MySQL Users and Security

CONTENT

- Introduction
- Getting Information about Users
- Creating a New MySQL User
- Deleting a MySQL User
- Renaming a MySQL User
- User Privileges
- Changing the Password for a MySQL User

Introduction

- We must take precautions for the security of our data and the management of users who have access to that data.
- In larger corporations, you have several different users with different access permissions.
- You don't want to give **root** access to every user, which gives each user complete control of your server.
- **Root** access is restricted to very few people, and then each additional user has its own unique permissions.

Introduction

- We do not want to give all users complete access to our databases.
- We need users to have the necessary permissions to run queries and perform daily tasks.
- With application development, one user is used to connect any number of end users to the database.
- Hundreds of users connect to the database using an application, so you just need one user for this application.

Introduction

- We can have database administrators of different levels.
- We might want to give access to one database to one administrator, and then give access to a second database to a different administrator.
- We might want to make your security more granular and only give certain users access to specific tables.
- The more sensitive our database server becomes, the higher level of security we'll use on our databases.

Introduction

- A mistake in security and user privileges can create huge problems for any business.
- The wrong security gives access to unauthorized users, users who have left the company, or you give access to data that should only be seen by specific personnel.
- When create users and grant privileges, we should be sure that we provide the right security without giving too many privileges that **create risks**.

Introduction

What is at stake?

- **Availability**. When a database or data disappear, business stops. Assuming you have working backups you can restore from, this is the least of all concerns.
- **Confidentiality**. Your secrets, private information of your customers and anything else that you decided not to make publicly available may be leaked.
- **Privacy**. Personal information which, according to certain laws, you may be legally liable for, may be stolen and used for malicious purpose.
- **Integrity**. Without your knowledge, data may be modified to serve someone else's purpose.

Getting Information about Users

The first step securing a database is to find out which users already have access. This information is stored, not surprisingly, in a MySQL database called **`mysql`**.

The **`mysql`** database contains a table called **`user`** which in turn contains a number of columns including the user login name and the users various privileges and connection rights. To obtain a list of users run the following command:

```
SELECT user FROM mysql.user;
```

Creating a New MySQL User

1) MySQL user names can be up to 16 characters long. User names don't need to be as complex as passwords, but they should be difficult for hackers to guess.

2) The MySQL user name is not the same as the operating system account. You can use the same but this isn't recommended.

3) Never use blank passwords for a database server.

4) MySQL uses its own encryption scheme for passwords. (PASSWORD() function)

Creating a New MySQL User

- One difference between MySQL and other database platforms is that the user and host name determine user permissions.

- The host name helps MySQL determine users who have access to specific hosts.The host name can be a fully qualified name, an IP or a wildcard host name that gives access to several databases.

- Database administrators can assign privileges to databases, tables and even columns. This is called column-level security.

Creating a New MySQL User

The creation of a new user account requires the user login name and an optional password. Regardless of the fact that the password is optional, it is unwise to add a new account without a password.

```
CREATE user_name IDENTIFIED BY 'password';
```

For example, to create a new account for a user called `ally` which is protected by a password we can issue the following statement:

```
CREATE USER 'ally'@'localhost' IDENTIFIED BY
'12345678';
```

Creating a New MySQL User

You may have noted that we specified that **ally** could only connect from 'localhost', in other words the same system on which the MySQL server is running. This means that if **ally** tries to connect to the MySQL server from a client running on a remote system, the connection will fail. In order to create an account that can connect from a particular host, simply specify the host name or IP address in place of the localhost in the above example. Alternatively, to allow a user to connect to the MySQL server from any remote host, simply use the '%' character in place of the host name:

```
CREATE USER 'ally'@'%' IDENTIFIED BY '12345678';
```

Deleting a MySQL User

An existing user account may be deleted using the DROP USER statement, the syntax for which is:

```
DROP USER user_name;
```

For example:

```
DROP USER 'ally'@'localhost';
```

Renaming a MySQL User

The account name of a MySQL user can be changed using the RENAME USER statement, the syntax of which is:

```
RENAME USER user_name TO new user_name;
```

For example:

```
RENAME USER 'ally'@'localhost' TO
'arx'@'localhost';
```

User Privileges

A newly created user can log into the MySQL server but by default has no privileges to do anything once connected. The next task after creating a new user account, therefore, is to add privileges to the account. This is achieved using the GRANT statement.

Before modifying a user's privileges it can be helpful to see what privileges are already set. This can be performed using the **SHOW GRANTS** statement in conjunction with the user's account name. For example:

```
mysql> show grants for ally@localhost;
+-----------------------------------------+
| Grants for ally@localhost               |
+-----------------------------------------+
| GRANT USAGE ON *.* TO 'ally'@'localhost' |
+-----------------------------------------+
1 row in set (0,00 sec)
```

User Privileges

- The statement 'USAGE ON *.*' indicates that the user has no privileges on any database or table. Simply put, the user cannot do anything once logged into the database server.
- There are numerous table and column level privileges that you can grant.
- You can also grant users the privileges to create other users or revoke other user privileges.

```
GRANT [type of permission] ON [database name].[table name] TO `[username]'@'localhost';
```

- The **type of permissions** is what you want to grant the user such as SELECT, INSERT, CREATE, UPDATE or DELETE.
- The **database name** is the MySQL database on which the user should have access.
- The **table name** is the table object, but you can also give the user access to all tables using a wildcard.
- The **user name** and host specify the user we set up earlier.

User Privileges

- To add a privilege, for example permission to query any table in a database named MySampleDB, we would issue the following command:

```
GRANT SELECT on MySampleDB.* TO ally@localhost;
```

- Once executed, the above statement will enable user 'ally' to perform SELECT statements on any table contained in the MySampleDB database. Similarly we could enable ally to INSERT rows into a table called product contained in the MySampleDB database as follows:

```
GRANT INSERT on MySampleDB.product TO
  'ally'@'localhost';
```

User Privileges

- It is also perfectly valid to specify multiple privileges in a single GRANT statement, for example:

```
GRANT INSERT, UPDATE on MySampleDB.product TO
  'ally'@'localhost';
```

- The following statement is how you grant all privileges (it is not recommended):

```
GRANT ALL ON *.* TO 'myuser'@'localhost' WITH GRANT
  OPTION;
```

- If we want to grant all privileges to a database:

```
GRANT ALL sampledatabase.* TO 'myuser'@'localhost'
  WITH GRANT OPTION;
```

- This statement also includes the GRANT OPTION statement. This means that the user can grant other users permissions to the database. The user can only pass on his own permissions

User Privileges

- GRANT SELECT ON sampledatabase.* TO 'myuser'@'localhost';
- GRANT SELECT ON sampledatabase.Customer TO 'myuser'@'localhost';
- GRANT SELECT(first_name) ON sampledatabase.Customer TO 'myuser'@'localhost';

  user only has access to one column on the Customer table.

- We need to use the FLUSH PRIVILEGES command. We tell the database to re-read the current user permission table, if not the newly granted permissions will go unnoticed until the next time you reboot the server.

User Privileges

- The REVOKE statement enables to revoke privileges from MySQL accounts. You must have the GRANT OPTION privilege, and you must have the privileges that you are revoking.

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] …
```

Example:

```
REVOKE DELETE ON sampledatabase.* FROM 'myuser'@'localhost';
```

- To revoke all privileges, which drops all global, database, table, column, and routine privileges for the named user or users:

```
REVOKE ALL [PRIVILEGES], GRANT OPTION FROM user [, user] …
```

User Privileges

MySQL supports a wide range of privileges which are outlined in the following table:

Setting	Description
ALL [PRIVILEGES]	Sets all simple privileges except GRANT OPTION
ALTER	Enable the use of ALTER TABLE statement
ALTER ROUTINE	Enable stored routines to be altered or dropped
CREATE	Enable the use of CREATE TABLE statement
CREATE ROUTINE	Enable creation of stored routines
CREATE TEMPORARY TABLES	Enable the use of CREATE TEMPORARY TABLE statement
CREATE USER	Enable the use of CREATE USER, DROP USER, RENAME USER, and REVOKE ALL PRIVILEGES.
CREATE VIEW	Enable the use of CREATE VIEW statement
DELETE	Enable the use of DELETE statement
DROP	Enable the use of DROP TABLE statement
EXECUTE	Enable the user to execute stored routines
FILE	Enable the use of SELECT INTO OUTFILE and LOAD DATA INFILE
INDEX	Enable the use of CREATE INDEX and DROP INDEX statements
INSERT	Enable the use of INSERT

User Privileges

MySQL supports a wide range of privileges which are outlined in the following table:

LOCK TABLES	Enable the use of LOCK TABLES on tables for which the user has the SELECT privilege
PROCESS	Enable the user to see all processes with SHOW PROCESSLIST
REFERENCES	Not implemented
RELOAD	Enable the use of the FLUSH statement
REPLICATION CLIENT	Enable the user to ask for slave or master server locations
REPLICATION SLAVE	Needed for replication slaves (reads binary log events from the master)
SELECT	Enable the use of SELECT
SHOW DATABASES	SHOW DATABASES shows all databases
SHOW VIEW	Enable the use of SHOW CREATE VIEW
SHUTDOWN	Enable the use of mysqladmin shutdown
SUPER	Enable the use of CHANGE MASTER, KILL, PURGE MASTER LOGS, and SET GLOBAL statements, the mysqladmin debug command; allows single connection if max_connections is reached
UPDATE	Enable the use of UPDATE
USAGE	Synonym for no privileges
GRANT OPTION	Enable privileges to be granted

User Privileges

In MySQL, you can limit the following server resources for individual accounts:

- **MAX_QUERIES_PER_HOUR**: The number of queries an account can issue per hour
- **MAX_UPDATES_PER_HOUR**: The number of updates an account can issue per hour
- **MAX_CONNECTIONS_PER_HOUR**: The number of times an account can connect to the server per hour
- **MAX_USERS_CONNECTIONS**: The number of simultaneous connections to the server by an account

- EXAMPLE:

```
ALTER USER 'ally'@'localhost'
WITH MAX_QUERIES_PER_HOUR 20
     MAX_UPDATES_PER_HOUR 10
     MAX_CONNECTIONS_PER_HOUR 5;
```

User Privileges

- The MySQL database stores most of its configurations in system tables. Users are also stored in system tables. For user accounts, you must query the **mysql.user** table to view a list of users and hosts with permissions to your database.
- The following command lets you view all users.

```
select user,host from mysql.user;
```

- If we want to know what access was given to myuser:

```
SHOW GRANTS for 'myuser'@'localhost';
```

- We can also view all privileges for all users by querying the **user_privileges** table. This is also a MySQL system table, and it has several records when you have numerous users.

```
SELECT * FROM information_schema.user_privileges;
```

User Privileges

- MySQL users and privileges are stored in system tables. Main tables are:

USER, DB, TABLES_PRIV, COLUMNS_PRIV

- **user**: User accounts, global privileges, and other non-privilege columns
- **db**: Database-level privileges
- **tables_priv**: Table-level privileges
- **columns_priv**: Column-level privileges

Changing the Password for a MySQL User

```
SET PASSWORD [FOR user] = password_option

password_option: {
    PASSWORD('auth_string')
  | 'auth_string'
}
```

For example:

```
SET PASSWORD FOR 'ally'@'localhost' = '1234';
```

END MySQL
Users and Security