Support de Cours : Algèbre Relationnelle & SQL

Séquence 5 : Grain 1

I- INTRODUCTION

Ce module vise à explorer un pilier essentiel de la sécurité en MySQL : la gestion des comptes utilisateurs. Nous aborderons la création de comptes, l'attribution de privilèges, l'utilisation des rôles, et les bonnes pratiques pour sécuriser votre base de données. L'objectif est de pouvoir protéger les données sensibles contre les accès non autorisés en limitant les droits au strict nécessaire pour réduire les risques.

II- NOMS D'UTILISATEURS ET MOTS DE PASSES

Un utilisateur MySQL est caractérisé par deux identifiants obligatoires son :

- Nom d'utilisateur (ou login) et
- Hôte.

Le nom d'utilisateur correspond à un identifiant librement choisi, tandis que l'hôte désigne précisément l'origine de la connexion.
Par convention, on utilise 'localhost' lorsque la connexion s'effectue depuis la même machine que le serveur MySQL. Pour les accès distants, l'hôte est généralement spécifié sous forme d'adresse IP ou de nom de domaine autorisé.

Par ailleurs, MySQL impose des exigences strictes en matière de sécurité des mots de passe. Ceux-ci doivent obligatoirement être robustes, combinant différents types de caractères : des lettres (majuscules et minuscules), des chiffres et des symboles spécifiques. Cette complexité vise à renforcer la protection des comptes contre les tentatives d'intrusion malveillantes.

Exemples de gestion des utilisateurs en langage SQL :

Créer un utilisateur : (login : Free, Hôte : localhost, Mot de passe : 'P@ssw0rd!2024'

- **CREATE USER 'Free'@'localhost' IDENTIFIED BY 'P@ssw0rd!2024';**

Supprimer un utilisateur :

- **DROP USER 'Free'@'localhost' ;**

Changer le nom du compte de 'Free' à 'FRED'

- **RENAME USER 'Free'@'localhost' TO 'Fred'@'localhost';**

III- LES PRIVILEGES

1. **DEFINITION :** Lorsqu'un utilisateur MySQL est créé via la commande CREATE USER, il ne possède initialement **aucun privilège**. Un privilège correspond à une autorisation précise permettant d'effectuer une action spécifique (comme lire, modifier ou supprimer) sur un élément de la base de données (tables, procédures, bases, etc.). Sans attribution explicite de droits, le compte reste entièrement restreint.

Dans cet état, l'utilisateur peut uniquement **établir une connexion** au serveur MySQL. Toute interaction avec les données ou les structures lui est interdite : il ne peut ni consulter des tables existantes, ni créer de nouveaux objets (bases de données, vues, procédures stockées), ni modifier ou exploiter aucun élément du système. L'octroi ultérieur de privilèges via GRANT est indispensable pour lui accorder des capacités opérationnelles.MySQL stocke l'ensemble des informations relatives privilèges des utilisateurs dans quatre tables spécialisées selon le niveau d'accès :

- **db** (niveau bases de données),
- **tables_priv** (niveau tables),
- **columns_priv** (niveau colonnes) et
- **proc_priv** (niveau procédures stockées).

2. ATTRIBUTION DES PRIVILEGES :

La syntaxe pour ajouter des privilèges à un utilisateur est la suivante :

GRANT PRIVILEGE [(LISTE_COLONNES)] [, PRIVILEGE [(LISTE_COLONNES)], ...]
ON [TYPE_OBJET] NIVEAU_PRIVILEGE TO UTILISATEUR
[IDENTIFIED BY MOT_DE_PASSE]

- **Privilege** est le privilège à accorder à l'utilisateur (comme SELECT, CREATE VIEW, EXECUTE,…) ;
- **Niveau_privilege** est le niveau auquel le privilège s'applique (nom_bdd.nom_table,…) ;
- **(liste_colonnes)** : est facultative et correspond à la liste des colonnes auxquelles le privilège s'applique ;
- **Type_objet** précise à quoi se rapporte le niveau en cas de noms ambigus.

Finalement, on peut accorder plusieurs privilèges en une fois : il suffit de séparer les privilèges par une virgule.

Exemples de gestion des privilèges :

- **GRANT SELECT, INSERT, DELETE ON BD-Test.Matable TO 'Free'@'localhost' ;**
- **GRANT UPDATE (col1, col2) ON BD-Test.Matable TO 'Free'@'localhost'**
- **GRANT CREATE ROUTINE, EXECUTE ON BD-Test.* TO 'Free'@'localhost'**

BD-Test.* : réfère à toutes les tables de la base de données BD-Test

Dans le premier exemple, on accorde à l'utilisateur 'Free' @'localhost' les privilèges SELECT, INSERT et DELETE sur la table MaTable de notre base de données BD-Test, Dans le deuxième exemple, on accorde au même utilisateur le privilège UPDATE sur les colonnes Col1 et Col2 de la même Table. Le troisième exemple accorde à l'utilisateur 'Free' le privilège de créer et d'exécuter des procédures stockées dans la base de données BD-Test,

3. RETRAIT DES PRIVILEGES :

La syntaxe pour retirer des privilèges à un utilisateur est la suivante :

```
REVOKE  PRIVILEGE [, PRIVILEGE, ...]
ON      NIVEAU_PRIVILEGE
FROM    UTILISATEUR;
```

Pour retirer un ou plusieurs priviléges à un utilisateur, on utilise la commande REVEOKE. L'exemple suivant retire pour l'utilisateur « Fred » les droits d'insertion et de mise à jour de la table « Matable » de notre base de données BD-TEST

- **REVOKE UPDATE, INSERT ON BD-Test.Matable FROM 'Fred'@'localhost'**

4. PRIVILEGES PARTICULIERS:

ALL : Le privilège ALL symbolise **l'ensemble des droits disponibles** dans MySQL. Lorsqu'il est accordé à un utilisateur via la commande GRANT, celui-ci obtient la totalité des autorisations d'action sur la base de données. Il est toutefois impératif de **préciser le niveau** (global, sur une base spécifique, une table, etc.) auquel ces droits s'appliquent. Une exception notable existe : le privilège GRANT OPTION **n'est pas inclus** dans le périmètre de ALL.

GRANT OPTION : Ce privilège autorise un utilisateur MySQL à **utiliser la commande** GRANT pour attribuer des droits à d'autres comptes. Sans cette autorisation spécifique, un utilisateur ne peut pas déléguer ses propres privilèges, même s'il dispose d'accès étendus. Toutefois, cette capacité est soumise à une **limitation fondamentale** : l'utilisateur ne peut accorder que les privilèges qu'il **possède lui-même explicitement**. Par exemple, s'il ne détient pas le droit DELETE sur une table, il sera incapable de l'octroyer à un autre utilisateur, même avec GRANT OPTION.

On peut accorder 'GRANT OPTION' de deux manières: Soit directement dans une commande GRANT ou bien en ajoutant WITH GRANT OPTION à la fin de la commande. Les deux exemples suivants sont équivalents et illustrent l'usage de cette commande. Dans ces deux exemples, l'utilisateur 'Free' a le droit de sélectionner, insérer et supprimer des enregistrements de la table « Matable ». Il a également le privilège d'attribuer ces mêmes droits aux autres utilisateurs

- **GRANT SELECT, INSERT, DELETE , GRANT OPTION ON BD-Test.Matable TO 'Free'@'localhost'**
- **GRANT SELECT, INSERT, DELETE ON BD-Test.Matable TO 'Free'@'localhost' WITH GRANT OPTION**

NB. Il faut noter que le privilège ALL doit être utilisé seul. Il n'est pas donc possible de combiner: GRANT ALL, GRANT OPTION….dans la même commande. Dans ce cas, il faut utiliser 'WITH GRANT OPTION'

IV- LES ROLES

Les rôles MySQL sont des **regroupements logiques de privilèges**, conçus pour simplifier l'administration des droits utilisateurs. En encapsulant des ensembles d'autorisations sous un identifiant unique (comme "administrateur" ou "lecture_seule"), ils permettent une gestion plus structurée et cohérente des accès sans avoir à attribuer individuellement chaque privilège. Plusieurs rôles peuvent être créés simultanément grâce à la commande **CREATE ROLE** en séparant leurs noms par des virgules. Par ailleurs, un même rôle est attribuable à plusieurs comptes via GRANT, offrant ainsi une **gestion centralisée** : modifier les privilèges du rôle impacte automatiquement tous les utilisateurs associés, optimisant les mises à jour de sécurité.

1. **On peut créer trois rôles: 'APP_DEVELOPER', 'APP_READ' et 'APP_WRITE**

   - **CREATE ROLE 'APP_DEVELOPER', 'APP_READ', 'APP_WRITE';**

2. **On accorde ensuite des privilèges différents aux trois rôles crées**

   - **GRANT ALL ON BD-Test .* TO 'APP_DEVELOPER';**
   - **GRANT SELECT ON BD-Test .* TO 'APP_READ';**
   - **GRANT INSERT, UPDATE, DELETE ON BD-Test .* TO 'APP_WRITE';**

Exemples : Les exemples suivants illustrent l'attribution des rôles préalablement créés aux différents utilisateurs : le premier assigne le rôle 'APP_DEVELOPER' à l'utilisateur 'Free' ; le second affecte simultanément les rôles 'APP_READ' et 'APP_WRITE' à 'FRED' ; enfin, le dernier applique le rôle 'APP_READ' aux deux utilisateurs 'Read_usr1' et 'Read_usr2', démontrant ainsi une gestion groupée des permissions.

POUR ACCORDER UN ROLE 'APP_DEVELOPER' A UN UTILISATEUR 'Free'@'localhost' :

- **GRANT 'APP_DEVELOPER' TO 'Free'@'localhost';**

POUR ACCORDER LES ROLE 'APP_READ' ET 'APP_WRITE à : 'Fred'@'localhost

- **GRANT 'APP_READ', 'APP_WRITE' TO 'Fred'@'localhost';**

POUR ACCORDER UN ROLE 'APP_READ' AUX UTILISATEURS : Read_usr1, Read_usr2

- **GRANT 'APP_READ' TO 'Read_usr1'@'localhost', 'Read_usr2'@'localhost**

IDEM, les exemples suivants illustrent cette fois-ci le retrait des rôles préalablement attribués aux différents utilisateurs. Dans ce cas, on utilise la commande REVOKE.. FROM

POUR RETIRER UN ROLE 'APP_DEVELOPER' A UN UTILISATEUR 'Free'@'localhost

- **REVOKE 'APP_DEVELOPER' FROM 'Free'@'localhost';**

V- CONCLUSION

Pour conclure : L'accès à MySQL repose sur une authentification par compte utilisateur, chaque compte disposant de privilèges hiérarchisés selon les différents niveaux (base de données, tables, colonnes, etc) définissant ainsi ses droits opérationnels. Le privilège ALL accorde l'ensemble des privilèges – à l'exception du privilège GRANT OPTION – qui autorise un utilisateur à déléguer uniquement les permissions qu'il détient lui-même. Enfin, les rôles peuvent être considérés comme des collections nommées de privilèges. Ils permettent d'optimiser la gestion centralisée des droits pour des groupes d'utilisateurs.
Nous approfondirons ces notions en classe à travers des exercices pratiques et des études de cas.