les dates au format "AAAA-MM-JJ" (plus pratique pour les classer de gauche à droite).

`DATETIME` est utilisé quand les valeurs doivent contenir l'heure en plus du jour.

La différente entre `DATETIME` et `TIMESTAMP` est que la taille des `TIMESTAMP` est limitée aux années 1970-2037.

Le type `TIME` peut stocker les heures du jour (HH:MM:SS) sans date. Il peut aussi représenter une période de temps (ex : -02:00:00 pour deux heures avant). Limité entre '-838:59:59' et '838:59:59'.

YEAR peut stocker des années.

Pour manipuler des dates, il faut préciser un jour et pas seulement une heure, car pourrait interpréter "HH:MM:SS" comme une valeur "YY:MM:DD".

Les exemples suivant montrent la plage de date précise pour les temps Unix, démarrant à l'époque Unix jusqu'à 2038 ($2^{31}-1$).

```
mysql> SET time_zone = '+00:00'; -- GMT
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT FROM_UNIXTIME(-1);
+-------------------+
| FROM_UNIXTIME(-1) |
+-------------------+
| NULL              |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT FROM_UNIXTIME(0); -- "Epoch"
+---------------------+
| FROM_UNIXTIME(0)    |
+---------------------+
| 1970-01-01 00:00:00 |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT FROM_UNIXTIME(2145916799);
+---------------------------+
| FROM_UNIXTIME(2145916799) |
+---------------------------+
| 2037-12-31 23:59:59       |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT FROM_UNIXTIME(2145916800);
+---------------------------+
| FROM_UNIXTIME(2145916800) |
+---------------------------+
| NULL                      |
+---------------------------+
1 row in set (0.00 sec)
```

Les fonctions de manipulation de date seront exposées dans un chapitre ultérieur.

SET et ENUM

SET est un type dont les valeurs sont prédéfinies dans une liste lors de la création de la table[4].

ENUM est similaire mais restreint à un seul membre, alors que SET autorise le stockage de n'importe lesquelles de ses valeurs ensemble.

Exemple :

```
SET("madame", "monsieur")    -- autorise un champ vide, "madame", "monsieur", "madame, monsieur", ou "monsieur, madame"

ENUM("madame", "monsieur")   -- autorise un champ vide, "madame" ou  "monsieur"
```

Références

1. https://dev.mysql.com/doc/refman/5.5/en/integer-types.html
2. Christian Soutou, *Apprendre SQL avec MySQL : Avec 40 exercices corrigés*, Editions Eyrolles, 7 juillet 2011 (lire en ligne (https://books.google.fr/books?id=4_5GPcYUh2AC&pg=PA29&lpg=PA29&dq=tinyblob+longblob&source=bl&ots=O-7uyDYHkl&sig=fJJJc0bXVjYNmKhJ2LUHvEjbm7I&hl=fr&sa=X&ved=0ahUKEwjq0uCx7eXKAhXMnRoKHVfkDPEQ6AEIYDAI#v=onepage&q=tinyblob%20longblob&f=false))
3. https://dev.mysql.com/doc/refman/5.7/en/numeric-type-attributes.html
4. http://dev.mysql.com/doc/refman/5.0/fr/set.html

Manipulation de base

Création

```
CREATE DATABASE Nom_de_la_base;
```

`mysqladmin create` permet de le faire en ligne de commande.

NB : dans MySQL, `CREATE SCHEMA` est un parfait synonyme de `CREATE DATABASE`, contrairement à d'autres SGBD comme Oracle ou SQL Server.

Suppression

```
DROP DATABASE Nom_de_la_base;
```

`mysqladmin drop` permet de le faire en ligne de commande. Le paramètre `-f` force celle-ci sans poser de question.

Renommage

Dans les versions 5.1.x il existait une commande `RENAME DATABASE db1 TO db2;`, mais elle a été retirée suite à des pertes de données[1].

Il reste toutefois la ligne de commande pour le faire en plusieurs étapes :

```
mysqladmin create Nom_de_la_nouvelle_base
mysqldump --opt Nom_de_la_base | mysql Nom_de_la_nouvelle_base
mysqladmin drop -f Nom_de_la_base
```

Une autre option avec les droits root, est de renommer le répertoire de la base :

```
cd /var/lib/mysql/
/etc/init.d/mysql stop
mv Nom_de_la_base/ Nom_de_la_nouvelle_base/
/etc/init.d/mysql start
```

Après renommage, il convient de migrer les permissions :

```
UPDATE mysql.db SET `Db`='Nom_de_la_nouvelle_base' WHERE `Db`='Nom_de_la_base';
FLUSH PRIVILEGES;
```

Copie

Avec mysqldump

`mysqldump` peut sauvegarder les bases, il suffit de réinjecter son résultat dans d'autres bases.

```
# Premièrement, nettoyer la base de destination :
mysqladmin drop -f base2
mysqladmin create base2
# Ensuite, copier la base1 dans la base2 :
mysqldump --opt base1 | mysql base2
```

Backup

Pour définir le backup automatique d'une base tous les soirs à minuit[2], sous Linux :

```
$ crontab -e
0 0 * * * /usr/local/bin/mysqldump -uLOGIN -PPORT -hHOST -pPASS base1 | gzip -c > `date “+\%Y-\%m-\%d”`.gz
```

Avec des outils de modélisation

Ces logiciels permettent de représenter les tables sous formes de diagrammes.

phpMyAdmin

> *Pour plus de détails voir : **phpMyAdmin**.*

MySQL Workbench

MySQL Workbench permet également la migration depuis d'autres bases de données, telles que Microsoft SQL Server[3].

Par rapport à phpMyAdmin, il a l'inconvénient de devoir être installé, mais a l'avantage de pouvoir modifier des tables en changeant de champ au clavier, comme dans un tableur.

DBDesigner

DBDesigner est en licence GNU GPL, mais ne peut pas être considéré comme un freeware car il requiert un compilateur Kylix non gratuit.

Il rencontre une erreur de connexion à MySQL sur la version 4 : *unable to load libmysqlclient.so*. Pour la résoudre :

- Installer les "Shared compatibility libraries" Télécharger (http://dev.mysql.com/downloads/mysql/5.0.html#downloads) MySQL pour version 5.0).

Sous Linux :

- Remplacer le fichier *libmysqlclient.so* de DBDesigner par le nouveau :

```
sudo ln -sf /usr/lib/libmysqlclient.so.10 /usr/lib/DBDesigner4/libmysqlclient.so
```

- Trouver et installer `kylixlibs3-unwind-3.0-rh.4.i386.rpm`
- Trouver un vieux xorg (ex : `xorg-x11-libs-6.8.2-37.FC4.49.2.1.i386.rpm` depuis FC4) et l'extraire :

```
rpm2cpio x.rpm | cpio -i
```

- Récupérer *libXft.so.1.1* dans ce package et l'installer :

```
sudo cp libXft.so.1.1 /usr/lib
ldconfig
```

Maintenant DBDesigner4 peut se connecter à MySQL5.

Kexi

Il existe aussi Kexi de Calligra Suite, téléchargeable sur http://userbase.kde.org/Calligra/Download/fr.

OpenOffice Base et ODBC

Configuration typique :

- Soit une base MySQL appelée `mysqlhost`.
- OpenOffice.org sur la machine cliente (Debian GNU/Linux dans l'exemple).
- Connexion via ODBC.

Sur le client, installer `mysql-client` :

```
aptitude install mysql-client
```

Sous Fedora/CentOS :

```
yum install mysql
```

Avant d'installer ODBC, test la connexion distante localement :

```
$ mysql -h mysqlhost -u user1 mysqldatabase -p
Enter password: PassUser1
```

Il faut créer la base `mysqldatabase` et l'utilisateur `user1` sur `mysqlhost`.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysqldatabase      |
+--------------------+
2 rows in set (0.00 sec)
....
mysql> quit;
Bye
```

Toujours sur la machine cliente :

```
aptitude install libmyodbc unixodbc
```

Pour Fedora/CentOS :

```
yum install mysql-connector-odbc unixODBC
```

Les fichiers `/etc/odbc.ini` et `/etc/odbcinst.ini` sont créés.

`odbcinst.ini` déclare le pilote ODBC disponible. Exemple pour Debian :

```
[MySQL]
Description     = MySQL driver
Driver          = /usr/lib/odbc/libmyodbc.so
Setup           = /usr/lib/odbc/libodbcmyS.so
CPTimeout       =
CPReuse         =
FileUsage       = 1
```

for CentOS:

```
[MySQL]
Description     = ODBC for MySQL
Driver          = /usr/lib/libmyodbc3.so
Setup           = /usr/lib/libodbcmyS.so
FileUsage       = 1
```

Maintenant `odbcinst` est utilisable :

```
# odbcinst -j
unixODBC 2.2.4
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
USER DATA SOURCES..: /root/.odbc.ini
```

Pour d'autres options : `man odbcinst`

Il faut créer au moins un DSN (*Data Source Name* ou *Data Set Name*), parce que chaque connexion ODBC avec OOo est initialisée avec.

Pour créer un DSN, il existe différente possibilités :

- Modifier `/etc/odbc.ini` (concerne tous les utilisateurs)
- Modifier `~/.odbc.ini` (concerne un seul utilisateur)
- Utilise les applications graphiques comme **ODBCConfig** (Debian : `unixodbc-bin`, Fedora : `unixODBC-kde`).

Finalement, ces applications graphiques modifient **/etc/odbc.ini** ou **~/.odbc.ini**.

Par exemple, un fichier `/etc/odbc.ini` (le nom du DSN est entre crochets []) :

```
[MySQL-test]
Description     =       MySQL ODBC Database
TraceFile       =       stderr
Driver          =       MySQL
SERVER          =       mysqlhost
USER            =       user1
PASSWORD        =
DATABASE        =       mysqldatabase
```

Dans ce cas, le DSN est appelé **MySQL-test**.

Ensuite pour tester, utiliser la commande **isql** :

```
$ isql -v MySQL-test user1 PassUser1
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> show databases;
+------------------+
| Database         |
+------------------+
| information_schema|
| mysqldatabase     |
+------------------+
2 rows affected
2 rows returned
SQL> quit;
```

Depuis OOo :

```
-> File
 -> New
  -> Database
-> Connecting to an existing database
 -> MySQL
   -> Next
-> Connect using ODBC
 -> Next
-> Choosing a Data Source
 -> MySQL-test
  -> Next
-> Username : user1 (tick password required)
-> Yes, register the database for me
-> Finish
```

A ce stade, le programme est connecté à la base *mysqldatabase* en tant que *user1*. Il reste donc le mot de passe à rentrer.

Ensuite, Java est requis dans les Wizards uniquement (lors de création directe JRE est inutile) :

- Wizard pour créer un formulaire.
- Wizard pour créer des rapports.
- Wizard pour créer des requêtes.
- Wizard pour créer tables.

Les distributions GNU/Linux fournissent généralement OpenOffice avec IcedTea (`openjdk-6-jre`/`java-1.6.0-openjdk`) ou GCJ (`java-gcj-compat`/`java-1.4.2-gcj-compat`) donc les fonctionnalités basées sur du Java fonctionnent.

Restauration

- Sous Linux, le mot de passe est demandé après entrée de la commande :

```
mysql -h localhost -u root -p MaBase < MaBase.sql
```

- Sous Windows, par défaut le compte root n'a pas de mot de passe et MySQL n'est pas dans les variables d'environnement donc on utilise son chemin absolu :

```
"C:\Program Files (x86)\EasyPHP\binaries\mysql\bin\mysql.exe" -h localhost -u root MaBase < MaBase.sql
```

Contrairement aux importations de PhpMyAdmin il n'y a pas de limite. Par exemple on peut charger une base de 2 Go en cinq minutes.

Références

1. https://dev.mysql.com/doc/refman/5.1/en/rename-database.html
2. http://stackoverflow.com/questions/6645818/how-to-automate-database-backup-using-phpmyadmin
3. http://www.thegeekstuff.com/2014/03/mssql-to-mysql/

Manipulation de table

CREATE TABLE

La syntaxe de création des tables d'une base est ainsi :

```
Create table tablename (FieldName1 DataType, FieldName2 DataType)
```

Les enregistrements de la requête `SELECT` peuvent être enregistrés dans une nouvelle table. Les types des données seront les mêmes que dans l'ancienne table. Exemple :

```
CREATE TABLE LearnHindi
select english.tag, english.Inenglish as english, hindi.Inhindi as hindi
FROM english, hindi
WHERE english.tag = hindi.tag
```

De plus, MySQL peut assurer l'auto-incrémentation des clés uniques grâce à l'option `AUTO_INCREMENT`. En cas de troncature de la table, le compteur peut être réinitialiser avec :

```
ALTER TABLE tablename AUTO_INCREMENT = 1
```

Tables d'archive

MySQL propose un type de table d'archive, prenant moins de place (par compression) mais dont on ne peut pas supprimer les enregistrements une fois ajoutés :

```
create table t1 (
 a int,
 b varchar(32))
ENGINE=ARCHIVE
```

Tables temporaires

Il est possible de créer des variables de type table, qui seront effacées à la fin de leurs scripts. On les appelle "tables temporaires" :

```
CREATE TEMPORARY TABLE IF NOT EXISTS MaTableTemp1 AS (SELECT * FROM MaTable1)
```

Exemple avec paramètre nommé :

```
CREATE TEMPORARY TABLE IF NOT EXISTS MaTableTemp1(id INT) AS (SELECT id FROM MaTable1)
```

Attention !

Si le nom de la colonne ne correspond pas au nom du champ sélectionné, la table temporaire se voit ajouter une colonne du nom de ce champ. Ex :

```
CREATE TEMPORARY TABLE IF NOT EXISTS MaTableTemp1(id1 INT) AS (SELECT id FROM MaTable1);
SHOW FIELDS FROM MaTableTemp1;
```

```
Field   Type    Null    Key     Default         Extra
id1     int(11)         YES             NULL
id      int(11)         NO              0
```

Attention !

Toutes les tables temporaires sont supprimées à la fin de la connexion MySQL qui les a créée[1].

Copier une table

Pour obtenir la même structure (noms et types des champs, index, mais aucun enregistrement) :

```
CREATE TABLE `new1` LIKE `old1`;
```

Pour dupliquer le contenu d'une table dans le résultat :

```
INSERT INTO `new1` SELECT * FROM `old1`;
```

Remarque : la limite de taille pour une table dépend du système de fichier, elle est généralement de 2 To[2]

ALTER TABLE

`ALTER TABLE` sert à ajouter, supprimer ou modifier la structure des tables (colonnes, index, propriétés).

Ajouter une colonne

```
ALTER TABLE awards
ADD COLUMN AwardCode int(2)
```

Modifier une colonne

Pour changer les caractéristiques :

```
ALTER TABLE awards
```

```
CHANGE COLUMN AwardCode VARCHAR(2) NOT NULL

ALTER TABLE awards
MODIFY COLUMN AwardCode VARCHAR(2) NOT NULL
```

Pour renommer une colonne :

```
ALTER TABLE awards CHANGE `AwardCode` `newcAwardCode` VARCHAR(2) NOT NULL;
```

Supprimer une colonne

```
ALTER TABLE awards
DROP COLUMN AwardCode
```

Reclasser les enregistrements d'une table

```
ALTER TABLE awards ORDER BY id
```

Remarque : cette opération n'est pas supportée par tous les moteurs de stockage. Elle peut accélérer certaines requêtes.

Renommer une table

Pour renommer une table, il faut préalablement retirer ses privilèges avec `ALTER` et `DROP`, puis `CREATE` et `INSERT` pour ceux à attribuer à la nouvelle table.

Renommage :

```
ALTER TABLE `old` RENAME `new`
```

Raccourci :

```
RENAME TABLE `old_name` TO `new_name`
```

Plusieurs :

```
RENAME TABLE `old1` TO `new1`, `old2` TO `new2`, ...
```

La différence entre `ALTER TABLE` et `RENAME` est que seul le premier peut renommer les tables temporaires, mais il n'en permet qu'un par requête.

DROP TABLE

```
DROP TABLE `awards`
```

Supprime toute la table (enregistrements et structure).

Plusieurs :

```
DROP TABLE `table1`, `table2`, ...
```

Avec vérification :

```
DROP TEMPORARY TABLE `table`;
DROP TABLE `table` IF EXISTS;
```

CASCADE

Certains enregistrements d'une base de données relationnelle peuvent devenir inutiles si ceux qui leur sont joints viennent à disparaitre.

C'est par exemple le cas dans une table "adresse de facturation" où il n'y n'aurait plus de personne physique ou morale associée, c'est-à-dire qu'il existerait en mémoire une ligne avec un id utilisateur pointant vers une ligne de la table "utilisateur" qui n'existe plus.

Pour éviter d'avoir à maintenir ces reliquats, MySQL offre la possibilité de les supprimer automatiquement "en cascade", au moment où ceux qui leur sont joints sont effacés. Cela se définit par dessus la contrainte d'intégrité `FOREIGN KEY`.

Exemple :

```
CREATE TABLE adresse_facturation (
  id int(11) NOT NULL AUTO_INCREMENT,
  id_utilisateur int(11) NOT NULL,
  adresse varchar(255),
  PRIMARY KEY (id),
  FOREIGN KEY (id_utilisateur)
  REFERENCES utilisateur (id)
  ON DELETE CASCADE
)
```

Unique

Une autre contrainte d'intégrité qui permet de forcer chaque valeur d'un champ à être différentes est UNIQUE :

Pour ajouter une contrainte unique du nom de la colonne concernée :

```
ALTER TABLE MaTable ADD UNIQUE (user_id)
```

Pour ajouter une contrainte unique nommée :

```
ALTER TABLE MaTable ADD UNIQUE KEY UNIQ_E6F03AD9A76ED395 (user_id)
```

Exemple pour travaux pratiques

Soit l'exemple suivant qui sera utilisé pour les sélections ensuite (toute ressemblance avec un framework connu est purement non fortuite : si vous avez déjà votre propre wiki, il est possible de sauter cette phase pour passer directement au paragraphe SELECT).

NB : le type *VARBINARY* est équivalent à *VARCHAR*, mais il faut savoir qu'il stocke la chaine de caractères sous sa forme binaire, et donc prend moins de place.

Création d'une base

```
CREATE DATABASE wiki1;
USE wiki1;

-- Liste des utilisateurs
CREATE TABLE IF NOT EXISTS `wiki1_user` (
  `user_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_name` varbinary(255) NOT NULL DEFAULT '',
  `user_real_name` varbinary(255) NOT NULL DEFAULT '',
  `user_password` tinyblob NOT NULL,
  `user_newpassword` tinyblob NOT NULL,
  `user_newpass_time` binary(14) DEFAULT NULL,
  `user_email` tinyblob NOT NULL,
  `user_touched` binary(14) NOT NULL DEFAULT '\0\0\0\0\0\0\0\0\0\0\0\0\0\0',
  `user_token` binary(32) NOT NULL DEFAULT '\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0',
  `user_email_authenticated` binary(14) DEFAULT NULL,
  `user_email_token` binary(32) DEFAULT NULL,
  `user_email_token_expires` binary(14) DEFAULT NULL,
  `user_registration` binary(14) DEFAULT NULL,
  `user_editcount` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `user_name` (`user_name`),
  KEY `user_email_token` (`user_email_token`),
  KEY `user_email` (`user_email`(50))
) ENGINE=InnoDB  DEFAULT CHARSET=binary AUTO_INCREMENT=41 ;

INSERT INTO `wiki1_user` (`user_id`, `user_name`, `user_real_name`, `user_password`, `user_newpassword`, `user_newpass_time`, `user_email`, `user_touched`, `user_token`,
`user_email_authenticated`, `user_email_token`, `user_email_token_expires`, `user_registration`, `user_editcount`) VALUES
(1, 'Utilisateur1', 'admin', '', '', NULL, '', '', '', '', '', '20130101', '20130101', 1000),
(2, 'Utilisateur2', '', '', '', NULL, '', '', '', '', '', '20130101', '20130101', 800),
(3, 'Bot1', 'admin', '', '', NULL, '', '', '', '', '', '20130101', '20130101', 5000),
(4, 'Utilisateur3', '', '', '', NULL, '', '', '', '', '', '20130102', '20130102', 500),
(5, 'Utilisateur4', '', '', '', NULL, '', '', '', '', '', '20130102', '20130102', 200);
(6, 'Utilisateur5', '', '', '', NULL, '', '', '', '', '', '20130103', '20130103', 200);

-- Liste des pages
CREATE TABLE IF NOT EXISTS `wiki1_page` (
  `page_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `page_namespace` int(11) NOT NULL,
  `page_title` varbinary(255) NOT NULL,
  `page_restrictions` tinyblob NOT NULL,
  `page_counter` bigint(20) unsigned NOT NULL DEFAULT '0',
  `page_is_redirect` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `page_is_new` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `page_random` double unsigned NOT NULL,
  `page_touched` binary(14) NOT NULL DEFAULT '\0\0\0\0\0\0\0\0\0\0\0\0\0\0',
  `page_latest` int(10) unsigned NOT NULL,
  `page_len` int(10) unsigned NOT NULL,
  PRIMARY KEY (`page_id`),
  UNIQUE KEY `name_title` (`page_namespace`,`page_title`),
  KEY `page_random` (`page_random`),
  KEY `page_len` (`page_len`),
  KEY `page_redirect_namespace_len` (`page_is_redirect`,`page_namespace`,`page_len`)
) ENGINE=InnoDB  DEFAULT CHARSET=binary AUTO_INCREMENT=8;

INSERT INTO `wiki1_page` (`page_id`, `page_namespace`, `page_title`, `page_restrictions`, `page_counter`, `page_is_redirect`, `page_is_new`, `page_random`, `page_touched`,
`page_latest`, `page_len`) VALUES
(1, 0, 'Accueil', '', 0, 0, 0, 0, '', 0, 0),
(2, 8, 'Sidebar', '', 0, 0, 0, 0, '', 0, 0),
(3, 0, 'MySQL', '', 0, 0, 0, 0, '', 0, 0),
(4, 0, 'PHP', '', 0, 0, 0, 0, '', 0, 0);


-- Propriétés des pages
CREATE TABLE IF NOT EXISTS `wiki1_page_props` (
  `pp_page` int(11) NOT NULL,
  `pp_propname` varbinary(60) NOT NULL,
  `pp_value` blob NOT NULL,
  UNIQUE KEY `pp_page_propname` (`pp_page`,`pp_propname`)
) ENGINE=InnoDB DEFAULT CHARSET=binary;

INSERT INTO `wiki1_page_props` (`pp_page`, `pp_propname`, `pp_value`) VALUES
(1, 'noindex', ''),
(2, 'defaultsort', ''),
(2, 'noindex', '');


-- Hyperliens dans les pages
CREATE TABLE IF NOT EXISTS `wiki1_pagelinks` (
  `pl_from` int(10) unsigned NOT NULL DEFAULT '0',
  `pl_namespace` int(11) NOT NULL DEFAULT '0',
  `pl_title` varbinary(255) NOT NULL DEFAULT '',
  UNIQUE KEY `pl_from` (`pl_from`,`pl_namespace`,`pl_title`),
  UNIQUE KEY `pl_namespace` (`pl_namespace`,`pl_title`,`pl_from`)
) ENGINE=InnoDB DEFAULT CHARSET=binary;

INSERT INTO `wiki1_pagelinks` (`pl_from`, `pl_namespace`, `pl_title`) VALUES
(1, 0, 'Lien1'),
(3, 0, 'Lien2');
```

Références

1. http://www.mysqltutorial.org/mysql-temporary-table/
2. http://dev.mysql.com/doc/refman/5.7/en/table-size-limit.html

Manipulation de données

INSERT

La syntaxe est la suivante :

```
INSERT INTO TableName (Column1, Column2, Column3)
VALUES (value1, value2, value3)
```

Ce qui effectue les opérations suivantes : `insert value1 into Column1, value2 into Column2, and value3 into Column3`.

Insérer un enregistrement (les valeurs sont insérées dans l'ordre où les colonnes apparaissent dans la base) :

```
INSERT INTO TableName
VALUES (value1, value2, value3)
```

Deux lignes :

```
INSERT INTO TableName
VALUES (value1, value2, value3), (value4, value5, value6)

INSERT INTO antiques VALUES (21, 01, 'Ottoman', 200.00);
INSERT INTO antiques (buyerid, sellerid, item) VALUES (01, 21, 'Ottoman');
```

Copier ceux d'une autre table :

```
INSERT INTO table1(field1, field2)
SELECT field1, field2
FROM table2

INSERT INTO World_Events SELECT * FROM National_Events
```

Astuces de performances :

- Pour insérer plusieurs lignes, utiliser `LOAD DATA INFILE` de préférence.
- Si un gros volume d'insertion est trop lent sur des tables indexées non vides, augmenter la valeur de `bulk_insert_buffer_size`.
- Avant des insertions en masse, retirer les clés.
- Verrouiller une table (`LOCK`) accélère les `INSERT`.

UPDATE

```
UPDATE table SET field1 = newvalue1, field2 = newvalue2 WHERE criteria ORDER BY field LIMIT n
```

Exemples :

```
UPDATE owner SET ownerfirstname = 'John'
  WHERE ownerid = (SELECT buyerid FROM antiques WHERE item = 'Bookcase');

UPDATE antiques SET price = 500.00 WHERE item = 'Chair';

UPDATE order SET discount=discount * 1.05

UPDATE tbl1 JOIN tbl2 ON tbl1.ID = tbl2.ID
  SET tbl1.col1 = tbl1.col1 + 1
  WHERE tbl2.status='Active'

UPDATE tbl SET names = REPLACE(names, 'aaa', 'zzz')

UPDATE products_categories AS pc
  INNER JOIN products AS p ON pc.prod_id = p.id
  SET pc.prod_sequential_id = p.sequential_id

UPDATE table_name SET col_name =
  REPLACE(col_name, 'host.domain.com', 'host2.domain.com')

UPDATE posts SET deleted=True
  ORDER BY date LIMIT 1
```

Avec `ORDER BY` il est possible de classer les enregistrements avant l'insertion, voire même sur un nombre donné de lignes (avec `LIMIT`).

Astuces de performances :

- La vitesse des `UPDATE` dépend du nombre d'index mis à jour.
- En cas d'`UPDATE` d'une table MyISAM au format dynamique, les colonnes larges causes des lectures de mémoire superflues. Il faut régulièrement lancer `OPTIMIZE TABLE` pour les réduire à la taille de leur contenu.
- Lancer plein d'`UPDATE` en même temps sur une table verrouillée est plus rapide qu'individuellement.

> **Attention !**
>
> La fonction `replace()` est sensible à la casse même avec les collations insensibles.

REPLACE

`REPLACE` fonctionne depuis MySQL 5.5[1], en remplaçant un enregistrement par un autre, exactement comme `DELETE` + `INSERT`, sauf que si l'ancien enregistrement a la même valeur que le nouveau en tant que `PRIMARY KEY` ou `UNIQUE index`, l'ancien est supprimé avant l'insertion du nouveau.

> **Attention !**

> Ne pas confondre avec `replace()`.

IGNORE

Pour éviter qu'une insertion soit interdite par une contrainte d'intégrité (ex : clé primaire en double), le mot `IGNORE` (dans "INSERT IGNORE" ou "REPLACE IGNORE") affiche juste des avertissements si une insertion est ignorée.

Avant MySQL 4.0.1, `INSERT ... SELECT` opérait implicitement en mode `IGNORE` : en ignorant les enregistrements qui causeraient des erreurs de valeur de clé dupliquée.

DELETE et TRUNCATE

```
DELETE [QUICK] FROM `table1`
TRUNCATE [TABLE] `table1`
```

Quelques précisions :

- Utiliser `DELETE` sans clause `WHERE`, supprime tous les enregistrements.
- Si une table contient beaucoup d'index, on peut agrandir le cache pour accélérer les `DELETE` (variable `key_buffer_size`).
- Pour les tables indexées MyISAM, parfois `DELETE` est plus rapide en spécifiant le mot `QUICK` (`DELETE QUICK FROM ...`). Cela permet de réutiliser les valeurs des index effacées.
- `TRUNCATE` efface également les lignes rapidement, en faisant `DROP` et `CREATE` (sur certains moteurs de stockage seulement).
- `TRUNCATE` ne garantit pas la transaction ou le verrouillage.
- `DELETE` informe de combien de lignes ont été supprimées, mais pas `TRUNCATE`.
- Après une suppression massive (au moins 30 % des lignes), il convient de lancer `OPTIMIZE TABLE` juste après pour accélérer la suite.
- Sur des tables InnoDB avec contraintes `FOREIGN KEY`, `TRUNCATE` se comporte comme `DELETE`.

```
DELETE FROM `antiques`
WHERE item = 'Ottoman'
ORDER BY `id`
LIMIT 1
```

Il est possible de classer les lignes avant leur suppression, tout en en choisissant le nombre.

Pour supprimer des enregistrements de plusieurs tables (*multi-table delete*[2] ou *cross table delete*) :

```
DELETE t1, t2
FROM t1
LEFT JOIN t2
WHERE t1.id=t2.id AND t1.value > 1;
```

Synonyme :

```
DELETE FROM t1, t2
USING t1
  LEFT JOIN t2
    ON t1.id = t2.id
WHERE t1.value > 1;
```

Toutefois dans les version inférieures à la 4 (et étonnamment constaté comme fausse alerte par PhpMyAdmin sur des versions ultérieures), les jointures sont interdites dans les suppressions, et on doit alors utiliser le `WHERE`[3] :

```
DELETE t1, t2
FROM table1 t1, table2 t2
WHERE t1.id=t2.id AND t1.value > 1;
```

> **Attention !**
>
> Comme la clause `LIMIT` ne fonctionne pas sur les suppressions multiples, il faut contourner en supprimant les enregistrements retournés par un `SELECT` avec `LIMIT`.

Références

1. http://dev.mysql.com/doc/refman/5.5/en/replace.html
2. https://dev.mysql.com/doc/refman/5.7/en/delete.html
3. https://www.electrictoolbox.com/article/mysql/cross-table-delete/

Requêtes

SELECT

La syntaxe de sélection est la suivante (chaque clause fera l'objet d'un paragraphe explicatif ensuite) :

```
SELECT *
FROM nom_table
WHERE condition
GROUP BY champ1, champ2
HAVING groupe condition
ORDER BY champ
LIMIT limite, taille;
```

Liste de champs

Il faut spécifier les données à récupérer avec SELECT :

```
SELECT DATABASE();      -- renvoie le nom de la base courante
SELECT CURRENT_USER();  -- l'utilisateur courant
SELECT 1+1; -- 2
```

L'étoile permet d'obtenir tous les champs d'une table :

```
SELECT * FROM `wiki1_page`;
```

Mais il est plutôt conseillé de nommer chaque champ (projection) pour accélérer la requête.

Noms des tables

Pour récupérer les champs d'une table ou d'une vue, il faut la placer dans la clause FROM :

```
USE wiki1;
SELECT page_id FROM `wiki1_page`;  -- renvoie les valeurs du champ "page_id" de la table "wiki1_page".

SELECT `wiki1`.`wiki1_page`.`page_id`;     -- idem
```

Autres exemples :

```
SELECT MAX(page_id) FROM `wiki1_page`; -- le nombre le plus élevé
SELECT page_id*2 FROM `wiki1_page`;    -- le double de chaque identifiant
```

WHERE

Cette clause permet de filtrer les enregistrements. Prenons pas exemple celui ou ceux dont le champ identifiant est égal à 42 :

```
SELECT * FROM `wiki1_page` WHERE `page_id`=42;
```

Ou bien ceux qui ne sont pas nuls :

```
SELECT * FROM `wiki1_page` WHERE page_id IS NOT NULL;
```

> **Attention !**
>
> Il est impossible d'utiliser le résultat d'une fonction calculée dans le SELECT dans le WHERE, car ce résultat n'est trouvé qu'à la fin de l'exécution, donc WHERE ne peut pas s'en servir au moment prévu.
>
> Pour ce faire il convient d'utiliser HAVING (voir-ensuite)

GROUP BY

Quand plusieurs enregistrements sont identiques dans le résultat, qu'ils ont les mêmes valeurs dans leurs champs sélectionnés, ils peuvent être groupés en une seule ligne.

Par exemple, en regroupant les enregistrements de la table utilisateurs sur le champ de date d'inscription au wiki, on peut obtenir pour chacune le nombre d'édition maximum, minimum et leurs moyennes :

```
SELECT user_registration, MAX(user_editcount), MIN(user_editcount), AVG(user_editcount)
FROM wiki1_user
GROUP BY `user_registration`;
```

Idem mais classé par nom et prénom d'utilisateur :

```
SELECT user_registration, user_real_name, MAX(user_editcount), MIN(user_editcount), AVG(user_editcount)
FROM wiki1_user
GROUP BY `user_registration`, `user_real_name`;
```

Cette instruction permet donc de réaliser des transpositions lignes en colonnes. Par exemple pour afficher les utilisateurs connus comme ayant le même e-mail :

```
SELECT user_email,
       MAX(CASE WHEN user_id = 1 THEN user_name ELSE NULL END) AS User1,
       MAX(CASE WHEN user_id = 2 THEN user_name ELSE NULL END) AS User2,
       MAX(CASE WHEN user_id = 3 THEN user_name ELSE NULL END) AS User3
FROM wiki1_user
GROUP BY `user_email`;
```

> **Attention !**

> Si on place le MAX dans le CASE la transposition ne s'effectue pas. 

Voir aussi `GROUP_CONCAT()` pour transposer les colonnes en lignes.

HAVING

`HAVING` déclare un filtre valable uniquement pour les enregistrements de la clause `GROUP BY`, ce qui le distingue de `WHERE` qui lui opère avant `GROUP BY`.

`HAVING` n'est pas optimisé et ne peut pas utiliser les index.

Voici un exemple d'erreur d'optimisation classique : l'ordonnancement des opérations ne filtre le gros des résultats (valeur *admin*) qu'à la fin de la requête (utilisant plus de mémoire, donc plus de temps qu'avec un `WHERE`) :

```
SELECT MAX(user_editcount), MIN(user_editcount), AVG(user_editcount)
FROM wiki1_user
GROUP BY user_real_name
HAVING user_real_name = 'admin';
```

Par contre, cet exemple ne peut pas être optimisé car le `HAVING` utilise le résultat du `MAX()` calculé après le `GROUP BY` :

```
SELECT MAX(user_editcount), MIN(user_editcount), AVG(user_editcount)
FROM wiki1_user
GROUP BY user_real_name
HAVING MAX(user_editcount) > 500;
```

ORDER BY

Il est possible de classer les résultat, par ordre croissant ou décroissant, des nombres ou des lettres.

```
SELECT * FROM `wiki1_page` ORDER BY `page_id`;
```

Par défaut, l'ordre est `ASCENDING` (croissant). Pour le décroissant il faut donc préciser `DESCENDING` :

```
SELECT * FROM `wiki1_page` ORDER BY `page_id` ASC;   -- ASC est facultatif
SELECT * FROM `wiki1_page` ORDER BY `page_id` DESC;  -- ordre inversé
```

Les valeurs NULL sont considérées comme inférieures aux autres.

Il est également possible de nommer la colonne à classer par son numéro :

```
SELECT `page_title`, `page_id` FROM `wiki1_page` ORDER BY 1;  -- nom
SELECT `page_title`, `page_id` FROM `wiki1_page` ORDER BY 2;  -- id
SELECT `page_title`, `page_id` FROM `wiki1_page` ORDER BY 1 DESC;
```

Les expressions SQL sont autorisées :

```
SELECT `page_title` FROM `wiki1_page` ORDER BY REVERSE(`page_title`)
```

La fonction `RAND()` classe de façon aléatoire :

```
SELECT `page_title` FROM `wiki1_page` ORDER BY RAND()
```

Quand un `GROUP BY` est spécifié, les résultats sont classés selon les champs qui y sont nommés, sauf avant un `ORDER BY`. Donc l'ordre décroissant peut aussi être précisé depuis le `GROUP BY` :

```
SELECT user_registration, user_real_name, MAX(user_editcount)
FROM wiki1_user
GROUP BY `user_registration` ASC, `user_real_name` DESC;
```

Pour éviter ce classement automatique du `GROUP BY`, utiliser `ORDER BY NULL` :

```
SELECT user_registration, user_real_name, MAX(user_editcount)
FROM wiki1_user
GROUP BY `user_registration`, `user_real_name` ORDER BY NULL;
```

LIMIT

Le nombre maximum d'enregistrements dans le résultat est facultatif, on l'indique avec le mot `LIMIT` :

```
SELECT * FROM `wiki1_page` ORDER BY page_id LIMIT 10;
```

Ce résultat retourne donc entre 0 et 10 lignes.

Généralement cela s'emploie après un `ORDER BY` pour avoir les maximums et minimums, mais voici un exemple pour en avoir trois au hasard :

```
SELECT * FROM `wiki1_page` ORDER BY rand() LIMIT 3;
```

Il est possible de définir une plage d'enregistrements, sachant que le premier est le numéro zéro :

```
SELECT * FROM `wiki1_page` ORDER BY page_id LIMIT 10;
SELECT * FROM `wiki1_page` ORDER BY page_id LIMIT 0, 10; -- synonyme
```

On peut donc paginer les requêtes dont les résultats peuvent saturer le serveur :

```
SELECT * FROM `wiki1_page` ORDER BY page_id LIMIT 0, 10;  -- première page
SELECT * FROM `wiki1_page` ORDER BY page_id LIMIT 10, 10; -- seconde page
SELECT * FROM `wiki1_page` ORDER BY page_id LIMIT 20, 10; -- troisième page
```

La seconde commande est équivalente à celle-ci :

```
SELECT * FROM `wikil_page` ORDER BY page_id LIMIT 10 OFFSET 10
```

Une astuce consiste à déboguer la syntaxe de sa requête rapidement en lui demandant un résultat vide, et observer ainsi s'il y a des messages d'erreur sans attendre :

```
SELECT ... LIMIT 0
```

Conseils d'optimisation

- SQL_CALC_FOUND_ROWS peut accélérer les requêtes[1][2].
- LIMIT est particulièrement pratique dans des SELECT avec ORDER BY, DISTINCT et GROUP BY, car leurs calculs n'impliquent pas toutes les lignes.
- Si la requête est résolue par le serveur en copiant les résultats dans une table temporaire, LIMIT aide MySQL à calculer combien de mémoire est requise par la table.

DISTINCT

Le mot DISTINCT peut être utilisé pour supprimer les doublons des lignes du résultat :

```
SELECT DISTINCT * FROM `wikil_page`    -- aucun doublon
SELECT DISTINCTROW * FROM `wikil_page` -- synonyme
SELECT ALL * FROM `wikil_page`         -- doublons (comportement par défaut)
```

Cela permet par exemple de récupérer la liste de toutes les valeurs différentes d'un champ :

```
SELECT DISTINCT `user_real_name` FROM `wikil_page` ORDER BY `user_real_name`
```

On peut également en sortir les différentes combinaisons de valeurs :

```
SELECT DISTINCT `user_real_name`, `user_editcount` FROM `wikil_page` ORDER BY `user_real_name`
```

Remarque : si une clé primaire ou un index unique fait partie de la sélection, le DISTINCT devient inutile. C'est également le cas avec GROUP BY.

> **Attention !**
>
> La fonction COUNT() a un comportement différent du GROUP BY avec SELECT DISTINCT COUNT(monChamp), qui renvoie un résultat différent de SELECT COUNT(DISTINCT monChamp).

IN and NOT IN

Équivalent du signe =, qui ne nécessite pas d'être répété quand il concerne plusieurs valeurs :

```
SELECT page_id
FROM wikil_page
WHERE page_namespace IN (0, 1);

-- Liste des pages qui ont des propriétés plus celles qui n'ont aucun hyperlien
SELECT page_id
FROM wikil_page as p, wikil_user as u WHERE p.page_id = u.user_id
UNION
SELECT page_id
FROM wikil_page WHERE page_id NOT IN (SELECT pp_page FROM wikil_page_props);
```

EXISTS

Fonction disponible depuis MySQL 4.

```
-- N'affiche la première sélection que si la seconde n'est pas nulle
SELECT page_title
FROM wikil_page
WHERE EXISTS (SELECT * FROM wikil_page_props WHERE pp_propname = 'noindex');
```

ALL

```
-- Ne renvoie que les pages dont le numéro est le seul de la seconde sélection
SELECT page_title
FROM wikil_page
WHERE page_id = ALL (SELECT pp_page FROM wikil_page_props WHERE pp_propname = 'defaultsort');
```

UNION et UNION All

Compatible MySQL 4 et plus. L'union de sélections nécessite qu'elles aient le même nombre de colonnes.

La requête suivante renvoie tous les enregistrements de deux tables :

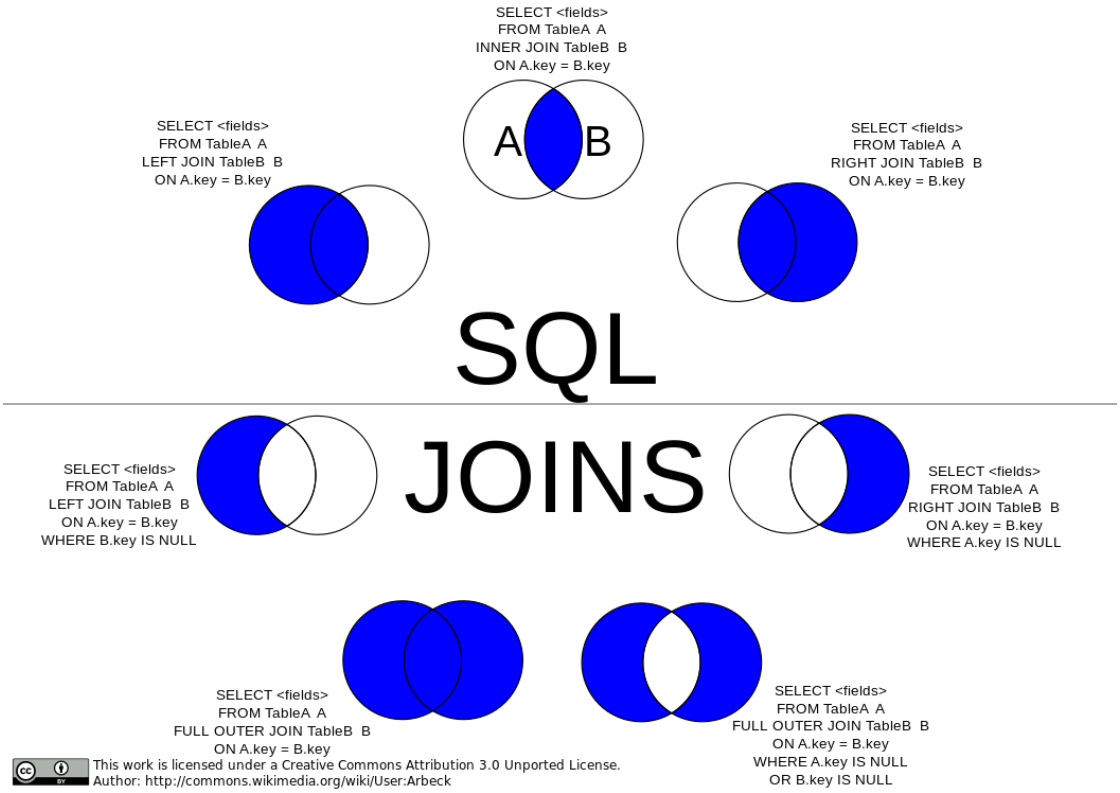
```
SELECT page_title FROM wikil_page
UNION ALL
SELECT user_name FROM wikil_user;
```

UNION est équivalent à UNION DISTINCT, ce qui le distingue de UNION ALL qui ne filtre pas les doublons.

```
SELECT page_id FROM wikil_page
UNION
SELECT page_id FROM wikil_page;
-- égal
(SELECT page_id FROM wikil_page)
```

```
UNION DISTINCT
(SELECT page_id FROM wikil_page)
ORDER BY page_id;
```

JOIN



Les relations entre les tables permettent de joindre intelligemment leurs résultats. La jointure naturelle est la plus rapide sur la plupart des plateformes SQL.

L'exemple suivant compare les nombres en anglais et en hindi.

```
CREATE TABLE english (Tag int, Inenglish varchar(255));
CREATE TABLE hindi (Tag int, Inhindi varchar(255));

INSERT INTO english (Tag, Inenglish) VALUES (1, 'One');
INSERT INTO english (Tag, Inenglish) VALUES (2, 'Two');
INSERT INTO english (Tag, Inenglish) VALUES (3, 'Three');

INSERT INTO hindi (Tag, Inhindi) VALUES (2, 'Do');
INSERT INTO hindi (Tag, Inhindi) VALUES (3, 'Teen');
INSERT INTO hindi (Tag, Inhindi) VALUES (4, 'Char');
```

select * from english

Tag	Inenglish
1	One
2	Two
3	Three

select * from hindi

Tag	Inhindi
2	Do
3	Teen
4	Char

INNER JOIN

```
SELECT hindi.Tag, english.Inenglish, hindi.Inhindi
FROM english, hindi
WHERE english.Tag = hindi.Tag
-- égal
SELECT hindi.Tag, english.Inenglish, hindi.Inhindi
FROM english INNER JOIN hindi ON english.Tag = hindi.Tag
```

Tag	Inenglish	Inhindi
2	Two	Do
3	Three	Teen

Remarque : le comportement d'un `JOIN` seul est `INNER JOIN`, qui est aussi synonyme de `CROSS JOIN`[3] (jointure cartésienne).

La jointure cartésienne décrit le cas où chaque ligne d'une table est jointe à toutes celles d'une autre.

```
SELECT * FROM english, hindi
-- égal
SELECT * FROM english CROSS JOIN hindi
```

3*3 = 9 lignes :

Tag	Inenglish	Tag	Inhindi
1	One	2	Do
2	Two	2	Do
3	Three	2	Do
1	One	3	Teen
2	Two	3	Teen
3	Three	3	Teen
1	One	4	Char
2	Two	4	Char
3	Three	4	Char

NATURAL JOIN

La jointure naturelle équivaut à `INNER JOIN` sur toutes les colonnes communes des deux tables.

USING

Le mot `USING` est compatible MySQL 4, mais change avec MySQL 5. La requête suivante est équivalente à celles `INNER JOIN` ci-dessus :

```
SELECT hindi.tag, hindi.Inhindi, english.Inenglish
FROM hindi NATURAL JOIN english
USING (Tag)
```

OUTER JOIN

```
SELECT hindi.Tag, english.Inenglish, hindi.Inhindi
FROM english OUTER JOIN hindi ON english.Tag = hindi.Tag
```

Tag	Inenglish	Tag	Inhindi
1	One		
2	Two	2	Do
3	Three	3	Teen
		4	Char

LEFT JOIN / LEFT OUTER JOIN

```
SELECT field1, field2 FROM table1 LEFT JOIN table2 ON field1=field2

SELECT e.Inenglish as English, e.Tag, h.Inhindi as Hindi
FROM english AS e
LEFT JOIN hindi AS h ON e.Tag=h.Tag
WHERE h.Inhindi IS NULL
```

```
English  tag   Hindi
One      1     NULL
```

Remarque : naturellement comme pour le `inner join`, s'il y a plusieurs lignes non NULL à droite, on les retrouve toutes en résultat.

RIGHT OUTER JOIN

```
SELECT e.Inenglish AS English, h.tag, h.Inhindi AS Hindi
FROM english AS e RIGHT JOIN hindi AS h
ON e.Tag=h.Tag
WHERE e.Inenglish IS NULL
```

```
English   tag     Hindi
NULL       4      Char
```

- S'assurer que le type des clés de jointes est le même dans les deux tables.
- Les mots clés LEFT et RIGHT ne sont pas absolus, ils opèrent selon le contexte : en intervertissant les tables le résultat sera identique.
- La jointure par défaut est `INNER JOIN` (pas `OUTER`).

FULL OUTER JOIN

MySQL n'a pas de jointure `FULL OUTER JOIN`. Voici comment l'émuler :

```
    (SELECT a.*, b*
        FROM tab1 a LEFT JOIN tab2 b
        ON a.id = b.id)
UNION
    (SELECT a.*, b*
        FROM tab1 a RIGHT JOIN tab2 b
        ON a.id = b.id)
```

Cette jointure permet d'ailleurs de comparer deux tables :

```
SELECT *
FROM table1
FULL OUTER JOIN table2 ON table2.id = table1.id
WHERE table1.id IS NULL OR table2.id IS NULL
```

Jointures multiples

Il est possible de joindre plus de deux tables :

```
SELECT ... FROM a JOIN (b JOIN c on b.id=c.id) ON a.id=b.id
```

Exemple :

```
mysql> SELECT group_type.type_id, group_type.nom, COUNT(people_job.job_id) AS count
       FROM group_type
        JOIN (groups JOIN people_job ON groups.group_id = people_job.group_id)
        ON group_type.type_id = groups.type
       GROUP BY type_id ORDER BY type_id
+---------+--------------------------------------+-------+
| type_id | nom                                  | count |
+---------+--------------------------------------+-------+
|       1 | Official GNU software                |   148 |
|       2 | non-GNU software and documentation   |   268 |
|       3 | www.gnu.org portion                  |     4 |
|       6 | www.gnu.org translation team         |     5 |
+---------+--------------------------------------+-------+
4 rows in set (0.02 sec)
```

Sous requêtes

Compatible MySQL 4.1 et plus.

- Les sous-requêtes SQL permettent aux résultats d'une requête d'être utilisés par une autre.
- Elles apparaissent toujours comme une partie de clause `WHERE` ou `HAVING`.
- Seul un champ peut être dans la sous-requête `SELECT`.
- Les `ORDER BY` ne sont donc pas autorisés (inutiles sur une seule colonne).

Par exemple, la "table" *RepOffice = OfficeNbr from Offices*, liste les bureaux où le quota de vente excède la somme des quotas des vendeurs individuels :

```
SELECT ville FROM Offices WHERE Target > ???
```

??? est la somme des quotas des vendeurs.

```
SELECT SUM(Quota)
FROM SalesReps
WHERE RepOffice = OfficeNbr
```

En combinant ces deux requêtes, les points d'interrogations disparaissent :

```
SELECT ville FROM Offices
WHERE Target > (SELECT SUM(Quota) FROM SalesReps
WHERE RepOffice = OfficeNbr)
```

Par exemple, tous les clients avec des commandes ou limites de crédits > 50000 €. En utilisant le mot `DISTINCT` pour ne lister les clients qu'une seule fois :

```
SELECT DISTINCT CustNbr
FROM Customers, Orders
WHERE CustNbr = Cust AND (CreditLimit > 50000 OR Amt > 50000);
```

Remarque : il y a donc trois types de filtre.

- `ON` : filtre les lignes d'une seule table.
- `WHERE` : filtre les lignes de toutes les tables.
- `HAVING` : filtre les lignes de toutes les tables après regroupements.

> **Attention !**
>
> Les sous-requêtes ne peuvent pas faire référence à un élément de la requête qui les contient. Si c'est nécessaire, il faut les transformer en jointures : 
> ... `JOIN (SELECT...) q ON q.id = ...`

References

1. http://www.mysqlperformanceblog.com/2007/08/28/to-sql_calc_found_rows-or-not-to-sql_calc_found_rows/
2. http://dev.mysql.com/doc/refman/5.0/en/information-functions.html
3. https://dev.mysql.com/doc/refman/5.7/en/join.html

- Official MySQL documentation (http://dev.mysql.com/doc)

NULL

Description

De nombreux langages de programmation ont deux valeurs logiques : `True` et `False`. SQL en possède une troisième pour les valeurs inconnue : `NULL`.

`NULL` étant une absence de valeur, il peut être assigné à des colonnes `TEXT`, `INTEGER` ou autres. Toutefois une colonne déclarée `NOT NULL` ne pourra pas en contenir.

```
INSERT into Singer
       (F_Name, L_Name, Birth_place, Language)
       values
       ("", "Homer", NULL, "Greek"),
       ("", "Sting", NULL, "English"),
       ("Jonny", "Five", NULL, "Binary");
```

`NULL` ne doit pas être entouré d'apostrophes ou de guillemets, ou bien il désignera une chaine de caractères contenant son nom.

> **Remarque :** `NULL` n'apparait pas dans les colonnes Varchar sous Windows XP mais sous Fedora oui.

L'exemple ci-dessous peut sélectionner des chanteurs avec prénom de taille zéro (""), par exemple pour Sting et Homer. Il vérifie si la date de naissance est nulle :

```
SELECT * from Singer WHERE Birth_place IS NULL;

SELECT * from Singer WHERE Birth_place IS NOT NULL;

SELECT * from Singer WHERE isNull(Birth_place)
```

> **Attention !**
>
> Les enregistrements X à NULL ne sont pas renvoyés par un WHERE X != 'Y'

`COUNT` ne tient pas compte des `NULL` :

```
select count(Birth_place) from Singer;
0
```

Par ailleurs, SUM(NULL) renvoie NULL.

Les opérations normales (comparaisons, expressions...) renvoient `NULL` si au moins un des éléments comparés est `NULL` :

```
SELECT (NULL=NULL) OR (NULL<>NULL) OR (NOT NULL) OR (1<NULL) OR (1>NULL) OR (1 + NULL) OR (1 LIKE NULL)
```

Deux valeurs inconnues ne sont donc pas égales (`NULL=NULL` renvoie `NULL`).

Gérer NULL

La fonction `COALESCE` peut simplifier le travail avec `NULL`.

Par exemple, pour éviter de montrer les valeurs nulles en les traitant comme des zéros :

```
SELECT COALESCE(colname,0) from table where COALESCE(colname,0) > 1;
```

Dans un champ date, les traiter comme celle actuelle :

```
ORDER BY (COALESCE(TO_DAYS(date),TO_DAYS(CURDATE()))-TO_DAYS(CURDATE()))
```

```
EXP(SUM(LOG(COALESCE(''*the field you want to multiply*'',1)))
```

La fonction `coalesce()` prévient des problèmes de calcul logarithmique d'une valeur nulle, et peut être optionnelle selon les circonstances.

```
SELECT t4.gene_name, COALESCE(g2d.score,0),
COALESCE(dgp.score,0), COALESCE(pocus.score,0)
FROM t4
LEFT JOIN g2d ON t4.gene_name=g2d.gene_name
LEFT JOIN dgp ON t4.gene_name=dgp.gene_name
LEFT JOIN pocus ON t4.gene_name=pocus.gene_name;
```

`IFNULL()` dans un `SELECT` fait de `NULL` n'importe quelle valeur désirée.

```
IFNULL(expr1,expr2)
```

Si *expr1* n'est pas nulle, `IFNULL()` renvoie *expr1*, sinon *expr2*.

`IFNULL()` renvoie une chaine ou un nombre, selon le contexte :

```
mysql> SELECT IFNULL(1,0);
-> 1
mysql> SELECT IFNULL(NULL,10);
-> 10
mysql> SELECT IFNULL(1/0,10);
-> 10
mysql> SELECT IFNULL(1/0,'yes');
-> 'yes'
```

Attention aux résultats peu prévisibles, par exemple la requête suivante efface toutes les entrées :

```
DELETE FROM ma_table1 WHERE field > NULL   -- fonctionne aussi avec une fonction renvoyant NULL
```

Pour obtenir les `NULL` en dernier lors d'un `ORDER BY` :

```
SELECT * FROM ma_table1 ORDER BY ISNULL(field), field [ ASC | DESC ]
```

Enfin, pour déterminer les champs d'une table qui ne peuvent pas être nuls :

```
SELECT *
FROM  `information_schema`.`COLUMNS`
WHERE IS_NULLABLE = 'NO' AND TABLE_NAME = 'ma_table1'
```

Opérateurs

MySQL propose plus que les standards des opérateurs SQL. Ils peuvent être utilisés pour rédiger des expressions contenant des constantes, variables, valeurs contenues dans des champs ou autres expressions.

Opérateurs d'assignation

L'opérateur = peut assigner une valeur à une colonne :

```
UPDATE `table1` SET `champ1`=0
```

Par contre pour assigner une valeur à une variable, l'opérateur est :=, car = est déjà utilisé pour la comparaison.

```
SELECT @variable1 := 1
```

`SELECT INTO` peut aussi remplir les variables.

```
SELECT 1 INTO @variable1
```

Opérateurs de comparaison

Égalité

Pour vérifier si deux valeurs sont égales, utiliser = :

```
SELECT True = True  -- 1
SELECT True = False -- 0
```

Pour vérifier si deux valeurs sont différentes, c'est <> ou != :

```
SELECT True <> False -- 1
SELECT True != True  -- 0
```

Comparaison IS NULL

Pour savoir si une valeur est nulle, utiliser IS :

```
SELECT (NULL = NULL)  -- NULL
SELECT (NULL IS NULL) -- 1
SELECT (1 IS NULL)    -- 0
SELECT (True IS True) -- erreur
```

Pour savoir si une valeur n'est pas nulle :

```
SELECT (True IS NOT NULL) -- 1
```

Il existe par ailleurs l'opérateur <=> qui considère NULL comme une valeur normale :

```
SELECT NULL <=> NULL -- 1
SELECT True <=> True -- 1
SELECT col1 <=> col2 FROM table1
```

Comparaison IS booléen

IS et IS NOT fonctionnent aussi avec TRUE, FALSE et UNKNOWN (qui est purement un synonyme de NULL).

```
SELECT 1 IS TRUE      -- 1
SELECT 1 IS NOT TRUE  -- 0
SELECT 1 IS FALSE     -- 0
SELECT (NULL IS NOT FALSE)    -- 1 : unknown n'est pas false
SELECT (NULL IS UNKOWN)       -- 1
SELECT (NULL IS NOT UNKNOWN)  -- 0
```

Plus grand et plus petit que

Avec des nombres :

```
SELECT 100 > 0  -- 1
SELECT 4 > 5    -- 0

SELECT 1 < 2    -- 1
SELECT 2 < 2    -- 0
```

Avec du texte dans l'ordre alphabétique :

```
SELECT 'a' < 'b'  -- 1
SELECT `a` >= `b` FROM `table1`
SELECT NOT (`a` < `b`) FROM `table1`
SELECT `a` <= `b` FROM `table1`
SELECT * FROM `table1` WHERE `a` >= `b`
```

Cet ordre alphabétique est définit par COLLATION (l'interclassement), pour un CHARACTER SET donné. Par exemple, une COLLATION peut être sensible à la casse ou pas (suffixe utf8_general_**cs** = case sensitive, utf8_general_**ci** = case insensitive).

Exemple :

```
SELECT _latin1'été', _utf8'été', _cp850'été', (_latin1'été' = _utf8'été'), (_latin1'été' LIKE _utf8'été')
-- Résultat :
été     Ã©tÃ©   ├®t├®   0   0
```

BETWEEN

l'opérateur BETWEEN ... AND ... permet de vérifier si une valeur appartient à une plage (bornes incluses) :

```
SELECT 2 BETWEEN 10 AND 100    -- 0
SELECT 10 BETWEEN 10 AND 100   -- 1
SELECT 20 BETWEEN 10 AND 100   -- 1

SELECT 8 NOT BETWEEN 5 AND 10  -- 0
```

IN

IN permet de s'assurer si une valeur est dans une liste :

```
SELECT 5 IN (5, 6, 7)  -- 1
SELECT 1 IN (5, 6, 7)  -- 0
SELECT 1 NOT IN (1, 2, 3)  -- 0
```

Attention : si la liste contient des nombres et des chaines, il faut tout mettre entre apostrophe pour obtenir le résultat escompté.

```
SELECT 4 IN ('a', 'z', '5')
```

Il n'y a aucune limite théorique au nombre de valeurs de la liste.

Opérateurs logiques

Booléens logiques

MySQL n'a pas vraiment de type BOOLEAN.

FALSE est un synonyme de 0. Les chaines vides sont considérées FALSE.

TRUE est un synonyme de 1. Tout ce qui n'est ni FALSE, ni NULL est considéré TRUE.

UNKNOWN est un synonyme de NULL. La date spéciale 0/0/0 est nulle.

NOT

NOT est le seul opérateur qui n'a qu'une seule opérande. Il renvoie 0 si l'opérande est TRUE, 1 si elle est FALSE, et NULL si elle est NULL.

```
SELECT NOT 1       -- 0
SELECT NOT FALSE   -- 1
SELECT NOT NULL    -- NULL
SELECT NOT UNKNOWN -- NULL
```

! est synonyme de NOT.

AND

AND renvoie 1 si les deux opérandes sont TRUE, sinon 0 ; si au moins l'une des deux opérandes est nulle, il renvoie NULL.

```
SELECT 1 AND 1      -- 1
SELECT 1 AND ''     -- 0
SELECT '' AND NULL  -- NULL
```

&& est synonyme de AND.

OR

OR renvoie TRUE si au moins une des opérandes est TRUE, sinon FALSE ; si les deux opérandes sont nulles, il renvoie NULL.

```
SELECT TRUE OR FALSE   -- 1
SELECT 1 OR 1          -- 1
SELECT FALSE OR FALSE  -- 0
SELECT NULL OR TRUE    -- NULL
```

|| est un synonyme de OR.

XOR

XOR (ou exclusif) renvoie :

- 1 si une seule des deux opérandes est TRUE et l'autre FALSE.
- 0 si les deux sont TRUE ou FALSE.
- NULL si au moins l'une des deux est NULL.

```
SELECT 1 XOR 0         -- 1
SELECT FALSE XOR TRUE  -- 1
SELECT 1 XOR TRUE      -- 0
SELECT 0 XOR FALSE     -- 0
SELECT NULL XOR 1      -- NULL
```

Opérateurs arithmétiques

Addition

```
SELECT +1 -- 1
SELECT 1 + 1 -- 2
```

Soustraction

```
SELECT -1 -- -1
SELECT -+1 -- -1
SELECT --1 -- 1
SELECT True - 1 -- 0
```

Multiplication

```
SELECT 1 * 1 -- 1
```

Divisions

Renvoie un nombre de type `FLOAT` :

```
SELECT 10 / 2  -- 5,0000
SELECT 1 / 1   -- 1,0000
SELECT 1 / 0   -- NULL
```

Pour retourner la valeur entière du résultat d'une division sous forme de type `INTEGER`, utiliser `DIV` :

```
SELECT 10 DIV 3 -- 3
```

Le reste de la division (modulo) se trouve avec '%' ou `MOD` :

```
SELECT 10 MOD 3 -- 1
```

Utiliser + pour convertir des données

Pour convertir un `INTEGER` en `FLOAT` :

```
SELECT 1 + 0.0       -- 1.0
SELECT 1 + 0.000     -- 1.000
SELECT TRUE + 0.000  -- 1.000
```

Il est impossible de convertir une valeur `FLOAT` en ajoutant 0.0, mais on peut forcer le type en `INTEGER` :

```
SELECT '1' + 0       -- 1
SELECT '1' + FALSE   -- 1
SELECT <nowiki>''</nowiki> + <nowiki>''</nowiki>  -- 0
```

Opérateurs de texte

Il n'y a pas d'opérateurs de concaténation en MySQL. Les opérateurs arithmétiques convertissent les valeurs en nombres et pour leurs opérations, donc la concaténation avec + est impossible.

La fonction `CONCAT()` pallie à cela.

LIKE

L'opérateur `LIKE` si la chaine recherchée est inclue dans une colonne :

```
SELECT * FROM articles WHERE titre LIKE 'hello world'
```

Généralement cette chaine est sensible à la casse, mais il y a deux exceptions, quand :

- une comparaison `LIKE` touche une colonne déclarée en `BINARY` ;
- l'expression contient une clause `BINARY` :

```
SELECT * 'test' LIKE BINARY 'TEST'  -- 0
```

Les comparaisons `LIKE` acceptent deux caractères spéciaux :

- `_` : n'importe quel caractère (un seul, ni zéro ni deux).
- `%` : n'importe quel séquence de caractères (par exemple zéro ou mille).

A noter que dans les expressions `LIKE`, `\` est aussi le caractère d'échappement pour `'`, et son comportement ne peut pas être changé par la clause `ESCAPE`. Il peut aussi échapper d'autres caractères, mais pas lui-même.

Utilisations courantes de `LIKE` :

- Trouver tous les titres commençant par "hello" :

```
SELECT * FROM articles WHERE titre LIKE 'hello%'
```

- Trouver tous les titres finissant par "world" :

```
SELECT * FROM articles WHERE titre LIKE '%world'
```

- Trouver tous les titres contenant la chaine "gnu" :

```
SELECT * FROM articles WHERE titre LIKE '%gnu%'
```

Ces caractères spéciaux peuvent être contenus dans le pattern lui-même. Par exemple, pour rechercher les symboles _ ou % dans la base :

```
SELECT * FROM articles WHERE titre LIKE '\_%'
SELECT * FROM articles WHERE titre LIKE '\%%'
```

/ peut-être une alternative à \ si on le précise :

```
SELECT * FROM articles WHERE titre LIKE '/_%' ESCAPE '/'
```

Quand on utilise l'opérateur =, les espaces des chaines sont ignorés, mais avec LIKE ils sont reconnus :

```
SELECT 'word' = 'word ';     -- 1
SELECT 'word' LIKE 'word ';  -- 0
```

De même, contrairement à "=", "LIKE" compare uniquement les caractères, même si leurs règles d'interclassement les regroupent[1] :

```
SELECT 'ä' = 'ae' COLLATE latin1_german2_ci;     -- 1
SELECT 'ä' LIKE 'ae' COLLATE latin1_german2_ci;  -- 0
```

LIKE fonctionne aussi avec les nombres :

```
SELECT 123 LIKE '%2%'  -- 1
```

Pour tester si un pattern ne fonctionne pas alors qu'il devrait, utiliser NOT LIKE :

```
SELECT 'a' NOT LIKE 'b' -- 1
```

SOUNDS LIKE

SOUNDS LIKE permet de vérifier si deux textes se prononcent pareils. Il utilise l'algorithme SOUNDEX, basé sur les règles de l'anglais, et peut s'avérer assez approximatif :

```
SELECT `word1` SOUNDS LIKE `word2` FROM `wordList`          -- forme courte
SELECT SOUNDEX(`word1`) = SOUNDEX(`word2`) FROM `wordList`  -- forme longue
```

SOUNDS LIKE est une extension apparue depuis MySQL 4.1.

Expressions régulières

Expressions rationnelles courantes

Caractère	Type	Explication
.	Point	n'importe quel caractère
[...]	crochets	classe de caractères : tous les caractères énumérés dans la classe
[^...]	crochets et circonflexe	classe complémentée : tous les caractères sauf ceux énumérés
^	circonflexe	marque le début de la chaine, la ligne...
$	dollar	marque la fin d'une chaine, ligne...
|	barre verticale	alternative - ou reconnaît l'un ou l'autre
(...)	parenthèses	groupe de capture : utilisée pour limiter la portée d'un masque ou de l'alternative
*	astérisque	0, 1 ou plusieurs occurrences
+	le plus	1 ou plusieurs occurrences
?	interrogation	0 ou 1 occurrence

Classes de caractères POSIX[2]

Classe	Signification
[[:alpha:]]	n'importe quelle lettre
[[:digit:]]	n'importe quel chiffre
[[:xdigit:]]	caractères hexadécimaux
[[:alnum:]]	n'importe quelle lettre ou chiffre
[[:space:]]	n'importe quel espace blanc
[[:punct:]]	n'importe quel signe de ponctuation
[[:lower:]]	n'importe quelle lettre en minuscule
[[:upper:]]	n'importe quelle lettre capitale
[[:blank:]]	espace ou tabulation
[[:graph:]]	caractères affichables et imprimables
[[:cntrl:]]	caractères d'échappement
[[:print:]]	caractères imprimables exceptés ceux de contrôle

Expressions rationnelles Unicode[3]

Expression	Signification
\A	Début de chaine
\b	Caractère de début ou fin de mot
\d	Chiffre
\D	Non chiffre
\s	Caractères espace
\S	Non caractères espace
\w	Lettre, chiffre ou underscore
\W	Caractère qui n'est pas lettre, chiffre ou underscore
\X	Caractère Unicode
\z	Fin de chaine

En MySQL 5.1, les expressions régulières fonctionnent sur des textes en octets et peuvent donc donner des résultats inattendus avec des textes en Unicode[4].

Syntaxe :

```
SELECT 'string' REGEXP 'pattern'
```

RLIKE est synonyme de REGEXP.

L'antislash fait office de caractère d'échappement.

Exemple, est-ce que la sélection est différente des lettres de A à Z :

```
SELECT 'a' REGEXP '^[a-z]';  -- 1
SELECT 'A' REGEXP '^[a-z]';  -- 1
SELECT '1' REGEXP '^[a-z]';  -- 0
```

Got error 'invalid character range'

L'utilisation de \ - est parfois proscrite.

Opérateur bit à bit

Il existe des opérateurs pour les opérations bit à bit.

Bit-NOT :

```
SELECT ~0  -- 18446744073709551615
SELECT ~1  -- 18446744073709551614
```

Bit-AND :

```
SELECT 1 & 1  -- 1
SELECT 1 & 3  -- 1
SELECT 2 & 3  -- 2
```

Bit-OR :

```
SELECT 1 | 0  -- 1
SELECT 3 | 0  -- 3
SELECT 4 | 2  -- 6
```

Bit-XOR :

```
SELECT 1 ^ 0  -- 1
SELECT 1 ^ 1  -- 0
SELECT 3 ^ 1  -- 2
```

Décalage de bit à gauche :

```
SELECT 1 << 2  -- 4
```

Décalage de bit à droite :

```
SELECT 1 >> 2  -- 0
```

Conditions

IF

La structure `IF ... THEN ... ELSE ... END IF;` ne fonctionne que dans les procédures stockées (contenant plusieurs requêtes). Pour gérer une condition en dehors d'elles, on peut utiliser[5] : `IF(condition, siVraie, siFausse);`.

Exemple : `SELECT IF(-1 < 0, 0, 1);` renvoie 0.

Exemple avec plusieurs conditions (switch)[6] [7] :

```
   IF n > m THEN SET s = '>';
   ELSEIF n = m THEN SET s = '=';
   ELSE SET s = '<';
   END IF;
```

CASE

```
SELECT CASE WHEN condition THEN siVraie ELSE siFausse END;
```

Exemple : `SELECT CASE WHEN '-1 < 0' THEN 0 ELSE 1 END;` renvoie 0.

Exemple avec plusieurs conditions[8] :

```
   CASE v
     WHEN 2 THEN SELECT v;
     WHEN 3 THEN SELECT 0;
     ELSE
       BEGIN
       END;
   END CASE;
```

Dans une seule requête :

```
SELECT CASE v
    WHEN 1 THEN 'a'
    WHEN 2 THEN 'b'
    WHEN 3 THEN 'c'
    WHEN 4 THEN 'd'
    ELSE 0
    END as value
```

Précédence

Précédence des opérateurs

Du plus au moins prioritaire :

```
INTERVAL
BINARY, COLLATE
!
-, ~
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
&&, AND
XOR
||, OR
:=
```

Modificateurs :

- *PIPES_AS_CONCAT* : si activé, || est prioritaire sur ^, mais - et ~ le reste sur ||.
- *HIGH_NOT_PRECEDENCE* : si activé, NOT est au niveau de !.

Utilisation des parenthèses

Tout comme en mathématiques, les parenthèses permettent d'évaluer des sous-expressions avant d'autres :

```
SELECT 1 + 1 * 5    -- = 6
SELECT (1 + 1) * 5  -- = 10
```

Cela peut aussi se faire pour rendre les requêtes plus lisibles aux humains :

```
SELECT 1 + (2 * 5)
```

Références

1. https://docs.oracle.com/cd/E17952_01/mysql-5.0-en/string-comparison-functions.html
2. https://www.regular-expressions.info/posixbrackets.html
3. http://www.regular-expressions.info/unicode.html
4. Pour se familiariser avec Unicode, on peut lire À la découverte d'Unicode
5. http://dev.mysql.com/doc/refman/5.7/en/control-flow-functions.html
6. https://dev.mysql.com/doc/refman/5.7/en/if.html
7. https://dev.mysql.com/doc/refman/5.7/en/case.html
8. https://dev.mysql.com/doc/refman/5.7/en/case.html

Fonctions

Syntaxe

A l'instar des mots réservés SQL, les noms des fonctions ne sont pas sensibles à la casse :

```
SELECT database() -- ok
SELECT DataBase() -- ok
SELECT DATABASE() -- ok
```

Si le SQL_MODE IGNORE_SPACE SQL_MODE n'est pas défini, il est impossible de placer un espace entre le no de la fonction et la première parenthèse, sous peine de voir une erreur 1064. IGNORE_SPACE est généralement à 0, car cela accélère le parseur. Donc :

```
SELECT DATABASE () -- déconseillé
SELECT DATABASE()  -- recommandé
```

Toutefois, cette restriction ne s'applique qu'aux fonctions natives de MySQL (pas aux procédures stockées).

Fonctions générales

Fonctions qui dépendent du type.

BENCHMARK(nombre, expression)

Exécute l'expression n fois et retourne toujours zéro[1], le chiffre pertinent est donc le temps pris par cette opération de simulation. Utile pour trouver les goulots d'étranglement des expressions SQL :

```
SELECT BENCHMARK(10000, 'Bonjour');  -- Traitement en 0.0010 sec
```

CAST(valeur AS type)

Renvoie la valeur convertie en chaine de caractères, comme les apostrophes.

```
SELECT CAST(20130101 AS date);  -- 2013-01-01
```

CHARSET(chaine)

Renvoie le type de caractères de la chaine :

```
SELECT CHARSET(20130101);  -- binary
SHOW CHARACTER SET;       -- montre tous les CHARACTER SET installés
```

COALESCE(valeur, ...)

Renvoie le premier paramètre non nul. S'ils sont tous nuls, renvoie NULL.

```
SELECT COALESCE(null, 'Bonjour', null);  -- bonjour
```

COERCIBILITY(chaine)

Renvoie la *coercibility* d'une chaine (entre 0 et 5) :

```
SELECT COERCIBILITY('bonjour');   -- 4
```

Coercibility[2]	Signification	Exemple
0	Explicit collation	Value with COLLATE clause
1	No collation	Concatenation of strings with different collations
2	Implicit collation	Column value
3	System constant	USER() return value
4	Coercible	Literal string
5	Ignorable	NULL or an expression derived from NULL

COLLATION(chaine)

Renvoie la *collation* d'une chaine :

```
SELECT COLLATION('bonjour');   -- utf8_general_ci
```

Pour obtenir celle par défaut d'une base :

```
SELECT DEFAULT_COLLATION_NAME FROM information_schema.SCHEMATA WHERE SCHEMA_NAME = 'maBase1'
```

CONNECTION_ID()

Renvoie l'identifiant du thread courant :

```
SELECT CONNECTION_ID();   -- 31
```

CONVERT(valeur, type)

Tout comme `CAST()`, retourne la valeur convertie dans le type mentionné :

```
SELECT CONVERT (20130101, date);  -- 2013-01-01
```

CONVERT(chaine USING charset)

Convertit la chaine string passée dans le CHARACTER SET spécifié :

```
SELECT CONVERT ('Voici une écriture' USING utf8);   -- Voici une écriture
SELECT CONVERT ('Voici une écriture' USING ASCII);  -- Voici une ?criture
```

CURRENT_USER()

Retourne les noms de l'utilisateur et de l'hôte courants :

```
SELECT CURRENT_USER();  -- root@localhost
```

DATABASE()

Retourne le nom de la base de données courante :

```
SELECT DATABASE();  -- wiki1
```

FOUND_ROWS()

Après un `SELECT` avec une `LIMIT` et le mot clé `SQL_CALC_FOUND_ROWS`, il est possible de lancer un autre `SELECT` avec `FOUND_ROWS()`. En effet il renvoie le nombre de ligne de la clause précédente, sans la limite :

```
SELECT FOUND_ROWS() AS n;  -- 0
SELECT SQL_CALC_FOUND_ROWS * FROM wiki1_page ORDER BY page_id LIMIT 10 OFFSET 2;  -- deux lignes
SELECT FOUND_ROWS() AS n;  -- 1
```

GREATEST(valeur1, valeur2, ...)

Renvoie la plus grande valeur des paramètres :

```
SELECT GREATEST(1, 2, 21, 3);  -- 21
```

IF(valeur1, valeur2, valeur3)

If *valeur1* est vraie, renvoie *valeur2*, sinon (fausse ou nulle) renvoie *valeur3*.

```
select if(1=2, 'irréel', 'réel');  -- réel
```

IFNULL(valeur1, valeur2)

Si *valeur1* est nulle, renvoie *valeur2*, sinon *valeur1*.

```
SELECT IFNULL('variable1', 'défaut');  -- variable1
```

ISNULL(valeur)

Si la valeur passée est nulle, renvoie 1, sinon 0.

```
SELECT ISNULL('variable1');  -- 0
```

NULLIF(valeur1, valeur2)

Renvoie `NULL` si valeur1 = valeur2, sinon valeur1.

```
SELECT NULLIF(10, 20);  -- 10
```

LAST_INSERT_ID()

Renvoie le dernier ID en AUTO_INCREMENT inséré dans la base, ce qui évite un SELECT lorsque l'on a besoin d'insérer deux enregistrements dont la deuxième avec une clé étrangère vers la première.

LEAST(valeur1, valeur2, ...)

Renvoie la plus petite valeur dans la liste des paramètres passés :

```
SELECT LEAST(1, 2, 21, 3, -1);  -- -1
```

INTERVAL(valeur1, valeur2, valeur3, ...)

Renvoie l'emplacement du premier argument supérieur au premier, en partant du zéro dans la liste des entiers en paramètres :

```
SELECT INTERVAL(10, 20, 9, 8, 7);  -- 0
SELECT INTERVAL(10, 9, 20, 8, 7);  -- 1
SELECT INTERVAL(10, 9, 8, 20, 7);  -- 2
SELECT INTERVAL(10, 9, 8, 7, 20);  -- 3
```

SUBSTR(chaine, début, taille)

Découpe une chaine de caractère :

```
SELECT SUBSTR('Hello World!', 7, 5); -- World
```

Date et heure

Il existe des dizaines de fonctions liées aux dates[3].

Pour trouver la date de l'an dernier :

```
SELECT CURDATE() - INTERVAL 1 YEAR
```

Sélectionner toutes les pages du wiki non lues depuis plus un an :

```
SELECT * FROM wiki1_page
WHERE page_touched <= (CURDATE() - INTERVAL 1 YEAR);
```

Autres exemples de sélections :

```
SELECT IF(DAYOFMONTH(CURDATE()) <= 15,
  DATE_FORMAT(CURDATE(), '%Y-%m-15'),
  DATE_FORMAT(CURDATE() + INTERVAL 1 MONTH, '%Y-%m-15')) AS next15
FROM table;

SELECT YEAR('2002-05-10'), MONTH('2002-05-10'), DAYOFMONTH('2002-05-10')

SELECT PurchaseDate FROM table WHERE YEAR(PurchaseDate) <= YEAR(CURDATE())

SELECT columns FROM table
WHERE start_time >= '2004-06-01 10:00:00' AND end_time <= '2004-06-03 18:00:00'

SELECT * FROM t1
WHERE DATE_FORMAT(datetime_column, '%T') BETWEEN 'HH:MM:SS' AND 'HH:MM:SS'

SELECT Start_time, End_time FROM Table
WHERE Start_time >= NOW() - INTERVAL 4 HOUR

SELECT NOW() + INTERVAL 60 SECOND

SELECT UNIX_TIMESTAMP('2007-05-01'); -- 1177970400
SELECT FROM_UNIXTIME(1177970400); -- 2007-05-01 00:00:00
```

> **Attention !**
>
> `convert('17/02/2016 15:49:03',datetime)` ou `convert('17-02-2016 15:49:03',datetime)` donne *null*, donc une requête d'insertion le remplace par le même résultat que `now()`. La syntaxe doit être `convert('2016-02-17 15:49:03',datetime)` ou `convert('2016/02/17 15:49:03',datetime)`.

DATE_ADD()

Pour additionner deux dates. Par exemple pour calculer le jour d'une livraison prenant 48 h :

```
SELECT DATE_ADD(NOW(), INTERVAL 2 DAY)
```

Pour la date d'hier :

```
SELECT DATE_ADD(NOW(), INTERVAL -1 DAY)
```

Les unités à additionner ou soustraire les plus courantes sont[4] :

```
SECOND
MINUTE
HOUR
DAY
WEEK
MONTH
YEAR
```

DATEDIFF()

Pour soustraire une date à une autre. Par exemple pour calculer un âge :

```
SELECT DATEDIFF(NOW(), birthday_date) / 365
FROM user
WHERE ISNULL(birthday_date) = 0 AND birthday_date != '0000-00-00'
```

Fonctions d'agrégation

COUNT(champ)

Si le paramètre est "*" au lieu d'un nom de colonne, `COUNT()` renvoie les nombre de lignes total de la requête. Cela peut permettre de savoir combien de lignes possède une table, par exemple le nombre de pages d'un wiki :

```
SELECT COUNT(*) FROM `wiki1_page`;
```

Si le mot DISTINCT est employé, cela ignore les doublons :

```
SELECT COUNT(DISTINCT *) FROM `wiki1_page`;
```

Si le nom d'un champ est précisé, cela renvoie le nombre de valeurs non nulles :

```
SELECT COUNT(`user_real_name`) FROM `wiki1_user`;

SELECT COUNT(DISTINCT `user_real_name`) FROM `wiki1_user`;
```

Cela fonctionne aussi pour des expressions, des combinaisons de champs :

```
SELECT COUNT(`user_name` + `user_real_name`) FROM `wiki1_user`;
```

Pour afficher le décompte de plusieurs tables non jointes :

```
SELECT
  (SELECT COUNT(*) FROM maTable1) as t1,
  (SELECT COUNT(*) FROM maTable2) as t2
```

MAX(champ)

MAX() renvoie la valeur maximum d'une expression issue du résultat d'une requête, ou NULL s'il n'y en a pas :

```
SELECT MAX(`user_editcount`) FROM `wiki1_user`;

SELECT MAX(LENGTH(CONCAT(`user_name`, ' ', `user_real_name`))) FROM `wiki1_user`;
```

Alternatives

Selon le contexte, la fonction MAX() n'est pas toujours la meilleure option pour obtenir un maximum. Par exemple en cas de sous-requêtes ou sans agrégation possible :

- `SELECT `user_editcount` FROM `wiki1_user` ORDER BY user_editcount DESC LIMIT 1;`
- ```
  SELECT `user_editcount`
  FROM `wiki1_user` wu1
  LEFT JOIN `wiki1_user` wu2 ON wu1.user_editcount > wu2.user_editcount
  WHERE wu2.user_editcount is null;
  ```

MIN(champ)

MIN() renvoie la valeur minimum d'une expression issue du résultat d'une requête, ou NULL s'il n'y en a pas :

```
SELECT MIN(`user_editcount`) FROM `wiki1_user`;

SELECT MIN(LENGTH(CONCAT(`user_name`, ' ', `user_real_name`))) FROM `wiki1_user`;
```

AVG(champ)

AVG() renvoie la valeur moyenne d'une expression, ou NULL s'il n'y en a pas :

```
SELECT AVG(`user_editcount`) FROM `wiki1_user`;
```

SUM(champ)

SUM() dresse la somme des valeurs d'une expression, ou NULL s'il n'y en a pas.

Si SUM(DISTINCT expression) est utilisé, les valeurs identiques ne sont ajoutées qu'une seule fois. Il a été ajouté après MySQL 5.1.

```
SELECT SUM( DISTINCT user_editcount )
FROM wiki1_user
```

GROUP_CONCAT(champ)

GROUP_CONCAT() concatène les valeurs de tous les enregistrements d'un groupe dans une seule chaine séparée par une virgule par défaut. En effet, le deuxième paramètre facultatif permet de définir un autre séparateur.

```
CREATE TEMPORARY TABLE product (
       id INTEGER, product_type VARCHAR(10), product_name VARCHAR(50)
   );

INSERT INTO product VALUES
    (1, 'mp3', 'iPod'),
    (2, 'mp3', 'Zune'),
    (3, 'mp3', 'ZEN'),
    (4, 'notebook', 'Acer Eee PC'),
    (4, 'notebook', 'Everex CloudBook');

SELECT * FROM product;

SELECT product_type, group_concat(product_name)
FROM product
GROUP BY product_type;
```
```

```
SELECT product_type, group_concat(' ', product_name)
FROM product
GROUP BY product_type;
```

Fonctions d'agrégation de bit

Syntaxe générale :

```
FUNCTION_NAME(''expression'')
```

Ces fonctions bit à bit calculent *expression* pour chaque ligne du résultat et entre les *expressions*. La précision est de 64 bit.

AND

```
SELECT BIT_AND(ip) FROM log
```

OR

```
SELECT BIT_OR(ip) FROM log
```

(retourne 0 s'il n'y a aucun résultat)

XOR

```
SELECT BIT_XOR(ip) FROM log
```

(retourne 0 s'il n'y a aucun résultat)

Références

1. http://dev.mysql.com/doc/refman/5.0/fr/information-functions.html
2. http://dev.mysql.com/doc/refman/5.0/en/information-functions.html#function_coercibility
3. https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html
4. https://www.w3schools.com/sql/func_mysql_date_add.asp

Procédures stockées

MySQL peut enregistrer des requêtes pour les rappeler comme les fonctions d'un programme. Elles peuvent intégrer des contrôles de flux, des boucles et des curseurs. Il en existe trois sortes :

- Déclencheurs (ou triggers) : programmes qui se déclenchent avant ou après un évènement impliquant une table (`DELETE, INSERT, UPDATE`) ;
- Évènements : programmes exécutés à une certaine date, régulièrement ;
- Procédures stockées : programmes invocable par la commande SQL `CALL`.

Les futures versions de MySQL pourraient même stocker des procédures écrites dans d'autres langages que SQL.

Déclencheurs

Gestion des TRIGGER

Disponibles depuis MySQL 5.0.2, ils fonctionnent sur les tables persistantes, mais pas les temporaires.

CREATE TRIGGER

```
CREATE TRIGGER `effacer_ancien` AFTER INSERT ON `wiki1_page`
    FOR EACH ROW
        DELETE FROM `wiki1_page` ORDER BY `page_id` ASC LIMIT 1
```

Cet exemple est une requête `DELETE` appelée `effacer_ancien`, qui se lance après qu'un nouvel enregistrement soit inséré dans la table. Si un INSERT ajoute plusieurs lignes à une table, le déclencheur est appelé plusieurs fois.

Les conditions de déclenchement des triggers doivent être des commandes LMD basiques :

- `INSERT`, dont `LOAD DATA` et `REPLACE` ;
- `DELETE`, incluant `REPLACE`, mais pas `TRUNCATE` ;
- `UPDATE`

Un cas particulier est `INSERT ... ON DUPLICATE KEY UPDATE`. Si `INSERT` est exécuté, `BEFORE INSERT` ou `AFTER INSERT` sont exécutés. Si `UPDATE` est exécuté à la place de `INSERT`, l'ordre des évènements est le suivant : `BEFORE INSERT, BEFORE UPDATE, AFTER UPDATE`.

Le déclencheur peut aussi s'appliquer à une table en particulier :

```
... ON `base1`.`table1` ...
```

Les noms des triggers doivent être unique pour chaque base.

Contrairement au standard SQL, tous les déclencheurs sont exécutés `FOR EACH ROW`, et non pour chaque commande.

Une procédure stockée doit être spécifiée entre les mots `BEGIN` et `END` sauf s'il ne contient qu'une seule commande. Le SQL dynamique ne peut pas y être utilisé (`PREPARE`) ; Une autre procédure stockée peut être appelée à la place.

Il est posible d'accéder à l'ancienne valeur d'un champ (avant l'exécution de la procédure) et à la nouvelle valeur :

```
CREATE TRIGGER `use_values` AFTER INSERT ON `example_tab`
    FOR EACH ROW BEGIN
        UPDATE `changelog` SET `old_value`=OLD.`field1`, `new_value`=NEW.`field1` WHERE `backup_tab`.`id`=`example_tab`.`id`
    END
```

DROP TRIGGER

Pour supprimer un déclencheur :

```
DROP TRIGGER `trigger1`
-- ou
DROP TRIGGER `base1`.`trigger1`
-- ou
DROP TRIGGER IF EXISTS `trigger1`
```

Pour modifier un trigger, il faut le supprimer puis le recréer.

Métadonnées des TRIGGER

SHOW CREATE TRIGGER

Disponible depuis MySQL 5.1. Affiche la commande pour recréer un déclencheur nommé :

```
SHOW CREATE TRIGGER effacer_ancien;
```

- **Trigger** : Nom du déclencheur.
- **sql_mode** : valeur du SQL_MODE au moment de l'exécution.
- **SQL Original Statement**
- **character_set_client**
- **collation_connection**
- **Database Collation**

SHOW TRIGGERS

Pour obtenir la liste des triggers de la base courante :

```
SHOW TRIGGERS;
```

Pour obtenir la liste des triggers d'une autre base :

```
SHOW TRIGGERS IN `base2`
```

```
-- ou
SHOW TRIGGERS FROM `base2`
```

D'autres filtres sont possibles :

```
SHOW TRIGGERS WHERE table='wiki1_page'
```

Remarque : il est impossible d'utiliser `LIKE` et `WHERE` ensemble.

Les colonnes du déclencheur sont :

- **Trigger** : nom
- **Event** : commande SQL qui le déclenche
- **Table** : table associée
- **Statement** : requête exécutée
- **Timing** : BEFORE ou AFTER
- **Created** : toujours NULL
- **sql_mode** : SQL_MODE définit lors de sa création
- **Definer** : créateur
- **character_set_client** : valeur de la variable `character_set_client` lors de la création
- **collation_connection** : valeur de la variable `collation_connection` lors de la création
- **Database Collation** : COLLATION utilisée par la base du trigger.

INFORMATION_SCHEMA.TRIGGERS

La base virtuelle INFORMATION_SCHEMA a une table `TRIGGERS` avec les colonnes suivantes :

- **TRIGGER_CATALOG** : catalogue contenant le trigger ;
- **TRIGGER_SCHEMA** : SCHEMA (DATABASE) contenant le trigger ;
- **TRIGGER_NAME** : nom du trigger ;
- **EVENT_MANIPULATION** : INSERT, UPDATE ou DELETE ;
- **EVENT_OBJECT_CATALOG** : pas encore implémenté ;
- **EVENT_OBJECT_SCHEMA** : schéma contenant la table associée au trigger ;
- **EVENT_OBJECT_NAME** : nom de la table associée au trigger ;
- **ACTION_ORDER** : pas encore implémenté ;
- **ACTION_CONDITION** : pas encore implémenté ;
- **ACTION_STATEMENT** : commande exécutée lors de l'activation du trigger ;
- **ACTION_ORIENTATION** : pas encore implémenté ;
- **ACTION_TIMING** : BEFORE ou AFTER ;
- **ACTION_REFERENCE_OLD_TABLE** : pas encore implémenté ;
- **ACTION_REFERENCE_NEW_TABLE** : pas encore implémenté ;
- **ACTION_REFERENCE_OLD_ROW** : pas encore implémenté ;
- **ACTION_REFERENCE_NEW_ROW** : pas encore implémenté ;
- **CREATED** : date et heure de création (pas encore implémenté) ;
- **SQL_MODE** : SQL_MODE valide pour l'exécution du trigger ;
- **DEFINER** : créateur du trigger, sous la forme 'utilisateur@hôte' ;
- **CHARACTER_SET_CLIENT** : valeur de la variable `character_set_client` lors de la création ;
- **COLLATION_CONNECTION** : valeur de la variable `collation_connection` lors de la création ;
- **DATABASE_COLLATION** : COLLATION utilisée par la base.

Évènements

Les évènements sont aussi appelé en anglais Scheduled Events ou Temporal Triggers. Ils sont planifiés pour s'exécuter à un moment donné, date ou intervalle de temps. Ils sont similaire aux crontab UNIX.

Quand un évènement est lancé, il doit être complètement exécuté. S'il est réactivé avant la fin de son exécution, une nouvelle instance du même évènement est créée. Donc il est conseillé d'utiliser `LOCK` pour éviter qu'ils interfèrent entre eux.

Le planificateur des évènements est un thread en permanence en exécution, afin d'être en mesure de lancer les évènements à tout moment. Il peut toutefois être désactivé en lançant MySQL avec ces options :

```
mysqld --event-scheduler=DISABLED
```

Ou bien en ajoutant une ligne dans le fichier de configuration (my.cnf) :

```
event_scheduler=DISABLED
```

Ou encore en cours d'utilisation :

```
SELECT event_scheduler -- valeurs : ON / OFF / DISABLED
SET GLOBAL event_scheduler = ON
SET GLOBAL event_scheduler = OFF
```

Quand il est lancé, on peut vérifier son status avec `SHOW PROCESSLIST`. Son utilisateur est 'event_scheduler'. Quand il est en sommeil, `State` est à 'Waiting for next activation'.

Gestion des EVENT

Les commandes sont `CREATE EVENT, ALTER EVENT, DROP EVENT`.

CREATE EVENT

Pour un évènement à exécuter le lendemain :

```
CREATE EVENT `évènement1`
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 DAY
  DO
    INSERT INTO `wiki1`.`news` (`title`, `text`) VALUES ('Example!', 'This is not a real news')
```

Son nom est obligatoire et doit être précisé après `CREATE EVENT`.

Pour créer une tâche à exécuter une seule fois, utiliser `AT`. Pour ne pas spécifier la date et l'heure de manière absolue, mais relativement après un intervalle, utiliser `AT CURRENT_TIMESTAMP + INTERVAL ...`.

Une tâche récurrente s'obtient avec `EVERY` :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

On peut aussi spécifier la date et l'heure du début et/ou de la fin. La tâche sera exécutée à intervalle régulier entre ces dates :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY INTERVAL 1 DAY
  DO
    OPTIMIZE TABLE `wiki1`.`news`
      STARTS CURRENT_TIMESTAMP + 1 MONTH
      ENDS CURRENT_TIMESTAMP + 3 MONTH
```

Les unités autorisées sont :

```
YEAR, QUARTER, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND, YEAR_MONTH, DAY_HOUR, DAY_MINUTE, DAY_SECOND, HOUR_MINUTE, HOUR_SECOND, MINUTE_SECOND
```

La clause `DO` spécifie la commande à exécuter.

Si la tâche est composée par plus d'une commande, utiliser `BEGIN ... END` :

```
 delimiter |
 CREATE EVENT `évènement3`
   ON SCHEDULE
     EVERY 1 DAY
   DO
     BEGIN
       DELETE FROM `logs`.`user` WHERE `deletion_time` < CURRENT_TIMESTAMP - 1 YEAR;
       DELETE FROM `logs`.`messages` WHERE `deletion_time` < CURRENT_TIMESTAMP - 1 YEAR;
       UPDATE `logs`.`activity` SET `last_cleanup` = CURRENT_TIMESTAMP;
   END |
 delimiter ;
```

Si un `EVENT` du même nom existe déjà, le serveur renvoie une erreur. On peut l'éviter avec `IF NOT EXISTS` :

```
CREATE EVENT `évènement2`
  IF NOT EXISTS
  ON SCHEDULE EVERY 2 DAY
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

Après expiration de l'évènement, MySQL le supprimer par défaut. Pour éviter cela afin de pouvoir le réutiliser son code ultérieurement, utiliser `ON COMPLETION` :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  ON COMPLETION PRESERVE
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

On peut aussi dire explicitement à MySQL de le supprimer :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  ON COMPLETION NOT PRESERVE
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

En précisant une date de lancement antérieure, l'évènement expire immédiatement après sa création, c'est pourquoi le serveur prévient avec un warning 1588, normalement.

Pour préciser si un évènement est activé lors de sa création, les mots sont `ENABLE, DISABLE, DISABLE ON SLAVES` (ce dernier ne se réplique pas sur les bases de données esclaves). Par défaut, il est activé :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  ON COMPLETION NOT PRESERVE
  DISABLE
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

Pour le modifier : `ALTER EVENT`.

On peut aussi commenter l'évènement dans une limite de 64 caractères :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  ON COMPLETION NOT PRESERVE
  DISABLE
  COMMENT 'let\'s optimize some tables!'
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

Par ailleurs, on peut modifier l'utilisateur de l'évènement pour obtenir d'autres permissions. Par exemple depuis celui voulu avec `CURRENT_USER` :

```
CREATE DEFINER = CURRENT_USER
  EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

Spécifier un autre utilisateur nécessite les droits root :

```
CREATE DEFINER = 'allen@localhost'
  EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

ALTER EVENT

Renommage d'un évènement :

```
CREATE EVENT `évènement2`
  ON SCHEDULE EVERY 2 DAY
  ON COMPLETION NOT PRESERVE
  RENAME TO `évènement3`
  DISABLE
  COMMENT 'let\'s optimize some tables!'
  DO
    OPTIMIZE TABLE `wiki1`.`news`
```

On peut aussi ne définir que la clause à modifier :

```
CREATE EVENT `évènement2` ENABLE;
```

DROP EVENT

Avec les permissions sur l'évènement à supprimer :

```
DROP EVENT `évènement3`
```

S'il n'existe pas l'erreur 1517 survient. Pour l'éviter :

```
DROP EVENT IF EXISTS `évènement3`
```

Métadonnées des EVENT

SHOW CREATE EVENT

Cette commande retourne la commande `CREATE EVENT` utilisée pour créer le trigger, et sur les paramètres l'impactant.

```
SHOW CREATE EVENT évènement2
```

Les colonnes du résultat sont :

- **Event** : nom
- **sql_mode** : mode SQL utilisé (ex : NO_ENGINE_SUBSTITUTION)
- **time_zone** : fuseau horaire du créateur (ex : SYSTEM)
- **Create Event** : code qui a généré l'évènement
- **character_set_client** (ex : utf8)
- **collation_connection** (ex : utf8_general_ci)
- **Database Collation** (ex : latin1_swedish_ci)

SHOW EVENTS

Pour afficher tous les évènements de la base courante :

```
SHOW EVENTS
```

Pour une base en particulier :

```
SHOW EVENTS FROM `wiki1`
-- Ou
SHOW EVENTS IN `wiki1`
```

Autres filtres :

```
SHOW EVENTS LIKE 'év%'
SHOW EVENTS WHERE definer LIKE 'admin@%'
```

Types de résultat :

- **Db** : nom de la base ;
- **Name** : nom de l'évènement ;
- **Definer** : créateur (user@host) ;
- **Time zone** : fuseau horaire ;
- **Type** : 'ONE TIME' ou 'RECURRING' selon la récurrence ;
- **Executed At** : date de l'exécution, ou `NULL` pour les récursifs ;
- **Interval Value** : nombre d'intervalle entre les exécutions, ou `NULL` pour les non récursifs ;
- **Interval Field** : unités de mesure de l'intervalle (ex : 'SECOND'), ou `NULL` pour les non récursifs ;
- **Starts** : date de première exécution, ou `NULL` pour les non récursifs ;
- **Ends** : date de dernière exécution, ou `NULL` pour les non récursifs ;
- **Status** : ENABLED, DISABLED, ou SLAVESIDE_DISABLED ;
- **Originator** : identifiant du serveur créateur (0 pour le courant). Disponible depuis MySQL 5.1 ;
- **character_set_client**
- **collation_connection**
- **Database Collation**

INFORMATION_SCHEMA.EVENTS

La base virtuelle INFORMATION_SCHEMA contient une table `EVENTS` depuis MySQL 5.1. Voici ses colonnes :

- **EVENT_CATALOG** : toujours NULL (les CATALOG ne sont pas encore implémentés par MySQL) ;
- **EVENT_SCHEMA** : nom de la base ;
- **EVENT_NAME** : nom de l'évènement ;
- **DEFINER** : créateur (user@host) ;
- **TIME_ZONE** : fuseau horaire ;
- **EVENT_BODY** : langage utilisé ;
- **EVENT_DEFINITION** : routine à exécuter ;
- **EVENT_TYPE** : 'ONE TIME' ou 'RECURRING' selon la récurrence ;
- **EXECUTE_AT** : date de l'exécution, ou NULL pour les récursifs ;
- **INTERVAL_VALUE** : nombre d'intervalle entre les exécutions, ou NULL pour les non récursifs ;
- **INTERVAL_FIELD** : unités de mesure de l'intervalle (ex : 'SECOND'), ou NULL pour les non récursifs ;
- **SQL_MODE** mode SQL ;
- **STARTS** : date de première exécution, ou NULL pour les non récursifs ;
- **ENDS** : date de dernière exécution, ou NULL pour les non récursifs ;
- **STATUS** : ENABLED, DISABLED, ou SLAVESIDE_DISABLED ;
- **ON_COMPLETION** : 'NOT PRESERVE' ou 'PRESERVE' ;
- **CREATED** : date de création ;
- **LAST_ALTERED** : date de dernière modification ;
- **LAST_EXECUTED** : date de dernière exécution ;
- **EVENT_COMMENT** : commentaires ;
- **ORIGINATOR** : identifiant du serveur créateur (0 pour le courant). Disponible depuis MySQL 5.1 ;
- **character_set_client**
- **collation_connection**
- **Database Collation**

Procédures stockées

Les procédures stockées sont des modules SQL exécutables avec `CALL`.

Il en existe deux types :

1. `FUNCTION` si elles retournent un résultat.
2. `PROCEDURE`s si elles ne retournent rien après leur traitement.

Avantages

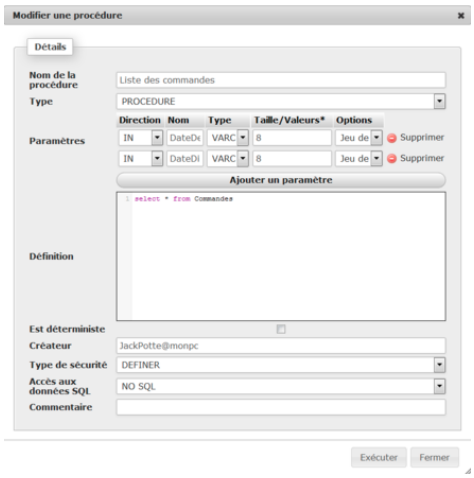
- Elles réduisent le trafic du réseau car une seule commande permet de leur en faire exécuter plusieurs. Les appeler est donc plus rapide.
- Ces modules peuvent être invoqués plusieurs fois depuis n'importe quel langage (PHP, Java...).
- Elles conservent une logique entre les bases : le DBA peut les modifier sans toucher aux programmes qui les appellent.
- Peut permettre aux utilisateurs qui n'ont pas accès à une table de récupérer ses données ou la modifier dans certaines circonstances.

Gestion des PROCEDURE et FUNCTION

CREATE PROCEDURE

Création de procédure stockée :

```
CREATE DEFINER = `root`@`localhost` PROCEDURE `Module1` ( ) NOT DETERMINISTIC NO SQL SQL SECURITY
DEFINER OPTIMIZE TABLE wiki1_page;
```



L'ajout de procédure stockée sous phpMyAdmin nécessite de remplir tous les champs. Le code s'obtient en cliquant sur *Exporter*.

CALL

Invocation :

```
CALL `Module1` ();
```

DROP PROCEDURE

Suppression :

```
DROP PROCEDURE `Module1` ;
```

Modification

On est obligé de supprimer et de recréer le module :

```
DROP PROCEDURE `Module1` ;
CREATE DEFINER = `root`@`localhost` PROCEDURE `Module1` ( ) NOT DETERMINISTIC NO SQL SQL SECURITY DEFINER
BEGIN
  OPTIMIZE TABLE wiki1_page;
  OPTIMIZE TABLE wiki1_user;
END
```

Métadonnées des PROCEDURE et FUNCTION

SHOW FUNCTION / PROCEDURE STATUS

```
SHOW PROCEDURE STATUS;
```

SHOW CREATE FUNCTION / PROCEDURE

```
SHOW CREATE PROCEDURE Module1;
```

INFORMATION_SCHEMA.ROUTINES

La base virtuelle INFORMATION_SCHEMA a une table `ROUTINES` avec les informations des procédures et fonctions.

INFORMATION_SCHEMA.PARAMETERS

Cette table contient toutes les valeurs des fonctions stockées.

Extensions au standard SQL

Délimiteur

MySQL utilise un caractère comme délimiteur pour séparer ses requêtes, par défaut ';'. Quand on crée des procédures stockées avec plusieurs requêtes, on en crée en fait une seule : `CREATE` de la procédure. Toutefois, si elles sont séparées par ';', il faut demander à MySQL de les ignorer pour estimer la fin du `CREATE`.

Dans l'exemple suivant, '|' joue ce rôle :

```
delimiter |
CREATE EVENT évènement1
    ON SCHEDULE EVERY 1 DAY
    DO
      BEGIN
        TRUNCATE `wiki1`.`wiki1_page`;
        TRUNCATE `wiki1`.`wiki1_user`;
      END
delimiter ;
```

Flow control

Les mots clés sont : `IF, CASE, ITERATE, LEAVE LOOP, WHILE, REPEAT`[1].

Loops

WHILE

```
DELIMITER $$
CREATE PROCEDURE compteur()
  BEGIN
    DECLARE x INT;
    SET x = 1;
    WHILE x  <= 5 DO
      SET  x = x + 1;
    END WHILE;
    SELECT x;  -- 6
  END$$
DELIMITER ;
```

LOOP

```
DELIMITER $$
CREATE PROCEDURE compteur2()
  BEGIN
    DECLARE x INT;
    SET x = 1;
    boucle1: LOOP
      SET  x = x + 1;
        IF x > 5 THEN
          LEAVE boucle1;
      END IF;
    END LOOP boucle1;
    SELECT x;  -- 6
  END$$
DELIMITER ;
```

REPEAT

```
DELIMITER $$
CREATE PROCEDURE compteur3()
  BEGIN
    DECLARE x INT;
    SET x = 1;
    REPEAT
      SET x = x + 1; UNTIL x > 5
    END REPEAT;
    SELECT x;  -- 6
  END$$
DELIMITER ;
```

Curseurs

Les curseurs permettent de traiter chaque ligne différemment, mais cela ralentit considérablement les requêtes.

```
DELIMITER $$
CREATE PROCEDURE curseur1()
BEGIN
    DECLARE resultat varchar(100) DEFAULT "";
    DECLARE c1 CURSOR FOR
```

```
        SELECT page_title
        FROM wiki1.wiki1_page
        WHERE page_namespace = 0;
    OPEN c1;
    FETCH c1 INTO resultat;
    CLOSE c1;
    SELECT resultat;
END;$$
DELIMITER ;
```

Ils doivent être déclaré puis ouvert avant le début de la boucle qui traite chaque enregistrement. Pour connaitre la fin de la table parcourue, il faut crée un *handler* après le curseur :

```
-- Concatène toutes les valeurs d'une colonne sur une ligne
DELIMITER $$
CREATE PROCEDURE curseur2()
BEGIN
    DECLARE resultat varchar(100) DEFAULT "";
    DECLARE total text DEFAULT "";
    DECLARE fin BOOLEAN DEFAULT 0;
    DECLARE c2 CURSOR FOR
        SELECT page_title
        FROM wiki1.wiki1_page
        WHERE page_namespace = 0;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;
    OPEN c2;
    REPEAT
        FETCH c2 INTO resultat;
        set total = concat(total, resultat);
    UNTIL fin END REPEAT;
    CLOSE c2;
    SELECT total;  -- AccueilMySQLPHPPHP
END;$$
DELIMITER ;
```

Gestion des erreurs

La déclaration d'un "handler" permet de spécifier un traitement en cas d'erreur[2] :

```
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
```

De plus, le type d'erreur peut être précisé :

```
DECLARE CONTINUE HANDLER FOR SQLSTATE [VALEUR]
DECLARE CONTINUE HANDLER FOR SQLWARNING
DECLARE CONTINUE HANDLER FOR NOT FOUND
```

mysql_affected_rows()

Cette fonction renvoie le nombre de lignes impactées par la requête précédente[3].

Références

1. http://dev.mysql.com/doc/refman/5.0/en/flow-control-statements.html
2. http://dev.mysql.com/doc/refman/5.7/en/declare-handler.html
3. https://dev.mysql.com/doc/refman/5.7/en/mysql-affected-rows.html

- http://www.convert-in.com/mssql-to-mysql-stored-procedures.htm

Bases de données spatiales

Principe

Lors du typage des champs, certains représentent des objets graphiques, et sont donc considérés comme étant de catégorie "Spatial" (base de données spatiales). Par conséquent, ils se manipulent par des requêtes différentes que pour le texte.

On distingue huit types de champs[1] :

- Geometry
- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

Et six relations entre eux[2] :

- Contains
- Disjoint
- Equals
- Intersects
- Overlaps
- Within

Requêtes

Cette section est vide, pas assez détaillée ou incomplète.

Références

1. (anglais) http://dev.mysql.com/doc/refman/5.0/en/spatial-datatypes.html
2. (anglais) http://dev.mysql.com/doc/refman/5.7/en/spatial-relation-functions-mbr.html

Importer et exporter

Exporter

phpMyAdmin propose un bouton "Export" permettant l'extraction de toute une base dans un fichier .sql. Pour ne sortir que quelques tables, il faut les cocher puis choisir "Export" dans le menu déroulant en bas.

Pour exporter en SQL, le mot clé est `INTO OUTFILE` :

```
SELECT * FROM destinataire INTO OUTFILE '/tmp/test' WHERE id IN (41, 141, 260, 317, 735, 888, 1207, 2211);
```

Le processus MySQL écrit lui-même le fichier, pas l'utilisateur. De plus, le fichier est stocké sur le serveur, pas le client.

Généralement le serveur a le droit d'écrire dans `/tmp`, donc même s'il n'est pas sécurisé ce répertoire est utilisé pour les exemples ci-dessous.

Par ailleurs, exporter est possible en ligne de commande :

```
mysql < query.txt > output.txt
```

Ou bien via `mysqldump`.

Importer

Pour importer un fichier :

```
LOAD DATA INFILE '/tmp/test' INTO TABLE destinataire;
```

Options additionnelles :

```
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
```

Pour spécifier la structure du document et la présence d'en-tête, on peut associer les colonnes de la base à des variables :

```
LOAD DATA LOCAL INFILE
'/tmp/test'
INTO TABLE destinataire
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(
@dummy,
name,
phone_number,
@dummy,
@dummy,
@dummy,
@dummy,
@dummy,
@dummy,
@dummy
)
```

Dans cet exemple, seule la seconde et troisième colonne du fichier sont stockées dans le champ *name* et *phone_number*.

Précisions sur le contenu

Pour importer un .sql créant un utilisateur et sa base de données, il faut savoir s'il existe déjà sur le serveur, car MySQL ne possède pas de `DROP USER IF EXISTS`. Par contre pour les bases ça fonctionne :

```
DROP DATABASE IF EXISTS `base1`;
CREATE DATABASE `base1` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `base1`;
--DROP USER `utilisateur1`@'localhost';
CREATE USER 'utilisateur1'@'localhost' IDENTIFIED BY 'p@ssword1';
GRANT USAGE ON *.* TO 'utilisateur1'@'localhost' IDENTIFIED BY 'p@ssword1';
GRANT ALL PRIVILEGES ON `utilisateur1`.* TO 'utilisateur1'@'localhost';
```

PS : si cette commande renvoie *la commande drop database est désactivée* avec PhpMyAdmin, modifier son `config.default.php` en passant `$cfg['AllowUserDropDatabase']` à `true`, et vider le cache du navigateur.

Réplication

Principe

La réplication signifie que les données écrites sur le master MySQL sont envoyées à des slaves faisant office de copies.

Applications :

- Sauvegardes : backup automatique alternatif à mysqldump.
- Distribution : accès en lecture de la même base depuis plusieurs serveurs pour augmenter les performances.
- Failover : système de secours.

Il y a deux types de réplication :

- Asynchrone (master/slave).
- Semi-asynchrone (réplication asynchrone plus avec un slave avant de terminer la requête).

Configurations des réplications :

- standard : master->slave
- double maître : master<->master

En Master-Master les deux hôtes sont tour à tour master et slave : le serveur A se réplique sur le serveur B qui se réplique sur le serveur A. Il n'y a pas de vérification de consistance des données, même si `auto_increment_increment/auto_increment_offset` est configuré les deux serveurs ne doivent pas être utilisés pour des accès concurrents.

Réplication asynchrone

C'est le cas le plus simple, un master écrit un fichier de log binaire, et les slaves peuvent lire ce dernier (potentiellement sélectivement) pour rejouer les commandes de la requête.

Étant asynchrone, le master et les slaves peuvent avoir différents états au même moment. Cette configuration peut résister aux coupures réseau.

Configuration du master

Dans `/etc/mysql/my.cnf`, section `[mysqld]` :

- Définir un identifiant de serveur ; par exemple 1 :

```
server-id = 1
```

- La réplication est basée sur les logs binaires, donc les activer :

```
log-bin
# ou log-bin = /var/log/mysql/mysql-bin.log
```

Créer un nouvel utilisateur pour que le slave puisse se connecter :

```
CREATE USER 'myreplication';
SET PASSWORD FOR 'myreplication' = PASSWORD('mypass');
GRANT REPLICATION SLAVE ON *.* to 'myreplication';
```

Vérifier l'identifiant de serveur :

```
SHOW VARIABLES LIKE 'server_id';
```

Configuration de chaque slave

Dans `/etc/mysql/my.cnf`, section `[mysqld]` :

- Définir un identifiant de serveur différent du master et des autres slaves :

```
server-id = 2
```

- Vérifier avec :

```
SHOW VARIABLES LIKE 'server_id';
```

- Il est aussi possible de déclarer le nom de la machine slave dans le master (cf. `SHOW SLAVE HOSTS`) :

```
report-host=slave1
```

Déclarer le master :

```
CHANGE MASTER TO MASTER_HOST='master_addr', MASTER_USER='myreplication', MASTER_PASSWORD='mypass';
```

Si la réplication sert de backup, spécifier le point de départ :

```
MASTER_LOG_FILE='<binary_log_from_master>', MASTER_LOG_POS=<master_binary_log_position>;
```

Démarrer la réplication :

```
START SLAVE;
```

Cela va créer un fichier `master.info`, typiquement dans `/var/lib/mysql/master.info` ; contenant la configuration et le statut.

Vérifier la réplication

Sur le slave

```
SHOW SLAVE STATUS;
```

Ou bien pour avoir un résultat formaté plus lisible :

```
SHOW SLAVE STATUS\G
```

Exemple :

```
*************************** 1. row ***************************
            Slave_IO_State:
               Master_Host: master_addr
               Master_User: myreplication
               Master_Port: 3306
...
```

Vérifier en particulier :

```
         Slave_IO_Running: Yes
         Slave_SQL_Running: Yes
```

On peut supposer une nature réplication asynchrone :

```
      Seconds_Behind_Master: 0
```

Voir aussi :

```
mysql> SHOW GLOBAL VARIABLES LIKE "%SLAVE%";
```

Sur le master

Vérifier les connexions des slaves :

```
mysql> SHOW PROCESSLIST\G
[...]
*************************** 6. row ***************************
     Id: 14485
   User: myreplication
   Host: 10.1.0.106:33744
     db: NULL
Command: Binlog Dump
   Time: 31272
  State: Has sent all binlog to slave; waiting for binlog to be updated
   Info: NULL

If you enabled <code>report-host</code>, the slave is also visible in:
mysql> SHOW SLAVE HOSTS;
+-----------+---------+------+------------------+-----------+
| Server_id | Host    | Port | Rpl_recovery_rank | Master_id |
+-----------+---------+------+------------------+-----------+
|         2 | myslave | 3306 |                0 |         1 |
+-----------+---------+------+------------------+-----------+
1 row in set (0.00 sec)
```

Consistance

La réplication est une simple copie, similaire aux sorties `mysqldump` dans le client `mysql`.

par conséquent, pour maintenir cette consistance :

- Ne pas écrire sur le slave ;
- Démarrer la réplication avec des données initiales identiques sur le master et le slave ;
- Utiliser les mêmes versions de MySQL sur eux peut aider.

Réparer

Par défaut, la réplication stoppe en cas d'erreur (provenant du réseau ou d'une requête).

Dans ce cas, regarder la trace dans le log (généralement `/var/log/syslog`) :

```
Oct 15 21:11:19 builder mysqld[4266]: 101015 21:11:19 [ERROR] Slave: Error 'Table 'mybase.form'
  doesn't exist' on query. Default database: 'mybase'.  Query:
  'INSERT INTO `form` (`form_id`,`timestamp`,`user_id`) VALUES ('abed',1287172429,0)',
  Error_code: 1146
```

La meilleure façon et de relancer la réplication entièrement.

On peut aussi tenter de réparer, par exemple faire sauter à MySQL la commande `1` :

```
STOP SLAVE;
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
START SLAVE;
```

Attention en définissant ce nombre car il contient toutes les commandes, pas seulement les erreurs.

Une autre façon est d'utiliser les outils Maatkit :

- mk-slave-restart (pour relancer la réplication du slave si SQL_SLAVE_SKIP_COUNTER ne peut pas aider)
- mk-table-checksum (pour faire un checksum des tables sur le master et le slave)
- mk-table-sync (pour synchroniser le slave avec le master basé sur des statistiques générés par `mk-table-checksum`).

Désinstaller

Pour supprimer une réplication :

```
mysql> RESET SLAVE;
```

- MySQL me le slave en pause et remplace la configuration avec les valeurs par défaut. `master.info` est effacé.
- Relancer MySQL pour effacer toute la configuration.

Attention : `STOP SLAVE` arrêt la réplication. Elle peut être relancée manuellement ensuite, ou bien automatiquement lors de la relance du serveur MySQL. Pour éviter ce lancement automatique :

```
slave-skip-start
```

Pour arrêter d'utiliser la réplication, vérifier que la configuration est bien vide :

```
mysql> SHOW SLAVE STATUS;
Empty set (0.00 sec)
```

Hints SQL

Avant chaque requête, des indications concernant la réplication peuvent être placées en commentaire. Par exemple via le plugin Mysqlnd de PHP[1].

- MYSQLND_MS_MASTER_SWITCH : force la requête sur le master.
- MYSQLND_MS_SLAVE_SWITCH : force la requête sur l'esclave.
- MYSQLND_MS_LAST_USED_SWITCH: force la requête sur le dernier serveur utilisé.

Références

1. https://dev.mysql.com/doc/connectors/en/apis-php-mysqlnd-ms.quickstart.sqlhints.html

Optimisation

Optimiser le serveur MySQL

Avant de démarrer l'optimisation

Quand la base est anormalement lente, vérifier les points suivants :

1. Trouver les goulots d'étranglements (processeur, RAM, I/O, requêtes)
2. Chercher à complexifier les requêtes gourmandes. Généralement on effectue les opérations suivantes dans cet ordre pour éliminer un maximum de données inutiles des résultats rapidement : projection, sélection, jointure.
3. Optimiser l'application en retirant les requêtes ou cache PHP des pages web.
4. Optimiser les requêtes (ajouter des index, des tables temporaires ou changer de jointure).
5. Optimiser la base du serveur (taille du cache, etc.).
6. Optimiser le système (les différents systèmes de fichier, le swap (mémoire virtuelle) et les versions du noyau).
7. Optimiser le hardware.

Des outils pour unixeries existent pour trouver les goulots d'étranglement :

`vmstat`
: monitore les utilisations du processeur, de la RAM et des I/O en les classant.

`mytop`
: trouve les requêtes lourdes[1].

`mysqlreport`
: checklist pas à pas[2] (nécessite Perl et son module DBD-MySQL installés).

`MySQL Workbench`[3]
: anciennement *mysql admin(istrator)*, monitore et personnalise MySQL de façon convenable.

On peut ensuite classer les applications en trois groupes par leurs nécessités :

- I/O et lecture (blogs, news).
- I/O et écriture (traqueur de connexion web, collection de données de compte).
- CPU (CMS, logiciel de business).

Variables de statut et serveur

MySQL peut être monitoré et personnalisé en surveillant les variables de statut, et définissant les variables de serveur qui peuvent être globales ou par session.

Les variables de statut peuvent être monitorées par *SHOW [GLOBAL|SESSION] STATUS [LIKE '%foo%']* ou *mysqladmin [extended-]status*.

Les variables de serveur peuvent être définies dans */etc/mysql/my.cnf* ou via *SET [GLOBAL|SESSION] VARIABLE foo := bar*, et affichées avec *mysqladmin variables* ou *SHOW [GLOBAL|SESSION] VARIABLES [LIKE '%foo%']*.

Généralement, les variables de statut commencent par une majuscule, et pas les variables de serveur.

Pour gérer les quotas de ces variables, il faut les multiplier par la valeur de *max_connections* pour avoir une estimation de la mémoire maximum utilisée. Cela permettra d'éviter des crashs lors des pics de connexions. Exemple :

```
min_memory_needed = global_buffers + (thread_buffers * max_connections)
```

```
global_buffers:
    key_buffer
    innodb_buffer_pool
    innodb_log_buffer
    innodb_additional_mem_pool
    net_buffer

thread_buffers:
    sort_buffer
    myisam_sort_buffer
    read_buffer
    join_buffer
    read_rnd_buffer
```

Expérience

Remarque : lors des tests, désactiver le cache (*query_cache_type=0* dans *my.cnf*) pour forcer un recalcul systématique des requêtes.

Lançons le programme Perl suivant :

```perl
#!/usr/bin/perl

use strict;

print "DROP TABLE IF EXISTS weightin;\n";
print "CREATE TABLE weightin (
    id INT PRIMARY KEY auto_increment,
    line TINYINT,
    date DATETIME,
    weight FLOAT(8,3)
);\n";

# 2 millions records, interval = 100s
for (my $timestamp = 1000000000; $timestamp < 1200000000; $timestamp += 100) {
    my $date = int($timestamp + rand(1000) - 500);
    my $weight = rand(1000);
    my $line = int(rand(3)) + 1;
    print "INSERT INTO weightin (date, line, weight) VALUES (FROM_UNIXTIME($date), $line, $weight);\n";
}
```

Rôle

Simule une entrée de données en quantité industrielle à intervalle régulier.

Utilisation

```
mysql> CREATE DATABASE industrial
$ perl generate_huge_db.pl | mysql industrial
real   6m21.042s
user   0m37.282s
sys    0m51.467s
```

Pour vérifier le nombre d'éléments :

```
mysql> SELECT COUNT(*) FROM weightin;
+----------+
| count(*) |
+----------+
|  2000000 |
+----------+
1 row in set (0.00 sec)
```

La taille doit être importante :

```
$ perl generate_huge_db.pl > import.sql
$ ls -lh import.sql
-rw-r--r-- 1 root root 189M jun 15 22:08 import.sql

$ ls -lh /var/lib/mysql/industrial/weightin.MYD
-rw-rw---- 1 mysql mysql 35M jun 15 22:17 /var/lib/mysql/industrial/weightin.MYD

$ time mysqldump industrial > dump.sql
real   0m9.599s
user   0m3.792s
sys    0m0.616s
$ ls -lh dump.sql
-rw-r--r-- 1 root root 79M jun 15 22:18 dump.sql

$ time mysqldump industrial | gzip > dump.sql.gz
real   0m17.339s
user   0m11.897s
sys    0m0.488s
$ ls -lh dump.sql.gz
-rw-r--r-- 1 root root 22M jun 15 22:19 dump.sql.gz
```

Incidemment, restaurer d'un dump est plus rapide car il y a moins d'insertions :

```
# time zcat dump.sql.gz | mysql industrial
real   0m31.772s
user   0m3.436s
sys    0m0.580s
```

La commande SQL scanne tous les enregistrements pour obtenir une somme :

```
mysql> SELECT SUM(*) FROM weightin;
```

Par exemple, pour compter le matériel depuis le premier janvier 2008 :

```
mysql> SELECT COUNT(*), SUM(poids) FROM pesee WHERE date >= '2008-01-01' AND date < '2008-01-02';
```

MySQL a besoin de lire toute la base même pour un petit nombre d'enregistrement, car rien ne garantit qu'ils soient classés. Pour améliorer ceci, on peut faire de la date un index. MySQL va donc créer une nouvelle table cachée avec les dates classées dans l'ordre, et stocker leur position dans la table 'weightin' afin de pouvoir faire le lien avec. Comme l'index est ordonné, MySQL peut plus rapidement localiser un enregistrement (ex : par dichotomie) plutôt que lors d'une lecture séquentielle.

Ajout de l'index :

```
ALTER TABLE weightin ADD INDEX (date);
```

On remarque que le fichier .MYD a grossi :

```
$ ls -lh /var/lib/mysql/industrial/
-rw-rw---- 1 mysql mysql  49M jun 15 22:36 weightin.MYI
```

C'est parce qu'il est utilisé pour stocker les index, par défaut toutes les clés primaires.

On constate aussi que l'ordre naturel des résultats change après ajout de l'index. Dans cet exemple, `SELECT * FROM weightin` renverra les valeurs de "date" les plus élevées à la fin, puis utilisera les ID comme clé de tri secondaire.

Autre exemple

Tentons d'optimiser la requête :

```
mysql> SELECT DISTINCT line FROM weightin;
```

Il suffit de faire de 'line' un index, afin qu'il puisse éviter les doublons regroupés ensemble, au lieu de rescanner toute la table pour les localiser :

```
ALTER TABLE weightin ADD INDEX (line);
```

Taille du fichier :

```
-rw-rw---- 1 mysql mysql  65M jun 15 22:38 weightin.MYI
```

On constate que l'ordre naturel des résultats change après ajout de l'index. Dans cet exemple, `SELECT * FROM weightin` renverra les valeurs de "line" les plus élevées à la fin, puis utilisera les ID comme clé de tri

secondaire.

> **Remarque :** il existe aussi la forme `CREATE INDEX my_index ON weightin (line);`, qui a l'avantage de baptiser l'index pour pouvoir le supprimer par son nom ultérieurement.

Considérations générales

La première question pour optimiser les sélections est toujours de savoir si les index sont configurés, et si oui s'ils sont utilisés.

1. Vérifier si les index sont utilisés

Les requêtes individuelles peuvent être détaillées par `EXPLAIN`. Pour tout le serveur les variables "Sort_%" peuvent être surveillés pour indiquer combien de fois MySQL doit aller les chercher dans le fichier de données en l'absence d'index.

2. Est-ce que les index sont stockés dans un tampon

Garder les index en mémoire vive augmente les performances de lecture. Le quotient des clés lues sur les requêtes de lecture de clés reflète les réels accès de MySQL au fichier d'index sur le disque quand il nécessitait une clé.

Idem avec les clés écrites, utiliser `mysqlreport` pour faire le calcul. Si le pourcentage est trop haut, key_buffer_size pour MyISAM et innodb_buffer_pool_size pour InnoDB sont les variables à régler.

Les variables Key_blocks_% peuvent être utilisées pour voir combien les clés tampons sont réellement utilisées. Une unité correspond à 1 ko, sauf si key_cache_block_size a été modifié. Comme MySQL utilise les blocs internes, key_blocks_unused doit être vérifié. Pour estimer la taille du tampon à définir, celle des fichiers .MYI doit être vérifiée. Pour InnoDB il y a innodb_buffer_pool_size qui concerne tous les types de données en tampon (pas seulement les index).

3. Configuration avancée

sort_buffer_size (par thread) est la mémoire utilisée pour `ORDER BY` et `GROUP BY`. Il set déconseillé par contre d'utiliser myisam_sort_buffer_size.

read_buffer_size (par thread) est la taille de mémoire allouée pour les scans complets de table (comme les tables volumineuses ne tiennent pas complètement en mémoire).

Query cache

La principale raison de ne pas rétrograder vers des versions antérieures à MySQL 4.0.1, est la faculté de stocker les requêtes `SELECT` jusqu'à ce que les tables soient modifiées.

Le *Query Cache* peut être configuré au travers des variables **query_cache_%**. La plus importante est la globale **query_cache_size** et **query_cache_limit** qui préviennent les requêtes uniques à résultats anormalement plus larges que la taille du cache.

Les blocs Query Cache ont une taille variable dont le minimum est définit par query_cache_min_res_unit, donc après reset du cache le nombre de bloc libre doit être idéalement de un. Une large valeur de Qcache_free_blocks engendrerait de la fragmentation.

Voir aussi les variables :

- **Qcache_free_blocks** : si la valeur est haute, cela indique une forte fragmentation.
- **Qcache_not_cached** : si la valeur est haute, il y a soit plus de requête hors du cache (par exemple parce qu'ils utilisent des fonctions comme `now()`) soit la valeur de query_cache_limit est trop basse.
- **Qcache_lowmem_prunes** : montant des anciens résultats purgés car le cache était plein, et les tables modifiées. query_cache_size doit être augmenté pour abaisser cette variable.

Exemple d'un cache vide :

```
mysql> SHOW VARIABLES LIKE 'query_cache_type';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| query_cache_type | ON    |
+------------------+-------+
1 row in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'query_cache_size';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| query_cache_size | 0     |
+------------------+-------+
1 row in set (0.00 sec)

mysql> SHOW STATUS LIKE 'Qcache%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_free_blocks      | 0     |
| Qcache_free_memory      | 0     |
| Qcache_hits             | 0     |
| Qcache_inserts          | 0     |
| Qcache_lowmem_prunes    | 0     |
| Qcache_not_cached       | 0     |
| Qcache_queries_in_cache | 0     |
| Qcache_total_blocks     | 0     |
+-------------------------+-------+
8 rows in set (0.00 sec)
```

Cache utilisé (savannah.gnu.org) :

```
mysql> SHOW VARIABLES LIKE "query_cache_size";
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| query_cache_size | 33554432 |
+------------------+----------+
1 row in set (0.00 sec)

mysql> SHOW STATUS LIKE "Qcache%";
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 1409     |
| Qcache_free_memory      | 27629552 |
| Qcache_hits             | 7925191  |
| Qcache_inserts          | 3400435  |
| Qcache_lowmem_prunes    | 2946778  |
| Qcache_not_cached       | 71255    |
```

```
| Qcache_queries_in_cache | 4546     |
| Qcache_total_blocks     | 10575    |
+-------------------------+----------+
8 rows in set (0.00 sec)
```

Le paramètre de `my.cnf` correspondant est :

```
query_cache_size = 32M
```

Pour nettoyer le cache afin d'améliorer les performances :

```
mysql> RESET QUERY CACHE;
Query OK, 0 rows affected (0.00 sec)
```

Attendre les locks

Les variables **Table_locks_%** affichent le nombre de requêtes en attente faute de pouvoir accéder aux tables actuellement verrouillées par d'autres requêtes. Ces situations peuvent être causées par la commande `LOCK TABLE` ou encore par plusieurs accès en écriture simultanés sur la même table.

Cache des tables

MySQL a besoin d'un certain temps pour ouvrir une table et lire ses métadonnées comme les noms de colonnes.

Si plusieurs threads tentent d'accéder à la même table, elle est ouverte plusieurs fois.

Pour accélérer ceci la métadonnée peut être stockée dans le **table_cache** (alias **table_open_cache** depuis MySQL 5.1.3).

Une bonne valeur est le nombre max_connections multiplié par le nombre de tables moyen par sélection.

Utiliser mysqlreport ou regarder les **Open_tables** après que les **Opened_tables** ou le **Uptime** nombre de tables ouvertes par seconde peut être calculé (hors des heures de pointe comme la nuit).

Connexions et threads

Pour chaque connexion de client, MySQL crée un thread séparé sous le processus principal mysqld. Pour les grands sites à plusieurs centaines de connexions par semaine, créer les threads eux-mêmes peut consommer un temps non négligeable. Pour l'accélérer, les threads en attente sont mis en cache après déconnexion de leur client. En règle générale, moins d'un thread par seconde peut être créé ensuite.

Les clients qui envoient plusieurs requêtes au serveur doit utiliser les connexions persistantes comme avec la fonction PHP `mysql_pconnect()`.

Ce cache peut être configuré par **thread_cache_size** et monitoré avec les variables **threads_%**.

Pour éviter les surcharges MySQL bloque les nouvelles connexions si plus que **max_connections** sont utilisé à cet instant. Commencer par **max_used_connections** et surveiller le nombre de connexion rejetées (**Aborted_clients**) et celle *time out* (**Aborted_connections**).

Ne pas déconnecter les clients aux connexions persistantes peut rapidement provoquer un déni de service. Les connexions normales sont fermées après le **wait_timeout** d'inactivité en seconde.

Tables temporaires

Il est parfaitement normal que MySQL crée des tables temporaires pendant les classements ou les résultats de regroupement. Ces tables sont soit en mémoire, soit trop larges et sont écrites sur le disque (plus lent).

Le nombre de tables sur le disque (variables **Created_tmp_%**) doit être négligeable ou la configuration de **max_heap_table_size** et **tmp_table_size** doit être reconsidérée.

Écritures différées

Pour rédiger les logs d'accès au serveur web dans une base, avec de nombreux `INSERT` subséquents dans la même table, les performances peuvent être améliorées en conseillant au serveur de mettre en cache les requêtes d'écriture un court moment, puis de tout envoyer comme batch sur le disque.

Attention au fait que toutes les méthodes mentionnées ne contreviennent pas à la recommandation ACID car les insertions sont reconnus avec OK au client avant leur écriture définitive sur le disque, et donc des données pourraient être perdues en cas de crash.

- Pour les tables MyISAM, les écritures différées peuvent être définies avec **DELAY_KEY_WRITE** dans un `CREATE` ou un `ALTER TABLE`. L'inconvénient est qu'après un crash la table est automatiquement marquée comme corrompue et doit être vérifiée voire réparée ce qui prend un certain temps.
- Pour InnoDB, c'est **innodb_flush_log_at_trx_commit**. En cas de crash seuls les index sont reconstruits.

INSERT DELAYED fonctionne sur les principaux moteurs de stockage.

Optimiser les tables

Index

Il convient d'utiliser le commande suivante régulièrement pour réorganiser les enregistrements sur le disque dur, afin de réduire la taille de la table (sans rien effacer) et d'accélérer les lectures par index (grâce aux enregistrements contigus nécessitant moins de déplacement des têtes de lectured des disques durs)[4] :

```
OPTIMIZE TABLE MaTable1
```

De plus, au moment de leurs créations, les types les plus petits possibles sont souhaités. Par exemple :

- si un nombre est toujours positif, choisir un type `unsigned` afin de pouvoir en stocker deux fois plus dans le même nombre d'octets.
- pour stocker des dates contemporaines (de 1970 à 2038) mieux vaut prendre un `timestamp` sur quatre octets qu'un `datetime` sur 8[5].

Optimiser les requêtes

Les requêtes en cours sont visibles avec :

```
show full processlist;
```

Comparer les fonctions avec BENCHMARK

BENCHMARK() permet de mesurer les rapidité des fonctions ou opérateurs MySQL :

```
mysql> SELECT BENCHMARK(100000000, CONCAT('a','b'));
+----------------------------------------+
| BENCHMARK(100000000, CONCAT('a','b')) |
+----------------------------------------+
|                                      0 |
+----------------------------------------+
1 row in set (21.30 sec)
```

Toutefois, on ne peut pas comparer des requêtes avec :

```
mysql> SELECT BENCHMARK(100, SELECT `id` FROM `lignes`);
ERROR 1064 (42000): You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for
the right syntax to use near 'SELECT `id` FROM `lignes`)' at line 1
```

En effet, sachant que MySQL doit parser la requête, on peut considérer que les benchmarks inférieurs à 10 s ne sont pas exploitables.

Analyse des fonctions avec EXPLAIN

En ajoutant `EXPLAIN` devant `SELECT`, MySQL détaille les différentes opérations qu'il effectue dans le cadre de cette sélection (comment les tables sont jointes et dans quel ordre). Cela permet de placer d'éventuels hints en fonction.

Exemple

Jointure de deux tables sans indice :

```
mysql> explain SELECT * FROM a left join b using (i) WHERE a.i < 2;
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | a     | ALL  | NULL          | NULL | NULL    | NULL |    4 | Using where |
|  1 | SIMPLE      | b     | ALL  | NULL          | NULL | NULL    | NULL |    3 |             |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
2 rows in set (0.01 sec)
```

Maintenant on ajoute un index sur la seconde table, ce qui fait descendre ensuite lors de la même sélection, la colonne *rows* de la seconde ligne : MySQL a donc effectué une lecture de moins pour le même résultat.

```
mysql> ALTER TABLE b ADD KEY(i);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> explain SELECT * FROM a left join b using (i) WHERE a.i < 2;
+----+-------------+-------+------+---------------+------+---------+----------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref      | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+----------+------+-------------+
|  1 | SIMPLE      | a     | ALL  | NULL          | NULL | NULL    | NULL     |    4 | Using where |
|  1 | SIMPLE      | b     | ref  | i             | i    | 5       | test.a.i |    2 |             |
+----+-------------+-------+------+---------------+------+---------+----------+------+-------------+
2 rows in set (0.00 sec)
```

Enfin, en ajoutant un index sur la première table, la condition `WHERE` est améliorée car MySQL sait qu'il ne lui faut qu'une ligne de la première table (au lieu de quatre) :

```
mysql> ALTER TABLE a ADD KEY(i);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> explain SELECT * FROM a left join b using (i) WHERE a.i < 2;
+----+-------------+-------+-------+---------------+------+---------+----------+------+-------------+
| id | select_type | table | type  | possible_keys | key  | key_len | ref      | rows | Extra       |
+----+-------------+-------+-------+---------------+------+---------+----------+------+-------------+
|  1 | SIMPLE      | a     | range | i             | i    | 5       | NULL     |    1 | Using where |
|  1 | SIMPLE      | b     | ref   | i             | i    | 5       | test.a.i |    2 |             |
+----+-------------+-------+-------+---------------+------+---------+----------+------+-------------+
2 rows in set (0.02 sec)
```

Hints d'optimisation

L'ordre des mots réservés est important si on applique plusieurs hints[6] :

```
SELECT [ALL | DISTINCT | DISTINCTROW ]
   [HIGH_PRIORITY] [STRAIGHT_JOIN]
   [SQL_SMALL_RESULT | SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
   [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
   ...
```

HIGH_PRIORITY

Généralement les commandes LMD (`INSERT, DELETE, UPDATE`) sont prioritaires sur le `SELECT`. Mais grâce à `HIGH_PRIORITY` un `SELECT` peut être traité avec elles.

STRAIGHT_JOIN

Force MySQL à évaluer les tables d'un `JOIN` dans l'ordre où elles sont nommées (de gauche à droite).

SQL_SMALL_RESULT

Lors d'un `DISTINCT` ou d'un `GROUP BY`, ce hint prévient l'optimiseur que la requête va renvoyer un petit nombre de lignes.

SQL_BIG_RESULT

Lors d'un `DISTINCT` ou d'un `GROUP BY`, dit à l'optimiseur que la requête renvoie un nombre élevé de résultats.

SQL_BUFFER_RESULT

Force MySQL à copier le résultat dans une table temporaire. Cela peut s'avérer utile par exemple pour supprimer des `LOCK` rapidement.

SQL_CACHE

Force MySQL à copier le résultat dans le cache. Ne fonctionne que si la valeur de `query_cache_type` est `DEMAND` ou 2.

SQL_NO_CACHE

Demande à MySQL de ne pas mettre le résultat en cache. C'est utile quand la requête survient très rarement, ou que le résultat change très souvent.

SQL_CALC_FOUND_ROWS

Si une requête contient `LIMIT`, ce hint dit au serveur de calculer combien de lignes auraient été retournées en cas d'absence de `LIMIT`. Pour récupérer le nombre il faut sélectionner `FOUND_ROWS()`.

```
SELECT SQL_CALC_FOUND_ROWS * FROM `wiki_page` LIMIT 2 OFFSET 100;
SELECT FOUND_ROWS();
```

Hints sur les index

Afin d'influer sur les déroulements vu précédemment, on peut utiliser les hints suivants :

- `USE INDEX` : spécifie de rechercher des enregistrements de préférence en parcourant les index des tables.
- `FORCE INDEX` : idem en plus restrictif. Une table ne sera parcourant sans index que si l'optimiseur n'a pas le choix.
- `IGNORE INDEX` : demande de ne pas favoriser les index.

Exemples :

```
SELECT *
FROM table1 USE INDEX (date)
WHERE date BETWEEN '20150101' AND '20150131'
```

```
SELECT *
FROM table1 IGNORE INDEX (date)
WHERE id BETWEEN 100 AND 200
```

Pour appliquer ces règles lors d'une jointure, d'un tri ou d'un regroupement, il suffit d'ajouter utiliser `FOR JOIN`, `FOR ORDER BY` ou `FOR GROUP BY`.

```
SELECT *
FROM table1 t1
JOIN table2 t2 FORCE INDEX ON JOIN (t1_id) ON t1.id = t2.t1_id
WHERE t1.id BETWEEN 100 AND 200
```

> **Attention !**
>
> Trop d'index dans une table ralentit toutes les requêtes qui y sont faites. Il est donc recommandé de ne pas en créer plus de deux ou trois dans chacune.

Liens externes

- *(en)* MySQL Optimization (http://investigacionit.com.ar/optimizacion-de-bases-de-datos-mysql/)
- *(en)* A guide to mysqlreport (http://hackmysql.com/mysqlreportguide)
- *(en)* High Performance MySQL (http://www.amazon.com/High-Performance-MySQL-Jeremy-Zawodny/dp/0596003064/) (livre)
- *(en)* Tuning tips from the company EZ (http://ez.no/community/articles/tuning_mysql_for_ez_publish)
- *(en)* MySysop (http://www.fillon.org/mysysop) (démo d'un script PHP pour l'optimisation et le tuning MySQL)
- Voir aussi les newsgroups et mailing lists MySQL

API

L'interface de programmation (API) MySQL permet aux applications d'interagir avec les bases de données.

Optimisation

Appels API

Connexions persistantes

En utilisant les connexions persistantes, plusieurs requêtes peuvent être exécutées sans reconnexion. C'est un gain de temps pour l'utilisateur, mais le serveur doit y allouer une partie de sa RAM pendant tout ce temps, ce qui peut le saturer, surtout quand tous ses sites le font.

Mémoire libre

Certaines requêtes enregistrent une ligne dans une variable. Cela peut donc avoir du sens de libérer cette mémoire un peu avant la fin du script, plutôt qu'à la toute fin.

Recherche de ligne

La plupart des APIs proposent deux types de recherches de ligne : en tableau (associatif ou pas) ou en objet.

Assigner les lignes dans un objet est plus lent, alors que le tableau non associatif est le plus rapide (et c'est le mieux si on ne prend qu'un seul champ par enregistrement).

API vs SQL

Généralement, les APIs ont des méthodes optimisées qui créent des commandes SQL et les envoient au serveur MySQL.

Réduire les communications client/serveur

- Certains scripts utilisent deux requêtes pour extraire une table pivot. Les communications client/serveur étant toujours le facteur ralentissant des applications, il faut préférer une seule jointure à la place.
- Si toutefois plusieurs requêtes s'avèrent nécessaires, utiliser les connexions persistantes.
- Ne sélectionner que le minimum de champs (éviter *).
- Éviter d'inclure dans les commandes SQL des caractères inutiles (espaces, tabs, commentaires...).

CREATE ... SELECT, INSERT ... SELECT

Lors de création de nouvelle table depuis une existante, `CREATE ... SELECT` peut être utilisé.

Pour remplir une table existante, c'est `INSERT ... SELECT` ou `REPLACE ... SELECT`.

INSERT DELAYED

Certains scripts n'ont pas besoin de vérifier si les insertions se sont bien déroulées.

Dans ce cas, faire appel à `INSERT DELAYED` pour que le serveur n'attende pas.

REPLACE

Lors d'un `DELETE` précédé d'un `INSERT`, le serveur reçoit deux commandes SQL.

En revanche, avec `REPLACE` il n'en reçoit qu'une.

De plus, `REPLACE DELAYED` est possible.

Autres techniques

Stocker les données dans des cookies

Parfois, des données de session sont stockées dans la base.

Cela nécessite un `UPDATE` et un `SELECT` à charge chargement de page.

Cela peut être évité avec les cookies (même si l'utilisateur peut refuser leur utilisation, ou bien lire leur contenu). Il faut éviter d'y stocker des mots de passe et leur attribuer un bref temps de vie sous peine de compromettre la vie privée de l'utilisateur.

Autre solution :

- Une fois l'utilisateur loggué sur le site, lancer `CURRENT_TIMESTAMP()` et un ID au hasard ;
- Définir un cookie avec cet ID ;
- Quand l'utilisateur fait quelque chose qui nécessite de vérifier son identification :

```
SELECT FROM `access` WHERE `id`=id_cookie AND `timestamp`>=CURRENT_TIMESTAMP() - login_lifetime
```

- Mettre à jour le *timestamp*.

Créer du contenu statique

Quand un utilisateur lit du contenu dynamique (d'une base), un document HTML est généré.

Souvent, cette page ne contient pas de variable mais du contenu inséré une seule fois, sans mise à jour.

D'où l'idée de stocker des pages HTML statiques, supprimée puis regénérées lors des mises à jour de la base.

PHP

Pilotes

PHP possède les pilotes officiels suivant pour MySQL :

- `mysql` : ancien mais toujours utilisé dans les applications web ; c'est un module PHP procédural.
- `mysqli` : plus rapide, peut être utilisé comme un ensemble de classes ou comme une bibliothèque procédurale.
- `PDO` (PHP Data Objects) : utilise PDO, une couche abstraite pour interaction avec les bases, avec des pilotes pour MySQL et ODBC.
- `PDO_MYSQL` : propose des fonctionnalités MySQL avancées, et les émulent si absentes.

Les fonctions de ces pilotes utilisent l'API C. Elles peuvent utiliser MySQL Client Library ou mysqlnd, comme pilotes natifs pour PHP.

Parfois, activer mysql et mysqli peut causer des problèmes. Il est donc préférable d'en activer qu'un.

De plus, PHP possède une extension ODBC qui fonctionne avec MySQL.

PEAR est un framework PHP important qui prend en charge MySQL.

register_globals et $_REQUEST

PHP a des variables d'environnement appelées `register_globals`. Depuis PHP 4.2, elles sont False par défaut, et ne doivent pas être activées. Dans PHP 5.3 la fonctionnalité est devenue obsolète et supprimée en PHP 5.4.0.

Cependant, si la version de PHP utilisée accepte register_globals, on peut vérifier s'il est activé lancer `ini_get()`. Si c'est le cas, `ini_set()` ne pourra pas le changer. Il y a deux façons de le faire :

- Éditer `php.ini`.
- Ajouter une ligne au .htaccess :

```
php_flag register_globals off
```

En fait si register_globals est True, un utilisateur peut arbitrairement ajouter des variables à votre script avec ce genre de commande :

```
your_script.php?new_variable=new_value
```

Ne jamais utiliser le tableau `$_REQUEST`, lui préférer :

- $_ENV
- $_GET
- $_POST
- $_COOKIE
- $_SERVER

Cet ordre est celui suivi par PHP, mais il peut être modifié par variables_order.

Cela signifie que si votre script définit une variable serveur appelée "userid" et que l'application tente de la lire dans `$_REQUEST`, l'utilisateur peut prévenir en ajoutant une variable à la requête.

Sécurité

Sécurité

Sur Internet, des pirates tentent de récupérer des données en violant leurs accès. Ils insèrent souvent du code dans une page HTML ou PHP, par injection de code, généralement par[7] :

- le biais d'un formulaire HTML qui permet d'entrer des données dans la base de données
- ou alors en insérant des données par le biais de la barre d'adresse de la fenêtre,
- par le dépôt d'un fichier sur le serveur (*upload*)
- ou toute autre voie d'accès au serveur possible.

Les bases de données permettent d'assurer le stockage des informations en liaison par exemple avec des pages HTML couplées avec PHP. Donc tout le monde peut en avoir accès par l'intermédiaire d'un client relié au serveur. MySQL basé sur SQL est notamment très utilisé avec PHP. Il est donc pertinent d'examiner quelques notions de sécurité en matière de SQL.

En ce qui concerne SQL, il faut savoir premièrement qu'une astuce de programmation ou un détournement de requête possible à partir d'un serveur a de bonnes chances de réussir sur un autre et il en est de même avec les applications web mêmes, qui peuvent être utilisées afin d'atteindre le serveur et d'en saccager les données.

Paramètres de connexion

Parfois, les paramètres de connexion (dont login et mot de passe) sont stockés dans un fichier texte non crypté, comme un .ini. Cela n'est donc pas recommandé : si un utilisateur devine son nom il peut le lire sans peine. S'il est situé en dehors du répertoire WWW c'est mieux, mais la meilleure façon est de les enregistrer dans un programme (ex : .php).

Il est toujours possible pour un utilisateur de trouver les accès FTP, donc il vaut mieux en utiliser d'autres pour MySQL.

Inutile de se souvenir des mots de passe MySQL car le programme est sensé le retrouver automatiquement. Il convient donc d'en choisir un robuste : très long, avec au moins une majuscule, une minuscule, un symbole (ex : '_') et un chiffre. Le tout sans contenir de mots du dictionnaire car ils sont utilisés pour accélérer les crackages lors d'attaque par force brute.

> **Attention !**
>
> Ne jamais les stocker autre part, y compris dans des e-mails.

Injections SQL

Définition

Normalement les valeurs stockées dans `$_POST` peuvent être insérées directement dans des requêtes SQL. Toutefois les injections SQL exploitent une faille de sécurité engendrée par le caractère d'échappement.

Exemple

Si on s'attend à recevoir un nombre (ex : 42), et qu'en fait `$_POST` contient une chaine avec échappement ("'42' OR 1") cela change complètement son résultat :

```
DELETE FROM `articles` WHERE `id`=42       -- supprime une ligne
DELETE FROM `articles` WHERE `id`=42 OR 1  -- supprime toutes les lignes (car 1 est toujours vrai)
```

L'utilisateur peut aussi insérer des commandes plus complexes séparées par des points-virgules :

```
SELECT * FROM `table1` WHERE title='bla bla'                          -- sélectionne
SELECT * FROM `table1` WHERE title='bla bla'; TRUNCATE TABLE `table1`  -- supprime tout
```

Par ailleurs, si un utilisateur découvre comment manipuler une base de cette façon, il peut très bien se créer un compte administrateur ensuite, et effectuer des modifications discrètes dans le but de récupérer de l'argent (fausses factures, phishing...).

Solution

Il faut vérifier que la variable stocke bien le type de données prévu :

- Pour les chaines de caractères : comme elles sont entourées d'apostrophes, tous ceux qu'elles contiennent doivent être convertis (ex : en '' ou \'). Par exemple, PHP propose `mysql_real_escape_string` pour gérer ces substitutions.
- Pour les dates : les entourer d'apostrophes comme les chaines.
- Pour les noms SQL (tables, champs...) : les entourer d'apostrophes comme les chaines.
- Pour les nombres : s'ils contiennent autre chose que des chiffres, ils ne conviennent pas.
- `NULL / UNKNOWN / TRUE / FALSE` : ne doivent jamais être rentrées par l'utilisateur.

Par ailleurs, aucun commentaire SQL ne devrait pourvoir être inséré par l'utilisateur.

Mots de passe

Généralement ils sont cryptés puis dans la base par les applications. Par contre si c'est fait en SQL, c'est qu'ils ont été écrits au moins une fois en clair et donc étés visibles :

- Dans les logs du serveur.
- Dans les logs MySQL.
- Dans `SHOW PROCESSLIST`.

Il est donc fortement déconseillé d'envoyer ce genre de commandes :

```
SELECT 1 FROM `users` WHERE `password`=MD5('abraxas')
```

Préférer depuis un .php par exemple :

```
$sql = "SELECT 1 FROM `users` WHERE `password`=MD5('".md5('abraxas')."')";
```

- Ne jamais utiliser de fonctions de cryptage non sécurisées, comme `PASSWORD()`.
- Ni de cryptage réversible (se contenter de comparer si un chaine cryptée est égale à une autre chaine cryptée comme authentification). Ou bien les mots de passe stockées dans une base doivent être impossibles à sélectionner (avec `SELECT`).
- Seulement du hachage cryptographique, comme SHA256, pas d'algorithmes plus vieux comme MD5.

SSL

Si le contenu de la base est publique, il n'y a pas de raison de crypter les communications.

Toutefois il peut y avoir un accès restreint pour les droits d'écriture, ce qui nécessite des mots de passe.

Le cryptage SSL s'avère une bonne solution pour cela. En effet, en plus de crypter les messages, il certifie que l'utilisateur est bien celui qui était prévu (même si ce dernier a donné son mot de passe par phishing).

Risques dans les manipulations de données

On en retrouve plusieurs types :

- Contournement d'une clause where, c'est-à-dire neutraliser la clause WHERE en injectant une condition constante (toujours vraie ou toujours fausse, comme par exemple : SELECT * FROM table WHERE 1;
- Modification sans limite, comme par exemple la destruction totale de données avec la commande DELETE : DELETE FROM table WHERE 1;
- Insertions indésirables, où le pirate peut injecter une ligne complète qui sera traitée à son insu par la base de données. L'application insère ainsi des valeurs dont elle n'est plus maîtresse, par exemple le pirate peut injecter : md5('secret') sans un formulaire, ce qui revient à crypter le mot 'secret', que la base de données ne pourra plus retrouver ou traiter.
- Exécutions multiples : par exemple une injection pour placer une requête dans une autre requête initiale, afin d'exécuter toutes les injections que veut le pirate.
- Surcharger le serveur : c'est-à-dire modifier une requête afin qu'elle engendre une charge de travail importante et inutile pour le serveur.
- Exportations cachées : en exportant des données, en les extrayant de leur dépôt habituel et les placer par la suite dans un réceptacle public plus facile à lire. Cela ce fait par exemple en contournant la clause WHERE, ou alors en insérant des fichier externes directement sur le serveur SQL.

Chiffrement d'un mot de passe

Le chiffrement d'un mot de passe ou d'une donnée consiste en sa transformation en une autre donnée par le biais d'une méthode extrêmement difficile à inverser voire impossible avec les fonctions de hachage. MySQL offre des méthodes de chiffrement des données et par la même du mot de passe. Nous avons par exemple :

```
INSERT INTO utilisateurs(identifiant, motdepasse) VALUES ('$id', MD5('$mdp'));

Vérification : SELECT id FROM utilisateurs WHERE identifiant = '$idTest' AND motdepasse = MD5('$mdpTest');

Pour du contenu sécurisé à afficher : Stockage : INSERT INTO dossiersSecrets(titre, contenu) VALUES ('$titre', AES_ENCRYPT('$contenu','motdepasse'));

Récupération : SELECT titre, AES_DECRYPT(contenu,'motdepasse') AS contenu FROM dossiersSecrets;
```

Sécurité PHP

Voici quelques types d'attaques que l'on retrouve fréquemment :

- l'injection de code distant.

Par exemple, c'est introduire une instruction PHP dans une page, qui ira chercher du code PHP sur un site distant pour le ramener sur le serveur local du pirate et l'exécuter comme code local, afin de prendre contrôle du serveur et des données du serveur distant piraté.

- l'exécution du code à la volée.

Cela ce fait par exemple à l'aide de : eval(), fonction qui prend en argument une chaîne de caractère pour l'exécuter comme du code PHP ; et par là exécuter des opérations destructrices de données par cette voie.

- le téléchargement de fichiers ; ce qui permet d'importer du code PHP sur un site.

Il suffit en effet que ce site ait des autorisations de lecture et qu'il soit accessible depuis le Web, sans besoin de droit d'exécution ou d'écriture pour exécuter tous les codes PHP que veut le pirate. En stockant le fichier, par la fonction "upload" (ex. image), le code est exécuté sur le serveur pirater et vient en saboter ou déranger le fonctionnement et même la lecture des autres documents.

- La directive register_globals;

Si elle est activée permet d'injecter dans les scripts toutes sortes de variables ; ce qui rend la programmation de script peu sûr. Heureusement que dans les versions supérieures à PHP 4.2 l'option register_globals est désactivée par défaut.

- filtrage des données ;

C'est-à-dire la gestion des erreurs, ou les mécanismes par lesquels l'application web déterminera la validité des données qui entrent et sortent de l'application. Il s'agit donc de s'assurer que : permièrement on ne puisse pas contourné le filtrage des données, que deuxièmement les données invalides ne puissent pas être confondues avec des données valides et enfin identifier l'origine des données.

Vous trouverez les solutions possibles à ces problèmes dans ce guide de sécurité PHP en ligne : http://phpsec.org/.

La solution des systèmes de gestion de contenu ou de création de site web

Une autre solution est d'utiliser des systèmes de création et de gestion de contenu Web, pour créer un site web personnel et l'agrémenter d'une petite bases de données. L'avantage ici est que si vous utilisez par exemple une application comme http://sites.google.com/.

L'avantage est que la sécurité est ici gérée par de grandes sociétés comme ici 'Google' qui engage des experts en la matière. Il devient dès lors très difficile pour des pirates de saccager les données de l'application web que vous créez, ainsi que la base de données.

Sites.google, par exemple, est une application en ligne qui permet le plus simplement possible de créer un petit site Internet perso agrémenter d'une base données. Pas besoin ici de sécuriser les données c'est la société Google, qui s'en charge ! Par exemple, Google.site permet de rassembler rapidement en un seul endroit une série d'informations, tout en sécurité :

- vidéos
- agendas,
- présentations,
- informations stockées dans une base de données,
- pièces jointes,
- texte.

Elle permet également en matière de sécurité d'autoriser leur accès ou leur modification à un groupe restreint, ou carrément au monde entier. Encore une fois, il y a ici la garantie de Google en matière de sécurité des informations.

Voici cependant quelques mesures de sécurité qui sont offertes à l'utilisateur :

- supprimer quelqu'un d'un site ;

- contrôler l'accès au site ;
- supprimer des commentaires et des pièces jointes ;
- système de login avec mot de passe.

En matière de mot de passe il est à noter qu'il s'agit d'avoir sous la main trois types de mot de passe selon leur utilité : un pour des applications banales sans importance que vous utilisez, l'autre pour des applications personnelles (ex. e-mail), et l'autre enfin si possible plus complexes pour des informations hautement importantes et confidentielles que vous désirez gérer depuis Internet (ex. accès pour un site bancaire).

Ouverture à un PC distant

Par défaut le serveur MySQL n'écoute pas sur le réseau. Pour changer cela, il faut préciser son IP publique dans la configuration[8] :

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

ajouter ou modifier la ligne :

```
bind-address = MonIPPublique
```

Si le problème persiste, vérifier que le port 3306 est bien ouvert sur le pare-feu.

Références

1. https://linuxfr.org/users/scurz/journaux/surveiller-un-serveur-mysql-avec-mytop
2. http://hackmysql.com/mysqlreport
3. https://dev.mysql.com/downloads/workbench/
4. http://dev.mysql.com/doc/refman/5.1/en/optimize-table.html
5. http://dev.mysql.com/doc/refman/5.1/en/datetime.html
6. http://dev.mysql.com/doc/refman/5.7/en/index-hints.html
7. http://phpsec.org/
8. http://www.cyberciti.biz/tips/how-do-i-enable-remote-access-to-mysql-database-server.html

Débogage

Introduction

Comme vu précédemment, il peut être utile d'activer les logs sur l'historique des requêtes lors du débogage d'une application qui utilise MySQL.

Gestion des exceptions

En MySQL, les anomalies du type "division par zéro" ne renvoient pas d'erreur mais NULL.

Toutefois il est possible de lever des exceptions lors des manipulations de table, par exemple pour éviter qu'une liste d'insertions s'arrête au milieu à cause d'une contrainte d'unicité. L'exemple ci-dessous fonctionne sur une table InnoDB (et pas MyISAM)[1] :

```
ALTER TABLE `MaTable1` ADD UNIQUE(`id`);

INSERT INTO MaTable1 (id) VALUES('1');
START TRANSACTION;
      INSERT INTO MaTable1 (id) VALUES('2');
      INSERT INTO MaTable1 (id) VALUES('3');
      INSERT INTO MaTable1 (id) VALUES('1');
IF condition THEN
  COMMIT;
ELSE
  ROLLBACK;
END IF;
```

Ici une erreur surgit lors de la deuxième insertion d'un id=1. Selon une condition, on peut donc annuler les insertions de 2 et 3, ou bien les soumettre.

> **Remarque :** par défaut, MySQL est en autocommit, cela signifie qu'un COMMIT est automatiquement effectué après chaque opération (rendant inutile les ROLLBACK). Pour le désactiver, lancer `SET autocommit = 0;`

> **Attention !**
>
> S'il y a plusieurs COMMIT avant un ROLLBACK (par exemple dans une boucle), ce dernier n'annulera que les opérations consécutives au dernier COMMIT.

Erreurs

1130: Host 'example.com' is not allowed to connect to this MySQL server

Dans le cas d'une connexion depuis un PC distant, le compte utilisé n'est pas autorisé. Il faut donc le configurer avec :

```
GRANT ALL PRIVILEGES ON *.* TO 'utilisateur'@'%' WITH GRANT OPTION;
```

au lieu ou en plus de :

```
GRANT ALL PRIVILEGES ON *.* TO 'utilisateur'@'localhost' WITH GRANT OPTION;
```

1093 - You can't specify target table '...' for update in FROM clause

Cela se produit quand on essaie de mettre à jour ou supprimer des lignes selon une sélection de ces mêmes lignes. En effet, il est impossible de mettre à jour une table pendant en même temps qu'elle une subit sous-requête. Par exemple, pour réinitialiser un mot de passe SPIP :

```
mysql> UPDATE spip_auteurs SET pass =
 (SELECT pass FROM spip_auteurs WHERE login='paul') where login='admin';
ERROR 1093 (HY000): You can't specify target table 'spip_auteurs' for update in FROM clause
```

- Passer par des `CREATE TEMPORARY TABLE` (voire `DECLARE` si cela rentre dans une variable scalaire).
- Sinon, il est possible de sélectionner les enregistrements à mettre à jour automatiquement en enveloppant la sous-requête dans une autre, grâce aux tables temporaires générées par les `FROM`[2].

1553: Cannot drop index 'UNIQ_XXX': needed in a foreign key constraint

Il faut supprimer la clé étrangère avant l'index duquel elle dépend :

```
ALTER TABLE `maTable` DROP FOREIGN KEY `FK_XXX`;
ALTER TABLE `maTable` DROP INDEX `UNIQ_XXX`;
```

2003: Can't connect to MySQL server

Changer le paramètre "host".

A new statement was found, but no delimiter between it and the previous one

Cette section est vide, pas assez détaillée ou incomplète.

Ajouter un ";" ?

Cannot add foreign key constraint

Cette section est vide, pas assez détaillée ou incomplète.

Survient lors d'un "CREATE TABLE", et ce n'est pas lié à la valeur de *foreign_key_checks*.

Erreur : fonctionnalités relationnelles désactivées !

Se produit dans le concepteur de diagramme de phpMyAdmin, il faut l'activer dans config.inc.php.

General error: 1215 Cannot add foreign key constraint

Ne pas créer de contrainte d'intégrité entre deux tables de moteur différent (ex : InnoDB vs MyISAM).

General error: 1267 Illegal mix of collations

Trois solutions :

- `ALTER TABLE` pour changer la structure d'au moins une des deux tables jointes, pour uniformiser leur collation.
- `CAST(monChamp AS CHAR CHARACTER SET utf8)`.
- `CONVERT(monChamp USING utf8)`.

Invalid use of group function

- Dans le cas d'un `SELECT`, il conviendrait d'utiliser `HAVING` au lieu de `WHERE` pour modifier des enregistrements en fonction d'autres d'une sous-requête.
- Pour un `UPDATE` ou un `DELETE`, les champs comparés par un `IN` ne sont peut-être pas du même type.

SQLSTATE[23000]: Integrity constraint violation: 1217 Cannot delete or update a parent row: a foreign key constraint fails

Il manque un "DROP FOREIGN KEY" avant un "DROP TABLE" ou un "DROP COLUMN".

Pour les supprimer, l'utilisateur root n'a pas le droit de modifier directement directement la table "information_schema". Il faut donc exécuter les requêtes générées par la suivante :

```
SELECT concat('alter table ',table_schema,'.',table_name,' DROP FOREIGN KEY ',constraint_name,';')
FROM information_schema.table_constraints
WHERE constraint_type='FOREIGN KEY'
AND table_name='maTable';
```

SQLSTATE[42000]: Syntax error or access violation

Utiliser phpMyAdmin pour trouver l'erreur de syntaxe.

This version of MySQL doesn't yet support 'LIMIT & IN/ALL/ANY/SOME subquery'

Remplacer les "IN" par des jointures, ou la sous-requête par une deuxième dont le résultat est stockée dans une table temporaire.

Type d'énoncé non reconnu

Certains mots clés ne sont reconnus que dans les procédures stockées, ou doivent être précédés d'un SELECT.

Références

1. http://stackoverflow.com/questions/2950676/difference-between-set-autocommit-1-and-start-transaction-in-mysql-have-i-misse
2. (anglais) http://www.xaprb.com/blog/2006/06/23/how-to-select-from-an-update-target-in-mysql/

Mots réservés

Langage

Liste des mots réservés MySQL :

```
ACCESSIBLE
ADD
ALL
ALTER
ANALYZE
AND
AS
ASC
ASENSITIVE
AUTO_INCREMENT
BDB
BEFORE
BERKELEYDB
BETWEEN
BIGINT
BINARY
BLOB
BOTH
BY
CALL
CASCADE
CASE
CHANGE
CHAR
CHARACTER
CHECK
COLLATE
COLUMN
COLUMNS
CONDITION
CONNECTION
CONSTRAINT
CONTINUE
CONVERT
CREATE
CROSS
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATABASE
DATABASES
DAY_HOUR
DAY_MICROSECOND
DAY_MINUTE
DAY_SECOND
DEC
DECIMAL
DECLARE
DEFAULT
DELAYED
DELETE
DESC
DESCRIBE
DETERMINISTIC
DISTINCT
DISTINCTROW
DIV
DOUBLE
DROP
DUAL
EACH
ELSE
ELSEIF
ENCLOSED
ESCAPED
EXISTS
EXIT
EXPLAIN
FALSE
FETCH
FIELDS
FLOAT
FLOAT4
FLOAT8
FOR
FORCE
FOREIGN
FOUND
FRAC_SECOND
FROM
FULLTEXT
GENERAL
GRANT
GROUP
HAVING
HIGH_PRIORITY
HOUR_MICROSECOND
HOUR_MINUTE
HOUR_SECOND
IF
IGNORE
IGNORE_SERVER_IDS
IN
INDEX
INFILE
INNER
INNODB
INOUT
INSENSITIVE
```

```
INSERT
INT
INT1
INT2
INT3
INT4
INT8
INTEGER
INTERVAL
INTO
IO_THREAD
IS
ITERATE
JOIN
KEY
KEYS
KILL
LEADING
LEAVE
LEFT
LIKE
LIMIT
LINEAR
LINES
LOAD
LOCALTIME
LOCALTIMESTAMP
LOCK
LONG
LONGBLOB
LONGTEXT
LOOP
LOW_PRIORITY
MASTER_HEARTBEAT_PERIOD
MASTER_SERVER_ID
MASTER_SSL_VERIFY_SERVER_CERT
MATCH
MAXVALUE
MEDIUMBLOB
MEDIUMINT
MEDIUMTEXT
MIDDLEINT
MINUTE_MICROSECOND
MINUTE_SECOND
MOD
MODIFIES
MySQL
NATURAL
NOT
NO_WRITE_TO_BINLOG
NULL
NUMERIC
ON
OPTIMIZE
OPTION
OPTIONALLY
OR
ORDER
OUT
OUTER
OUTFILE
PRECISION
PRIMARY
PRIVILEGES
PROCEDURE
PURGE
RANGE
READ
READS
READ_WRITE
REAL
REFERENCES
REGEXP
RELEASE
RENAME
REPEAT
REPLACE
REQUIRE
RESIGNAL
RESTRICT
RETURN
REVOKE
RIGHT
RLIKE
SCHEMA
SCHEMAS
SECOND_MICROSECOND
SELECT
SENSITIVE
SEPARATOR
SET
SHOW
SIGNAL
SLOW
SMALLINT
SOME
SONAME
SPATIAL
SPECIFIC
SQL
SQLEXCEPTION
SQLSTATE
SQLWARNING
SQL_BIG_RESULT
SQL_CALC_FOUND_ROWS
SQL_SMALL_RESULT
SQL_TSI_DAY
SQL_TSI_FRAC_SECOND
SQL_TSI_HOUR
SQL_TSI_MINUTE
SQL_TSI_MONTH
```

```
SQL_TSI_QUARTER
SQL_TSI_SECOND
SQL_TSI_WEEK
SQL_TSI_YEAR
SSL
STARTING
STRAIGHT_JOIN
STRIPED
TABLE
TABLES
TERMINATED
THEN
TIMESTAMPADD
TIMESTAMPDIFF
TINYBLOB
TINYINT
TINYTEXT
TO
TRAILING
TRIGGER
TRUE
The
UNDO
UNION
UNIQUE
UNLOCK
UNSIGNED
UPDATE
USAGE
USE
USER_RESOURCES
USING
UTC_DATE
UTC_TIME
UTC_TIMESTAMP
VALUES
VARBINARY
VARCHAR
VARCHARACTER
VARYING
WHEN
WHERE
WHILE
WITH
WRITE
XOR
YEAR_MONTH
ZEROFILL
```

Logiciel

En Unix, ce service se lance avec la commande "$ mysql ".

Voici le résultat de la première commande avec MySQL5.4.3 :

```
mysql> ?

For information about MySQL products and services, visit:
   http://www.mysql.com/
For developer information, including the MySQL Reference Manual, visit:
   http://dev.mysql.com/
To buy MySQL Network Support, training, or other products, visit:
   https://shop.mysql.com/

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
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.
notee     (\t) Don't write into outfile.
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.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog
with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'

mysql>
```

Il en existe aussi d'autres :

```
describe   Describe the table in argument.
substring  (or substr) Convert a data type to another in the result.
trim       Suppress the prefixes and suffixes of a string of characters.
alter table
...
```

Références

- http://dev.mysql.com/doc/refman/5.5/en/reserved-words.html
- http://www.htmlite.com/mysql002a.php
