Authentification Simple et administration des droits

MySQL implémente un système sophistiqué de contrôle d'accès et de privilèges qui vous permet de créer des règles d'accès complètes pour la gestion des opérations client et d'empêcher efficacement les clients non autorisés d'accéder au système de base de données.

Le contrôle d'accès MySQL comporte deux étapes lorsqu'un client se connecte au serveur:

- Vérification de la connexion: un client qui se connecte au serveur de base de données MySQL doit avoir un nom d'utilisateur et un mot de passe valides. De plus, l'hôte à partir duquel le client se connecte doit correspondre à l'hôte dans la table de droits MySQL.
- Vérification de la demande: une fois la connexion établie avec succès, MySQL vérifie, pour chaque instruction émise par le client, si le client dispose des privilèges suffisants pour exécuter cette instruction. MySQL peut vérifier un privilège au niveau de la base de données, de la table et du champ.

Création de comptes d'utilisateurs à l'aide de l'instruction MySQL CREATE USER

MySQL fournit l'instruction CREATE USER qui vous permet de créer un nouveau compte utilisateur.
La syntaxe de l'instruction CREATE USER est la suivante:
CREATE USER ''compte_utilisateur'' IDENTIFIED BY ''mot de passe'';

Le compte utilisateur au format 'nom_utilisateur' @ 'nom_hôte' est suivi de la clause CREATE USER.

Le mot de passe est spécifié dans la clause IDENTIFIED BY. Le mot de passe doit être en texte clair. MySQL chiffrera le mot de passe avant de sauvegarder le compte utilisateur dans la table user.

```
CREATE USER myuser@localhost IDENTIFIED BY 'myuser';
Query OK, 0 rows affected (0,06 sec)
```

Puis la connexion peut s'établir ainsi:

```
./bin/mysql -u myuser -h localhost -p
```

L'utilisateur peut se connecter mais n'a pour l'instant aucun droit

```
show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0,00 sec)
```

Les droits de notre utilisateurs sont ainsi:

```
SHOW GRANTS FOR 'myuser'@'localhost';
+-----------------------------------------+
| Grants for myuser@localhost |
+-----------------------------------------+
| GRANT USAGE ON *.* TO `myuser`@`localhost` |
+-----------------------------------------+
1 row in set (0,00 sec)
```

creation d'un utilisateur avec un plugin d'authentification

Il est possible d'utiliser un plugin d'authentification spécifique ici un plugin sha256

```
CREATE USER 'sha256user'@'localhost'
-> IDENTIFIED WITH sha256_password BY 'password';
```

Augmentation de la sécurité via le plugin de validation de password

Le plugin validate_password sert à tester les mots de passe et à améliorer la sécurité. Le plugin expose un ensemble de variables système qui vous permettent de définir une politique de mot de passe.

```
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';
Query OK, 0 rows affected, 1 warning (0,07 sec)

mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS
-> FROM INFORMATION_SCHEMA.PLUGINS
-> WHERE PLUGIN_NAME LIKE 'validate%';
+-------------------+--------------+
| PLUGIN_NAME | PLUGIN_STATUS |
+-------------------+--------------+
| validate_password | ACTIVE |
+-------------------+--------------+
1 row in set (0,00 sec)
```

On teste maintenant la création d'un mot de passe pauvre

```
create user 'bob'@'%' IDENTIFIED BY 'test';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

Il est possible de charger une strategie de validation:

- La stratégie LOW teste uniquement la longueur du mot de passe. Les mots de passe doivent comporter au moins 8 caractères. Pour changer cette longueur, modifiez validate_password_length.

- La stratégie MEDIUM ajoute les conditions selon lesquelles les mots de passe doivent contenir au moins un caractère numérique, un caractère minuscule, un caractère majuscule et un caractère spécial (non alphanumérique). Pour modifier ces valeurs, modifiez validate_password_number_count, validate_password_mixed_case_count et validate_password_special_char_count.
- La stratégie STRONG ajoute la condition selon laquelle les sous-chaînes de mot de passe de longueur égale ou supérieure à 4 ne doivent pas correspondre aux mots du fichier de dictionnaire, s'il en a été spécifié. Pour spécifier le fichier de dictionnaire, modifiez validate_password_dictionary_file.

Ici on selectionne la stratégie medium par défaut

```
SET GLOBAL validate_password_policy = 1;
Query OK, 0 rows affected (0,00 sec)

mysql> create user 'bob'@'%' identified by 'aA!12345678';
Query OK, 0 rows affected (0,05 sec)
```

Assignation d'un droit a un utilisateur

La commande GRANT permet d'associer un droit à un utilisateur. Nous donnons ici le droit show databases à l'utilisateur myuser

```
mysql> grant show databases on *.* TO 'myuser'@'localhost';
Query OK, 0 rows affected (0,13 sec)

mysql> \q
Bye
```

Nous pouvons maintenant le tester

```
pilou@lubuntu:~/mysql80/mysql-8.0.13-linux-glibc2.12-x86_64$ ./bin/mysql -u myuser -h
localhost -p
mysql> show databases;
+--------------------+
| Database |
```

```
+--------------------+
| information_schema |
| mysql |
| performance_schema |
| sys |
| testUtf8 |
| testdb |
+--------------------+
6 rows in set (0,00 sec)
```

Mise en place de Rôle

En règle générale, vous avez plusieurs utilisateurs avec le même ensemble de privilèges. Le moyen d'octroyer et de révoquer des privilèges à plusieurs utilisateurs de modifier les privilèges de chaque utilisateur individuellement demande beaucoup de temps.

Pour faciliter les choses, MySQL a fourni un nouvel objet appelé role, qui est une collection nommée de privilèges.

Si vous souhaitez accorder le même ensemble de privilèges à plusieurs utilisateurs, procédez comme suit:

- Tout d'abord, créez un nouveau rôle.
- Deuxièmement, accordez des privilèges au rôle.
- Troisièmement, accordez le rôle aux utilisateurs.

Si vous souhaitez modifier les privilèges des utilisateurs, vous devez modifier uniquement les privilèges du rôle attribué. Les modifications prendront effet pour tous les utilisateurs auxquels le rôle a été attribué.

Nous allons créer une base de données exemple:

```
CREATE DATABASE crm;
Query OK, 1 row affected (0,05 sec)

mysql> use crm;
Database changed
mysql> CREATE TABLE customer(
```

```
-> id INT PRIMARY KEY AUTO_INCREMENT,
-> first_name varchar(255) NOT NULL,
-> last_name VARCHAR(255) NOT NULL,
-> phone VARCHAR(15) NOT NULL,
-> email VARCHAR(255)
-> );
Query OK, 0 rows affected (0,11 sec)

mysql> INSERT INTO customer(first_name,last_name,phone,email)
-> VALUES('John','Doe','(408)-987-7654','john.doe@mysql.org'),
-> ('Lily','Bush','(408)-987-7985','lily.bush@mysql.org');
Query OK, 2 rows affected (0,06 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

Puis nous créons 3 rôle permettant respectivement d^'etre un super utilisateur de la base, de pouvoir la lire et de pouvoir l'écrire

```
mysql> CREATE ROLE crm_dev, crm_read, crm_write;
Query OK, 0 rows affected (0,05 sec)

mysql> GRANT ALL ON crm.* TO crm_dev;
Query OK, 0 rows affected (0,07 sec)

mysql> GRANT SELECT ON crm.* TO crm_read;
Query OK, 0 rows affected (0,11 sec)

mysql> GRANT INSERT, UPDATE, DELETE ON crm.* TO crm_write;
Query OK, 0 rows affected (0,05 sec)
```

Puis nous créons des utilisateurs

```
mysql> -- developer user
mysql> CREATE USER crm_dev1@localhost IDENTIFIED BY 'Secure$1782';
Query OK, 0 rows affected (0,03 sec)

mysql> -- read access user
mysql> CREATE USER crm_read1@localhost IDENTIFIED BY 'Secure$5432';
```

```
Query OK, 0 rows affected (0,01 sec)

mysql> -- read/write users
mysql> CREATE USER crm_write1@localhost IDENTIFIED BY 'Secure$9075';
Query OK, 0 rows affected (0,01 sec)

mysql> CREATE USER crm_write2@localhost IDENTIFIED BY 'Secure$3452';
Query OK, 0 rows affected (0,14 sec)
```

Enfin nous associons nos utilisateurs a nos rôle

```
mysql> GRANT crm_dev TO crm_dev1@localhost;
Query OK, 0 rows affected (0,08 sec)

mysql>
mysql> GRANT crm_read TO crm_read1@localhost;
Query OK, 0 rows affected (0,01 sec)

mysql>
mysql> GRANT crm_read, crm_write TO crm_write1@localhost, crm_write2@localhost;
Query OK, 0 rows affected (0,11 sec)
```

Il est possible de voir les droits d'un utilisateur, qui sont en fait les droits de l'utilisateur associé aux roles:

```
SHOW GRANTS FOR crm_dev1@localhost;
+-----------------------------------------------+
| Grants for crm_dev1@localhost |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `crm_dev1`@`localhost` |
| GRANT `crm_dev`@`%` TO `crm_dev1`@`localhost` |
+-----------------------------------------------+
2 rows in set (0,00 sec)
```

Pour voir les droits d'un utilisateur en fonction d'un rôle, il faut le demander explicitement via la clause USING

```
mysql> SHOW GRANTS FOR crm_write1@localhost USING crm_write;
+-----------------------------------------------------------------+
| Grants for crm_write1@localhost |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `crm_write1`@`localhost` |
| GRANT INSERT, UPDATE, DELETE ON `crm`.* TO `crm_write1`@`localhost` |
| GRANT `crm_read`@`%`,`crm_write`@`%` TO `crm_write1`@`localhost` |
+-----------------------------------------------------------------+
3 rows in set (0,00 sec)
```

Les droits sont associé a un rôle. Lors de la connection à MySQL, l'utilisateur doit spécifier le rôle qu'il souhaite utiliser

```
pilou@lubuntu: ~/mysql80/mysql-8.0.13-linux-glibc2.12-x86_64$ ./bin/mysql -u crm_read1 -h
localhost -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.13 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select current_role();
+---------------+
| current_role() |
+---------------+
| NONE |
+---------------+
1 row in set (0,00 sec)

mysql> SET ROLE crm_read;
Query OK, 0 rows affected (0,00 sec)

mysql> select current_role() ;
```

```
+----------------+
| current_role() |
+----------------+
| `crm_read`@`%` |
+----------------+
1 row in set (0,00 sec)

mysql> use crm;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

connexion via PAM

Les plugins d'authentification sont multiples en version entreprise et sont souvent en version GPL avec mariadb.
Il est possible d'authentifier les utilisateurs via PAM en utilisant le systeme unix pour les authentifier en lieu et place de MySQL.
Sous mariadb installer le plugin auth_pam

```
INSTALL SONAME 'auth_pam';
```

Puis créer un utilisateur qui pourras se connecter via pam

```
CREATE USER 'test_pam'@'localhost' IDENTIFIED VIA pam USING 'mariadb';
```

Créer l'utilisateur sous Linux (ici ubuntu)

```
addUser test_pam
```

Dans le repertoire /etc/pam.d on va editer la configuration pam de MySQL en demandant a ce que la vérification des utilisateurs se fasse via les comptes Unix

```
#%PAM-1.0
@include common-auth
@include common-account
@include common-session-noninteractive
```

Revision #1
Created 24 November 2019 13:54:11 by Admin
Updated 24 November 2019 14:06:15 by Admin