MySQL - Introduction

## La gestion des usagers

Les droits d’un usager sur une base de données peuvent varier dépendamment de l’endroit (machine) où il se connecte.

 Par exemple :

CREATE USER 'usr\_facturation'@***'localhost'*** IDENTIFIED BY 'xyz\_pwd';

GRANT ALL ON ma\_bd\_facturation.\* TO 'usr\_facturation'@***'localhost'***;

## Les engins de tables de MySQL

Voici une liste des engins de tables les plus populaires :

### innoDB

* Engin par défaut de MySQL
* Supporte les transactions (donc les clés étrangères également)
* *Row-lock*

### MyISAM

* *Table-lock*
* Jusqu’à tout récemment, c’était l’engin par défaut de MySQL
* Ne supporte pas les transactions, ni les clés étrangères
* Peut, dans certains cas, être plus rapide si la table est intense au niveau des SELECT

### MEMORY

* Table dont le contenu est gardé en mémoire uniquement.
* Comme toutes les données sont en mémoire (incluant les index), c’est extrêmement rapide. Il faut cependant faire attention. Si le serveur plante ou se ferme, les données seront perdues.

## Certaines clauses et particularités de MySQL

### LIMIT

Afin de limiter les résultats d’une requête, la clause LIMIT est utilisée. Par exemple :

SELECT

 id, name

FROM

users

ORDER BY

name

*LIMIT 10, 20;*

Ceci retourne les résultats à partir de la ligne #11 jusqu’à la ligne #30 (inclusivement). Pour retourner les 10 premières lignes = LIMIT 0,10;

### AUTO\_INCREMENT

Il n’est pas nécessaire de faire des séquences comme avec Oracle (version inférieure à 12c).

CREATE TABLE users (

 id INT NOT NULL **AUTO\_INCREMENT**,

 …

 PRIMARY KEY pk\_users (id)

) ENGINE = innoDB;

### FULLTEXT SEARCH

CREATE TABLE products (

 id INT NOT NULL AUTO\_INCREMENT,

 description text,

 PRIMARY KEY pk\_products (id),

 FULLTEXT idx\_prod\_desc (description)

) ENGINE = innoDB;

SELECT \* FROM products WHERE MATCH(description) AGAINST ("table rouge");

## Syntaxe

### Création d’une base de données

CREATE {DATABASE | SCHEMA } xyz\_db CHARACTER SET = utf8 (ou ascii, greek, …)

*Database et schema sont synonymes pour MySQL.*

### Création d’un usager

CREATE USER 'foo'@'192.168.0.1' IDENTIFIED BY 'pwd';

foo : Le nom de l’usager

192.168.0.1 : Nom de l’hôte d’où il se connecte. Pour n’importe quel hôte, on utilise « % » (ceci ne

remplace pas le « localhost » cependant.

pwd : Mot de passe de l’usager

### Assignation de droits d’un usager à une base de données

GRANT v ON w.x TO 'y'@'z';

V: Privilège donné

* Exemple :
	+ ALL - Tous les privilèges
	+ INSERT - Privilège d’insertion
	+ UPDATE - Privilège de mise à jour
	+ SELECT - Privilège de SELECT
	+ EXECUTE - Permet d’exécuter des procédures stockées
	+ ALTER - Permet de modifier la structure d’une table avec ALTER TABLE

W: Nom de la base de données (ex : xyz\_db)

X: Nom de la table, ou \* pour toutes les tables

Y: nom de l’usager (ex: foo)

Z: hôte (ex: %)

Exemple complet : GRANT SELECT, INSERT ON xyz\_db.\* TO 'foo'@'192.168.0.1';

### Suppression de droits d’un usager sur une base de données

REVOKE INSERT ONT xyz\_db.\* TO 'foo'@'192.168.0.1';

### Exemple de création d’une table simple

La colonne “status” permet une énumération de valeurs permises.

CREATE TABLE users (

 id INT NOT NULL AUTO\_INCREMENT,

 status ENUM("pending", "inactive", "active") DEFAULT "pending",

 password VARCHAR(255) NOT NULL,

 email VARCHAR(255) NOT NULL,

 data JSON not null,

 PRIMARY KEY pk\_users(id),

 INDEX idx\_users\_email (email)

) ENGINE = innoDB;

### Exemple de création d’une table avec une clé étrangère

CREATE TABLE forgot\_passwords (

 id INT NOT NULL AUTO\_INCREMENT,

 id\_user INT NOT NULL,

 access\_key VARCHAR(70),

 PRIMARY KEY pk\_temporary\_passwords (id),

 CONSTRAINT fk\_temporary\_passwords\_id\_user FOREIGN KEY (id\_user) REFERENCES users (id)

) ENGINE = innoDB;

## « Collation »

Le terme anglais « collation » signifie un ensemble de règles sur la manière dont les données sont comparées et triées.

Voici un exemple :

Si une table #1 est avec la collation « utf8\_general\_ci », elle sera insensible aux accents et à la casse, dû au suffixe « \_ci » et supportera le character set utf8.

Au lieu de « general », la collation aurait pu être « utf8\_unicode\_ci ». La différence est la suivante :

* Général :
	+ Plus rapide
* Unicode
	+ Supporte les combinaisons de caractères
		- ß = ss (alors qu’avec général, ß = s)

## Outils d’administration de MySQL

### La commande SHOW

En utilisant la commande show, on peut avoir plusieurs informations sur la base de données. Voici quelques-unes des options de la commande SHOW.

SHOW DATABASES

* Permet de voir toutes les bases de données sur le serveur

SHOW TABLES

* Permet de voir toutes les tables pour la base de données sélectionnées

SHOW WARNINGS

* Il arrive qu’une opération se termine avec un warning. Il est donc possible d’afficher plus de détails sur l’avertissement

Exemple :

DROP TABLE IF EXISTS blabla\_existe\_pas;

Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;

Note | 1051 | Unknown table 'blabla'

SHOW PROCESSLIST

* Permet de voir tous les processus de connexion sur la base de données de MySQL. Ainsi on peut voir s’il y a des requêtes qui ne se terminent pas, ou qui sont lentes.

### L’optimisation de requêtes

Les logs

Il existe plusieurs types de logs, dont le « slow query log ». Celui-ci doit être activé, mais permet de savoir quelles sont les requêtes qui sont dangereusement lentes pour le serveur.

Pour activer les logs en général, on peut utiliser MySQL Workbench 🡪 Server administration 🡪 Log files 🡪 Enable General Log

Un fichier sera maintenant disponible qui contient toutes requêtes exécutées sur la base de données. Ceci peut aider lors de l’administration, mais c’est un fichier qu’il faut absolument protéger, car on pourrait y voir des mots de passes et de l’information confidentielle.

La commande EXPLAIN

* Permet d’avoir des informations sur une table (comme DESCRIBE), ou d’avoir des informations sur l’analyse d’une requête SELECT.
* EXPLAIN est particulièrement pratique pour les requêtes avec des jointures ou des unions. Ou encore pour les requêtes lourdes, afin de les optimiser.

Exemple : EXPLAIN SELECT name FROM users WHERE email LIKE ‘fr%’;

Analyse d’une requête avec EXPLAIN



Le « select\_type » fournit une information sur le type de requête. Dans ce cas-ci, il n’y a pas d’union et ce n’est pas une requête imbriquée.

« Type » est la colonne la plus importante. Voici les valeurs possibles (du mieux au pire):

* Const : On utilise la clef primaire ou une clef unique au complet (WHERE id = ‘…’);
* Eq\_ref : Un peu comme conf, mais lorsqu’il y a une jointure de tables.
* Ref : Lorsqu’il y a une colonne indexée et utilisée avec l’opérateur = ou <=>
* Fulltext : Lorsque le scan est avec un index « FULL TEXT »
* Ref\_or\_null : Comme ref, mais la requête possède « OR email IS NULL »
* Index\_merge : Utilisation de 2 index dans la même table
* Unique\_subquery : SELECT … WHERE email **IN** (SELECT ID FROM … ) où la sous-requête retourne

une valeur unique

* Index\_subquery : SELECT … WHERE email **IN** (SELECT first\_name FROM … ) où la sous-requête

retourne une colonne indexée

* Range : Lors de <=, >=, between, …
* Index : Souvent utilisé lorsqu’il y a l’opérateur LIKE ‘af%’
* All : Scan complet de la table. Mauvais… très mauvais

« possible\_keys » montre les clés possible pour la requête.

« Rows » montre le nombre de lignes qui ont fait partie de la requête.

Création de copies de sauvegarde avec mysqldump

Grâce à l’option mysqldump, il est possible de facilement exporter le contenu des bases de données mysql dans des fichiers textes (CSV, XML, …). Il est également possible d’utiliser mysqldump pour transférer le contenu d’une base de données vers une autre base de données.

La commande suivante permet de se connecter avec l’usager root, de lire le contenu de la base de données xyz\_db et d’en faire une copie dans fichier-backup.sql

mysqldump –u root –p xyz\_db > fichier-backup.sql

On peut ensuite restaurer la base de données en utilisant :

mysql –u root –p xyz\_db < fichier-backup.sql