



Administration et optimisation de MySQL

V 0.1 (2 mai 2020) - IT-Akademy (Lyon, mai 2020)

Benoît Prieur - SoartheC - CC-BY-SA 4.0



Tour de table

- Connaissance sur les bases de données relationnelles ?
- Connaissances sur MySQL ?
- Connaissances sur le langage SQL ?



TP Final (à faire au fur et à mesure si possible)

- Créer une base simple avec une table STUDENTS, une table TEACHERS, une table d'association S-T pour gérer cette relation many-to-many.
- Pour chaque étape abordée dans les slides :
 - Faire une action sur la base de données.
 - Documenter l'action.



Présentation de MySQL

- Système de gestion de bases de données relationnelles (SGBDR).
- Depuis la fin des années 1990.
- Michael Widenius (principal auteur).
- MySQL racheté par Sun Microsystems (2008). Sun Microsystems rachetée par Oracle Corporation (2009). Système Oracle et MySQL appartiennent à la même entreprise.



Clients MySQL en ligne de commande

- *mysql* : exécution des requêtes.
- *mysqldump* : sauvegardes logiques.
- *mysqladmin* : opérations d'administration.
 - Obtenir les statistiques.
 - Interrompre les requêtes.
 - Gérer les mots de passe.



Protocoles de communication

- *TCP/IP* : local ou distant.
- Unix Socket : local uniquement (sous Unix, MacOS).
- Shared Memory : local uniquement (sous Windows).
- Named Pipes : local uniquement (sous Windows).



Les outils d'utilisation

- PhpMyAdmin (solution web).
- SQL Workbench.
- La ligne de commande.



Installation de MySQL et de divers outils

- Selon les plateformes (Windows, Linux, MacOS) de chacun.e, installer MySQL.
- Installer un client. PhpMyAdmin ou SQL Workbench.
- Vérifier en ligne de commande que l'installation est correct.
 - (localhost, user, pwd) doit permettre de se connecter à MySQL.



Persistance sur le disque

- Les bases de données, appelées également schémas.
Répertoire dédié du nom de la base de données.
- Dans ce répertoire, des fichiers .frm définissant la structure des tables.
- Les journaux.
- Les déclencheurs appelés triggers.



Les bases de données proches de MySQL

- **MariaDB** : système (open source) lancé par Michael Widenius. Migrations de l'un à l'autre pas si simple.
- **Percona Server** : production de patches puis d'un produit.
- **Amazon RDS/Aurora** : préconfiguration sur EC2.
- **WebScaleSQL** : co-développé par plusieurs grands acteurs (Google, FB, etc.).



Moteur de stockage (1)

- Moteur de base de données (synonymes).
- Contrôle, lit, enregistre et trie les informations.
- Surcouche intégrant un système de fichiers propre au moteur de stockage.
- Par défaut avec MySQL : InnoDB.



Moteur de stockage (2)

On précise le moteur de stockage à la création de la table (InnoDB par défaut). On peut le connaître avec SHOW CREATE TABLE;

```
> create database ITAKADEMY;
```

```
> use ITAKADEMY;
```

```
> CREATE TABLE t5 (id INT AUTO_INCREMENT, NOM CHAR (20), PRIMARY KEY (id))  
ENGINE=InnoDB;
```

```
> SHOW CREATE TABLE t5;
```



Moteur de stockage (3)

On peut également connaître le moteur de stockage grâce à INFORMATION_SCHEMA.

```
> SELECT TABLE_NAME, ENGINE FROM INFORMATION_SCHEMA.TABLES WHERE  
TABLE_NAME = 't5';
```

```
+-----+-----+  
| TABLE_NAME | ENGINE |  
+-----+-----+  
| t5          | InnoDB |  
+-----+-----+
```

1 row in set (0.00 sec)



Moteur de stockage (4)

Le choix du moteur de stockage est réversible et peut être modifié avec ALTER TABLE. Ici on passe de InnoDB à MyISAM.

```
> ALTER TABLE t5 ENGINE = MyISAM;
```



Moteur de stockage InnoDB

- Transactionnel ACID (Atomicité, Cohérence, Isolation, Durabilité)
 - Consultation de l'article Wikipédia en français :
https://fr.wikipedia.org/wiki/Propri%C3%A9t%C3%A9s_ACID
- MVCC (Multi Version Concurrency Control)
 - Les lectures ne bloquent les écritures et réciproquement.



Moteur de stockage MyISAM

Moteur historique lié à MySQL. Ne pas utiliser MyISAM. Un arrêt subitement du serveur peut faire perdre les données alors manipulées. Ce qui n'est pas le cas avec InnoDB.

- nom_table.frm : structure de la table.
- nom_table.MYD : les données.
- nom_table.MYI : les index.



Autres exemples de moteurs de stockage utilisables avec MySQL

- Archive.
- XTraDB.
- TokuDB.
- ... Il y en a d'autres, pertinents chacun dans des contextes particuliers.



La notion de verrou

- Exemple d'un accès à un fichier (analogie).
 - Ouverture en lecture.
 - Ouverture en écriture.
- Verrouillage de toute une table ou des quelques lignes concernées par l'écriture. Cette dernière fonctionnalité est permise par InnoDB.



La notion de verrou implicite

- Implémentée par InnoDB ou TokuDB pour SELECT, INSERT, UPDATE.
- MyISAM ou Memory verrouillent la table entière.
- InnoDB permet à plusieurs clients (grâce à ce système de verrou implicite) :
 - de lire des lignes, identiques ou différentes.
 - de lire des lignes et écrivent d'autres lignes.
 - d'écrire des lignes différentes.



La notion de verrou explicite

- Certaines opérations ont besoin d'un verrou explicite. Fonctionne avec InnoDB.
 - LOCK TABLES.
 - UNLOCK TABLES.

> *LOCK TABLES A READ, B READ, C WRITE;*

- Autres types de verrous avec InnoDB : verrou coopératif (par exemple).



La notion de transaction

- Transaction \Leftrightarrow ensemble de requêtes qui forme une unité logique.
- Doit respecter COMMIT/ROLLBACK mais pas seulement.
- ACID :
 - Atomicité : toutes les requêtes sont exécutées OK ou toutes annulées.
 - Cohérence : si BDD est cohérente avant, elle l'est après.
 - Isolation : les requêtes d'une transaction n'affectent pas les autres transactions.
 - Durabilité : On s'assure qu'aucune donnée n'est perdue.



Exemple de transactions (1)

- START TRANSACTION
- COMMIT

> *START TRANSACTION;*

> *INSERT INTO t5 (NOM, NUMERO) VALUES ('JOHN DOE', '5345353');*

> *INSERT INTO t5 (NOM, NUMERO) VALUES ('JOHN DOE 2', '999998');*

> *COMMIT;*



Exemple de transactions (2)

- Première session cliente (MySQL)

```
> START TRANSACTION;  
> DELETE FROM MA_TABLE;  
> SELECT COUNT(*) FROM MA_TABLE;  
>>>>>>>> 0
```

- Deuxième session cliente (MySQL)

```
> SELECT COUNT(*) FROM MA_TABLE;  
>>>>>>>> 12
```

- Première session cliente (MySQL)

- On fait un COMMIT ou un ROLLBACK pour confirmer ou non.



Configuration du serveur (1)

- Plusieurs manières de configurer le serveur MySQL :
 - En éditant le fichier `my.cnf` (`my.ini` sous Microsoft Windows).
 - N'existe pas sous MacOS par défaut. Il faut le créer.
 - Chemin habituel : `/usr/local/mysql/etc/my.cnf`
- Possibilité de le configurer de manière dynamique :
 - “à chaud” avec la commande `SQL SET`



Configuration du serveur (2)

- Dynamiquement, on peut :
 - Changer la configuration pour toutes les sessions.
 - Seulement pour la session courante.
- On utilise pour cela :
 - *SET GLOBAL*
 - *SET SESSION*



Configuration du serveur : exemple dynamique

```
mysql> SHOW GLOBAL VARIABLES LIKE 'sort_buffer_size';  
mysql> SHOW SESSION VARIABLES LIKE 'sort_buffer_size';  
mysql> SET SESSION sort_buffer_size=1055555;  
mysql> SHOW GLOBAL VARIABLES LIKE 'sort_buffer_size';  
mysql> SHOW SESSION VARIABLES LIKE 'sort_buffer_size';
```



Connaître avec certitudes les paramètres du serveur

Le fichier .cnf peut ne pas être à jour car surchargé par des SET dynamiques. Pour connaître le paramétrage du serveur il faut donc le requêter.

> *SHOW GLOBAL VARIABLES*



Statut courant du serveur

> *STATUS;*

```
mysql Ver 8.0.19 for macos10.15 on x86_64 (MySQL Community Server - GPL)
Connection id:          16
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:         stdout
Using outfile:         ''
Using delimiter:      ;
Server version:        8.0.19 MySQL Community Server - GPL
Protocol version:     10
Connection:            Localhost via UNIX socket
Server characteraset:  utf8mb4
Db characteraset:     utf8mb4
Client characteraset:  utf8mb4
Conn. characteraset:   utf8mb4
UNIX socket:          /tmp/mysql.sock
Binary data as:       Hexadecimal
Uptime:               15 days 20 hours 55 min 56 sec
```



Journalisation

- Journalisation ou historique des événements ou log (en anglais).
 - Journal binaire.
 - Journal des requêtes lentes.
 - Journal des erreurs.
 - Journal général (pas détaillée ici).



Journal binaire

- Log de toutes les opérations de type INSERT, UPDATE, DELETE, DROP, CREATE, ALTER, journalisées en binaire.
- > *SHOW BINARY LOGS;*
 - Permet d'obtenir tous les journaux binaires en cours.
- On peut empêcher “à chaud” la journalisation binaire pour une session en modifiant *SQL_LOG_BIN*.
 - > *SELECT @@session.sql_log_bin;*
 - > *SET SESSION sql_log_bin=1;*



Journal des requêtes lentes

- En session ou en global.
- *slow_query_log = 1*
- *slow_query_log_file = /var/lib/mysql/lenteur.log*
- *long-query-time = 1*
- On peut également indiquer une table comme cible de journalisation :
 - *log_output = TABLE*



Un mot sur les requêtes lentes

“Il est plus avantageux d’optimiser une requête rapide souvent utilisée plutôt que d’optimiser une requête lente rarement utilisée.”



journal des erreurs

- Ce journal est systématiquement activé par défaut.
- Obtenir le fichier dans lequel sont enregistrées les erreurs :
 - > *SHOW GLOBAL VARIABLES LIKE 'log_error';*



Le mode SQL

- Pour connaître le mode courant de la session et le mode global :
 - `SELECT @@SESSION.sql_mode;`
 - `SELECT @@GLOBAL.sql_mode;`
- Exemple de comportement selon l'utilisation de `STRICT_ALL_TABLES` et `NO_ENGINE_SUBSTITUTION`.
- Changer le mode SQL :
 - `SET sql_mode = 'NO_ENGINE_SUBSTITUTION';`



Utiliser le cache de requêtes

- `SHOW VARIABLES LIKE 'query_cache_type';`
- Différentes options :
 - **ON** : le serveur tente de mettre en cache toutes les requêtes `SELECT` (sauf celle en `SQL_NO_CACHE => SELECT SQL_NO_CACHE`).
 - **OFF** : aucune requête n'est mise en cache.
 - **DEMAND** : le serveur met en cache les requêtes `SELECT` avec le mot-clé `SQL_CACHE => SELECT SQL_CACHE`.



Rappel de la notion de vue

- *Une vue est une sorte de table virtuelle.*

```
CREATE VIEW ToutLeMonde AS
  SELECT e.nom as Employe, d.nom as Departement
  FROM Employes e,Departements d
  WHERE e.departement = d.id;
```

(exemple Wikipédia)



Rappel de la notion de procédure stockée

- *Une procédure stockée (ou stored procedure) : ensemble d'instructions SQL précompilées, stockées dans une base de données et exécutée sur demande.*

Utilisation de CREATE PROCEDURE.



Sécurité : chiffrement des données dans MySQL

- AES_ENCRYPT()/AES_DECRYPT()
- DES_ENCRYPT()/DES_DECRYPT()
- ENCODE()/DECODE()

> *INSERT INTO TABLE_A VALUES (1, AES_ENCRYPT('Phrase','crypt_key'));*

> *SELECT j, k FROM TABLE_A;*

?z#wdg,<

> *SELECT j, AES_DECRYPT(j,'crypt_key') AS data_crypt FROM TABLE_A;*



Plusieurs options de sécurité

- Se déclare préférentiellement dans le fichier de configuration .cnf :
 - *skip-networking* : pour empêcher les connexions distantes TCP/IP.
 - *bind-address* : pour spécifier une adresse IP unique d'accès à la base de données.
 - *skip-name-resolve* : empêche l'utilisation de DNS => adresse IP ou localhost.
 - Quelques autres, très spécialisées.



Gestion des utilisateurs (1)

- Schéma système mysql, les tables suivantes :
 - *user, db, tables_priv, columns_priv, procs_priv*

Table db : droits spécifiques à une base de données (ou schémas) de chaque utilisateur.

Table tables_priv : droits spécifiques à une table ou à une vue.

Table columns_priv : droits spécifiques à une colonne.

Table procs_priv : droits pour les procédures stockées.



Gestion des utilisateur (2)

- > *SELECT user, host FROM mysql.user WHERE user = 'root';*
- *CREATE USER* : créé un compte utilisateur et donne un mot de passe.
- *GRANT* : permet de donner des droits à un user.
- *REVOKE* : permet de supprimer les droits d'un compte utilisateur mais pas de supprimer le compte.
- *DROP USER* : permet de supprimer un compte utilisateur.
- *SHOW GRANTS* : permet de visualiser les droits.



Gestion des utilisateurs (3) - perte du mot de passe admin

- Perte du mot de passe admin/root :
 - Utiliser l'option *skip-grant-tables*.
 - Connection sans chargement des tables utilisateurs.
 - Permet de changer le mdp du compte admin.
 - Grande vulnérabilité pendant l'opération.
 - Utiliser *FLUSH PRIVILEGES*;
 - > *SET PASSWORD FOR 'root'@'localhost' = PASSWORD('Youpi');*



Gestion des utilisateurs (4)

- > *CREATE USER 'toto' IDENTIFIED BY 'TutuPwd'*;
- Jusqu'à version MySQL 5.6.x, le mot de passe peut apparaître en clair dans les journaux. Maîtriser les accès des fichiers concernés.
 - Essayer de donner des droits à un user puis les lui retirer partiellement.
 - Enfin supprimer le user.



Sécurité des vues

- Exécution avec les droits de l'utilisateur :
> **CREATE SQL SECURITY INVOKER** VIEW MAVUE AS SELECT name FROM STUDENTS WHERE name='toto';
- Exécution avec les droits du créateur de la vue :
> **CREATE SQL SECURITY DEFINER** VIEW MAVUE AS SELECT name FROM STUDENTS WHERE name='toto';



Aspects sauvegarde et restauration (1)

- Sauvegarde dans un fichier :
 - > ***SELECT * FROM MATABLE INTO OUTFILE '/var/tmp/matable.txt';***

- Restaurer un fichier :
 - > ***LOAD DATA INFILE 'file/.dat' INTO TABLE MATABLE;***



Aspects sauvegarde et restauration (2)

- Outils *mysqldump* fourni en standard avec MySQL.
- Se lance en ligne de commande.
- Nombreuses options de sauvegarde :

<https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html>



Premiers aspects d'optimisation

- Des variables à modifier pour optimisation :
 - *cpufreq-info* (vitesse CPU)
 - *InnoDB_buffer_pool_reads* (mémoire vive)
- Des considérations quant au disque :
 - RAID vs SSD
- L'OS du serveur de bases de données :
 - Historiquement Windows pouvait être un peu moins performant (vrai pour les versions < 5.5).



Optimisation dans la conception : les types (1)

- Choisir les types de colonnes les plus adéquats. Par exemple :
 - Préférer *TINYINT UNSIGNED* pour un numérique plutôt petit.
 - Réserver *INT UNSIGNED* pour les nombres très grands.
 - Pour les chaînes de caractères : bien évaluer les tailles supposées avant de choisir un *VARCHAR 255*.
- En termes d'encodage, *UTF8* n'est pas toujours judicieux :
 - Par exemple, la table contient essentiellement des textes en chinois ou en russe.



Optimisation dans la conception : les types (2)

- Une ressource qui liste de façon claire l'ensemble des types de colonnes en SQL :
 - https://www.w3schools.com/sql/sql_datatypes.asp



Optimisation : les indexes (1)

- Lors de la création :

```
> CREATE TABLE MaTable(  
  
    id INT NOT NULL,  
    col1 VARCHAR(30) NOT NULL,  
    col2 VARCHAR(30) NOT NULL,  
    KEY idx_col1 (col1)  
);
```



Optimisation : les indexes (2)

- Après la création de la table (deux façons de faire) :

> **CREATE INDEX** *idx_id* **ON** *MaTable* (*id*);

> **ALTER TABLE** *MaTable* **ADD INDEX** (*col2*);



Optimisation : les indexes (3)

- Pour aller plus loin : beaucoup d'optimisations possibles avec les indexes (*Hash*, *B-Tree*).
- Une optimisation facile (voire une bonne pratique) est d'indexer les colonnes impliquées dans les clauses *WHERE* fréquemment utilisées.



Le plan d'exécution ; comment le connaître ?

- *EXPLAIN* permet d'en savoir plus.
- > *EXPLAIN SELECT COUNT(*) FROM MaTable\G*



Défragmentation des tables

- > *OPTIMIZE TABLE t;*
- > *ALTER TABLE t ENGINE=INNODB;*



La réplication (1)

- La réplication signifie que les données écrites sur le "maître" MySQL sont copiées sur des "esclaves" faisant office de copies (de sauvegarde).



La réplication (2)

- On crée un compte de réplication :
 - > *GRANT REPLICATION SLAVE, REPLICATION CLIENT on *.* to 'replication_utilisateur'@'IP_esclave' IDENTIFIED BY 'mdp';*
- Dans le fichier .cnf on active le journal binaire et on déclare un id du master

```
[mysqld]
log_bin = /var/lib/mysql/mysql-bin
server_id = 100
```
- Procédure complète :
<http://www.responsive-mind.fr/replication-mysql-master-master/>
- Différence entre Master-Slave et Master-Master



Rappel du TP final

- Créer une base simple avec une table STUDENTS, une table TEACHERS, une table d'association S-T pour gérer cette relation many-to-many.
- Pour chaque étape abordée dans les slides :
 - Faire une action sur la base de données.
 - Documenter l'action.