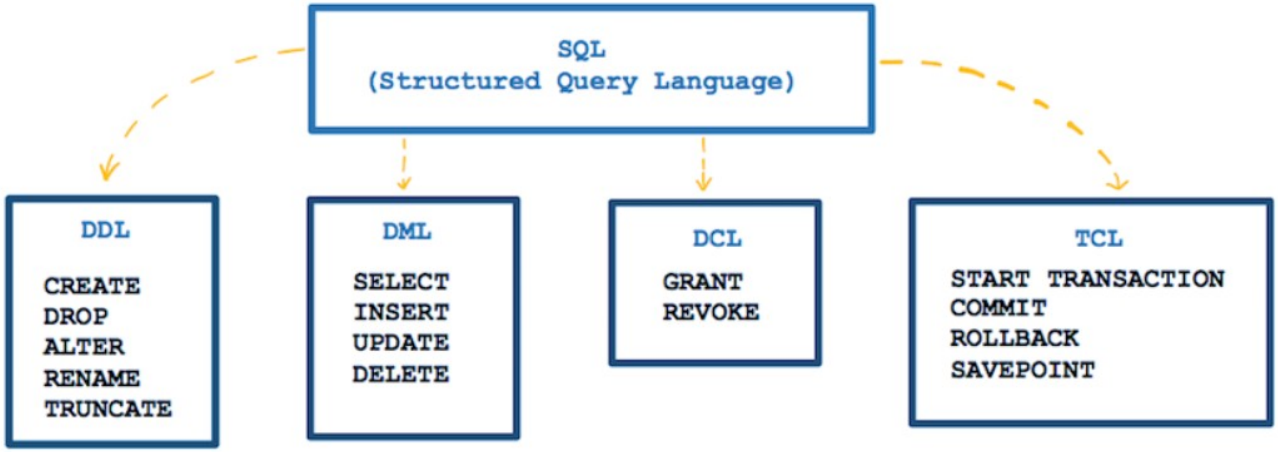
DATABASES

GS WEB APPLICATION DEVELOPMENT

Teacher: Vanesa Maldonado Guerrero
Curs 2025/26

The SQL DCL language



The **Data Control Language (DCL)** is the part of SQL dedicated to controlling access to data in a database. The most commonly used DCL statements are the following:

GRANT: is used to assign permissions to a user or role.

REVOKE: is used to remove permissions that have been assigned to a user or role.

User Management in MySQL Server

Create a new user: The simplified syntax for creating a user in MySQL is as follows:

CREATE USER creates an account and you can add "extras" for:

- login (auth)
- whether to use SSL
- limits
- Expiration
- blocking

```
CREATE USER [IF NOT EXISTS]
    user [auth_option] [, user [auth_option]] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH resource_option [resource_option] ...]
    [password_option | lock_option] ...

user:
    (see Section 6.2.3, “Specifying Account Names”)

auth_option: {
    IDENTIFIED BY 'auth_string'
  | IDENTIFIED WITH auth_plugin
  | IDENTIFIED WITH auth_plugin BY 'auth_string'
  | IDENTIFIED WITH auth_plugin AS 'hash_string'
  | IDENTIFIED BY PASSWORD 'hash_string'
}

tls_option: {
   SSL
 | X509
 | CIPHER 'cipher'
 | ISSUER 'issuer'
 | SUBJECT 'subject'
}

resource_option: {
    MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
}

password_option: {
    PASSWORD EXPIRE
  | PASSWORD EXPIRE DEFAULT
  | PASSWORD EXPIRE NEVER
  | PASSWORD EXPIRE INTERVAL N DAY
}

lock_option: {
    ACCOUNT LOCK
  | ACCOUNT UNLOCK
}
```

User Management in MySQL Server

Create a new user: Its simplest form:

```
CREATE USER username IDENTIFIED BY ' password ' ;
```

Example:

You will need to replace username and password with the details of the new user you wish to create:

```
CREATE USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

Once we have created the user, we need to assign permissions so that they can access the database(s) we want.

User Management in MySQL Server

DROP USER: Deletes the user(s) you specify.
IF EXISTS: Prevents MySQL from returning an error if the user does not exist.

'username'@'localhost': This is the exact account you are deleting:

- **username**: username
- **allowed host**: localhost (only from the local machine)

You can delete multiple users in a single line:

```
DROP USER IF EXISTS 'u1'@'localhost', 'u2'@'%', 'u3'@'172.16.%';
```

User Management in MySQL Server

Delete a user: delete the user and host from the system.
In MySQL, a user is actually: **'username'@'host'**
So, simply using "the name" isn't enough if there are multiple variations.

The syntax for deleting a user in MySQL is as follows:

```
DROP USER [IF EXISTS] nombre_usuario [, nombre_usuario] ...
```

Example:

```
DROP USER IF EXISTS 'nombre_usuario'@'localhost';
```

User Management in MySQL Server

Obtain the user list:

MySQL users are stored in the table **mysql.user**. The primary key of this table consists of the values userand host, so each row will be identified by a username and the host from which it can connect.

The following query returns the list of users we have in MySQL and from which host they can connect:

```
SELECT user,host FROM mysql.user;
```

```
+------------------+--------------+
| user             | host         |
+------------------+--------------+
| root             | localhost    |
| debian-sys-maint | localhost    |
| mysql.session    | localhost    |
| mysql.sys        | localhost    |
+------------------+--------------+
```

The column hostindicates which host the user can connect from. Some of the values we can find in this column are the following:

localhost The user can only connect from the local machine.

%The user can connect from any IP address.

A specific IP address, to indicate that the user can only connect from that IP address.

An IP address with the wildcard % to indicate a range of IP addresses.

User Management in MySQL Server

Example:

Imagine that the table mysql.userhas the following values:

```
+------------------+--------------+
| user             | host         |
+------------------+--------------+
| user1            | localhost    |
| user2            | %            |
| user3            | 172.16.0.11  |
| user4            | 172.16.%     |
+------------------+--------------+
```

- In this example, the user user1 can only connect from localhost.
- The user user2 can connect from any IP address.
- The user user3 can only connect from the IP address 172.16.0.11.
- The user user4 can only connect from IP addresses that begin with 172.16., which would be equivalent to the network 172.16.0.0/16.

Privilege or permission management

Privileges or permissions in MySQL are used to determine what operations a user can perform.

The operations a user can perform include:

- MySQL server administration operations,
- operations on all databases and all objects they contain,
- operations on specific objects in a database.

MySQL makes a distinction between static and dynamic privileges:

- **Static privileges:** These are built into the server.
- **Dynamic privileges:** These are available only when the component that implements them has been enabled.

Types of static permissions that we can apply

The list of static permissions that we can assign to a user in MySQL is very extensive. Below is a list of some of the permissions we will use most frequently:

- **ALL PRIVILEGES** With this option we can assign all privileges at once.
- **CREATE**: allows you to create new tables or databases.
- **DROP**: allows you to delete tables or databases
- **DELETE**: allows you to delete records from tables.
- **INSERT**: allows inserting records into tables.
- **SELECT**: allows reading records in the tables.
- **UPDATE**: allows updating selected records in tables.
- **WITH GRANT OPTION**: allows a user to assign their privileges to other users.

Types of static permissions that we can apply

Assigning permissions to a user

The simplified syntax for assigning permissions to a user in MySQL is as follows:

```
GRANT permiso ON nombre_base_de_datos.nombre_tabla TO 'nombre_usuario'@'localhost';
```

Example 1:

```
GRANT ALL PRIVILEGES ON *.* TO 'nombre_usuario'@'localhost';
```

In this command, the asterisks indicate that we are applying the permission ALL PRIVILEGES to the user nombre_usuariofor all tables in each of the databases.

After this command, the following command must be executed to refresh all user privileges.

```
FLUSH PRIVILEGES;
```

Types of static permissions that we can apply

Example 2:

In this example we are going to create the database prestashopand the user user@localhost, and we are going to assign all permissions on the database.

```
-- Creamos la base de datos prestashop
DROP DATABASE IF EXISTS prestashop;
CREATE DATABASE prestashop CHARACTER SET utf8mb4;
USE prestashop;

-- Creamos un usuario y le asignamos todos los persisos
DROP USER IF EXISTS 'user'@'localhost';
CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON prestashop.* TO 'user'@'localhost';

-- Refrescamos los privilegios si queremos que se apliquen los cambios inmediatamente
FLUSH PRIVILEGES;
```

Types of static permissions that we can apply

Example 3:

In this example we are going to create the database prestashopand the user user@localhost, but this time we are only going to assign the user the permissions of SELECT, INSERT, UPDATE and DELETE.

```
-- Creamos la base de datos prestashop
DROP DATABASE IF EXISTS prestashop;
CREATE DATABASE prestashop CHARACTER SET utf8mb4;
USE prestashop;

-- Creamos un usuario y le asignamos todos los persisos
DROP USER IF EXISTS 'user'@'localhost';
CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT, UPDATE, DELETE ON prestashop.* TO 'user'@'localhost';

-- Refrescamos los privilegios si queremos que se apliquen los cambios inmediatamente
FLUSH PRIVILEGES;
```

Types of static permissions that we can apply

Example 4:

In this example we will use the option WITH GRANT OPTION so that the user can assign their privileges to other users.

```
-- Creamos la base de datos prestashop
DROP DATABASE IF EXISTS prestashop;
CREATE DATABASE prestashop CHARACTER SET utf8mb4;
USE prestashop;

-- Creamos un usuario y le asignamos todos los persisos
DROP USER IF EXISTS 'user'@'localhost';
CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON prestashop.* TO 'user'@'localhost' WITH GRANT OPTION;

-- Refrescamos los privilegios si queremos que se apliquen los cambios inmediatamente
FLUSH PRIVILEGES;
```

Since the user 'user'@'localhost'has the option WITH GRANT OPTION and has all permissions (ALL PRIVILEGES) on all database tables prestashop, they will be able to assign these same permissions to other users on the same database tables prestashop.

Types of static permissions that we can apply

Example 5:

In the following example, the user 'user'@'localhost' will only be able to assign other users the permission on the database SELECT table .customer prestashop

```
GRANT SELECT ON prestashop.customer TO 'user'@'localhost' WITH GRANT OPTION;
```

Types of static permissions that we can apply

Remove permissions from a user:

The simplified syntax for removing permissions from a user in MySQL is as follows:

```
REVOKE permiso ON nombre_base_de_datos.nombre_tabla FROM 'nombre_usuario'@'localhost';
```

Example:

INSERT In this example we are going to remove the permissions UPDATE that DELETE the user has user@localhoston all tables in the database prestashop.

```
REVOKE INSERT, UPDATE, DELETE ON prestashop.* FROM 'user'@'localhost';
```

If we want the changes to take effect immediately, we will need to execute the following statement:

```
FLUSH PRIVILEGES;
```

Types of static permissions that we can apply

How to check a user's permissions:

We can also check what specific permissions a given user has. The following query returns the permissions the user has root:

```
SHOW GRANTS FOR root@localhost;
```

```
+---------------------------------------------------+
| Grants for root@localhost                         |
+---------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' |
+---------------------------------------------------+
```

Role Management

A role is a set of privileges that can be assigned to one or more users.

Create a role
The syntax for creating a role in MySQL is as follows:

```
CREATE ROLE [IF NOT EXISTS] nombre_rol [, nombre_rol ] ...
```

To assign privileges to a role, the following statement is used `GRANT`:

```
GRANT permiso ON nombre_base_de_datos.nombre_tabla TO nombre_rol;
```

To assign a role to a user, the following statement is also used `GRANT`:

```
GRANT nombre_rol TO nombre_usuario;
```

Role Management

Example:

In this example we are going to create three roles: rol_lectura_escritura, rol_lectura and rol_escritura.

```
DROP ROLE IF EXISTS 'rol_lectura_escritura', 'rol_lectura', 'rol_escritura';
CREATE ROLE 'rol_lectura_escritura', 'rol_lectura', 'rol_escritura';
```

Now we will assign the privileges that each role will have and on what database it will have these privileges.

```
GRANT ALL ON base_de_datos.* TO 'rol_lectura_escritura';
GRANT SELECT ON base_de_datos.* TO 'rol_lectura';
GRANT INSERT, UPDATE, DELETE ON base_de_datos.* TO 'rol_escritura';
```

Role Management

Example:

Finally, we're going to create several users and assign them the roles we've created:

```
-- Creamos los usuarios
DROP USER IF EXISTS admin;
CREATE USER admin@'localhost' IDENTIFIED BY 'password1';

DROP USER IF EXISTS usuario_lectura_1;
CREATE USER usuario_lectura_1@'localhost' IDENTIFIED BY 'password2';

DROP USER IF EXISTS usuario_lectura_2;
CREATE USER usuario_lectura_2@'localhost' IDENTIFIED BY 'password3';

DROP USER IF EXISTS usuario_escritura_1;
CREATE USER usuario_escritura_1@'localhost' IDENTIFIED BY 'password4';

DROP USER IF EXISTS usuario_escritura_2;
CREATE USER usuario_escritura_2@'localhost' IDENTIFIED BY 'password5';

-- Asignamos los roles a los usuarios
GRANT 'rol_lectura_escritura' TO admin@'localhost';
GRANT 'rol_lectura' TO usuario_lectura_1@'localhost', usuario_lectura_2@'localhost';
GRANT 'rol_escritura' TO usuario_escritura_1@'localhost', usuario_escritura_2@'localhost';
```

Role Management

Removing a user role:

```
REVOKE nombre_rol FROM nombre_usuario;
```

Delete a role:

The syntax for deleting a role in MySQL is as follows:

```
DROP ROLE [IF NOT EXISTS] nombre_rol [, nombre_rol ] ...
```

Role Management

Obtain a list of existing roles:

Roles are stored in the table mysql.useras if they were users, but with some specific values in the columns account_locked, password_expiredand authentication_string.

- **account_locked**: Indicates if the user account is locked.
- **password_expired**: Indicates if the user's password has expired.
- **authentication_string**: Contains the user's password hash.

To obtain the list of roles in MySQL, we can run the following query:

```
SELECT mysql.user.user
FROM mysql.user
WHERE mysql.user.account_locked='Y' AND mysql.user.password_expired='Y' AND mysql.user.authentication_string='';
```