SQL – Backups

Afin d'éviter de perdre toutes les modifications de schémas de données, il est recommandé de procéder régulièrement à des sauvegardes de notre DB.

Il en existe plusieurs types qui sont plus ou moins complexes.

SQL – mysqldump

Pour un simple backup, nous utiliserons (dans une session système, pas MySQL) la fonction `mysqldump`.

La requête générique est la suivante :

```
mysqldump [options_connexion] {choix_db}
```

SQL – mysqldump

- Les options de connexion sont les mêmes que celles de `mysql` (`-u`, `-p`, `--default-character-set`, etc.)
- Le choix de la db se fait soit :
  - Avec le nom seul (pour ne sauvegarder que certaines tables de cette DB, il faut simplement les indiquer à la suite du nom de la DB)
  - Avec l'option `--databases` suivie d'une liste de DB séparées par des espaces
  - Avec l'option `--all-databases` pour toutes les DB

Rem : il existe une multitude d'autres options pour, par exemple, ajouter les requêtes de `DROP DATABASE`, `DROP TABLE`, etc.

SQL – mysqldump

Pour préciser le fichier de sauvegarde, il faut ajouter à la fin de la commande :

- … > fichier_bu.sql
- … `--result-file=fichier_bu.sql`

SQL – mysqldump

Sauvegarde de toute la DB clicom

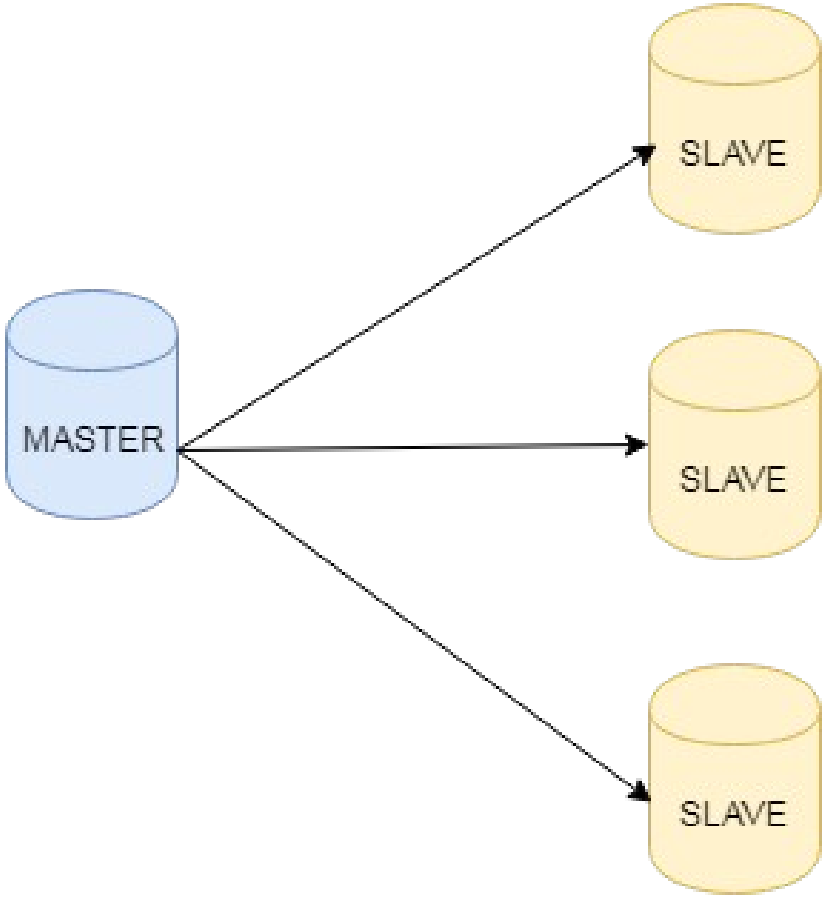
```
mysqldump -u root -p clicom > clicom.sql
```

Sauvegarde de la table client de la DB clicom

```
mysqldump -u root -p clicom client > clicom.sql
```

SQL – Réplication

La réplication repose sur un principe MASTER-SLAVE(S).



SQL – Réplication

Le but recherché et d'avoir (généralement en temps réel) deux (ou plusieurs) DB identiques pour nous permettre de basculer sur un deuxième serveur en cas de défaillance du premier.

Ce principe est basé sur le fait que le serveur MySQL garde la trace de toutes les modifications (insertions, mises à jour, suppressions, etc.) dans un fichier de log binaire.

Les SLAVES vont accéder à ce fichier de log sur le MASTER afin de pouvoir exécuter les mêmes requêtes.

SQL – Réplication

Le fichier de log binaire enregistre les modifications depuis un certain TIMESTAMP (démarrage du log).

Tous les SLAVES auront besoin de la copie des données qui existaient au moment de ce TIMESTAMP (principe des sauvegardes incrémentales).

Sans une copie conforme des données du MASTER au moment du démarrage du log binaire sur le SLAVE, la réplication échouera.

SQL – Réplication

La réplication est une solution qui comprend plusieurs avantages :

- La sauvegarde en temps réel est possible
- Amélioration des performances
  - Par le découplage des types de requête (Les requêtes de modification – schémas et données – doivent être exécutées sur le MASTER et celles de lecture sur le MASTER ou un SLAVE)
  - Exécution de requêtes statistiques et/ou d'analyse sur un SLAVE

SQL – Réplication

La réplication est une solution qui comprend plusieurs avantages :

- Sécurité des données : en réalisant des backups sur un SLAVE (en interrompant la réplication durant ces opérations) → pas de corruption des données sur le MASTER
- Le SLAVE n'a pas besoin de garantir une connexion permanente car sauvegardes incrémentales
- Sauvegardes distantes

SQL – Réplication

Déclaration d'un MASTER

Il faut commencer par modifier le fichier de configuration (généralement `my.ini` ou `my.cnf`).

La première chose à faire est d'autoriser les connexions à distance en commentant la ligne de l'option `bind-address` :

```
[mysqld]
# bind-address 127.0.0.1
```

SQL – Réplication

Déclaration d'un MASTER

Ensuite, il faut préciser quelques paramètres (lorsque c'est fait, vous redémarrez le serveur) :

```
[mysqld]
# identifiant unique du serveur
server-id = 1
# emplacement du fichier de log binaire
log_bin = c:\\chemin_log\mysql-bin.log
# temps d'expiration des logs
expire_logs_days = 14
# taille du fichier de log
max_binlog_size = 500M
```

SQL – Réplication

Déclaration d'un MASTER

Nous devons ensuite créer un utilisateur pour la réplication :

```
GRANT REPLICATION SLAVE
ON *.*
TO replication@'%'
IDENTIFIED BY 'replication';
```

SQL – Réplication

Suivi de :

```
FLUSH TABLES WITH READ LOCK;
```

afin de forcer un `LOCK` des tables et de vider tous les caches utilisées pour s'assurer de ne plus modifier le fichier de log car nous devons prendre note deux informations via un :

```
SHOW MASTER STATUS;
```

Les informations à noter sont le nom du fichier et la position dans celui-ci. Nous aurons ainsi notre point de départ (équivalent au TIMESTAMP).

SQL – Réplication

Nous pouvons maintenant libérer les tables :

```
UNLOCK TABLES;
```

SQL – Réplication

Déclaration d'un SLAVE

Nous allons également modifier le fichier de configuration (et ensuite relancer le serveur) :

```
[mysqld]
# identifiant unique du serveur
server-id = 2
# emplacement du fichier de log binaire
log_bin = c:\\chemin_log\mysql-bin.log
# temps d'expiration des logs
expire_logs_days = 14
# taille du fichier de log
max_binlog_size = 500M
# relay-log = c:\\chemin_log\mysql-relay-bin.log
```

SQL – Réplication

Déclaration d'un SLAVE

Nous devons ensuite indiquer au SLAVE sur quel MASTER il doit pointer:

```
CHANGE MASTER TO
MASTER_HOST='adresse ip ou nom DNS du MASTER',
MASTER_USER='replication',
MASTER_PASSWORD='replication',
MASTER_LOG_FILE='mysql-bin.000001', # selon relevé sur MASTER
MASTER_LOG_POS=779; # selon relevé sur MASTER
```

SQL – Réplication

Déclaration d'un SLAVE

On peut enfin démarrer l'esclave :

```
START SLAVE;
SHOW SLAVE STATUS \G;
```

SQL – Réplication

Déclaration d'un SLAVE

S'il ne démarre pas ou si une erreur de type « logging » apparaît, il peut être nécessaire de stopper le log du serveur actif (avant de le relancer) :

```
STOP SLAVE;

SET GLOBAL slow_query_log = OFF;
SET GLOBAL sql_log_bin = 0;

START SLAVE;

SET GLOBAL slow_query_log = ON;
SET GLOBAL sql_log_bin = 1;
```

SQL – Réplication

Il vous reste à tester la réplication en créant une DB sur le MASTER par exemple ou en insérant un enregistrement dans une table.