Fichiers et bases de données

## Gestion des utilisateurs

### Tables système

information\_schema : infos sur les tables, les colonnes, le type des colonnes et les procédures.

performance\_schema : infos sur les actions faites sur le serveur.

user : infos sur les utilisateurs et leurs privilèges.

db : privilèges par bd.

tables\_priv : privilèges au niveau des tables.

columns\_priv : privilèges au niveau des colonnes.

proc\_priv : privilèges au niveau des procédures.

### Gestion des utilisateurs et des privilèges

* Créer un utilisateur :

CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';

* Création d'un utilisateur nommé "newuser" avec un mot de passe "password".

Remarques :

* IDENTIFIED BY pas obligatoire. L'utilisateur n'a pas de mdp si rien n'est précisé.
* Mdp hashé avant d'être stocké dans mysql.user.
* Peut accepter différentes options :
* Méthodes d'authentificat° : IDENTIFIED [WITH MYSQL\_NATIVE\_PASSWORD/SHA256\_PASSWORD/CACHING\_SHA2\_PASSWORD] BY 'password';
* Rôle : DEFAULT ROLE role1, [role2]. L'utilisateur aura tous les droits liés à ce rôle. Pour créer un rôle et ses droits : CREATE ROLE role1; + GRANT SELECT ON \*.\* TO role1;
* Options WITH : MAX\_QUERIES\_PER\_HOUR, MAX\_UPDATES\_PER\_HOUR, MAX\_CONNECTIONS\_PER\_HOUR, MAX\_USER\_CONNECTIONS. Ttes suivient par une valeur (0 = pas de limite). Pour mettre +ieurs options, les mettre les unes après les autres sans les séparer par une virgule.
* PASSWORD EXPIRE INTERVAL x DAY, PASSWORD HISTORY x (Retenir les x derniers mdp afin que l'utilisateur ne réutilise pas le même), FAILED\_LOGIN\_ATTEMPTS x (x tentatives de connexion avant blocage du compte), PASSWORD\_LOCK\_TIME x (Nbre de jours de blocage du compte).
* ACCOUNT LOCK/UNLOCK
* Modifier mdp d'un utilisateur :

ALTER USER 'newuser'@'localhost' IDENTIFIED BY 'new\_password';

* Modification du mdp de "newuser" en "new\_password".
* Supprimer un utilisateur:

DROP USER 'user1'@'localhost';

* Renommer un utilisateur :

RENAME USER 'anciennom'@'localhost' TO 'nouveaunom'@'localhost';

* Attribution de privilèges :

GRANT ALL PRIVILEGES ON \*.\* TO 'user1'@'localhost';

* Attribution de tous les privilèges sur ttes les bases de données à "user1".

GRANT ALL PRIVILEGES ON dbexercices\_q2.\* TO 'user1'@'localhost';

* Attribution de tous les privilèges sur la base de données "db\_exercices\_q2" à "user1".

GRANT SELECT, INSERT, UPDATE, DELETE ON dbexercices\_q2.\* TO 'user1'@'localhost';

* Attribution de des privilèges spécifiques SELECT, INSERT, UPDATE et DELETE sur la base de données "db\_exercices\_q2" à "user1".

GRANT SELECT, INSERT ON dbexercices\_q2.client TO 'user1'@'localhost';

* Attribution de des privilèges spécifiques SELECT et INSERT sur la table "client" de la base de données "db\_exercices\_q2" à "user1".

GRANT SELECT, UPDATE ON dbexercices\_q2.client, dbexercices\_q2.employe TO 'user1'@'localhost';

* Attribution de des privilèges spécifiques SELECT et INSERT sur les tables "client" et "employe" de la base de données "db\_exercices\_q2" à "user1".

Remarques :

* Privilèges pouvant être attribués :
* Gestion des enregistrements : SELECT, INSERT, UPDATE, DELETE.
* Gestion de la structure des tables : CREATE, CREATE TEMPORARY TABLES, CREATE VIEW, ALTER, DROP.
* Autres éléments de la db : CREATE ROUTINE, ALTER ROUTINE, EXECUTE, INDEX, TRIGGER, LOCK TABLES, CREATE USER.
* Lors de la créat° d'une procédure stockée/trigger/vue, possibilité de définir l'user responsable de la définit° de celle-ci : CREATE DEFINER = 'user1'.'localhost' PROCEDURE/TRIGGER/VIEW… Pour que la procédure, le trigger ou la view vérifie les droits de celui qui invoque celle-ci plutôt que celui qui la définit : SQL SECURITY INVOKER.
* Retrait de privilèges :

REVOKE INSERT, UPDATE ON dbexercices\_q2.client FROM 'user1'@'localhost';

* Attribution de des privilèges spécifiques INSERT et UPDATE sur la table "client" de la base de données "db\_exercices\_q2" à "user1".

## Transactions

Regroupe +ieurs requêtes en un bloc. Si une des requêtes échoue ou si interrompue => Annulation de tte la transaction.

MySQL valide chaque requête automatiquement. Pour désactiver cela : SET autocommit = 0. Il faut que autocommit soit à 1 pour utiliser les transactions.

START TRANSACTION;

requête1;

requête2;

COMMIT; (Valider) / ROLLBACK; (Annuler)

Remarques:

* Possibilité de placer un point de sauvegarde :

SAVEPOINT nom\_pt\_de\_sauvegarde;

Pour annuler les requêtes jusqu'à ce point de sauvegarde :

ROLLBACK TO nom\_pt\_sauvegarde;

* Si une transaction est en cours et qu'elle modifie des lignes => un verrou est posé sur celles-ci tant qu'elle n'est pas terminée.
* Les transactions doivent respecter le principe ACID (Atomicité, cohérence, isolation et durabilité).

## Verrous

* Verrouiller des tables

LOCK TABLES nom\_table [READ/WRITE] [, nom\_table2];

READ => Lecture dans la table tjrs possible, mais pas d'écriture.

WRITE => Lecture et ecriture impossible pour les autres utilisateurs.

Remarques :

* LOCK libère ts les verrous en cours de l'utilisateur => Il faut tous les placer en une fois s'il y en a +ieurs.
* START TRANSACTION retire aussi ts les verrous de l'utilisateur.
* LOCK TABLES valide une transaction en cours d'écriture (Apres un START TRANSACTION).
* Si l'on veut utiliser LOCK TABLES ds une transaction => SET autocommit = 0;
* Déverrouiller les tables :

UNLOCK TABLES;

Remarques :

* On ne peut déverouiller les tables une à une.