Administración de usuarios y permisos en MySQL®

La administración de usuarios y permisos en MySQL® no puede dejarse librada solo a DBAs ni mucho menos al simple azar. Por ello, en este artículo, aprenderemos a gestionar usuarios y permisos en MySQL®, de forma simple y comenzado desde lo más básico.

Escrito por: **Eugenia Bahit** (GLAMP Hacker & eXtreme Programmer)

Eugenia es **Arquitecta de Software**, docente e instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y **eXtreme Programming**. Miembro de la **Free Software Foundation**, **The Linux Foundation** y **Debian Hackers**. Creadora de python-printr, Europio Engine y colaboradora de Vim.

Webs:
Cursos de programación: www.cursosdeprogramacionadistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:
Twitter / Identi.ca: **@eugeniabahit**

Cualquier informático que por diversos motivos suela trabajar con MySQL® conectándose como `root`, debe tener la capacidad de administración suficiente y necesaria para tomar medidas con respecto a la seguridad de las bases de datos, que hasta incluso podrían "sacar las papas del fuego" en situaciones de suma emergencia.

Al contrario de lo que generalmente se cree, gestionar usuarios y permisos en MySQL®, es una tarea sumamente sencilla que está al alcance de cualquier programador o administrador de sistemas. Lo más complejo, es tomar las medidas indicadas realizando el análisis propio de un DBA.

Pero en este artículo, nos centraremos en el cómo se gestiona y respecto a lo demás, nos limitaremos a decir que:

La medida más apropiada es aquella que define una política clara sobre las necesidades reales de los usuarios del sistema.

Dicho esto, concentrémonos en cómo llevar a cabo las políticas elegidas.

Creación de usuarios

Para agregar un nuevo usuario , es tan simple como ejecutar la siguiente sentencia:

```
CREATE USER     nombre_de_usuario
IDENTIFIED BY   'clave en texto plano';
```

Por ejemplo, para agregar al usuario `juanperez` con la clave `123456`, se utilizaría:

```
CREATE USER     juanperez
IDENTIFIED BY   '123456';
```

Una muy buena práctica es limitar la conexión de los usuarios por *host*, para **prevenir conexiones desde *hosts* no deseados**. Para ello, se utiliza:

```
'nombre_de_usuario'@'host'
```

Por ejemplo, para crear al usuario `beatrix` y solo permitir su conexión local, la sentencia sería la siguiente:

```
CREATE USER     'beatrix'@'localhost'
IDENTIFIED BY   '123456';
```

Si luego se deseara **cambiar (o asignar en caso de inexistencia) una contraseña a un determinado usuario**, se utilizará `SET PASSWORD` como se muestra a continuación:

```
SET     PASSWORD
FOR     usuario = PASSWORD('nueva clave en texto plano');

# o también:
SET     PASSWORD
FOR     'usuario'@'host' = PASSWORD('nueva clave en texto plano');
```

Por ejemplo, para agregar `p1r0m4n14c0` como contraseña del usuario `root`, la sentencia sería:

```
SET     PASSWORD
FOR     'root'@'localhost' = PASSWORD('p1r0m4n14c0');
```

Otorgar permisos

Para otorgar permisos en MySQL®, se deben considerar:

Permiso: El tipo de consultas que se permitirá efectuar al usuario (`SELECT`, `INSERT`, `DELETE`, `UPDATE`);

Database: Las bases de datos y/o tablas sobre las cuáles aplicarán dichos permisos;

Usuario: El o los usuarios a los cuáles serán otorgados los permisos aplicados.

La configuración de permisos se realizará con la siguiente sentencia:

```
GRANT     permiso
ON        database
TO        usuario;
```

Por ejemplo, para otorgar permisos de selección sobre la tabla `categoria` de la base de datos `weblibros` al usuario `juanperez`, se ejecutará:

```
GRANT  SELECT
ON     weblibros.categoria
TO     juanperez;
```

Suponiendo que la base de datos `weblibros` contenga dos tablas:

```
mysql> show tables from weblibros;
+--------------------+
| Tables_in_weblibros |
+--------------------+
| categoria          |
| libro              |
+--------------------+
2 rows in set (0.00 sec)
```

Si en vez de `root` la consulta fuese realizada por el usuario `juanperez`, obtendría lo siguiente:

```
mysql> show databases;
+--------------------+
```

```
| Database           |
+--------------------+
| information_schema |
| weblibros          |
+--------------------+
2 rows in set (0.00 sec)

mysql> show tables from weblibros;
+----------------------+
| Tables_in_weblibros |
+----------------------+
| categoria           |
+----------------------+
1 row in set (0.00 sec)
```

Y si el usuario `juanperez` intentara acceder a una tabla no permitida, el acceso le sería negado:

```
mysql> select * from weblibros.libro;
ERROR 1142 (42000): SELECT command denied to user 'juanperez'@'localhost' for table
'libro'
```

Al momento de escribir las sentencias para otorgar permisos, disponemos de diferentes opciones.

Opciones para indicar el tipo de permisos:

```
GRANT   SELECT                  # un permiso específico
GRANT   SELECT, INSERT, UPDATE  # varios permisos
GRANT   ALL                     # todos los permisos
```

Opciones para indicar a qué bases de datos/tablas aplicarán los permisos:

```
ON      database.table          # a una tabla
ON      database.table1,        # a varias tablas
        database.table2,
        database2.table
ON      database.*              # a todas las tablas de la misma DB
```

Opciones para indicar a quién/quiénes aplican los permisos:

```
TO      usuario                 # a un usuario
TO      usuario1, usuario2      # a varios usuarios
```

Un ejemplo completo

Suponiendo que en sistema existen las siguientes bases de datos y tablas:

```
DATABASE            TABLAS
europioengine       categoria, materiaprima, producto
webcursos           categoria, libro
```

Crearemos primero 3 usuarios, a los que solo se les permita conectarse de forma local:

```
CREATE USER     'vanina'@'localhost'
IDENTIFIED BY   'v4n1n41974';

CREATE USER     'roxana'@'localhost'
IDENTIFIED BY   'r0x4n4-1275';

CREATE USER     'silvana'@'localhost'
IDENTIFIED BY   '33885409sil';
```

Y ahora, otorgaremos permisos de selección y actualización, a todos los usuarios recién creados, para todas las tablas de la base de datos `europioengine`:

```
GRANT     SELECT, UPDATE
ON        europioengine.*
TO        vanina, roxana, silvana;
```

Conceder permiso de conceder permisos

Cuando otorgamos permisos a un determinado usuario, es posible además, conceder un permiso extra que le permita asignar sus mismos privilegios a otros usuarios. Una situación de ejemplo podría ser la siguiente:

> *Al usuario Ana le asignamos permisos de selección sobre la base de datos ventas. Entonces, el usuario Ana, puede asignar permisos de selección sobre la base de datos ventas a cualquier otro usuario.*

De eso se trata la cláusula WITH GRANT OPTION que se especifica de la siguiente manera:

```
GRANT    permisos
ON       tablas
TO       usuario
WITH GRANT OPTION
```

Si seguimos con el ejemplo anterior, ahora, al usuario `vanina`, le otorgaremos todos los

permisos para la tabla `categoria` de la base de datos `weblibro`, permitiéndole conceder dichos permisos a cualquier otro usuario existente:

```
GRANT     ALL
ON        weblibros
TO        vanina
WITH      GRANT OPTION;
```

Revocando permisos

Revocar permisos es algo similar a concederlos. Solo cambia un poco la sintaxis:

```
GRANT => REVOKE
TO    => FROM
```

La sintaxis básica, se vería como la siguente:

```
REVOKE    privilegio
ON        tabla
FROM      usuario
```

Por ejemplo:

```
REVOKE     SELECT, INSERT
ON         midatabase.tabla_productos
FROM       anitamayer
```

Puedo además, revocar privilegios en cascada, para los casos en los cuáles, el usuario en cuestión, haya otorgado privilegios a terceras personas:

```
REVOKE     SELECT, INSERT
ON         midatabase.tabla_productos
FROM       anitamayer
CASCADE
```

O puedo optar por la alternativa contraria de recibir un error en caso en el cuál, pretender revocar privilegios a un usuario, afecte a otros usuarios:

```
REVOKE     SELECT, INSERT
ON         midatabase.tabla_productos
FROM       anitamayer
RESTRICT
```