



TEMA

Tema 1. Lenguaje DCL: Usuarios y permisos en MYSQL

Administración de sistemas  
informáticos en red

**Administración de sistemas gestores  
de bases de datos**

Autor: Joan Pou



# Tema 1: Lenguaje DCL: Usuarios y permisos en MYSQL

## ¿Qué aprenderás?

- Crear y gestionar usuarios.
- Asignar permisos y privilegios.
- Aplicar la composición del diccionario de datos.
- Crear roles.

## ¿Sabías que...?

- MySQL es el gestor de base de datos más utilizado para el desarrollo de aplicaciones web.
- MySQL fue comprado por Oracle en el año 2010.
- MySQL es utilizado por grandes empresas como Google, Wikipedia, Twitter, Facebook.



## 1. Lenguaje DCL

El lenguaje de control de datos (Data Control Language) permite al administrador de la base de datos controlar el acceso a los datos almacenados en la base de datos. Para controlar la seguridad el administrador de la base de datos deberá crear usuarios permitiendo o denegando privilegios para que puedan acceder a los diferentes objetos de la base de datos.

### 1.1. Introducción

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.

Podemos destacar las siguientes bases 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

- **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

- **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.`x$memory_global_total`;
```

total_allocated
136909064

## 1.2. Niveles de seguridad en MySQL

MySQL incorpora un sistema de seguridad para especificar qué 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\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATM...	localhost
mysql.session	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATM...	localhost
mysql.sys	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATM...	localhost
root	*A4B6157319038724E3560894F7F932C8886EBFCF	localhost
NULL	NULL	NULL



- **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	

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

**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

- **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	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



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

### 1.3. Gestión de usuarios

#### 1.3.1. Creación de usuarios

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

**`CREATE USER nombre_usuario IDENTIFIED BY 'password';`**

Donde `nombre_usuario` tiene el formato `usuario@host`.

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;
```

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 simultaneas que se permiten a cada usuario. Por defecto ilimitadas. Debemos tener en cuenta que si utilizamos MySQLWorkBench o PhpMyAdmin, para cada usuario se crean dos conexiones.

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;
```

Para iniciar la sesión de con un nuevo usuario desde la línea de comandos utilizamos la instrucción **mysql** con el parámetro **-u** el nombre del usuario y el parámetro **-p** para indicar que se debe poner una contraseña. Por ejemplo:

***Mysql -u user1 -p***

### 1.3.2. Modificación y borrado 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';
```

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;
```



## 1.4. 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:

Permiso	Función
ALL [PRIVILEGES]	Otorgamos todos los privilegios a nivel global, de base de datos o de tabla.
CREATE	Permite crear nuevas tablas o bases de datos.
DROP	Permite borrar tablas y bases de datos.
DELETE	Permite eliminar registros en las tablas.
INSERT	Permite insertar registros en las tablas.
SELECT	Permite la lectura registros en las tablas.
UPDATE	Permite la actualización de registros en las tablas.
GRANT OPTION	Permite quitar privilegios de usuarios.
CREATE USER	Permite crear, modificar, borrar, renombrar y quitar los permisos de los usuarios en el gestor.
ALTER	Modifica una tabla es necesario tener el privilegio CREATE.
LOCK TABLES	Permite bloquear una tabla.
USAGE	Se asigna al crear un usuario y que permite conectarse a MySQL.
EXECUTE	Ejecutar procedimientos y funciones.



### 1.4.1. Asignación de permisos. Orden GRANT

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. 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.

Los permisos son globales cuando indicamos en el parámetro ON \*.\* Por ejemplo 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;
```

Para otorgar permisos a nivel de base de datos hemos de especificar el nombre de ésta en la cláusula ON del GRANT. 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;
```



Para otorgar permisos a nivel de tabla o de columna, debemos indicar en el parámetro ON el nombre de la base de datos y de la tabla sobre la que afecta el permiso. Si solo se aplica sobre una columna deberemos indicar el nombre de la columna.

Ejemplo 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;
```

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)
```

#### 1.4.2. Quitar permisos. Orden REVOKE

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.

Los permisos que se pueden quitar son los mismos que se utilizan con la orden GRANT. Pero para poder quitar un permiso debemos tener en cuenta:

- Debemos poseer ese permiso.
- Lo hemos obtenido con la opción "with gran option".
- El usuario tiene el permiso CREATE USER.



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;
```

#### 1.4.3. 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)
```



## 1.5. 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;
```

Para asignar y retirar privilegios a los roles utilizamos la misma orden GRANT vista anteriormente. A los roles se les asignar 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;
```



Video Usuarios MySQL





## Recursos y enlaces

- [Crear un usuario en MySQL.](#)



- [Permisos en MySQL.](#)



- [Descargar MySQL.](#)



## Conceptos clave

- **Diccionario de datos:** Conjunto de metadatos que guarda la información sobre los objetos de la base de datos.
- **GRANT:** Instrucción para asignar privilegios a un usuario.
- **REVOKE:** Instrucción para quitar privilegios a un usuario.
- **Rol:** Conjunto de privilegios que se puede otorgar a un determinado usuario.



## Test de autoevaluación

---

¿En qué base de datos se guarda la información relativa a los usuarios?

- a) SYS.
- b) MYSQL.
- c) Performance\_schema.
- d) Information\_schema.

Para ver los permisos de un determinado usuario utilizamos:

- a) SELECT privileges FROM mysql.user;
- b) SHOW PRIVILEGES;
- c) SHOW GRANTS.
- d) LIST PRIVILEGES.

El privilegio USAGE permite:

- a) Conectar al gestor de MySQL
- b) Hacer un SELECT en todas las bases de datos;
- c) Hacer SELECT, INSERT, DELETE, UPDATE en todas las bases de datos.
- d) Crear usuarios.

## Ponlo en práctica

---

### Actividad 1

---

Crea una base de datos llamada actividad 1 con una tabla de alumnos. Inserta 3 registros. Crea dos usuarios gestor con todos los privilegios de lectura, insert, modificar y borrar registros y lector con los privilegios de solo lectura.



## SOLUCIONARIOS

### Test de autoevaluación

---

¿En qué base de datos se guarda la información relativa a los usuarios?

- e) SYS.
- f) **MYSQL.**
- g) Performance\_schema.
- h) Information\_schema.

Para ver los permisos de un determinado usuario utilizamos:

- e) SELECT privileges FROM mysql.user;
- f) SHOW PRIVILEGES;
- g) **SHOW GRANTS.**
- h) LIST PRIVILEGES.

El privilegio USAGE permite:

- e) **Conectar al gestor de MySQL**
- f) Hacer un SELECT en todas las bases de datos;
- g) Hacer SELECT, INSERT, DELETE, UPDATE en todas las bases de datos.
- h) Crear usuarios.



## Ponlo en práctica

### Actividad 1

Crea una base datos llamada actividad 1 con una tabla de alumnos. Inserta 3 registros. Crea dos usuarios gestor con todos los privilegios de lectura, insert, modificar y borrar registros y lector con los privilegios de solo lectura.

```
CREATE DATABASE actividad1;
USE actividad1;
CREATE TABLE alumnos (
    numero mediumint NOT NULL PRIMARY KEY,
    nombre varchar(30) NOT NULL,
    curso varchar(5) NOT NULL
) ENGINE=InnoDB;

INSERT INTO alumnos VALUES (1, 'Marta Roca', 'DAM'),
(2, 'Jose Ruiz', 'ASIX'), (3, 'Maria Comas', 'DAW');
CREATE USER gestor@localhost identified by 'gestor1234';
CREATE USER lector@localhost identified by 'lector1234';
GRANT SELECT, INSERT,DELETE, UPDATE ON actividad1.* TO gestor@localhost;
GRANT SELECT ON actividad1.* TO lector@localhost;
```