TEMA 10: CREACIÓN Y MANIPULACIÓN DE OTROS OBJETOS DE LA BASE DE DATOS

1. Índices.

Los índices son estructuras de datos que aceleran las operaciones de búsqueda basadas en los campos sobre los que se definen los índices.

Además de crear índices en el momento en que se crea una tabla (dentro de la instrucción CREATE TABLE) escribiendo *index (atr1, atr2, …)*, también es posible hacerlo con posterioridad mediante el comando CREATE INDEX. MySQL crea índices automáticamente para los campos únicos, clave primaria y claves ajenas.

Los índices no tienen por qué abarcar la totalidad de un atributo, sino que pueden ser parciales, lo que se lleva a cabo básicamente sobre campos largos de tipo texto.

Además, los índices pueden ser multicolumna, es decir, pueden abarcar más de un atributo de una tabla, lo que puede resultar de utilidad cuando se usan esas diversas columnas en cláusulas WHERE de sentencias SELECT.

Podemos hablar de los siguientes tipos de índices:

- *Unique*: Un índice único es aquel formado por atributos cuyo valor no se repite en la tabla.
- *Full-text*: Un índice de este tipo está formado por uno o varios campos de texto utilizados para la búsqueda de palabras dentro de un campo. Solo sirve para tablas de tipo MyISAM y para campos de tipo *char*, *varchar* y *text*.
- *Spatial*: Se trata de índices que se usan para campos de tipo espacial como *line* y *curve*.

La instrucción para crear un índice es CREATE INDEX, que requiere de la siguiente sintaxis:

```
CREATE [{UNIQUE | FULLTEXT | SPATIAL}] INDEX nombre_índice
ON nombre_tabla (atributo_índice, .…)

atributo_índice: nombre_atributo [(longitud)] [{ASC | DESC}]
```

Como se puede observar, se puede indicar después de la palabra CREATE opcionalmente si se trata de un índice único, *full-text* o *spatial*. Después de la palabra INDEX se debe indicar el nombre que se le asigna al índice. Hay que indicar después de la palabra ON la tabla sobre la que se crea el índice y el o los atributos correspondientes. Los índices no tienen por qué ser completos, sino que pueden ser parciales, indicando en tal caso la longitud que abarcan entre paréntesis. Además, se puede indicar si se desea que el índice sea en orden ascendente o descendente. El orden por defecto es ascendente.

Por ejemplo, vamos a crear un índice parcial sobre los 10 primeros caracteres de la descripción de un artículo:

```
mysql> create index IDesArt
    -> on Articulo(DesArt(10));
Query OK, 0 rows affected (0.51 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Creemos ahora otro índice sobre el atributo *PVPArt* en orden descendente. Usaremos la siguiente orden:

```
mysql> create index IPVPArt
    -> on Articulo (PVPArt DESC);
Query OK, 0 rows affected (0.19 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Para visualizar los índices definidos sobre una tabla se puede usar el comando SHOW INDEX, cuya sintaxis es la siguiente:

```
SHOW {INDEX | INDEXES | KEYS}
{FROM | IN} nombre_tabla
[{FROM | IN} nombre_BD]
```

Como se puede observar, se puede indicar después de SHOW tanto INDEX, como INDEXES, como KEYS. Se debe indicar después FROM o IN y la tabla cuyos índices se desean mostrar, y si no es una tabla de la base de datos actual, FROM o IN y la base de datos en cuestión. Así, para visualizar los índices que hay definidos sobre la tabla *Articulo* de la base de datos actual, escribiremos:

```
mysql> show indexes from Articulo;
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| articulo |          0 | PRIMARY  |            1 | CodArt      | A         |           5 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| articulo |          1 | IDesArt  |            1 | DesArt      | A         |           4 |       10 |   NULL |      | BTREE      |         |               | YES     | NULL       |
| articulo |          1 | IPVPArt  |            1 | PVPArt      | D         |           5 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.00 sec)
```

Se puede observar cómo además de los dos índices que se acaban de crear explícitamente con la instrucción CREATE INDEX (*IDesArt* e *IPVPArt*), MySQL creó implícitamente un índice para el atributo clave primaria de la tabla (*CodArt*).

No existe una instrucción ALTER INDEX para realizar modificaciones sobre los índices creados.

Por su parte, la instrucción para eliminar un índice es la instrucción DROP INDEX, cuya sintaxis es la siguiente:

```
DROP INDEX nombre_índice ON nombre_tabla
```

Como se puede observar, se debe indicar el nombre del índice y el nombre de la tabla sobre la que se ha definido. Así, para borrar el índice *IPVPArt* definido sobre el atributo *PVPArt* de la tabla *Articulo*, deberemos emplear la siguiente instrucción:

```
drop index IPVPArt on Articulo;
```

2. Vistas.

Las vistas son objetos de la base de datos que incluyen mediante una consulta un subconjunto de datos de la base de datos. A veces se llama a las vistas "tablas virtuales" porque se puede trabajar con ellas en la mayoría de los casos como si fuesen tablas, pero no lo son porque las vistas no contienen datos almacenados. De hecho, cada vez que se realiza una operación sobre una vista, se ejecuta la sentencia SELECT asociada a la vista para obtener los datos de la misma.

Las vistas permiten que el administrador de la base de datos solo ponga a disposición de determinados usuarios aquellos datos a los que estos deben poder acceder creando las vistas correspondientes. Para los usuarios ver los datos contenidos en tablas o en vistas es exactamente igual, pero no para el administrador, quien debe velar por la integridad de los datos. Además, es posible dar permisos sobre vistas como si se tratase de tablas, permitiendo de esta manera proteger las tablas originales.

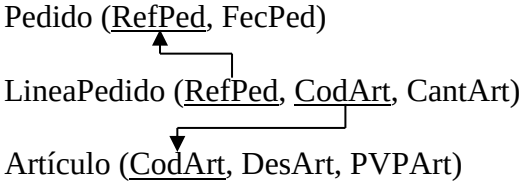
Para crear una vista se debe emplear la sentencia CREATE VIEW, cuyo formato es el siguiente:

```
CREATE [OR REPLACE]
VIEW Nombre_Vista [(campo1, campo2, …, campon)]
AS sentencia_select
```

La opción OR REPLACE sirve para que en el caso de que ya exista una vista con el nombre indicado, no se produzca un mensaje de error y esa vista sea sustituida por la que se está creando.

A toda vista es necesario asignarle un nombre, que no puede coincidir con el nombre de ninguna otra vista ni tabla dentro de la base de datos actual. Después de indicar el nombre de la vista se pueden especificar entre paréntesis y separados por comas los nombres de sus atributos. En caso de omitir esta lista, se asignan a los atributos de la vista los mismos nombres que aparecen en la sentencia SELECT con la que se crea la vista.

Vamos a trabajar con vistas a partir de algunas tablas de la base de datos *Pedidos*, cuyo esquema relacional se muestra a continuación:



A modo de ejemplo, se va a crear una vista llamada *ArticulosBaratos* con la descripción y precio de los artículos cuyo precio es inferior a 0,50 €. En este caso no se van a especificar los nombres de los atributos de la vista, por lo que coincidirán con los de la tabla *Articulo* sobre la que se define:

```
mysql> create view ArticulosBaratos
    -> as select DesArt, PVPArt from Articulo where PVPArt<0.50;
Query OK, 0 rows affected (0.07 sec)

mysql> select * from ArticulosBaratos;
+----------------+--------+
| DesArt         | PVPArt |
+----------------+--------+
| Goma de borrar |   0.15 |
| Sacapuntas     |   0.25 |
+----------------+--------+
2 rows in set (0.00 sec)
```

En la sentencia SELECT asociada a la vista pueden aparecer varias tablas y/o vistas en su cláusula FROM.

Si a algún atributo de la vista se le desea asignar un nombre diferente de lo que viene especificado en la cláusula SELECT de la consulta con la que se crea la vista, se puede obrar de una de las dos siguientes maneras:

- Asignar un alias al atributo en la sentencia SELECT.
- Escribir los nombres de todos los atributos de la vista entre paréntesis y entre comas después del nombre de la vista.

A modo de ejemplo, vamos a crear una vista que contenga por cada artículo solicitado en más de un pedido, su código, descripción, el número de pedidos en que ha sido solicitado y el número de unidades totales solicitadas del artículo. Para obtener estos dos últimos datos tenemos que realizar agrupamientos y aplicar funciones de grupo (count y sum) a dos atributos. La podemos crear con la siguiente instrucción:

```
mysql> create view ArticulosPedidos as
    -> select A.CodArt, DesArt, count(RefPed), sum(CantArt)
    -> from LineaPedido L join Articulo A on L.CodArt = A.CodArt
    -> group by A.CodArt, DesArt
    -> having count(RefPed) > 1;
Query OK, 0 rows affected (0.04 sec)

mysql> select * from ArticulosPedidos;
+--------+----------------+---------------+--------------+
| CodArt | DesArt         | count(RefPed) | sum(CantArt) |
+--------+----------------+---------------+--------------+
| A0043  | Bolígrafo azul |             3 |           20 |
+--------+----------------+---------------+--------------+
1 row in set (0.00 sec)
```

Pero de esta manera hay dos atributos en la vista (los dos últimos), cuyos nombres pueden no resultar muy adecuados. Para asignarles otros nombres podríamos haber ejecutado una de las dos siguientes opciones:

```
create view ArticulosPedidos (CodArt, DesArt, NumPed, Unidades)as
select A.CodArt, DesArt, count(RefPed), sum(CantArt)
from LineaPedido L join Articulo A on L.CodArt = A.CodArt
group by A.CodArt, DesArt
having count(RefPed) > 1;

create view ArticulosPedidos as
select A.CodArt, DesArt, count(RefPed) NumPed, sum(CantArt) Unidades
from LineaPedido L join Articulo A on L.CodArt = A.CodArt
group by A.CodArt, DesArt
having count(RefPed) > 1;
```

Sobre las vistas no solo se pueden realizar consultas, sino también inserciones, borrados y modificaciones, que afectarán a los datos reales almacenados en la/s tabla/s sobre la/s que se ha definido la vista. No obstante, para poder llevar a cabo este tipo de operaciones se deben cumplir ciertas condiciones.

Para poder realizar inserciones, borrados y modificaciones sobre una vista, se deben cumplir las siguientes condiciones:

- Debe haber una relación uno a uno entre los registros de la vista y los registros de la tabla subyacente.
- No debe haber GROUP BY ni DISTINCT en la consulta asociada a la vista.
- No debe haber uniones ni reuniones externas en la consulta asociada a la vista.

Para poder realizar modificaciones, además de las condiciones anteriores, se debe cumplir que ninguna de las columnas que se va a actualizar sea una columna derivada, es decir, definida como una expresión en la vista.

Para poder realizar inserciones en una vista, además de las condiciones enumeradas dos párrafos antes, se deben cumplir todas las condiciones indicadas a continuación:

- La vista debe contener todos los atributos obligatorios de la tabla que no tengan definido en la tabla valor por defecto.
- Las columnas de la vista deben ser referencias a columnas simples y no a columnas derivadas, entendiendo que una columna derivada es aquella a la que nos referimos mediante una expresión.

A modo de ejemplo, se va a crear a partir de la tabla *Emple* de la base de datos *Empresa*, una vista con el apellido, oficio, salario y comisión de los empleados del departamento número 20.

```
mysql> create view emp20
    -> as select apellido, oficio, salario, comision
    -> from emple where dept_no=20;
Query OK, 0 rows affected (0.06 sec)
```

Vamos a realizar una inserción sobre esta vista. Obviamente el SGBD intentará realizar la inserción sobre la tabla *Emple*, que es la tabla sobre la que se ha creado la vista.

```
mysql> insert into emp20
    -> values ('GARCÍA', 'EMPLEADO', 1200, 100);
ERROR 1423 (HY000): Field of view 'empresa.emp20' underlying table doesn't
have a default value
```

No se ha podido llevar a cabo la inserción porque hay dos atributos de la tabla *Emple* (*emp_no* y *fecha_alt*) que tienen asignada la restricción not null y no tienen asignado valor por defecto.

Ahora vamos a ver si es posible eliminar al empleado apellidado SÁNCHEZ de la tabla *Emple* a través de la vista. Se debería poder por no incumplirse ninguna de las condiciones especificadas anteriormente para poder efectuar borrados.

```
mysql> delete from emp20
    -> where apellido='SÁNCHEZ';
Query OK, 1 row affected (0.10 sec)
```

Y podemos ver como consultando la tabla *Emple* el empleado SÁNCHEZ ha sido en efecto eliminado.

Vamos a crear ahora una vista llamada *SalariosAnuales* con los números de empleado, apellidos y salarios anuales de los empleados de la tabla *Emple* que trabajan en el departamento número 20. Hemos de tener en cuenta que los salarios en la tabla *Emple* son mensuales, por lo que para obtener los anuales, si no consideramos pagas extra, habrá que multiplicarlos por 12. Crearemos la vista con la siguiente sentencia:

```
mysql> create view SalariosAnuales (numemp, apeemp, salemp)
    -> as select emp_no, apellido, salario * 12
    -> from emple
    -> where dept_no = 20;
Query OK, 0 rows affected (0.06 sec)

mysql> select * from SalariosAnuales;
+--------+---------+----------+
| numemp | apeemp  | salemp   |
+--------+---------+----------+
|   7566 | JIMÉNEZ | 27600.00 |
|   7788 | GIL     | 28200.00 |
|   7876 | ALONSO  | 10320.00 |
+--------+---------+----------+
4 rows in set (0.00 sec)
```

A través de esta vista podremos realizar modificaciones de atributos de la tabla *Emple* con la condición de que el atributo que se modifica en la vista *SalariosAnuales* no sea un

atributo calculado o derivado, como ocurre con el atributo *salemp*. Así, por ejemplo, podremos poner en minúsculas excepto la primera letra el apellido de ALONSO:

```
mysql> update SalariosAnuales
    -> set apeemp='Alonso'
    -> where apeemp='ALONSO';
Query OK, 1 row affected (0.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from emple;
+--------+----------+-------------+------+------------+---------+----------+---------+
| emp_no | apellido | oficio      | dir  | fecha_alt  | salario | comision | dept_no |
+--------+----------+-------------+------+------------+---------+----------+---------+
|   7499 | ARROYO   | VENDEDOR    | 7698 | 2009-02-20 | 1200.00 |   240.00 |      30 |
.  .  . .  .  . .  .  .
    7876 | Alonso   | EMPLEADO    | 7788 | 2013-08-09 |  860.00 |     0.00 |      20 |
|   7900 | JIMENO   | EMPLEADO    | 7698 | 2010-12-03 |  725.00 |     0.00 |      30 |
+--------+----------+-------------+------+------------+---------+----------+---------+
11 rows in set (0.00 sec)
```

Pero si intentamos modificar su salario, al ser este un campo calculado, se producirá un error:

```
mysql> update SalariosAnuales
    -> set salemp=11500
    -> where apeemp='Alonso';
ERROR 1348 (HY000): Column 'salemp' is not updatable
```

Se puede modificar la sentencia SELECT asociada a una vista mediante la sentencia ALTER VIEW, cuya sintaxis es la siguiente:

```
ALTER VIEW Nombre_Vista [(campo1, campo2, …, campon)]
AS sentencia_select
```

Por ejemplo, vamos a realizar una modificación sobre la vista *SalariosAnuales* creada anteriormente, de manera que el salario anual tenga en cuenta también dos pagas extras al año (14 pagas mensuales en lugar de 12):

```
mysql> alter view SalariosAnuales (numemp, apeemp, salemp)
    -> as select emp_no, apellido, salario * 14
    -> from emple
    -> where dept_no=20;
Query OK, 0 rows affected (0.04 sec)

mysql> select * from SalariosAnuales;
+--------+---------+----------+
| numemp | apeemp  | salemp   |
+--------+---------+----------+
|   7566 | JIMÉNEZ | 32200.00 |
|   7788 | GIL     | 32900.00 |
|   7876 | Alonso  | 12040.00 |
+--------+---------+----------+
4 rows in set (0.00 sec)
```

Para eliminar una o varias vistas de la base de datos, se usa la sentencia DROP VIEW, cuya sintaxis es la siguiente:

```
DROP VIEW [IF EXISTS] vista1, vista2, …
```

La opción IF EXISTS sirve para que si no existe alguna de las vistas indicadas no se muestre el mensaje de error que aparece por defecto. Por ejemplo, para borrar la vista *SalariosAnuales*, deberemos escribir:

```
mysql> drop view SalariosAnuales;
Query OK, 0 rows affected (0.00 sec)
```

3. Sinónimos.

El concepto de sinónimo no existe en MySQL, pero sí en otros SGBDs como Oracle. Así, en Oracle, cuando un usuario tiene acceso a una tabla de otro usuario y quiere realizar cualquier operación sobre la misma, debe anteponer al nombre de la tabla el nombre del usuario y un punto. Por ejemplo, si el usuario *jose* tiene acceso a la tabla *Depart* de *santi*, para consultar dicha tabla tendrá que teclear la orden:

```
select * from santi.Depart;
```

Mediante el uso de sinónimos *jose* puede crear un sinónimo para referirse a la tabla de *santi* sin necesidad de incluir su nombre.

Un sinónimo es un nuevo nombre que se puede dar a una tabla o a una vista. Con los sinónimos se pueden utilizar dos nombres para referirse a un mismo objeto de la base de datos. Resulta útil sobre todo para acceder a tablas y vistas de otros usuarios. Para crear un sinónimo se utiliza la sintaxis:

```
CREATE SYNONYM nombre_sinónimo FOR [nombre_usuario.]nombre_tabla;
```

Para eliminar sinónimos se empleará la siguiente sintaxis:

```
DROP SYNONYM [nombre_usuario.]nombre_sinónimo;
```

4. Usuarios.

Las bases de datos normalmente son empleadas por varios usuarios y en entornos multiusuario es necesario realizar un control de acceso para garantizar que solo puedan acceder a la base de datos los usuarios autorizados. Además, es preciso garantizar que cada usuario que accede a la base de datos solo pueda realizar las operaciones que le correspondan de acuerdo con su perfil.

En MySQL existe la instrucción CREATE USER, que permite crear usuarios para la base de datos. La sintaxis de esta instrucción es la siguiente:

```
CREATE USER Nombre_usuario1 [IDENTIFIED BY [PASSWORD] 'contraseña1'],
[Nombre_usuario2 [IDENTIFIED BY [PASSWORD] 'contraseña2'],…
[DEFAULT ROLE NombreRol]
[WITH opción_with]
[PASSWORD EXPIRE {DEFAULT | NEVER | INTERVAL n DAY}];
```

Se debe especificar después de la palabra USER el nombre del usuario que se desea crear, el cual puede constar solo de un nombre de usuario o puede tener la forma nombre_usuario@nombre_host, donde nombre_host es el nombre de la máquina desde la que se efectúa la conexión. Puede tratarse de una dirección IP (que se debe indicar entre comillas) o del servidor local, en cuyo caso hay que escribir *localhost*.

Se puede crear un usuario sin contraseña, lo cual no es aconsejable, o, por el contrario, se le puede asignar una contraseña especificando después de IDENTIFIED BY y PASSWORD (esta palabra es opcional) la contraseña entre comillas simples.

En la misma instrucción se pueden crear varios usuarios separando la información de un usuario de la de otro usuario por el símbolo coma.

Se puede indicar opcionalmente el rol que se desea asignar al usuario escribiendo después de DEFAULT ROLE el nombre de dicho rol.

Es posible especificar cuándo expirará la contraseña del/de los usuario/s creados/s mediante la opción PASSWORD EXPIRE. Se pueden indicar 3 valores:

- *Default* indica que expira según está especificado en la variable del sistema *default_password_lifetime* (número de días).
- *Never* indica que no expira nunca.
- *Interval n day* indica que expira dentro de n días.

Por ejemplo, mediante la siguiente instrucción creamos dos usuarios: uno llamado *jose*, con contraseña *1234*, que se va a poder conectar desde cualquier ordenador, y otro llamado *ana*, con contraseña *5678*, que solo se va a poder conectar desde el ordenador local.

```
mysql> create user jose identified by '1234',
    -> ana@localhost identified by '5678';
Query OK, 0 rows affected (0.00 sec)
```

Cuando se crean usuarios, MySQL añade dichos usuarios a la tabla *User* de la base de datos *MySQL*. Esta tabla presenta el siguiente formato:

```
mysql> use mysql
Database changed
mysql> desc 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                     |       |
| Process_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| File_priv               | enum('N','Y')                     | NO   |     | N                     |       |
| Grant_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| References_priv         | enum('N','Y')                     | NO   |     | N                     |       |
| Index_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| Alter_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| Show_db_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Super_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| Create_tmp_table_priv   | enum('N','Y')                     | NO   |     | N                     |       |
| Lock_tables_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Execute_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Repl_slave_priv         | enum('N','Y')                     | NO   |     | N                     |       |
| Repl_client_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Create_view_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Show_view_priv          | enum('N','Y')                     | NO   |     | N                     |       |
| Create_routine_priv     | enum('N','Y')                     | NO   |     | N                     |       |
| Alter_routine_priv      | enum('N','Y')                     | NO   |     | N                     |       |
| Create_user_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Event_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| Trigger_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Create_tablespace_priv  | enum('N','Y')                     | NO   |     | N                     |       |
| ssl_type                | enum('','ANY','X509','SPECIFIED') | NO   |     |                       |       |
| ssl_cipher              | blob                              | NO   |     | NULL                  |       |
| x509_issuer             | blob                              | NO   |     | NULL                  |       |
| x509_subject            | blob                              | NO   |     | NULL                  |       |
| max_questions           | int(11) unsigned                  | NO   |     | 0                     |       |
| max_updates             | int(11) unsigned                  | NO   |     | 0                     |       |
| max_connections         | int(11) unsigned                  | NO   |     | 0                     |       |
| max_user_connections    | int(11) unsigned                  | NO   |     | 0                     |       |
| plugin                  | char(64)                          | NO   |     | caching_sha2_password |       |
| authentication_string   | text                              | YES  |     | NULL                  |       |
| password_expired        | enum('N','Y')                     | NO   |     | N                     |       |
| password_last_changed   | timestamp                         | YES  |     | NULL                  |       |
| password_lifetime       | smallint(5) unsigned              | YES  |     | NULL                  |       |
| account_locked          | enum('N','Y')                     | NO   |     | N                     |       |
| Create_role_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Drop_role_priv          | enum('N','Y')                     | NO   |     | N                     |       |
| Password_reuse_history  | smallint(5) unsigned              | YES  |     | NULL                  |       |
| Password_reuse_time     | smallint(5) unsigned              | YES  |     | NULL                  |       |
| Password_require_current| enum('N','Y')                     | YES  |     | NULL                  |       |
| User_attributes         | json                              | YES  |     | NULL                  |       |
+-------------------------+-----------------------------------+------+-----+-----------------------+-------+
51 rows in set (0.15 sec)
```

Los campos que más nos interesan ahora de esta tabla son los siguientes:

- *Host*: Contiene el nombre de la máquina desde la que se puede conectar el usuario.
- *User*: Contiene el nombre del usuario que se ha creado.
- *Authentication_string*: Es la contraseña con la que accede el usuario correspondiente.

Veamos el contenido de estos campos para todas las filas de esta tabla:

```
mysql> select host, user, authentication_string from user;
+-----------+------------------+-------------------------------------------------------------------------+
| host      | user             | authentication_string                                                   |
+-----------+------------------+-------------------------------------------------------------------------+
| %         | jose             | *A4B6157319038724E3560894F7F932C8886EBFCF                               |
| localhost | ana              | *F13ACB16013CCF84877CED35B9DB9105C0C98A43                               |
| localhost | mysql.infoschema | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.session    | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.sys        | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | root             | *53890EDEEA2F1FE635CC0EC35A76445350A17909                               |
+-----------+------------------+-------------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

Hay varios usuarios, concretamente *root*, *mysql.infoschema*, *mysql.session* y *mysql.sys*, que son creados por MySQL cuando se realiza la instalación. Podemos observar

que están además los usuarios *jose* y *ana* que acabamos de crear. Para el usuario *jose* el campo host toma el valor %, lo que quiere decir que este usuario se puede conectar desde cualquier ordenador; sin embargo, en el caso de *ana*, en el atributo *host* pone *localhost*, lo que quiere decir que solo se podrá conectar desde el ordenador local. Ambos tienen asignada una contraseña, la cual está encriptada.

Todos los campos cuyo nombre finaliza en *priv* hacen referencia a si el usuario en cuestión tiene otorgado o no el privilegio correspondiente a nivel global, según si el valor del atributo en cuestión es 'Y' (sí) o 'N' (no), respectivamente.

La opción *with* dentro de la instrucción CREATE USER puede contener uno o varios de los valores que se especifican a continuación:

- MAX_QUERIES_PER_HOUR número: Permite especificar el número de consultas y actualizaciones que ese usuario puede realizar por hora.
- MAX_UPDATES_PER_HOUR número: Permite especificar el número de actualizaciones que ese usuario puede realizar por hora.
- MAX_CONNECTIONS_PER_HOUR número: Permite indicar el número de conexiones o logueos que puede realizar el usuario como máximo en una hora.
- MAX_USER_CONNECTIONS número: Permite especificar el número máximo de conexiones simultáneas que el usuario puede establecer con el servidor.

Si en cualquiera de los cuatro valores presentados no se asigna ningún valor, se supone un cero, que quiere decir que no hay límite.

Por ejemplo, podemos crear un tercer usuario llamado *santi* que solo se pueda conectar desde el ordenador local y vamos a limitarle el número de consultas y actualizaciones por hora a 100 y el número de conexiones simultáneas a 10:

```
mysql> create user santi@localhost identified by '4321'
    -> with max_queries_per_hour 100 max_user_connections 10;
Query OK, 0 rows affected (0.05 sec)
```

La información sobre las restricciones impuestas al usuario se almacena también en la tabla *User*, concretamente en los atributos *max_questions*, *max_updates*, *max_connections* y *max_user_connections*. Así, si consultamos para los diferentes usuarios del servidor su nombre de usuario y estos atributos, vemos el siguiente contenido:

```
mysql> select host, user, max_questions, max_updates, max_connections, max_user_connections
    -> from user;
+-----------+------------------+---------------+-------------+-----------------+----------------------+
| host      | user             | max_questions | max_updates | max_connections | max_user_connections |
+-----------+------------------+---------------+-------------+-----------------+----------------------+
| %         | jose             |             0 |           0 |               0 |                    0 |
| localhost | ana              |             0 |           0 |               0 |                    0 |
| localhost | mysql.infoschema |             0 |           0 |               0 |                    0 |
| localhost | mysql.session    |             0 |           0 |               0 |                    0 |
| localhost | mysql.sys        |             0 |           0 |               0 |                    0 |
| localhost | root             |             0 |           0 |               0 |                    0 |
| localhost | santi            |           100 |           0 |               0 |                   10 |
+-----------+------------------+---------------+-------------+-----------------+----------------------+
7 rows in set (0.00 sec)
```

Se pueden cambiar las restricciones asignadas a un usuario y la opción de expiración de la contraseña mediante el comando ALTER USER con la siguiente sintaxis:

```
ALTER USER Nombre_usuario
[DEFAULT ROLE NombreRol]
[WITH opción_with]
[PASSWORD EXPIRE {DEFAULT | NEVER | INTERVAL n DAY}];
```

Así, con la siguiente orden cambiaremos el número máximo de conexiones simultáneas del usuario santi@localhost a 15:

```
mysql> alter user santi@localhost
    -> with max_user_connections 15;
Query OK, 0 rows affected (0.03 sec)
```

También se pueden asignar contraseñas a usuarios creados mediante el comando SET PASSWORD, cuya sintaxis es la siguiente:

```
SET PASSWORD FOR Nombre_usuario = 'contraseña' [REPLACE 'contraseña_actual'];
```

Si se especifica la opción REPLACE, se debe indicar a continuación la contraseña actual que se desea modificar.

Así, para cambiar la contraseña del usuario *jose* a ‘4321’ tendremos que ejecutar la orden:

```
mysql> set password for jose = '4321';
Query OK, 0 rows affected (0.00 sec)
```

También se puede cambiar el nombre de un usuario mediante la instrucción RENAME USER, cuya sintaxis es la siguiente:

```
RENAME USER nombre_antiguo1 TO nombre_nuevo1,
[ ,nombre_antiguo2 TO nombre_nuevo2]…
```

Solo se debe indicar el nombre antiguo del usuario y el nuevo nombre y en la misma orden se puede cambiar el nombre a varios usuarios. Por ejemplo, para cambiar el nombre del usuario *ana@localhost* por *anita@localhost* pondremos:

```
mysql> rename user ana@localhost to anita@localhost;
Query OK, 0 rows affected (0.00 sec)
```

Para eliminar un usuario se usa la orden DROP USER, cuya sintaxis es la siguiente:

```
DROP USER Nombre_usuario1 [, Nombre_usuario2]…;
```

Como se puede observar, solo se debe indicar el nombre del usuario que se desea borrar y en la misma instrucción DROP USER se pueden eliminar varios usuarios. Por ejemplo, con la siguiente orden se borrará al usuario *anita*, que se conecta desde el ordenador local.

```
mysql> drop user anita@localhost;
Query OK, 0 rows affected (0.00 sec)
```

5. Permisos.

Los permisos en MySQL se pueden referir a varios niveles:

- Nivel global: Los permisos globales se aplican a todas las bases de datos de un servidor dado.
- Nivel de base de datos: Los permisos a nivel de base de datos se aplican a todos los objetos de una base de datos dada.
- Nivel de tabla: Los permisos a nivel de tabla se aplican a todos los atributos de la tabla.
- Nivel de columna: Los permisos a nivel de columna se aplican a columnas específicas en una determinada tabla.
- Nivel de rutina: Los permisos a nivel de rutina se aplican a rutinas específicas (procedimientos o funciones).

De hecho, en la base de datos *MySQL*, además de la tabla *User*, que contiene información sobre los usuarios y los permisos a nivel global que tienen otorgados, se encuentran las siguientes tablas:

- *Db*: Contiene privilegios a nivel de base de datos.
- *Tables_priv*: Contiene privilegios a nivel de tabla.
- *Columns_priv*: Contiene privilegios a nivel de atributo.
- *Procs_priv*: Contiene privilegios para procedimientos y funciones.

Permiso	Significado
ALL [PRIVILEGES]	Todos los permisos excepto GRANT OPTION
ALTER	Permite modificar el diseño de una tabla con ALTER TABLE
ALTER ROUTINE	Permite modificar o borrar rutinas almacenadas (procedimientos y funciones).
CREATE	Permite el uso de CREATE TABLE y CREATE DATABASE
CREATE ROLE	Permite crear roles
CREATE ROUTINE	Permite crear rutinas almacenadas
CREATE TABLESPACE	Permite crear tablespaces, alterarlos y borrarlos.
CREATE TEMPORARY TABLES	Permite el uso de CREATE TEMPORARY TABLE
CREATE USER	Permite el uso de CREATE USER, DROP USER, RENAME USER y REVOKE ALL PRIVILEGES
CREATE VIEW	Permite el uso de CREATE VIEW
DELETE	Permite el uso de DELETE
DROP	Permite el uso de DROP TABLE, DROP DATABASE y DROP VIEW
DROP ROLE	Permite eliminar roles
EVENT	Permite crear, modificar, eliminar y visualizar eventos.
EXECUTE	Permite ejecutar rutinas almacenadas
FILE	Permite el uso de SELECT … INTO OUTFILE (exportar) y LOAD DATA INFILE (importar)
GRANT OPTION	Permite conceder o retirar a otros usuarios los privilegios poseídos.
INDEX	Permite el uso de CREATE INDEX y DROP INDEX
INSERT	Permite el uso de INSERT
LOCK TABLES	Permite el uso de LOCK TABLES (bloquear tablas) en tablas para las que tenga el permiso SELECT
PROCESS	Permite el uso de SHOW FULL PROCESSLIST y de SHOW ENGINE.
REFERENCES	Permite crear claves ajenas al atributo especificado
RELOAD	Permite el uso de FLUSH (limpiar la memoria caché)
REPLICATION CLIENT	Permite al usuario preguntar dónde están los servidores maestro o esclavo en caso de replicación
REPLICATION SLAVE	Es necesario para los esclavos de replicación (para leer eventos del log binario desde el maestro)
SELECT	Permite el uso de SELECT
SHOW DATABASES	Permite el uso de SHOW DATABASES, que muestra todas las bases de datos
SHOW VIEW	Permite el uso de SHOW CREATE VIEW
SHUTDOWN	Permite el uso de mysqladmin shutdown (para parar el servidor)
SUPER	Permite el uso de CHANGE MASTER, KILL, PURGE BINARY LOGS, SET GLOBAL y el comando mysqladmin admin debug, que permite conectar incluso si se llega *a max_connections*.
TRIGGER	Permite crear, eliminar, ejecutar y mostrar triggers.
UPDATE	Permite el uso de UPDATE
USAGE	Es un sinónimo de ningún privilegio

Figura 1: Permisos en MySQL.

Los permisos o privilegios que se pueden especificar en MySQL se muestran en la tabla de la página 14.

El comando para asignar privilegios en MySQL es GRANT, cuya sintaxis es la siguiente:

```
GRANT privilegio1 [(lista_atributos1)] [, privilegio2 [(lista_atributos2)]]…
ON {nombre_tabla | * | *.* | nombre_BD.* | nombre_BD.nombre_tabla |
    nombre_BD.nombre_rutina}
TO nombre_usuario1
[, nombre_usuario2 …]
[WITH GRANT OPTION];
```

Los privilegios que se desean otorgar se deben indicar después de la palabra GRANT y pueden ser los que se han especificado en la tabla de la figura 1. Se pueden conceder varios privilegios en la misma orden separados por comas y cada privilegio se puede otorgar sobre todos los atributos, que es lo que se lleva a cabo por defecto, o sobre uno o varios atributos en concreto, que se deben especificar entre paréntesis y separados por comas. Además de privilegios, se pueden conceder roles.

Se pueden otorgar privilegios a nivel de tabla, en cuyo caso hay que especificar después de ON el nombre de la tabla. Si la tabla se encuentra en una base de datos diferente de la actual, al nombre de la tabla se le debe anteponer el nombre de la base de datos y un punto. Para otorgar un privilegio a nivel de la base de datos actual, es decir, a todas las tablas de la base de datos actual, se especificará después de ON un asterisco (*). Si el privilegio se desea otorgar sobre todas las tablas de una base de datos diferente de la actual, se deberá especificar nombre_BD.*. Se pueden asignar privilegios globales, es decir, a nivel de todas las bases de datos, especificando después de ON *.*. Finalmente, se pueden conceder privilegios a nivel de una rutina (procedimiento o función), en cuyo caso se debe especificar nombre_BD.nombre_rutina.

Los privilegios CREATE ROLE, CREATE TABLESPACE, CREATE USER, DROP ROLE, FILE, PROCESS, RELOAD, REPLICATION CLIENT, REPLICATION SLAVE, SHOW DATABASES, SHUTDOWN y SUPER son permisos administrativos que solo se pueden otorgar globalmente, es decir, empleando la sintaxis *.*.

Los únicos privilegios que se pueden especificar para una tabla son SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT_OPTION, REFERENCES, INDEX, CREATE VIEW, SHOW VIEW, TRIGGER y ALTER. Los únicos privilegios que se pueden especificar para un atributo son SELECT, INSERT, UPDATE Y REFERENCES.

Se pueden conceder los privilegios indicados a uno o varios usuarios, que se especifican después de la palabra TO. Para conceder el/los privilegio/s a varios usuarios, se deben separar sus nombres por el símbolo coma (,).

La opción WITH GRANT OPTION concede al usuario que recibe el permiso la opción de dar a otros usuarios cualquiera de sus permisos. Por este motivo, es conveniente asignar este permiso con cautela.

Veamos algunos ejemplos de órdenes GRANT. En primer lugar, creemos el usuario *dba* y asignémosle todos los permisos para hacer cualquier cosa. Además, le vamos a dar la opción de pasar sus permisos a cualquier otro usuario:

```
mysql> create user dba identified by 'aaaa';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on *.* to dba
    -> with grant option;
Query OK, 0 rows affected (0.01 sec)
```

Concedamos al usuario *jose* la opción de consultar cualquier tabla de la base de datos *Pedidos*:

```
mysql> grant select on Pedidos.* to jose;
Query OK, 0 rows affected (0.09 sec)
```

Ahora entremos en una sesión con el usuario *jose*. Veremos cómo podemos consultar tablas de la base de datos *Pedidos*, pero no podemos por ejemplo insertar datos en la tabla *Pedido*.

```
C:\Users\Jose>cd c:\Program Files\MySQL\MySQL Server 8.0\bin

c:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u jose -P 33060 -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
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> use pedidos;
Database changed
mysql> select * from pedido;
+--------+------------+
| refped | fecped     |
+--------+------------+
| P0001  | 2018-02-16 |
| P0002  | 2018-02-18 |
| P0003  | 2018-02-23 |
| P0004  | 2018-02-25 |
+--------+------------+
4 rows in set (0.14 sec)
```

```
mysql> insert into pedido values ('P0005', '2019-09-26');
ERROR 1142 (42000): INSERT command denied to user 'jose'@'localhost' for
table 'pedido'
```

Ahora desde la sesión del usuario *root* concedámosle al usuario *jose* la opción de insertar, modificar y borrar datos sobre la tabla *Pedido*:

```
mysql> use pedidos;
Database changed
mysql> grant insert, update, delete on Pedido to jose;
Query OK, 0 rows affected (0.05 sec)
```

Desde la sesión del usuario *jose* ahora sí que podremos añadir un pedido y luego eliminarlo sin problema:

```
mysql> insert into pedido values ('P0005', '2019-06-26');
Query OK, 1 row affected (0.06 sec)

mysql> select * from pedido;
+--------+------------+
| refped | fecped     |
+--------+------------+
| P0001  | 2018-02-16 |
| P0002  | 2018-02-18 |
| P0003  | 2018-02-23 |
| P0004  | 2018-02-25 |
| P0005  | 2019-06-26 |
+--------+------------+
5 rows in set (0.00 sec)

mysql> delete from pedido where refped='P0005';
Query OK, 1 row affected (0.04 sec)
```

Concedámosle a este usuario la opción de crear tablas, alterarlas y eliminarlas en la base de datos *Pedidos*:

```
mysql> grant create, alter, drop on pedidos.* to jose;
Query OK, 0 rows affected (0.05 sec)
```

Para poder visualizar los privilegios de un determinado usuario se puede hacer uso del comando SHOW con la siguiente sintaxis:

```
SHOW GRANTS FOR Nombre_usuario;
```

Así, para visualizar los permisos que tiene otorgados el usuario *jose* escribiremos lo siguiente:

```
mysql> show grants for jose;
+-----------------------------------------------------------------+
| Grants for jose@%                                               |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'jose'@'%'                                |
| GRANT SELECT, CREATE, DROP, ALTER ON `pedidos`.* TO 'jose'@'%'  |
| GRANT INSERT, UPDATE, DELETE ON `pedidos`.`pedido` TO 'jose'@'%' |
+-----------------------------------------------------------------+
3 rows in set (0.03 sec)
```

Por otro lado, los privilegios concedidos a los usuarios también se pueden retirar o revocar, para lo que se emplea el comando REVOKE, cuya sintaxis es la misma que la del comando GRANT con las siguientes salvedades:

- En vez de GRANT hay que escribir REVOKE.
- En vez de escribir TO antes de los usuarios a los que se les conceden los privilegios, se debe escribir FROM antes de los usuarios a los que se les retiran los privilegios.
- No se puede emplear la opción with.

Por tanto, su sintaxis queda como sigue:

```
REVOKE privilegio1 [(lista_atributos1)] [, privilegio2 [(lista_atributos2)]]…
ON {nombre_tabla | * | *.* | nombre_BD.* | nombre_BD.nombre_tabla |
    nombre_BD.nombre_rutina}
FROM nombre_usuario1 [, nombre_usuario2] …;
```

Así, mediante la siguiente orden retiramos al usuario *jose* la posibilidad de añadir datos en la tabla *Pedido*:

```
mysql> revoke insert on pedido from jose;
Query OK, 0 rows affected (0.00 sec)
```

Si ahora mostramos los privilegios de este usuario, veremos que ya no aparece el privilegio que le acabamos de retirar:

```
mysql> show grants for jose;
+-----------------------------------------------------------------+
| Grants for jose@%                                               |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'jose'@'%'                                |
| GRANT SELECT, CREATE, DROP, ALTER ON `pedidos`.* TO 'jose'@'%' |
| GRANT UPDATE, DELETE ON `pedidos`.`pedido` TO 'jose'@'%'        |
+-----------------------------------------------------------------+
3 rows in set (0.00 sec)
```

6. Roles.

Un rol abarca un conjunto de privilegios sobre el sistema y/o sobre objetos de la base de datos. Para crear un rol se utiliza la orden CREATE ROLE con la siguiente sintaxis:

```
CREATE ROLE nombre_rol;
```

Una vez creado el rol, es posible asignarle privilegios con la orden GRANT estudiada, de igual manera que se otorgan privilegios a usuarios. Después de asignar al rol los privilegios que se considere que debe tener asignados, es posible asignar ese rol mediante una orden GRANT a uno o varios usuarios. También es posible asignar este rol a un usuario al crearlo mediante CREATE USER y la opción DEFAULT ROLE. Así, los usuarios tendrán a partir de ese momento todos los privilegios que se concedieron al rol asignado. Otra opción para

asignar un rol a un usuario ya creado es utilizar la instrucción ALTER USER con la opción DEFAULT ROLE.

Por ejemplo, podemos crear un rol llamado *RolVentas* para ser utilizado por los usuarios del departamento de Ventas. Los usuarios de este departamento deben poder realizar consultas, inserciones, modificaciones y borrados sobre las tablas *Pedido* y *LineaPedido* y deben poder consultar la tabla *Articulo*. Por ello, vamos a crear el rol y asignarle los permisos necesarios para poder realizar todas estas operaciones:

```
mysql> create role RolVentas;
Query OK, 0 rows affected (0.07 sec)

mysql> grant select, insert, update, delete on Pedidos.Pedido to RolVentas;
Query OK, 0 rows affected (0.07 sec)

mysql> grant select, insert, update, delete on Pedidos.LineaPedido to RolVentas;
Query OK, 0 rows affected (0.07 sec)

mysql> grant select on Pedidos.Articulo to RolVentas;
Query OK, 0 rows affected (0.06 sec)
```

También se pueden visualizar los privilegios otorgados a un rol mediante la orden *show grants*:

```
mysql> show grants for RolVentas;
+-------------------------------------------------------------------------------------+
| Grants for RolVentas@%                                                              |
+-------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `RolVentas`@`%`                                               |
| GRANT SELECT ON `pedidos`.`articulo` TO `RolVentas`@`%`                             |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `pedidos`.`lineapedido` TO `RolVentas`@`%` |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `pedidos`.`pedido` TO `RolVentas`@`%`      |
+-------------------------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

Ahora cada vez que se incorporen nuevos empleados al departamento de Ventas, con asignarles el rol *RolVentas*, recibirán de forma conjunta todos los privilegios que tiene concedidos este rol. Así, si se incorporan a este departamento dos usuarios (*luis* y *loli*), solo los tendremos que crear y asignarles este rol para que puedan trabajar:

```
mysql> create user loli identified by '1111', luis identified by '2222'
    -> default role RolVentas;
Query OK, 0 rows affected (0.04 sec)
```

Podemos visualizar los privilegios del usuario *luis* con la siguiente orden:

```
mysql> show grants for luis;
+-----------------------------------+
| Grants for luis@%                 |
+-----------------------------------+
| GRANT USAGE ON *.* TO `luis`@`%`  |
| GRANT `RolVentas`@`%` TO `luis`@`%` |
+-----------------------------------+
2 rows in set (0.00 sec)
```

Esto nos indica que el usuario *luis* tiene concedido el rol *RolVentas*. Si no sabemos qué privilegios conlleva este rol, los podemos consultar mediante la orden:

```
mysql> show grants for RolVentas;
+-----------------------------------------------------------------------------------------+
| Grants for RolVentas@%                                                                  |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `RolVentas`@`%`                                                   |
| GRANT SELECT ON `pedidos`.`articulo` TO `RolVentas`@`%`                                 |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `pedidos`.`lineapedido` TO `RolVentas`@`%` |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `pedidos`.`pedido` TO `RolVentas`@`%`       |
+-----------------------------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

La concesión de un rol a un usuario mediante la orden GRANT no surtirá efecto automáticamente porque por defecto los roles concedidos de esta manera no están activos. Se puede consultar el rol activo en una determinada sesión ejecutando la instrucción SELECT CURRENT_ROLE(); Si se indica NONE, querrá decir que no está activo ningún rol.

Si queremos que todas las asignaciones de roles con el comando GRANT se activen automáticamente, debemos asignar el valor 1 a la variable del sistema *activate_all_roles_on_login* en el archivo *my.ini*.

Para eliminar un rol se debe hacer uso de la instrucción DROP ROLE, cuyo formato es el siguiente:

```
DROP ROLE [IF EXISTS] Nombre_rol1 [, Nombre_rol2]…;
```