



Linkia FP

Formación Profesional Oficial a Distancia



ASIX – M10 – Clase 01

UF1: Lenguaje SQL: DCL y extensión procedimental

Lenguaje DCL

Usuarios y permisos - MySQL

CLASE

Bases de datos

Se puede decir que una **base de datos** es una colección de datos relacionados entre sí y que tienen un significado implícito.

Deben cumplir características del tipo: mínima redundancia; seguridad; integridad; respaldo; tiempo de respuesta adecuado, entre otras.

Un sistema gestor de bases de datos es un conjunto de programas que permiten el almacenamiento, modificación y extracción de los datos pertenecientes a una base de datos (BBDD). Algunos de los sistemas comerciales más conocidos son:

- Oracle
- SQL Server
- mongoDB
- PostgreSQL
- MySQL



Instalación MySQL

- Página oficial para descargar e instalar MySQL:
<https://www.mysql.com/downloads/>
- En esta página encontraremos *MySQL Community* de licencia pública

MySQL Community Server 8.0.17

Select Operating System:

Microsoft Windows

Looking for previous GA versions?


Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

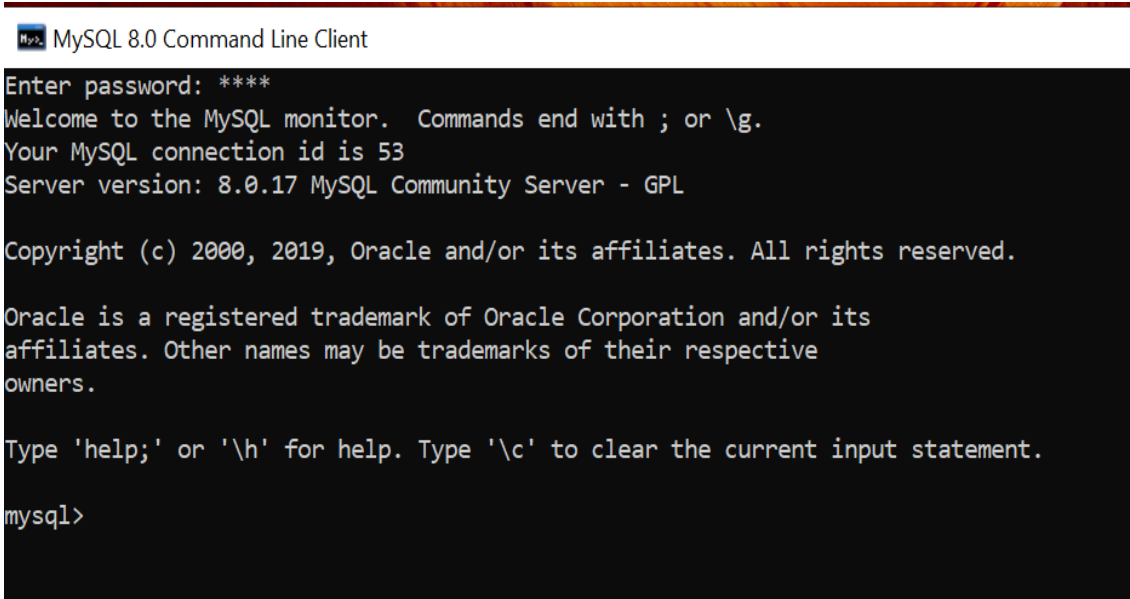


[Go to Download Page >](#)

- **MySQL** es un sistema gestor de base de datos relacional de código abierto que se basa en un lenguaje de consulta estructurado (SQL) y que puede ser ejecutado prácticamente en todas las plataformas
- Existe la posibilidad de instalar el entorno gráfico **MySQL Workbench**
- Una vez instalado MySQL, tenemos dos accesos directos, uno para trabajar desde el entorno gráfico MySQL Workbench y otro para trabajar desde la línea de comandos



Desde la línea de comandos (el terminal) accedemos a MySQL con la contraseña de administrador que se introdujo al realizar la instalación: **root**



```
MySQL 8.0 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 53
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2019, 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>
```

Algunas instrucciones básicas:

`select versión();`

```
mysql> select version();
+-----+
| version() |
+-----+
| 8.0.17    |
+-----+
1 row in set (0.00 sec)

mysql>
```

- Para ver las bbdd: **show databases;**
- Para ver las tablas de la base de datos que tengamos seleccionada: **show tables;**
- Para ver las columnas de una tabla:
describe nombre_de_la_tabla;
- Para salir de MySQL: **exit**

Diccionario de datos

Cuando se instala MySQL uno de los componentes que se crea es el Diccionario de datos. Contiene una serie de metadatos que almacenan información sobre los objetos de la base de datos. En una base de datos relacional el diccionario de datos almacena:

- La estructura lógica y física de la BD.
- La definición de las tablas, vistas, índices, disparadores, procedimientos, etc.
- La cantidad de espacio utilizado por cada objeto de la base de datos.
- Información sobre los usuarios y los privilegios sobre los objetos.

Diccionario de datos

- **Information_Schema.** Las tablas que forman esta base de datos en realidad son vistas y guarda toda la información sobre las bases de datos, tablas, tipos de columnas, privilegios de acceso, etc. Podemos realizar consultas sobre esta base de datos pero nunca cambiar su estructura y datos.

Por ejemplo, podemos consultar los campos que forman la tabla “actor” de la base de datos sakila de la siguiente forma:

```
use information_schema;
SELECT column_name, column_type, column_key, extra FROM columns
WHERE table_schema = 'sakila' and table_name='actor';
```

column_name	column_type	column_key	extra
actor_id	smallint(5) unsigned	PRI	auto_increment
first_name	varchar(45)		
last_name	varchar(45)	MUL	
last_update	timestamp		on update CURRENT_TIMESTAMP

Diccionario de datos

- **MySQL.** También forma parte del diccionario y es donde se almacenan la información referente a los usuarios, las bases de datos y sus permisos (privilegios) de acceso a las mismas.

Por ejemplo, para consultar los usuarios creados en el gestor utilizamos la siguiente instrucción:

```
select Host,User
from mysql.user;
```

Host	User
%	joan
localhost	joan
localhost	mysql.sys
localhost	root
NULL	NULL

Diccionario de datos

- **Sys y performance_schema.** La base de datos performance_schema almacena los datos de rendimiento del gestor. Y la base de datos sys incluye una serie de objetos que permite al administrador de la base de datos analizar el rendimiento de MySQL utilizando los datos que recopila la base de datos performance_schema. Esta información la podemos obtener gráficamente desde la opción Performance Reports del MysqlWorkBench.

Por ejemplo, para consultar la memoria total utilizada por el gestor utilizamos la siguiente instrucción:

```
select * from sys.`memory_global_total`;
```

	total_allocated
▶	136909064

Niveles de Seguridad

Mysql incorpora un sistema de seguridad para especificar que operaciones puede realizar cada usuario con los datos almacenados. Para ello utiliza diferentes niveles. Primero se comprueba el acceso al servidor, después el acceso a la base de datos y finalmente el acceso a la tabla o columna indicada.

- **En el primer nivel** se determina que usuarios tienen permiso de acceso al servidor. La tabla user almacena la lista de usuarios donde podemos destacar los siguientes campos: host (nombre del host), user (nombre de usuario) y authentication_string (contraseña cifrada).

```
SELECT user, authentication_string, host from mysql.user;
```

User	authentication_string	Host
joan	*A4B6157319038724E3560894F7F932C8886EBFCF	%
mysql.infoschema	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMU...	localhost
mysql.session	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMU...	localhost
mysql.sys	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMU...	localhost
root	*A4B6157319038724E3560894F7F932C8886EBFCF	localhost
NULL	NULL	NULL

Niveles de Seguridad

- **En el segundo nivel** se definen los permisos que tienen los usuarios sobre todas las bases de datos del sistema o sobre una base de datos en particular.

Permisos globales: Son los que se aplican a todas las bases de datos y se almacenan en la tabla user de la base de datos mysql. Los usuarios que tengan el valor Y en las columnas select_priv, insert_priv, update_priv y delete_priv pueden realizar operaciones SQL de SELECT, INSERT, UPDATE y DELETE sobre todas las tablas de todas las bases de datos de MySQL.

`desc mysql.user;`

Field	Type	Null	Key	Default	Extra
Host	char(255)	NO	PRI		
User	char(32)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	
Create_priv	enum('N','Y')	NO		N	
Drop_priv	enum('N','Y')	NO		N	
Reload_priv	enum('N','Y')	NO		N	
Shutdown_priv	enum('N','Y')	NO		N	

Niveles de Seguridad

- **En el segundo nivel** se definen los permisos que tienen los usuarios sobre todas las bases de datos del sistema o sobre una base de datos en particular.

Permisos globales: Son los que se aplican a todas las bases de datos y se almacenan en la tabla user de la base de datos mysql. Los usuarios que tengan el valor Y en las columnas select_priv, insert_priv, update_priv y delete_priv pueden realizar operaciones SQL de SELECT, INSERT, UPDATE y DELETE sobre todas las tablas de todas las bases de datos de MySQL.

```
desc mysql.user;
```

Field	Type	Null	Key	Default	Extra
Host	char(255)	NO	PRI		
User	char(32)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	
Create_priv	enum('N','Y')	NO		N	
Drop_priv	enum('N','Y')	NO		N	
Reload_priv	enum('N','Y')	NO		N	
Shutdown_priv	enum('N','Y')	NO		N	

Niveles de Seguridad

Ejemplo:

Consultando esta tabla podemos comprobar que el usuario root y tienen permisos globales, en cambio el usuario alumno no dispone de estos permisos.

```
SELECT user,select_priv,insert_priv,update_priv,delete_priv
from mysql.user;
```

user	select_priv	insert_priv	update_priv	delete_priv
joan	Y	Y	Y	Y
alumno	N	N	N	N
mysql.infoschema	Y	N	N	N
mysql.session	N	N	N	N
mysql.sys	N	N	N	N
root	Y	Y	Y	Y

Niveles de Seguridad

Permisos locales: Son los permisos que se aplican sobre una base de datos en particular y están almacenados en la tabla db de la base de datos mysql. La estructura de la tabla incorpora la columna db para indicar sobre qué base de datos tiene los permisos.

`desc mysql.db;`

Field	Type	Null	Key	Default
Host	char(255)	NO	PRI	
Db	char(64)	NO	PRI	
User	char(32)	NO	PRI	
Select_priv	enum('N','Y')	NO		N
Insert_priv	enum('N','Y')	NO		N
Update_priv	enum('N','Y')	NO		N
Delete_priv	enum('N','Y')	NO		N
Create_priv	enum('N','Y')	NO		N

Consultando esta tabla podemos comprobar a modo de ejemplo que el usuario alumno tiene permisos de select y de insert en la base de datos sakila (se le han otorgado previamente).

`SELECT user,db,select_priv,insert_priv,update_priv,delete_priv
from mysql.db;`

user	db	select_priv	insert_priv	update_priv	delete_priv
mysql.session	performance_schema	Y	N	N	N
alumno	sakila	Y	Y	N	N
mysql.sys	sys	N	N	N	N

Niveles de Seguridad

- **En el tercer nivel** se definen los permisos de los usuarios sobre una determinada tabla de una base de datos o sobre una determinada columna.

Si el permiso es a nivel de tabla lo tendrá sobre todas las columnas de la tabla. Estos permisos se almacenan en la tabla `tables_priv` de la base de datos `mysql`. En esta tabla podemos destacar las siguientes columnas: `Table_Name` (tabla sobre la que se otorga el permiso), `Table_Priv` (permiso que se otorga: `select`, `insert`,...), `Grantor` (usuario que ha otorgado el permiso), `TimeStamp` (no se utiliza), `column_priv` (almacena un permiso a nivel de columna).

En la siguiente captura observamos que el usuario user1 tiene permisos de insert y select sobre la tabla actor de la base de datos sakila y sobre una determinada columna tiene update.

```
select * from mysql.tables_priv;
```

Host	Db	User	Table_name	Grantor	Timestamp	Table_priv	Column_priv
localhost	mysql	mysql.session	user	boot@	0000-00-00 00:00:00	Select	
localhost	sakila	user1	actor	root@localhost	0000-00-00 00:00:00	Select,Insert	Update
localhost	sys	mysql.sys	sys_config	root@localhost	2020-08-17 10:13:42	Select	

Niveles de Seguridad

Para consultar los permisos a nivel de columna utilizamos la tabla `columns_priv` de la base de datos `mysql`. En el siguiente ejemplo observamos que el usuario `user1` tiene el privilegio de `update` sobre la columna `first_name`.

```
select * from mysql.columns_priv;
```

Host	Db	User	Table_name	Column_name	Timestamp	Column_priv
localhost	sakila	user1	actor	first_name	0000-00-00 00:00:00	Update

En la tabla `procs_priv` de la base de datos `mysql` se almacenan los permisos que se aplican a nivel de procedimientos y funciones. Básicamente son tres: **Execute** (para ejecutar), **Alter routine** (para modificar o borrar) y **GRANT** (para poder otorgar los permisos anteriores).

```
desc mysql.procs_priv;
```

Field	Type	Null	Key	Default
Host	char(255)	NO	PRI	
Db	char(64)	NO	PRI	
User	char(32)	NO	PRI	
Routine_name	char(64)	NO	PRI	
Routine_type	enum('FUNCTION','PROCEDURE')	NO	PRI	NULL
Grantor	varchar(288)	NO	MUL	
Proc_priv	set('Execute','Alter Routine','Grant')	NO		
Timestamp	timestamp	NO		CURRENT_TIMESTAMP

Ejemplos.

```
show databases;
```

```
use information_schema;
```

```
SELECT column_name, column_type, column_key, extra FROM columns  
WHERE table_schema= 'sakila' and table_name='actor';
```

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

```
select * from sys.x$memory_global_total;
```

```
-- niveles de seguridad
```

```
-- acceso al servidor
```

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

```
-- permisos globales.
```

```
select user, select_priv, insert_priv, update_priv, delete_priv from mysql.user;
```

```
-- permisos locales a nivel de base de datos.
```

```
select user, db, select_priv, insert_priv, update_priv, delete_priv from mysql.db;
```

```
-- permisos a nivel de tabla o columna.
```

```
select * from mysql.tables_priv;
```

```
select * from mysql.columns_priv;
```

Creación de usuarios en MySQL

Para crear un usuario en MYSQL utilizamos la instrucción CREATE USER. El formato en su opción más simple seria el siguiente:

CREATE USER nombre_usuario IDENTIFIED BY 'password';

Donde nombre_usuario tiene el formato usuario@host.

```
-- ejemplos  
CREATE USER 'user1'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE;  
CREATE USER 'user2'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE INTERVAL 30 DAY;  
CREATE USER 'user3'@'localhost' IDENTIFIED BY '12345678' ACCOUNT LOCK;
```

Creación de usuarios en MySQL

Disponemos de diversas opciones para gestionar la caducidad de la contraseña.

- PASSWORD EXPIRE. Obligamos al usuario a cambiar la contraseña en el próximo inicio de sesión.
- PASSWORD EXPIRE INTERVAL n DAY. Obligamos a cambiar la contraseña cada cierto número de días.
- PASSWORD EXPIRE NEVER. La contraseña nunca caduca.
- PASSWORD EXPIRE DEFAULT. La contraseña caduca en el número de días que indica la variable del sistema **default_password_lifetime** que es de 360 días en las versiones de MYSQL inferiores a la 5.7.11 y no caduca a partir de esta versión.

Con la opción ACCOUNT LOCK o ACCOUNT UNLOCK (por defecto) podemos crear un usuario con la cuenta bloqueada.

```
-- ejemplos
CREATE USER 'user1'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE;
CREATE USER 'user2'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE INTERVAL 30 DAY;
CREATE USER 'user3'@'localhost' IDENTIFIED BY '12345678' ACCOUNT LOCK;
```

Creación de usuarios en MySQL

Cuando creamos usuarios también podemos limitar los recursos con las siguientes opciones:

- MAX_QUERIES_PER_HOUR. Número máximo de consultas por hora.
- MAX_UPDATES_PER_HOUR. Número máximo de actualizaciones por hora.
- MAX_CONNECTIONS_PER_HOUR. Máximo de conexiones por hora.
- MAX_USER_CONNECTIONS. Máximo de conexiones simultáneas que se permiten a cada usuario. Por defecto ilimitadas.

Por ejemplo para crear el usuario user1 con un máximo de 10 conexiones simultáneas y 100 selects por hora utilizaríamos la instrucción:

```
CREATE USER 'user1@localhost' IDENTIFIED BY '12345678'  
WITH MAX_QUERIES_PER_HOUR 100 MAX_USER_CONNECTIONS 10;
```

Modificación de usuarios

La instrucción **ALTER USER IF EXISTS** nos permite modificar las opciones de un determinado usuario, tiene las mismas opciones que el **CREATE USER**. Por ejemplo para bloquear al usuario `user1` utilizaríamos:

```
ALTER USER IF EXISTS user1@localhost ACCOUNT LOCK;
```

Para modificar la contraseña de un usuario utilizamos la sintaxis:

```
ALTER USER IF EXISTS user1@localhost IDENTIFIED BY 'NuevaContraseña';
```

Modificación de usuarios

Para modificar el nombre de un usuario debemos utilizar la instrucción `RENAME USER` indicando el nombre antiguo y el nombre nuevo. Por ejemplo para cambiar el nombre de `user1@localhost` a `usuario1@localhost` utilizamos la orden:

```
RENAME USER user1@localhost TO usuario1@localhost;
```

Para borrar a un determinado usuario se utiliza la orden `DROP USER`. Por ejemplo para borrar al `usuario1@localhost` utilizamos la orden:

```
DROP USER usuario1@localhost;
```



```
/*Crea la BD empresa */
CREATE DATABASE empresa;
/* CREATE DATABASE IF NOT EXISTS empresa; */

USE empresa;

DROP TABLE IF EXISTS departamento;

CREATE TABLE IF NOT EXISTS departamento (
    numdep int(4),
    nombredep varchar(20) not null unique,
    primary key (numdep));

INSERT INTO departamento VALUES ('1','Contabilidad');
INSERT INTO departamento VALUES ('2','Recursos Humanos');
INSERT INTO departamento VALUES ('3','Informática');
INSERT INTO departamento VALUES ('4','Comercial');
INSERT INTO departamento VALUES ('5','Facturación');

/*Tabla trabajadores */

DROP TABLE IF EXISTS trabajadores;
CREATE TABLE IF NOT EXISTS trabajadores (
    dni VARCHAR(9) PRIMARY KEY ,
    nombre VARCHAR(50),
    ciudad VARCHAR(40),
    antigüedad INT(2),
    salario decimal (8,2),
    departamento INT(4),
    foreign key (departamento) REFERENCES departamento(numdep));
```

Ejemplos.

Ejemplos.

```
INSERT INTO trabajadores VALUES ("11112222A","Rojo Iglesias, Marta", "Barcelona",12, 2, 2);
INSERT INTO trabajadores VALUES ("11112233A","Perez Carrillo, Iván","Bilbao",5, 48000, 2);
INSERT INTO trabajadores VALUES ("11112244B","Torres Marqués, Fernando","Madrid",11, 55000, 2);
INSERT INTO trabajadores VALUES ("11112255B","Rubio Sánchez, María","Sevilla",4, 36000, 3);
INSERT INTO trabajadores VALUES ("11112266C","Llamas Rocasolano, Isabel","Barcelona",13, 28000, 3);
INSERT INTO trabajadores VALUES ("11112222C","Gomez Corachán, Manuel","Madrid",15, 22000, 1);
INSERT INTO trabajadores VALUES ("22112222A","Roca Benítez, Elena","Sevilla",6, 35000, 4);
INSERT INTO trabajadores VALUES ("22112222V","Roca Benítez, Marta","Zaragoza",6, 28000, 1);

select * from trabajadores;

-- creamos tres usuarios.
create user 'user1'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE NEVER;
create user 'user2'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE NEVER;
create user 'user3'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE NEVER;
-- modificar un usuario
alter user 'user3'@'localhost' IDENTIFIED BY 'user1234';
```

Gestión de permisos.

Una vez creados los usuarios debemos asignar los permisos para que se puedan conectar al gestor y trabajar con las bases de datos.

Los permisos mas habituales que se pueden asignar a los usuarios son:

- ALL PRIVILEGES: Permite a un usuario de MySQL acceder a todas las bases de datos asignadas en el sistema.
- CREATE: permite crear nuevas tablas o bases de datos.
- DROP: permite eliminar tablas o bases de datos.
- DELETE: permite eliminar registros de tablas.
- INSERT: permite insertar registros en tablas.
- SELECT: permite leer registros en las tablas.
- UPDATE: permite actualizar registros seleccionados en tablas.
- GRANT OPTION: permite remover privilegios de usuarios.
- CREATE USER: Permite gestionar los usuarios.
- EXECUTE: Permite ejecutar procedimientos y funciones.

Orden Grant. Asignar Permisos

La instrucción para asignar permisos a un determinado usuario es la instrucción GRANT que tiene la siguiente sintaxis:

GRANT permiso ON [nombreBaseDatos].[nombredetabla] TO usuario.

Debemos tener en cuenta respecto al ON:

- *.* hace referencia a todas las bases de datos y tablas del sistema.
- Basedatos.* hace referencia a todas las tablas de la base de datos indicada.
- Basedatos.tabla solo hacer referencia a la tabla de la base de datos.

Una vez asignados los privilegios no es necesario ejecutar la orden FLUSH PRIVILEGES para que los cambios tengan efecto ya que el gestor recarga nuevamente los permisos en memoria. Sólo es necesaria esta instrucción si modificamos los valores utilizando un update, delete o insert.

Orden Grant. Ejemplos

En la primera instrucción damos todos los permisos al usuario USER1, en la segunda damos solo el permiso de SELECT sobre todas las bases de datos de MYSQL al usuario user2.

```
GRANT ALL ON *.* TO user1@localhost;  
GRANT SELECT ON *.* TO user2@localhost;
```

Por ejemplo para que el usuario user3 pueda hacer un INSERT y un SELECT en todas las tablas de la base de datos sakila utilizamos la siguiente instrucción.

```
GRANT SELECT,INSERT ON sakila.* TO user3@localhost;
```

El user1 solo puede hacer select en la tabla film de la base de datos sakila y el usuario user2 solo puede hacer select en los campos title y description.

```
GRANT SELECT ON sakila.film TO user1@localhost;  
GRANT SELECT (title,description) ON sakila.film to user2@localhost;
```

Mostrar privilegios.

Para ver los privilegios que tiene un determinado usuario puede utilizar la instrucción **SHOW GRANTS;**

```
mysql> show grants;
+-----+
| Grants for user2@localhost |
+-----+
| GRANT USAGE ON *.* TO `user2`@`localhost` |
| GRANT SELECT (`description`, `title`) ON `sakila`.`film` TO `user2`@`localhost` |
+-----+
2 rows in set (0.00 sec)
```

Debemos tener en cuenta las siguientes consideraciones para que los cambios en el sistema de permisos tengan efecto:

- Los cambios a nivel global tienen efecto cuando el usuario vuelve a conectarse.
- Los cambios a nivel de base de datos cuando el usuario vuelve a seleccionarla.
- Los cambios a nivel de tabla y columna tienen un efecto inmediato.

Orden REVOKE. Quitar privilegios

La instrucción para quitar permisos a un determinado usuario es la instrucción REVOKE que tiene la siguiente sintaxis:

REVOKE permiso ON [nombreBaseDatos].[nombredetabla] FROM usuario.

Debemos tener en cuenta respecto al ON:

- *.* hace referencia a todas las bases de datos y tablas del sistema.
- Basedatos.* hace referencia a todas las tablas de la base de datos indicada.
- Basedatos.tabla solo hacer referencia a la tabla de la base de datos.

Orden REVOKE. Ejemplos

- Quitar el permiso de borrar en la tabla film de la base de datos sakila al usuario user1.

```
REVOKE DELETE ON sakila.film FROM user1@localhost;
```

- Quitar el permiso de hacer select en la columna description de la tabla film de la base de datos sakila al usuario user3.

```
REVOKE SELECT (description) ON sakila.film FROM user2@localhost;
```

- Quitar todos los permisos al usuario user3.

```
REVOKE ALL ON *.* FROM user3@localhost;
```


Ejemplos

```
-- el usuario1 puede acceder a todas las bases de datos.  
grant all on *.* to user1@localhost;  
flush privileges;  
revoke all on *.* from user1@localhost;  
flush privileges;  
  
grant all on empresa.* to user1@localhost;  
flush privileges;  
  
grant select, insert on empresa.* to user2@localhost;  
flush privileges;  
revoke select, insert on empresa.* from user2@localhost;  
grant select(dni, nombre) on empresa.trabajadores to user2@localhost;  
flush privileges;
```

Opción WITH GRANT OPTION

Si otorgamos permisos a usuarios utilizando la opción WITH GRANT OPTION, habilitamos a ese usuario para que pueda otorgar ese mismo permiso a otros usuarios.

Por ejemplo si queremos otorgar el permiso de SELECT, INSERT al usuario user1 sobre la base de datos sakila y que este usuario lo pueda otorgar a otros usuarios debemos utilizar la siguiente instrucción.

```
GRANT SELECT,INSERT ON sakila.* TO user1@localhost WITH GRANT OPTION;
```

Ahora podemos comprobar como el user1 puede asignar este permiso a otros usuarios. En el siguiente ejemplo el user1 asigna el permiso de lectura sobre la tabla sakila.actor al usuario user2.

```
mysql> show grants;
+-----+
| Grants for user1@localhost |
+-----+
| GRANT USAGE ON *.* TO `user1`@`localhost` |
| GRANT SELECT, INSERT ON `sakila`.* TO `user1`@`localhost` WITH GRANT OPTION |
| GRANT SELECT ON `sakila`.`film` TO `user1`@`localhost` |
+-----+
3 rows in set (0.00 sec)

mysql> GRANT SELECT ON sakila.actor TO user2@localhost;
Query OK, 0 rows affected (0.02 sec)
```

Ejemplo WITH GRAN OPTION

```
-- asignar privilegios a otros usuarios
grant select, insert on empresa.trabajadores to user1@localhost with grant option;
flush privileges

-- usuario 1
show grants;
grant select on empresa.trabajadores to user3@localhost;
flush privileges;

-- usuario 3
show grants;
use empresa;
select * from trabajadores;
select * from departamento;
```

Gestión de roles

Los roles son un conjunto de privilegios que se pueden otorgar a un usuario o a otro Rol. De esta manera se simplifica el trabajo de DBA. Los privilegios del rol pueden ser de nivel global, de base de datos o de tabla y columnas. Podemos asignar diferentes roles a un mismo usuario, pero solo puede tener activado uno solo por defecto.

En MySQL es posible crear roles a partir de la versión 8. Para crear un rol utilizamos la instrucción.

CREATE ROLE [IF NOT EXISTS] nombrerol;

Ejemplos:

- Crear un rol llamado consultaSakila

```
CREATE ROLE IF NOT EXISTS consultaSakila;
```

Gestión de roles

Para asignar y retirar privilegios a los roles utilizamos la misma orden GRANT vista anteriormente. A los roles se les asignan privilegios igual que a los usuarios. Para eliminar los privilegios de un rol utilizamos REVOKE.

En el siguiente ejemplo asignamos al rol anterior, el select de todas las tablas de la base de datos Sakila, el insert en la tabla film y el Update y delete en la tabla actor.

```
GRANT SELECT ON sakila.* TO consultaSakila;  
GRANT INSERT ON sakila.film TO consultaSakila;  
GRANT UPDATE,DELETE ON sakila.actor TO consultaSakila;
```

El siguiente paso es asignar el rol a los usuarios con la instrucción GRANT, en el ejemplo anterior.

```
GRANT consultaSakila TO user1@localhost,user2@localhost;
```

Un usuario puede tener diferentes roles asignados. En el último paso seleccionamos cual es el rol por defecto que va a utilizar con la instrucción SET DEFAULT ROLE.

```
SET DEFAULT ROLE consultaSakila TO user1@localhost,user2@localhost;
```

Ejemplos. Gestión de roles

```
-- Ejemplo: trabajo con roles.  
create user 'user4'@'localhost' IDENTIFIED BY '12345678' PASSWORD EXPIRE NEVER;  
create role roleempresa;  
grant select on empresa.* to roleempresa;  
grant insert,update on empresa.departamento to roleempresa;  
grant update on empresa.trabajadores to roleempresa;  
grant roleempresa to user4@localhost;  
set default role roleempresa to user4@localhost;  
flush privileges;  
  
-- comprobaciones usuario4  
show grants;  
use empresa;  
select * from departamento;  
delete from departamento where numdep=6;
```

Pregunta de la 1a clase virtual.

Que permisos se definen
en el tercer nivel de
Seguridad de MySQL





Linkia FP

Formación Profesional Oficial a Distancia

