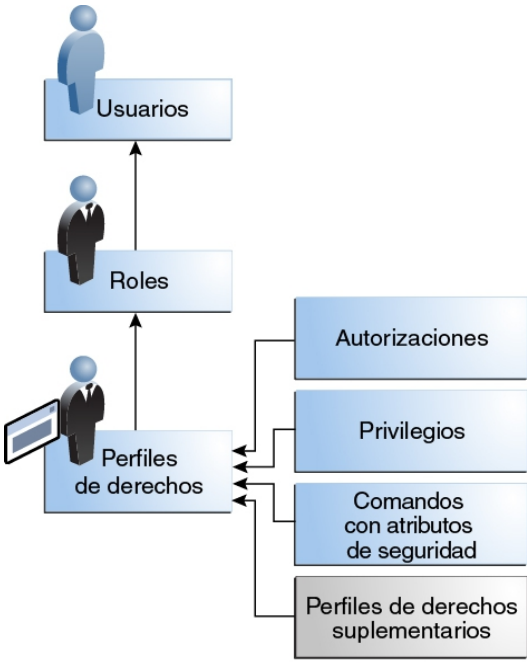
CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Tarea 7: Administración de Usuarios en MySQL



Bases de Datos
CFGS DAW

Alumno: David Barberá Zahonero

Tarea a realizar:

Imaginad que comenzáis a trabajar en una empresa y tenéis que dar de alta a nuevos usuarios en la BD, crear nuevos perfiles, y modificar roles y permisos para otros usuarios ya existentes.

En esta empresa utilizan MySQL como SGBD y en los apuntes que tenéis de BBDD no había nada de Administración de Usuarios para MySQL!!!! Horror!!!

No os queda más remedio que **buscar documentación en la red** sobre la administración de usuarios en MySQL y **redactar** un **documento** similar al de **teoría** de Oracle, para poder entender perfectamente las similitudes y diferencias entre ambos SGBD.

Para practicar, antes de hacer los cambios en la empresa y no meter la pata, realizaréis los **ejercicios de la UD7 en MySQL** y comentaréis las diferencias que encontráis con Oracle.

ÍNDICE

1 TAREAS FUNDAMENTALES DE UN DBA.

DBA es la persona encargada de la operación del sistema, y es el responsable de decidir:

- Los datos que se deben almacenar en la BD.
- La política de mantenimiento, tratamiento de los datos y la seguridad de la información.

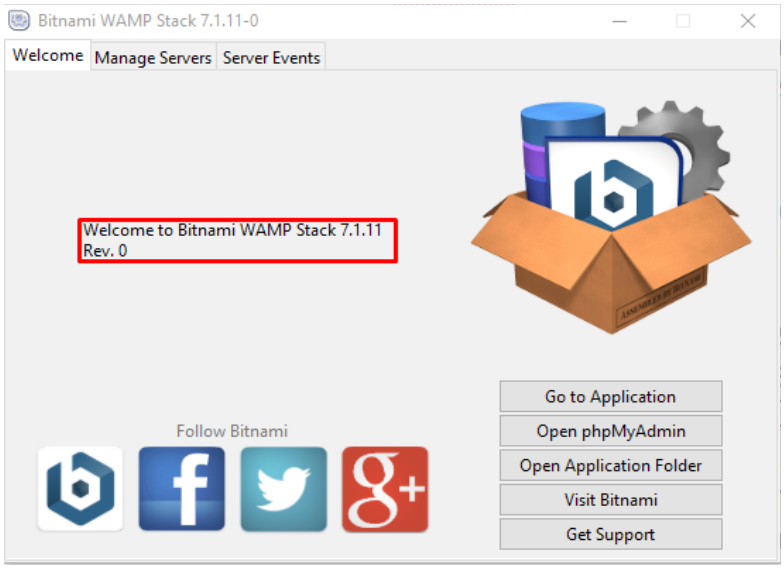
Entre las principales funciones de un DBA encontramos:

- Apoyar y asesorar respecto a que SGBD adquirir por parte de la Organización.
- Definir la información que contendrán las BBDD corporativas.
- Diseñar la estructura de almacenamiento y la estrategia de accesos a las BBDD.
- Ser el enlace entre los Usuarios de la Organización.
- Definir la política de copias de respaldo y recuperación de la información contenida en las BBDD.
- Proporcionar soporte a programadores y analistas que se encuentren desarrollando aplicaciones que crean y/o accedan a las BBDD.

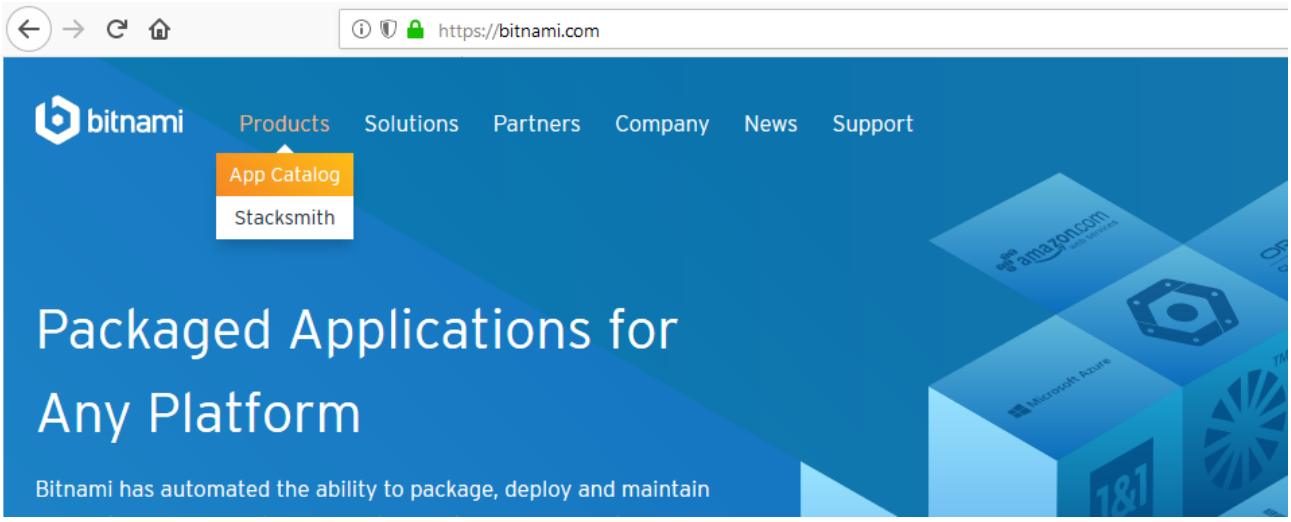
2 ACTUALIZACIÓN DEL SGBD.

Una de las tareas, no menos importante que las descritas en el punto anterior, para un DBA es mantener la propia BD y el Software para la Gestión de la Base de Datos actualizados. Por ello, primero debemos comprobar cual es la versión que tenemos instalada y comprobar si existen actualizaciones de la misma. En mi caso tengo instalado el paquete *Bitnami WAMP Stack*, que provee de un completo entorno de desarrollo listo para ejecutarse.

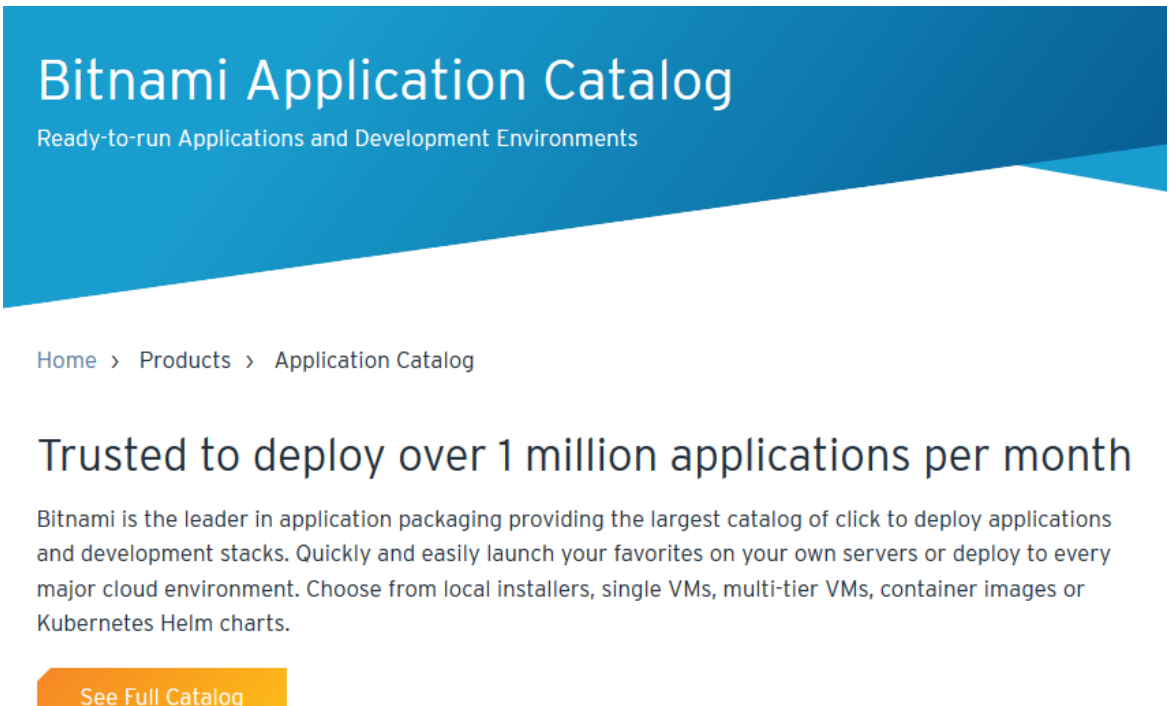
Además de PHP, MySQL y Apache, incluye FastCGI, OpenSSL, phpMyAdmin, ModSecurity, SQLite, ImageMagic, xDebug, Xcache, OpenLDAP y otros componentes. Para ver la versión del software ejecutamos la herramienta Bitnami WAMP Stack, una vez abierta podemos ver la versión de la misma.



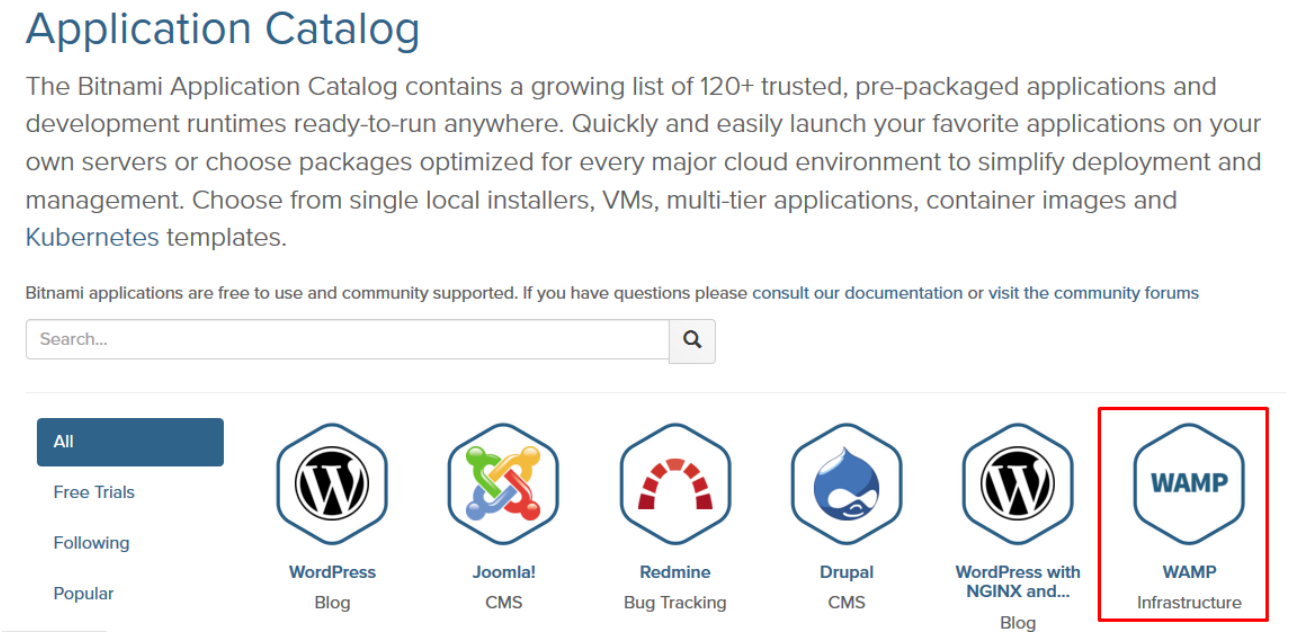
En mi caso la versión es la 7.1.11. Posteriormente comprobamos si en el sitio oficial hay alguna versión superior a la que tenemos instalada, para ello podemos acceder al sitio directamente pulsando sobre el botón *Visit Bitnami*. Una vez tenemos abierto en el navegador el siti de Bitnami accedemos al catálogo de apps.



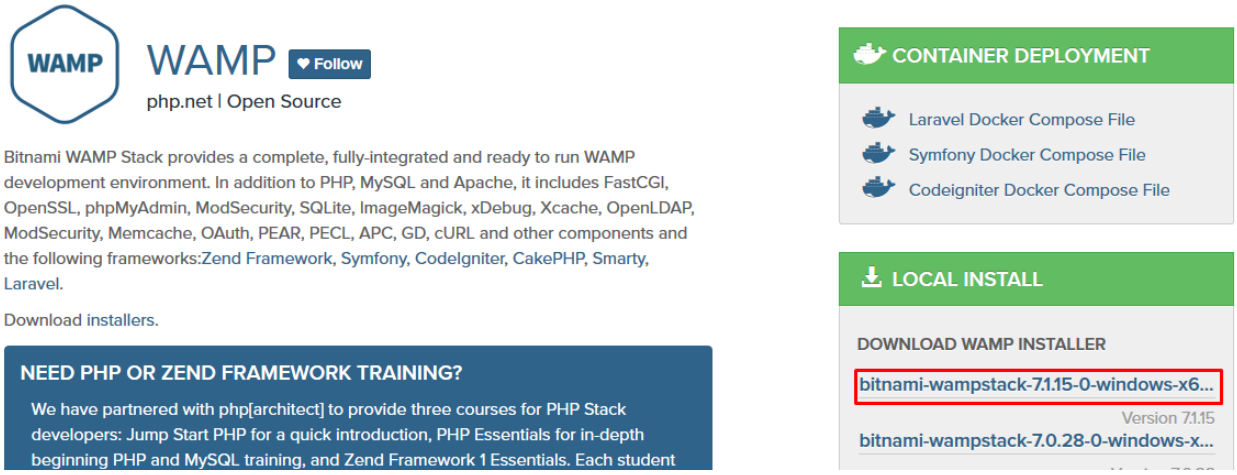
Y posteriormente pulsamos sobre el botón *See full catalog*.



Una vez accedido al catálogo completo de apps se puede ver el acceso paquete WAMP.



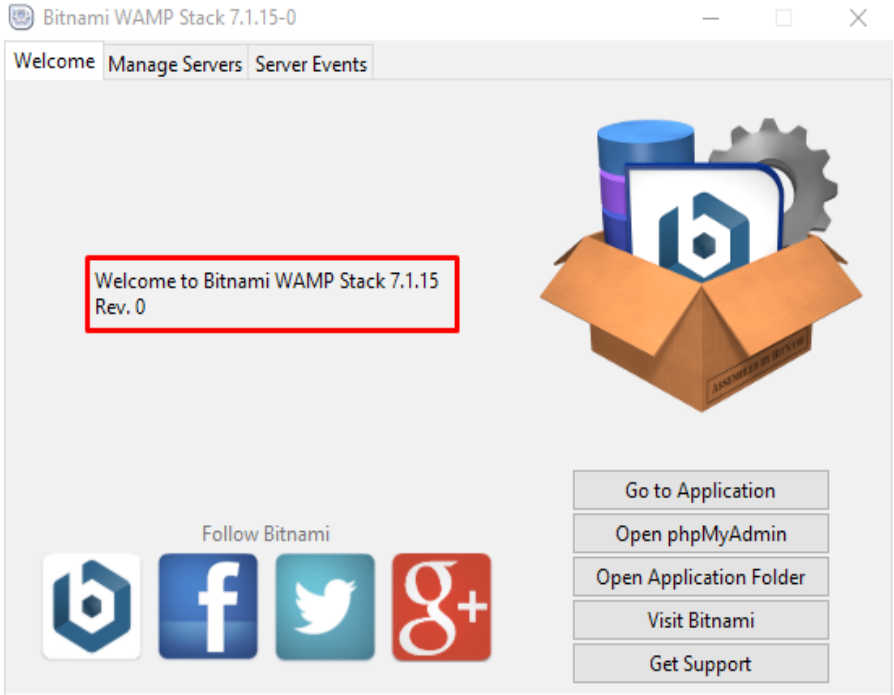
Pulsando sobre el mismo, accedemos a la página del paquete WAMP. Una vez en ella, podemos ver cual es la última versión del software.



Como podemos comprobar, la versión actual disponible en el sitio oficial de *Bitnami* es superior a la que tenemos instalada. Pulsando sobre el enlace iniciamos el proceso de descarga de la app.

Una vez descargado el archivo de instalación, y antes de proceder a iniciar la instalación del software, realizamos una copia de respaldo de todas las BD y todos los usuarios creados en la versión anterior de WAMP y desinstalamos la versión actual de nuestro sistema.

Una vez desinstalada la versión antigua de WAMP procedemos a la instalación de la nueva versión ejecutando el archivo descargado anteriormente. Seguimos el asistente de instalación hasta que finalice la misma.



Por último, una vez instalada la nueva versión, procedemos a importar las BD y usuarios exportados anteriormente a través de la herramienta phpMyAdmin desde la opción *Importar*. Una vez

finalizado el proceso de importación, ya tenemos nuestra BD y SGBD actualizados.

3 ADMINISTRACIÓN DE USUARIOS Y PRIVILEGIOS.

Cualquier informático que trabaje con MySQL y se conecte como *root*, debe de ser capaz de poder realizar la administración suficiente para tomar las medidas de seguridad oportunas para sus BBDD.

Gestionar usuarios y permisos en MySQL es una tarea sencilla, al alcance de cualquier programador o administrador de sistemas. Lo más complejo, es tomar las medidas indicadas realizando un análisis propio de un DBA.

MySQL permite definir diferentes usuarios, y además, asignar a cada uno de ellos determinados privilegios en distintos niveles o categorías.

3.1 Niveles de Privilegios.

Los privilegios otorgados a una cuenta MySQL determinan que operaciones puede realizar. Los privilegios de MySQL se diferencian por el contexto en el que se aplican y en diferentes niveles de operación:

Privilegios de Administración: Permiten al usuario gestionar operaciones del servidor MySQL.
Privilegios de Datos: Estos privilegios pueden concederse para bases de datos específicas, o globalmente para que se apliquen a todas las BBDD.
Privilegios de Estructura: Se aplican a tablas, índices, vistas y rutinas almacenadas dentro de una BD, para todos los objetos de un tipo dado de una BD, o globalmente para todos los objetos de un tipo dado dentro de una BD.

La información sobre los privilegios de una cuenta se almacena en las tablas de usuario db, tables_priv, columns_privs y procs_privs

3.2 Lista de Privilegios.

La siguiente tabla muestra los nombres de privilegio utilizados en las sentencias GRANT y REVOKE, junto con el nombre de columna asociada a cada privilegio en las tablas de concesión (mysql.user) y el contexto en el que se aplica el privilegio.

Privilegio	Columna	Contexto
ALL [PRIVILEGIOS]	Sinónimo de “todos los privilegios”	Administración de Servidor
ALTER	Alter_priv	Tablas

Privilegio	Columna	Contexto
ALTER ROUTINE	Alter_routine_priv	Rutinas Almacenadas
CREATE	Create_priv	BD, tablas, o índices
CREATE ROUTINE	Create_routine_priv	Rutinas Almacenadas
CREATE TABLESPACE	Create_tablespace_priv	Administración de Servidor
CREATE TEMPORARY TABLES	Create_tmp_table_priv	Tablas
CREATE USER	Create_user_priv	Administración de Servidor
CREATE VIEW	Create_view_priv	Vistas
DELETE	Delete_priv	Tablas
DROP	Drop_priv	BD, tablas, o vistas
EVENT	Event_priv	BD
EXECUTE	Execute_priv	Rutinas Almacenadas
FILE	File_priv	Acceso a archivo en el host del servidor
GRANT OPTION	Grant_priv	BD, tablas, o rutinas almacenadas
INDEX	Index_priv	Tablas
INSERT	Insert_priv	Tablas o columnas
LOCK TABLES	Lock_tables_priv	BD
PROCESS	Process_priv	Administración de Servidor
PROXY	See proxies_priv table	Administración de Servidor
REFERENCES	References_priv	BD o tablas
RELOAD	Reload_priv	Administración de Servidor
REPLICATION CLIENT	Repl_client_priv	Administración de Servidor
REPLICATION SLAVE	Repl_slave_priv	Administración de Servidor
SELECT	Select_priv	Tablas or columnas
SHOW DATABASES	Show_db_priv	Administración de Servidor
SHOW VIEW	Show_view_priv	Vistas
SHUTDOWN	Shutdown_priv	Administración de Servidor
SUPER	Super_priv	Server administration
TRIGGER	Trigger_priv	Tablas
UPDATE	Update_priv	Tablas o columnas
USAGE	Sinónimo de “sin privilegios”	Administración de Servidor

3.3 Creación de Usuarios.

Una cuenta se defien en terminos de un Nombre de Usuario y el/los host/s clientes desde donde se puede conectar. Su sintaxis es:

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

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

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

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

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

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

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

A partir de la versión 5.0.2 de MySQL existe la sentencia para crear usuarios *CREATER USER*, en versiones anteriores se usa exclusivamente la sentencia *GRANT* para crearlos.

En general es preferible usar la sentencia *GRANT*, ya que si crea un usuario mediante *CREATE USER*, posteriormente se tendrá que usar una sentencia *GRANT* para concederle los privilegios.

Usando *GRANT* podemos crear un usuario y al mismo tiempo concederle los privilegios que tendrá. La sintaxis simplificada para hacerlo es:

```
GRANT tipo_priv[(lista_col)],…(tipos de privilegios)
ON nombre_tbl.nombre_BD | nombre_BD.*(a tablas o BD)
TO nombre_usuario (para usuario)
IDENTIFIED BY 'password_usuario'; (password usuario)
```

Un ejemplo de creación de usuario mediante la sentencia *GRANT* sería:

```
GRANT SELECT, INSERT, UPDATE ON test.*
TO usuario1 IDENTIFIED BY 'password';
```

Con esta sentencia hemos creado el usuario: usuario1 y concedido los privilegios **SELECT**, **INSERT** e **UPDATE** en la BD **test** y a todos sus objetos.

La otra forma mucho más sencilla de crear usuarios en MySQL sería utilizando la sentencia *CREATE USER*. Por ejemplo, para crear el usuario *usuario1* y solo permitir su conexión local, la sentencia sería la siguiente:

```
CREATE USER 'usuario1'@'localhost'
IDENTIFIED BY 'pass_usuario1';
```

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_usuario'@'nombre_host'
```

3.4 Eliminar Usuarios.

Para eliminar una cuenta, se usa la declaración *DROP USER*. Su sintaxis es la siguiente:

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

Por ejemplo:

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

3.5 Cuentas de Usuario Reservadas.

Una parte de la instalación de MySQL es la inicialización del directorio de datos. Durante esta inicialización, MySQL crea cuentas de usuario que se deben considerar reservadas:

- **'root'@'localhost'**: Usado para propósitos administrativos.
- **'mysql.sys'@'localhost'**: Utilizado como *DEFINER* para objetos

del esquema *sys*. El uso de la cuenta mysql.sys evita problemas que ocurren si un DBA cambia el nombre o quita la cuenta root. Esta cuenta está bloqueada, por lo que no se puede usar para conexiones de clientes.

- **'mysql.session'@'localhost'**: Utilizada internamente por plugins para acceder al servidor. Esta cuenta está bloqueada, por lo que no se puede usar para conexiones de clientes.

3.6 Gestión de Contraseñas.

MySQL permite a los DBA caducar las contraseñas de las cuentas de usuario manualmente. Es posible establecer una política de caducidad global, así como también por cuenta.

Para expirar la contraseña de una cuenta manualmente, se usa la declaración *ALTER USER*:

```
ALTER USER 'nom_usuario'@'nom_host' PASSWORD EXPIRE;
```

Esta operación establece la contraseña caducada en la fila correspondiente de la tabla mysql.user.

Para establecer una política global de caducidad de contraseñas, se usa la variable de sistema ***DEFAULT_PASSWORD_LIFETIME***. Su valor por defecto es , que desactiva la expiración de la contraseña. Si se establece un valor positivo a esta variable de sistema, se indica el tiempo de vida permitido para la contraseña, de modo que la contraseña debe cambiar en N días. Ejemplos:

- Para establecer una politica global de contraseñas y establecer un tiempo de vida de la contraseña de 6 meses, iniciar el servidor con estas líneas en un archivo my.cnf del servidor:

```
[mysqld]
default_passwor_lifetime=180
```

- Para establecer una política global de contraseñas y que estas nunca expiren, establecer *default_password_lifetime=0:*

```
[mysqld]
default_passwor_lifetime=0
```

- *default_password_lifetime* también se puede cambiar en tiempo de ejecución:

```
SET  default_password_lifetime=0;
SET  default_password_lifetime=180;
```

Para establecer una política de caducidad de contraseñas en cuentas individuales se usa las opción de *PASSWORD EXPIRE* de las declaraciones *CREATE USER* y *ALTER USER*.

Ejemplos de declaración de expedición de contraseñas de una cuenta individual:

- Requiere que la contraseña sea cambiada cada 90 días:

```
CREATE USER 'usuario'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;
ALTER USER 'usuario'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;
```

- Desactivar la expiración de la contraseña:

```
CREATE USER 'usuario'@'localhost' PASSWORD EXPIRE NEVER;
ALTER USER 'usuario'@'localhost' PASSWORD EXPIRE NEVER;
```

- Establecer la política de expedición global:

```
CREATE USER 'usuario'@'localhost' PASSWORD EXPIRE DEFAULT;
ALTER USER 'usuario'@'localhost' PASSWORD EXPIRE DEFAULT;
```

3.7 Otorgar Privilegios.

Para otorgar permisos en MySQL se deben considerar:

Permiso: El tipo de consulta que podrá ejecutar el usuario.
Database: Las BD y/o tablas sobre las cuales se aplican los permisos.
Usuario: Los usuarios a los cuales le son otorgados los permisos.

La configuración de permisos se realizará mediante la siguiente sintaxis:

```
GRANT
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO user [auth_option] [, user [auth_option]] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH {GRANT OPTION | resource_option} ...]

GRANT PROXY ON user
    TO user [, user] ...
    [WITH GRANT OPTION]

object_type: {
    TABLE
  | FUNCTION
  | PROCEDURE
}

priv_level: {
    *
  | *.*
  | db_name.*
  | db_name.tbl_name
  | tbl_name
  | db_name.routine_name
}

user:
    (see Section 6.2.3, "Specifying Account Names")

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

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

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

Por ejemplo, para otorgar permisos de selección sobre la tabla categoria de la BD weblibros al usuario *usuario1*, se ejecutará:

```
GRANT SELECT
ON weblibros.categoria
TO usuario1;
```

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 que BD/tablas aplican los permisos:

```
ON database.tabla          # a una tabla
ON database.tabla1,
   database.tabla2,
   database.tabla3         # a varias tablas
ON database.*              # a todas las tablas de BD
```

Opciones para indicar a quien/quienes aplican los permisos:

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

3.8 Revocar Privilegios.

Para revocar privilegios se usa la sentencia REVOKE, su sintaxis es:

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] ...

REVOKE ALL [PRIVILEGES], GRANT OPTION
    FROM user [, user] ...

REVOKE PROXY ON user
    FROM user [, user] ...
```

Su sintaxis es muy similar a la de GRANT, por ejemplo para revocar el privilegio de selección del usuario: *usuario1* de la tabla categoria de la BD weblibros, la sentencia sería:

```
REVOKE SELECT
ON weblibros.categoria
FROM usuario1;
```

4 CONFIGURACIÓN DE RECURSOS DE SISTEMA.

Un medio de restricción del uso del servidor MySQL por un cliente es establecer la variable global del sistema *MAX_USER_CONNECTIOS* a un valor distinto de cero.

Esto limita el número de conexiones simultáneas que puede realizar cualquier cuanta dada, pero no limita a lo que un cliente puede hacer una vez conectado. Además, la configuración de *MAX_USER_CONNECTIOS* no habilita la gestión de cuentas individuales.

Para abordar estas preocupaciones, MySQL permite límites para cuentas individuales en el uso de estos recursos de servidor:

Variable Global	Descripción
MAX_QUERIES_PER_HOUR	El número de consultas que una cuenta puede emitir por hora.
MAX_CONNECTIONS_PER_HOUR	El número de veces que una cuenta se puede conectar al servidor por hora.
MAX_UPDATES_PER_HOUR	El número de actualizaciones que una cuenta puede emitir por hora.
MAX_USER_CONNECTIONS	El número de conexiones al servidor similtáneas puede realizar una cuenta.

Cualquier declaración que un cliente puede emitir cuenta contra el límite de consulta, a menos que sus resultados se publiquen desde el caché de consultas.

Para establecer los límites de los recursos del servidor para una cuenta en tiempo de creación, se usa la declaración *CREATE USER*, para modificar los límites de una cuenta existente, se usa la declaración *ALTER USER*, proporcionando una clausula *WITH* que indica que cada recurso está limitado. El valor por defecto de cada límite es cero (sin límite). Por ejemplo, para crear una nueva cuenta que puede acceder la la BD *Customer*, pero solo de manera limitada, emitimos esta declaración:

```
CREATE USER 'usuario1'@'localhost' IDENTIFIED BY 'usuario1'
WITH MAX_QUERIES_PER_HOUR 20
     MAX_UPDATES_PER_HOUR 10
     MAX_CONNECTIONS_PER_HOUR 5
     MAX_USER_CONNECTIONS 2;
```

Para modificar una cuenta existente, se usa la declaración *ALTER USER*. La siguiente declaración cambia el límite de consultas que puede realizar *usuario1* por hora a 100:

```
ALTER USER 'usuario1'@'localhost' WITH MAX_QUERIES_PER_HOUR 100;
```

5 TABLAS GRANT.

El sistema de bases de datos MySQL incluye varias tablas GRANT que contienen la información de las cuentas de usuario y los privilegios que estas soportan.

Para manipular el contenido de las tablas GRANT, se pueden modificar indirectamente usando declaraciones de gestión de cuentas como *CREATE USER*, *GRANT* y *REVOKE* para configurar y controlar los privilegios de cada una.

Las tablas de la base de datos MySQL que contienen la información GRANT son:

- **users**: cuentas de usuario, privilegios globales, y otras columnas sin privilegios.
- **db**: privilegios de nivel de BD.
- **tables_priv**: privilegios de nivel de tabla.
- **columns_priv**: privilegios de nivel de columna.
- **procs_priv**: privilegios de procedimientos almacenados.
- **proxies_priv**: privilegios de Proxy-user.

6 MOSTRAR PRIVILEGIOS DE USUARIO.

La siguiente instrucción muestra los privilegios concedidos a una cuenta de usuario:

```
SHOW GRANTS FOR usuario;
```

El nombre de cuenta de usuario para la decalaración *SHOW GRANTS* usa el mismo formato que para la declaración *GRANT*, por ejemplo: 'usuario1'@'localhost'.

```
SHOW GRANTS FOR 'usuario1'@'localhost';
```

Para mostrar los privilegios concedidos de la cuenta de usuario actual, puedes utilizar cualquiera de las siguientes declaraciones:

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

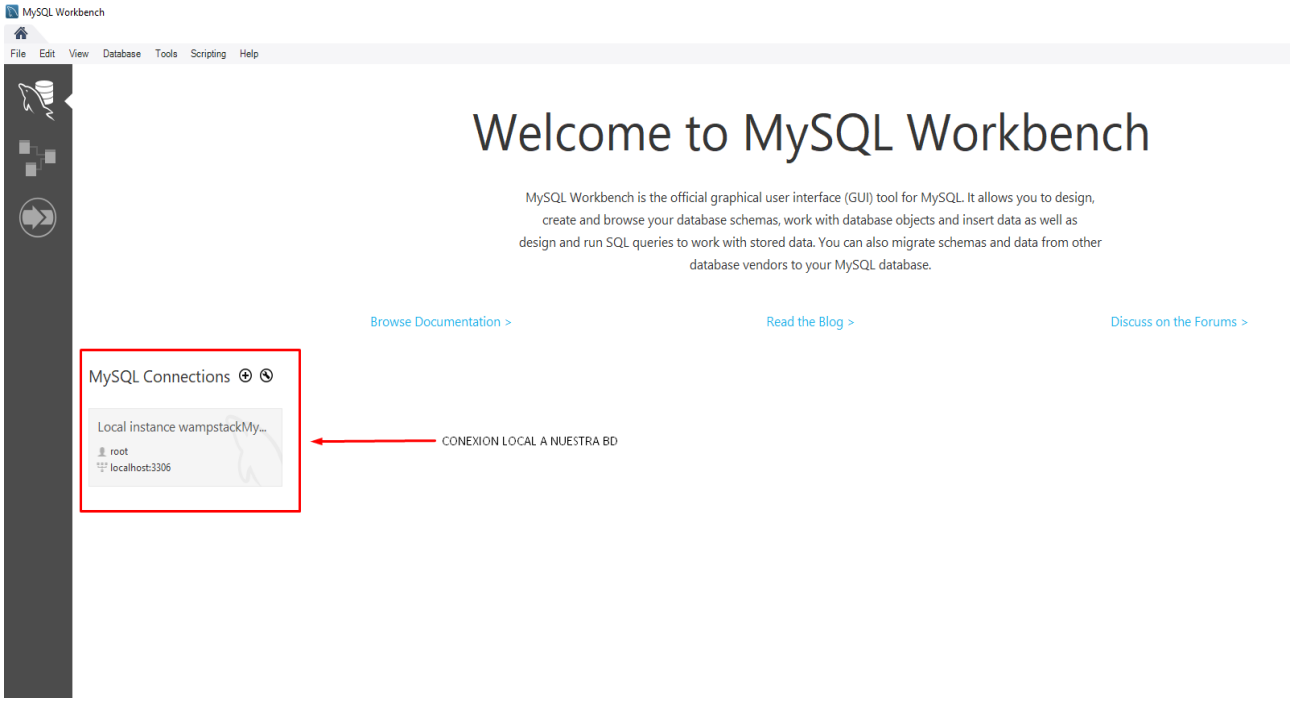
7 ROLES EN MYSQL.

7.1 ***Versión 7.21 (versión estable).***

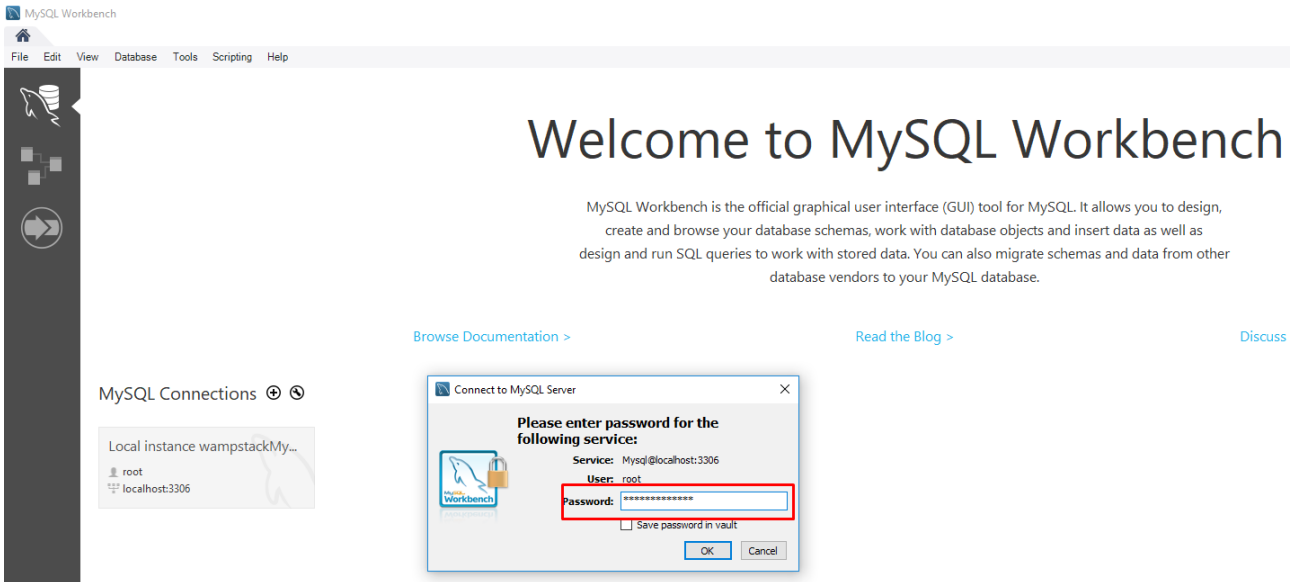
El concepto de Rol en MySQL es completamente diferente a Oracle. En la BD de Oracle el Rol (ROLE) es un objeto que podemos crear, modificar y borrar y en MySQL no existe tal objeto, es un concepto de configuración que se puede gestionar a través de determinados SGBD. Por ejemplo, desde la herramienta phpMyAdmin incluida en el software que hemos instalado anteriormente (WAMP), no existe la posibilidad de asignación de roles a la configuración de usuarios, pero la herramienta gestión de BD que nos provee la sitio oficial de MySQL, el concepto de rol, puede configurarse y ser asignado a las cuentas de usuario que estén creadas en nuestro sistema de BD. Esta herramienta se llama MySQL Workbench y la podemos descargar desde la url:

https://dev.mysql.com/downloads/workbench/

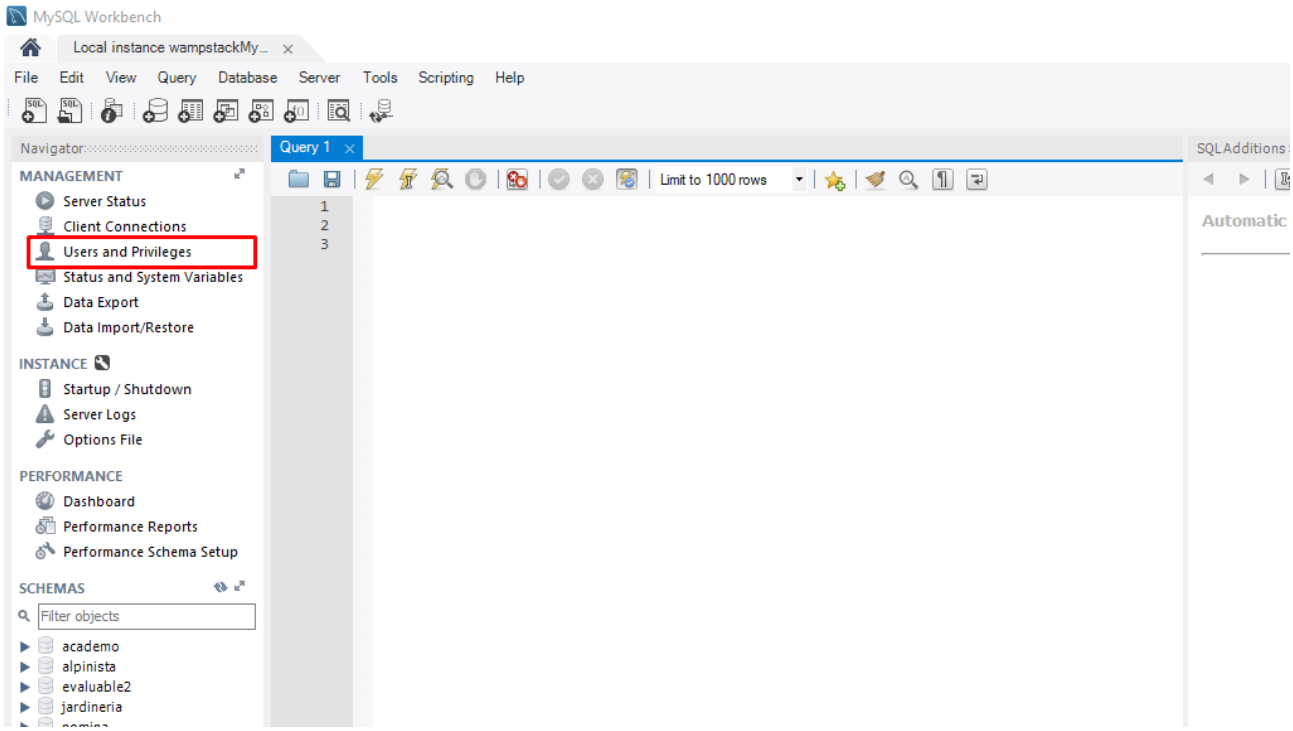
El proceso de instalación de la herramienta es muy sencillo, solo hay que seguir el asistente de instalación. Una vez instalado y teniendo los servicios de nuestra BD MySQL arrancados en nuestro ordenador, iniciamos el programa y este ya detecta nuestra BD y simplemente debemos conectarnos con nuestro usuario y contraseña.



Si pulsamos sobre nuestra conexión local, nos aparecerá la ventana de login con el usuario por defecto que configuramos cuando instalamos la BD *root*, solo tendremos que indicar la contraseña del usuario y accederemos a la herramienta.

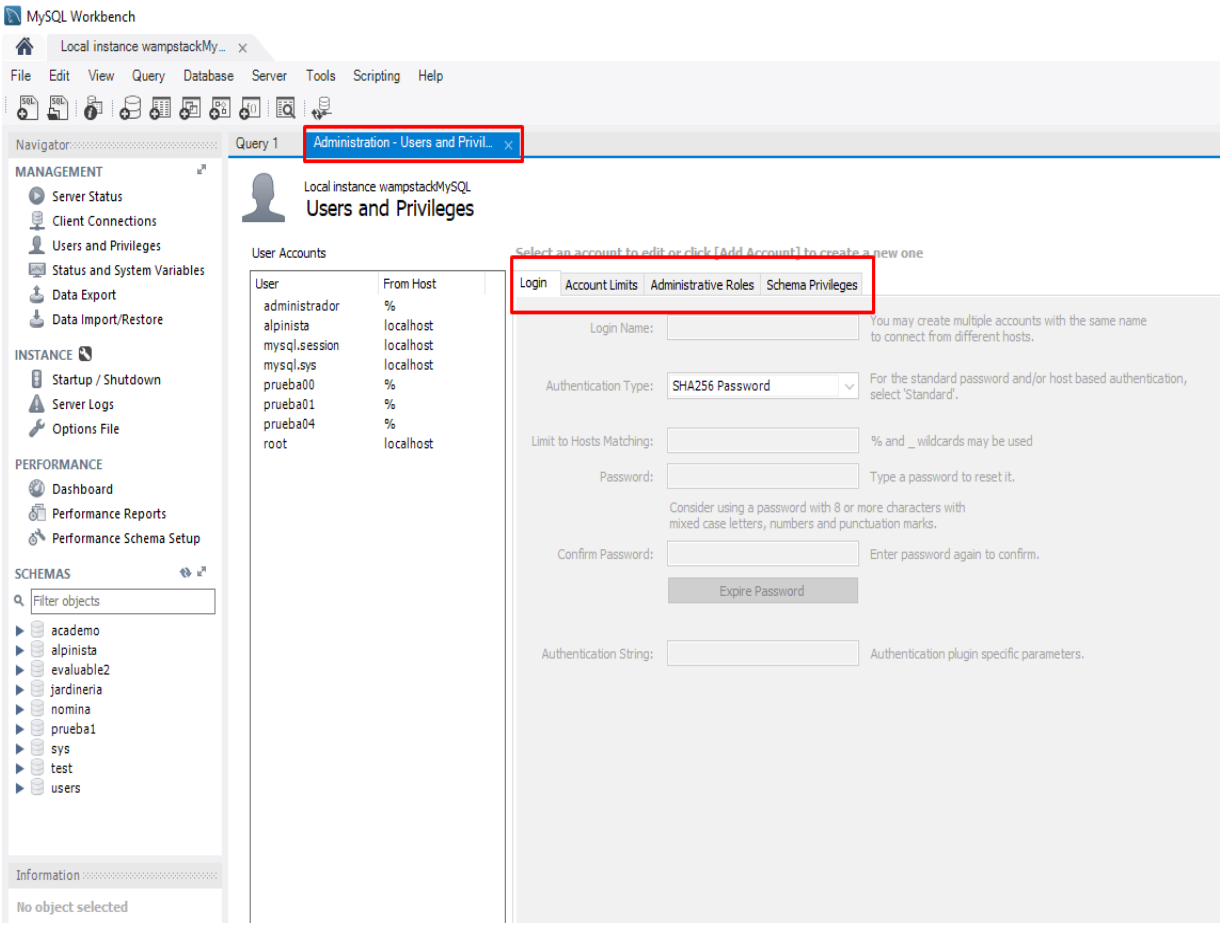


Una vez dentro de la herramienta, para acceder a la configuración de roles debemos pulsar sobre la opción *User and Privileges* que encontraremos a la izquierda de la pantalla, en el árbol de gestión.

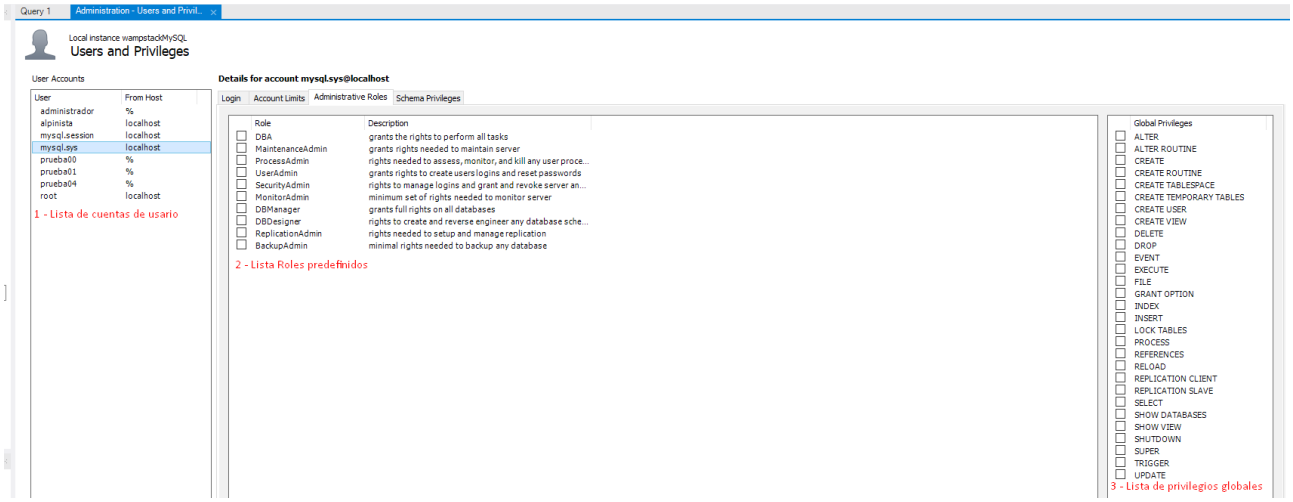


Esta opción nos abrirá en la ventana de trabajo una nueva pestaña donde podremos acceder a toda la configuración de usuarios, roles y privilegios.

Para poder configurar los privilegios, seleccionamos una cuenta de la lista de cuentas de usuario y pulsamos sobre la



pestaña *Administrative Roles*.



Como se puede observar:

1. En la opción 1 elegimos la cuenta de usuario a la que se le va asignar la configuración de rol pertinente.
2. En la opción 2 elegimos el tipo de rol predefinido que sea oportuno, esto activará los privilegios globales que tenga asignados ese rol en la lista de la opción 3.
3. En la opción 3 aparecerán los privilegios globales seleccionados de cada rol. Si hacemos una selección propia de privilegios, aparecerá en el listado de Roles predefinidos un nuevo Rol llamado *Custom*.

7.2 ***Versión 8.0.3 rc (versión Alpha).***

En MySQL 8.0.3 un *ROLE* es una colección de nombres de privilegios. Como las cuentas de usuario, *ROLE* puede tener concedidos privilegios y estos pueden ser revocados. A una cuenta se le pueden otorgar roles, lo que otorga a la cuenta los privilegios asociados a cada *ROLE*. Esto posibilita la asignación de conjuntos de privilegios a las cuentas y provee una conveniente alternativa a la concesión de privilegios individuales.

La siguiente lista resume la capacidad de gestión de *ROLE en MySQL:*

- **CREATE ROLE** y **DROP ROLE** posibilita la creación y la eliminación de *ROLE*.
- **GRANT** y **REVOKE** posibilita al asignación y revocación de privilegios para cuentas de usuario y roles.
- **SHOW GRANTS** muestra los privilegios y roles asignados a cuentas de usuario y roles.
- **SET DEFAULT ROLE** establece que *ROLE* tiene una cuenta de usuario por defecto.
- **SET ROLE** Cambia el ROLE activo en la sesión actual.

- El **CURRENT ROLE()** muestra el *ROLE* actual dentro de la sesión actual.
- Las variables de sistema **mandatory_roles** y **activate all roles on login** permite definir roles obligatorios y la activación automática de los mismos cuando los usuarios inician sesión en el servidor.

7.2.1 Creando roles y concediéndoles privilegios.

Para crear roles se usa *CREATE ROLE*:

Su sintaxis es:

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

Como ejemplo podemos ejecutar la siguiente sentencia:

```
CREATE ROLE 'role1', 'role2', 'role3';
```

Para conceder privilegios se ejecuta *GRANT* usando la misma sintaxis como para asignar privilegios a una cuenta de usuario. Como ejemplo podemos ejecutar la siguiente sentencia:

```
GRANT ALL ON *.* TO 'role1';
GRANT SELECT ON bdnombre.* TO 'role2';
GRANT INSERT, UPDATE, DELETE ON bdnombre.* TO 'role3';
```

Para asignar un *ROLE* a una cuenta de usuario debemos usar sentencias GRAN como se ha mostrado anteriormente.

```
GRANT 'role1' TO 'usuario1';
```

7.2.2 Definiendo roles obligatorios.

Es posible definir como obligatorios nombres de roles como valor de la variable de sistema *mandatory_roles*. El servidor trata un rol obligatorio como otorgado a todos los usuarios, así no necesita ser concedido explícitamente a una cuenta.

Para especificar roles obligatorios al iniciar el servidor, hay que definir la variable de sistema *mandatory_roles* en el archivo de servidor *my.cnf*.

```
[mysqld]
mandatory_roles='role1,role2@localhost,r3@%.example.com'
```

También podemos establecer la variable de sistema *mandatory_roles* en tiempo de ejecución mediante la sentencia:

```
SET PERSIST mandatory_roles = 'role1,role2@localhost,r3@%.example.com';
```

SET PERSIST establece el valor para la instancia en ejecución de MySQL. Para cambiar el valor solo para la instancia en ejecución de MySQL, sin guardarlo para reinicios posteriores, se utilizará la palabra *GLOBAL* sustituyendo a *PERSIST*.

7.2.3 Comprobando los privilegios de los roles.

Para verificar los privilegios otorgados a una cuenta de usuario utilizamos la declaración *SHOW GRANTS*. Por ejemplo:

```
mysql> SHOW GRANTS FOR administrador;
+-----------------------------------------------------------------------+
| Grants for administrador@%                                            |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `administrador`@`%`                             |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `users`.* TO `administrador`@`%` |
| GRANT `ADMIN`@`%` TO `administrador`@`%`                              |
+-----------------------------------------------------------------------+
3 rows in set (0.03 sec)

mysql>
```

Para mostrar también los privilegios de *ROLE* agregamos la declaración *USING* a la sentencia junto con el nombre del *ROLE*. Por ejemplo:

```
mysql> SHOW GRANTS FOR 'administrador' USING 'ADMIN';
+-----------------------------------------------------------------------+
| Grants for administrador@%                                            |
+-----------------------------------------------------------------------+
| GRANT CREATE USER, CREATE ROLE ON *.* TO `administrador`@`%`          |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `users`.* TO `administrador`@`%` |
| GRANT `ADMIN`@`%` TO `administrador`@`%`                              |
+-----------------------------------------------------------------------+
3 rows in set (0.03 sec)

mysql>
```

7.2.4 Activando roles.

Los roles concedidos a un usuario pueden activarse y desactivarse dentro de la sesión. Si un rol concedido está activo dentro de una sesión, sus privilegios estarán activos, de otra forma no. Para ver que roles están activos en la sesión actual usamos la función *CURRENT_ROLE()*.

Por defecto, otorgar un rol a una cuenta o nombrarlo en el valor de la variable de sistema *mandatory_roles* no hace que el rol se active automáticamente. Por ejemplo, si en una sesión en la que no se han activado ningún rol, si invocamos la función *CURRENT_ROLE()*, el resultado es NINGUNO (sin roles activos).

```
mysql> SELECT CURRENT_ROLE();
+----------------+
| CURRENT_ROLE() |
+----------------+
| NONE           |
+----------------+
```

Para especificar que roles deben volver a activarse cada vez que un usuario inicie sesión con el servidor usamos la declaración *SET DEFAULT ROLE*. Por ejemplo:

```
SET DEFAULT ROLE ALL TO 'usuario1';
```

Con esta sentencia, establecemos todos los roles otorgados al *usuario1* como predeterminados.

7.2.5 Revocando roles o privilegios de roles.

Como los roles pueden ser otorgados a una cuenta de usuario, también pueden ser revocados. La sintaxis para revocar un rol es:

```
REVOKE role FROM user;
```

Los roles nombrados en el valor de la variable de sistema *mandatory_roles* no pueden ser revocados.

La declaración *REVOKE* también puede ser aplicada para modificar los privilegios del propio *ROLE*. Esto no solo afecta al rol en si, sino también a las cuentas que tengan otorgado dicho rol. Para hacer esto mostramos el siguiente ejemplo:

```
REVOKE INSERT, UPDATE, DELETE ON bdnombre.* TO 'role3';
```

7.2.6 Eliminando roles.

Para eliminar un *ROLE* usamos *DROP ROLE*:

```
DROP ROLE 'app_read', 'app_write';
```

Eliminando un rol, revocamos todos sus privilegios en las cuentas de usuario que los tengan concedidos. Los roles nombrados en el valor de la variable de sistema *mandatory_roles* no pueden ser eliminados.

8 PLUGINS EN MYSQL.

Existe la posibilidad de instalar en MySQL Plugins que ofrecen la posibilidad de configurar diferentes variable de sistema que permiten al DBA establecer limites y mejorar la seguridad de nuestra información y administración del sistema. Por ejemplo, el plugin Connection-Control permite a los administradores introducir un retraso creciente en la respuesta del servidor a los clientes después de una cierta cantidad de intentos de conexión fallidos consecutivos. Esta capacidad proporciona un elemento de disuasión que ralentiza los posibles ataques de fuerza bruta que puedan sufrir nuestras BD de MySQL. La librería de este plugin contiene dos plugins:

CONNECTION_CONTROL que comprueba las conexiones entrantes y añade un retraso a la respuesta del servidor si es necesario.

CONTROL_FAILED_LOGIN_ATTEMPTS que implementa una tabla en la BD *INFORAMTION_SCHEMA* que expone la información de la monitorización de los intentos de conexión fallidos.

Para configurar el plugin Connection-Control, este expone varias variables de sistema:

CONNECTION_CONTROL_FAILDE_CONNECTIONS_THRESHOLD: El número consecutivo de intentos permitidos a un cliente antes de que el servidor le agregue un retraso para intentos de conexión posterirores.

CONNECTION_CONTROL_MIN_CONNECTION_DELAY: La cantidad de retraso para agregar por cada fallo de conexión consecutiva por encima del umbral.

CONNECTION_CONTROL_MAX_CONNECTION_DELAY: La cantidad máxima de retraso a añadir.

Otros plugins de interés son: Transparent Data Encription (Para la protección de datos críticos), Audit (Provee de una auditoría de controles de seguridad), Firewall (Protección en tiempo real contra amenazas cibernéticas), etc...

9 EJERCICIOS.

9.1 Ejercicio 1

Conectarse como usuario SYSTEM a la base y crear un usuario llamado “administrador” autentificado por la base de datos. Indicar como "tablespace" por defecto USERS y como "tablespace" temporal TEMP; asignar una cuota de 500K en el "tablespace" USERS.

El sistema de BD MySQL no soporta el uso de TABLESPACE, por lo que la declaración para crear el usuario tal cual nos solicita el enunciado nos dará error en MySQL.

```
mysql> CREATE USER administrador
    -> IDENTIFIED BY 'password'
    -> DEFAULT TABLESPACE USERS
    -> TEMPORARY TABLESPACE TEMP
    -> QUOTA 500K ON USERS;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA 500K ON USERS' at line 3
```

Para realizar este ejercicio en MySQL, nos limitaremos a la siguiente declaración:

```
mysql> CREATE USER administrador
    -> IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.10 sec)

mysql>
```

9.2 Ejercicio 2

Abrir una sesión e intentar conectarse como usuario “administrador”, ¿qué sucede?, ¿por qué?.

```
Microsoft Windows [Versión 10.0.16299.309]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.

C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u administrador -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, 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>
```

MySQL, a diferenia de la BD de Oracle, al crear una nueva cuenta de usuario le otorga por defecto el privilegio *USAGE* por lo que, inicialmente, puede establecer conexión con el sistema MySQL pero no podrá acceder a ninguna BD del mismo, ya que no tiene establecidos ningún privilegio a ninguna de estas.

```
mysql> use test
ERROR 1044 (42000): Access denied for user 'administrador'@'%' to database 'test'
```

9.3 Ejercicio 3

Averiguar qué privilegios de sistema, roles y privilegios sobre objetos tiene concedidos el usuario “administrador”.

```
mysql> show grants for administrador;
+----------------------------------------+
| Grants for administrador@%             |
+----------------------------------------+
| GRANT USAGE ON *.* TO 'administrador'@'%' |
+----------------------------------------+
1 row in set (0.00 sec)
```

El privilegio concedido para el usuario *administrador* es el que MySQL concede por defecto: USAGE.

9.4 **Ejercicio 4**

Otorgar el privilegio “CREATE SESSION” al usuario “administrador” e intentar de nuevo la conexión.

En MySQL no existe el privilegio *CREATE SESSION*. El usuario si puede conectarse al sistema MySQL ya que se le concede por defecto el privilegio *USAGE (*este privilegio es equivalente a *CREATE SESSION* de Oracle*)*, lo que no puede es conectarse a ninguna BD contenida en el sistema.

Para que pueda conectarse a una BD contenida en el sistema creamos una BD llamada USERS

```
mysql> CREATE DATABASE users CHARACTER SET utf8 COLLATE utf8_spanish_ci;
Query OK, 1 row affected (0.06 sec)
```

y concedemos los privilegios para que el usuario pueda conectarse a la BD.

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON users.* TO administrador;
Query OK, 0 rows affected (0.01 sec)
```

Ahora el usuario *administrador* si puede conectarse a la BD *users*.

```
C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u administrador -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, 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> SHOW GRANTS FOR CURRENT_USER;
+-------------------------------------------------------------------------+
| Grants for administrador@%                                              |
+-------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'administrador'@'%'                               |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `users`.* TO 'administrador'@'%' |
+-------------------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> use users;
Database changed
mysql>
```

9.5 **Ejercicio 5**

Conectarse como usuario “administrador” y crear un usuario llamado “prueba00” que tenga como "tablespace" por defecto USERS y como "tablespace" temporal TEMP;
Asignar una cuota de 0K en el "tablespace" USERS. ¿Es posible hacerlo?

Tal y como está planteado el enunciado, en MySQL nos debe dar un error si intentamos crear el usuario debido a que, en MySQL las cuentas de usuario se conectan a BD ya creadas y no son estas cuentas por si solas una BD como en Oracle, por lo que no podemos incluir la declaración *TABLESPACE* en una sentencia para crear una cuenta de usuario en MySQL.

```
mysql> CREATE USER prueba00
    -> IDENTIFIED BY 'prueba00'
    -> DEFAULT TABLESPACE users
    -> TEMPORARY TABLESPACE temp
    -> QUOTA 0K ON users;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
QUOTA 0K ON users' at line 3
mysql>
```

La creación del usuario *prueba00* en MySQL sería:

```
mysql> CREATE USER prueba00
    -> IDENTIFIED BY 'prueba00';
ERROR 1227 (42000): Access denied; you need (at least one of) the CREATE USER privilege(s) for this operation
mysql>
```

En este caso, el usuario *administrador* no tiene concedido el privilegio *CREATE USER*, por lo que no puede crear usuarios.

9.6 ***Ejercicio 6***

Conectado como usuario SYSTEM, otorgar el privilegio “create user” al usuario “administrador” y repetir el ejercicio anterior.

CREATE USER es un privilegio de nivel global, por lo que su concesión debe realizarse sobre todo el sistema, por lo que su declaración será:

```
mysql> GRANT CREATE USER ON *.* TO administrador;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

A diferencia que en la BD Oracle, en MySQL hay que indicar en que BD aplica cada privilegio. En este caso al ser de nivel global, en la declaración indicamos que aplique a todas las BD y sus objetos con “*.*”.

Ahora si intentamos crear el usuario *prueba00*, el resultado es satisfactorio.

```
mysql> CREATE USER prueba00
    -> IDENTIFIED BY 'prueba00';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

9.7 ***Ejercicio 7***

Averiguar qué usuarios de la base de datos tienen asignado el privilegio “create user” de forma directa, ¿qué vista debe ser consultada?.

Al igual que en la BD de Oracle, se ejecuta una sentencia a una tabla del sistema donde guarda la información de los privilegios que tiene cada cuenta de usuario. En MySQL a estas tablas se les llama TABLAS GRANT y en concreto para este cometido la consulta se realiza a la tabla ***user*** de la BD ***mysql***. En Oracle se realiza sobre la tabla del sistema ***DBA_SYS_PRIVS***.

```
mysql> SELECT User, Create_user_priv FROM mysql.user WHERE Create_user_priv = 'Y';
+---------------+------------------+
| User          | Create_user_priv |
+---------------+------------------+
| root          | Y                |
| alpinista     | Y                |
| administrador | Y                |
+---------------+------------------+
3 rows in set (0.00 sec)

mysql>
```

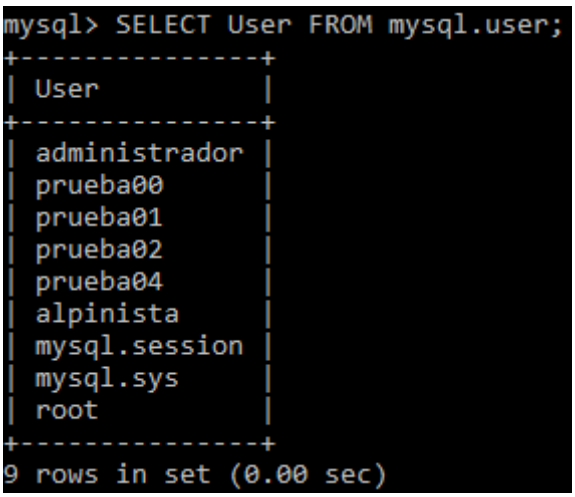
La columna donde se especifica el privilegio CREATE USER es

“Create_user_priv”, donde “Y” es que si tiene concedido dicho privilegio y “N” no lo tiene concedido.

9.8 Ejercicio 8

Hacer lo mismo para el privilegio “create session”.

En MySQL, a diferencia de la BD de Oracle, todos los usuarios creados tienen concedido por defecto el privilegio *USAGE*, que sería el mismo privilegio que *CREATE SESSION* de Oracle, por lo que la sentencia para listar los usuarios será:

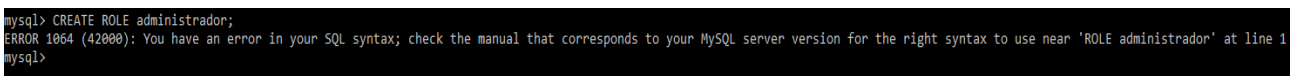


9.9 Ejercicio 9

Estando conectado como usuario “administrador” probar a crear un rol llamado “administrador” ¿qué ocurre?

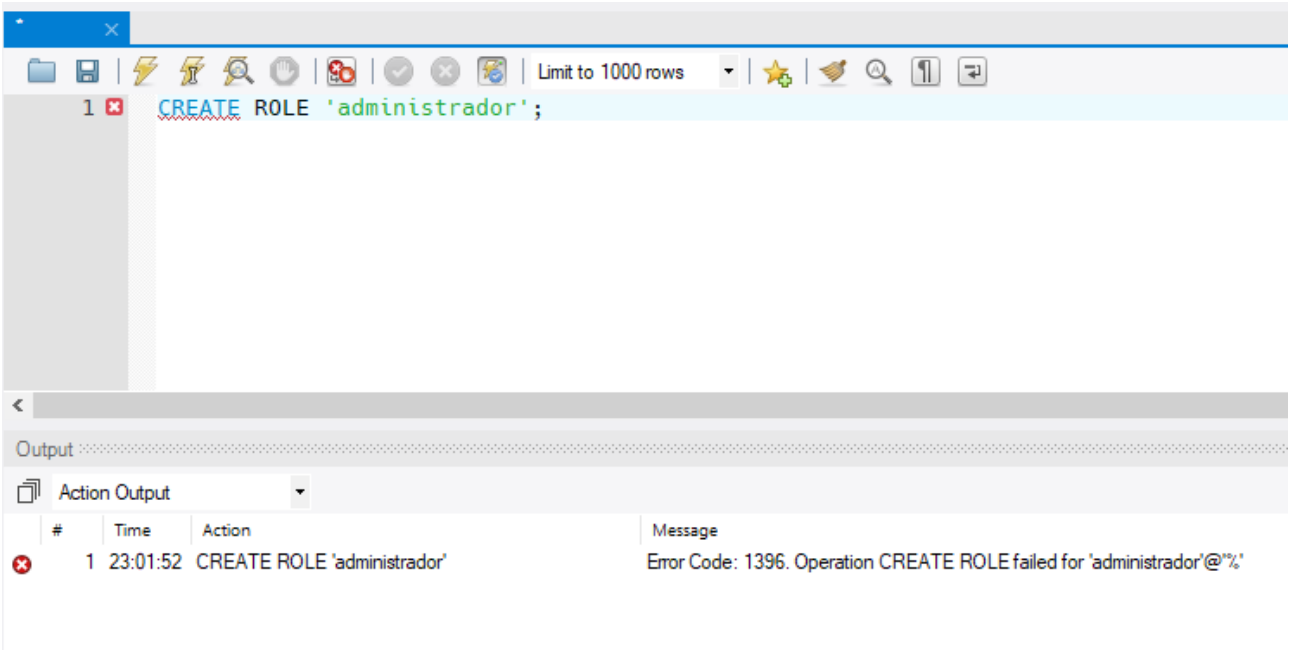
Version MySQL 7.21

MySQL no soporta ni la creación ni el uso de la declaración *ROLE* como en la BD de Oracle. Si intentamos crear un *ROLE* como lo haríamos en Oracle, nos da el siguiente error:



Versión MySQL 8.0.3 rc

Al crear el ROLE “administrador” sale el siguiente error por que existe un usuario que tiene el mismo nombre.

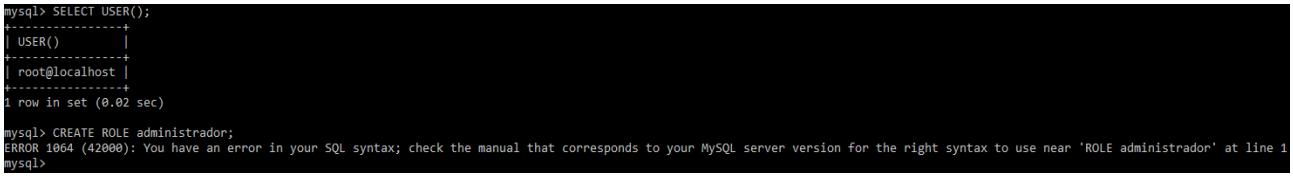


9.10 Ejercicio 10

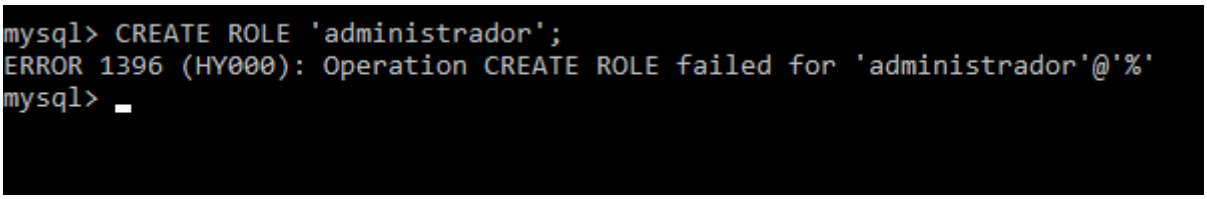
Ídem estando conectado como usuario SYSTEM, ¿qué sucede?, ¿por qué?

Ocurre exactamente lo mismo que en el ejercicio anterior.

Versión MySQL 7.21



Versión MySQL 8.0.3 rc



9.11 Ejercicio 11

Comprobar en el diccionario de datos los usuarios o roles que poseen el privilegio “CREATE ROLE”.

Versión MySQL 7.21

En MySQL, a diferecia de la BD de Oracle, no se puede realizar dicha comprobación debido a que no está soportado ni la creación ni el uso de la declaración *ROLE*. No existe ninguna tabla/vista que guarde la información respecto a *ROLE* y cuentas de usuario. Esta información si se guarda en Oracle en la vista: *DBA_ROLE*

Versión MySQL 8.0.3 rc

En la tabla *mysql.user* el campo *Create_role_priv* indica con “Y” los usuarios que tienen otorgado el privilegio *CREATE ROLE* la sentencia para ver que usuarios tienen concedido este privilegio es:

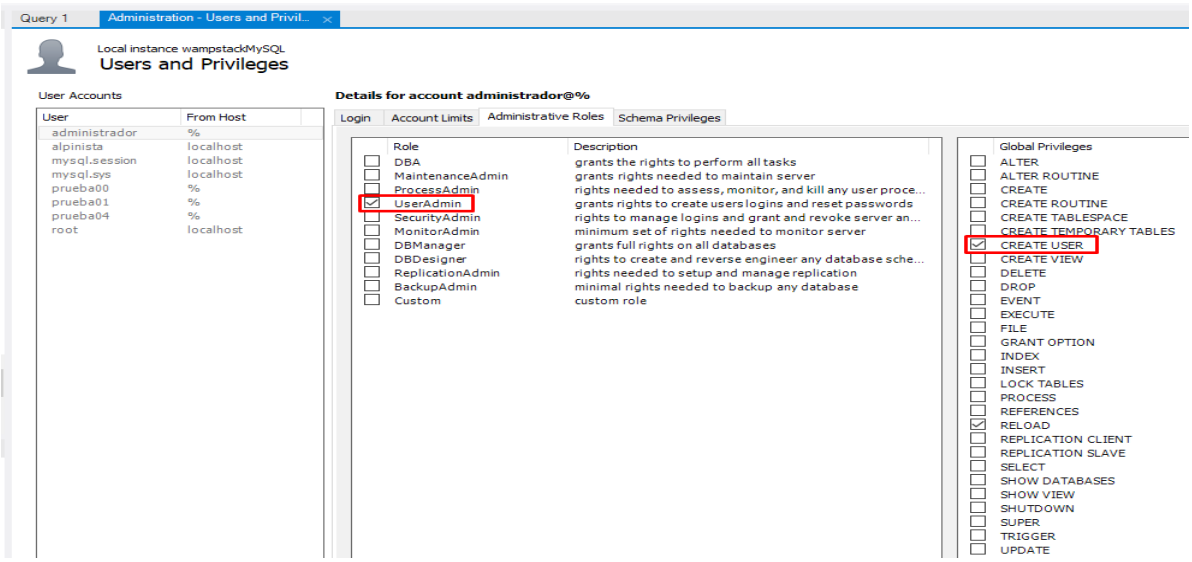
```
mysql> SELECT USER, Create_role_priv FROM user WHERE Create_role_priv = 'Y';
+------+------------------+
| USER | Create_role_priv |
+------+------------------+
| root | Y                |
+------+------------------+
1 row in set (0.00 sec)

mysql>
```

9.12 Ejercicio 12

Crear un rol llamado “ADMIN”, asignarle los privilegios “create session”, “create user” y “CREATE ROLE”. Asignarlo al usuario administrador.

Versión MySQL 7.21

MySQL no soporta el uso de ROLE, pero se puede realizar una configuarción similar a través de la herramienta MySQL Workbench. Desde la opción *Users and Privileges* seleccionamos el usuario administrador y le asisgnamos el Rol predefinido UserAdmin, para aplicar dicho Rol al usuario pulsamos el botón *Apply*.



Versión MySQL 8.0.3 rc

En la sentencia no se ha incluido el ROLE USAGE (*Create Session* de Oracle) debido a que este privilegio ya está otorgado por defecto en la creación de la cuenta de usuario.

```
mysql> CREATE ROLE 'ADMIN';
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> GRANT CREATE USER, CREATE ROLE ON *.* TO 'ADMIN';
Query OK, 0 rows affected (0.11 sec)

mysql>
```

9.13 Ejercicio 13

Consultar los privilegios de sistema que tiene asignados de forma directa el usuario “administrador”, revocarlos y asignarle el rol “admin”.

Versión MySQL 7.21

```
mysql> SHOW GRANTS FOR administrador;
+-----------------------------------------------------------------------+
| Grants for administrador@%                                            |
+-----------------------------------------------------------------------+
| GRANT CREATE USER ON *.* TO 'administrador'@'%'                       |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `users`.* TO 'administrador'@'%' |
+-----------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

En la siguiente imagen podemos ver sobre la tabla mysql.user los privilegios globales que tiene el usuario *administrador*. Esto correspondería a la información que obtenemos de la primera fila del resultado mostrado en la imagen anterior.

```
mysql> SELECT Select_priv, Insert_priv, Update_priv, Delete_priv, Create_priv, Drop_priv, Reload_priv, Shutdown_priv, Process_priv, File_priv, Grant_priv, References_priv, Index_priv, Alter_priv, Show_db_priv, Super_priv, Create_tmp_tabl
e_priv, Lock_tables_priv, Execute_priv, Repl_slave_priv, Repl_client_priv, Create_view_priv, Show_view_priv, Create_routine_priv, Alter_routine_priv, Create_user_priv, Event_priv, Trigger_priv, Create_tablespace_priv FROM mysql.user
    -> WHERE User = 'administrador';
+-------------+--------------+--------------+--------------+-------------+-------------+-------------+---------------+--------------+-----------+-------------+-----------------+------------+------------+--------------+------------+-----------+
| Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Reload_priv | Shutdown_priv | Process_priv | File_priv | Grant_priv | References_priv | Index_priv | Alter_priv | Show_db_priv | Super_priv | Create_tmp_
table_priv | Lock_tables_priv | Execute_priv | Repl_slave_priv | Repl_client_priv | Create_view_priv | Show_view_priv | Create_routine_priv | Alter_routine_priv | Create_user_priv | Event_priv | Trigger_priv | Create_tablespace_priv |
+-------------+--------------+--------------+--------------+-------------+-------------+-------------+---------------+--------------+-----------+-------------+-----------------+------------+------------+--------------+------------+-----------+
| N           | N            | N            | N           | N           | N           | N           | N             | N            | N         | N           | N               | N          | N          | N            | N          | N
          | N                | N            | N               | N                | N                | N              | N                   | N                  | Y                | N          | N            | N                      |
+-------------+--------------+--------------+--------------+-------------+-------------+-------------+---------------+--------------+-----------+-------------+-----------------+------------+------------+--------------+------------+-----------+
1 row in set (0.00 sec)
```

Como se puede observar, el privilegio *CREATE USER* mostrado en la columna *Create_user_priv* tiene el valor “Y”.

En la siguiente imagen podemos ver sobre la tabla mysql.db los privilegios de BD que tiene el usuario *administrador*. Esto correspondería a la información que obtenemos de la segunda fila del resultado de la primera imagen.

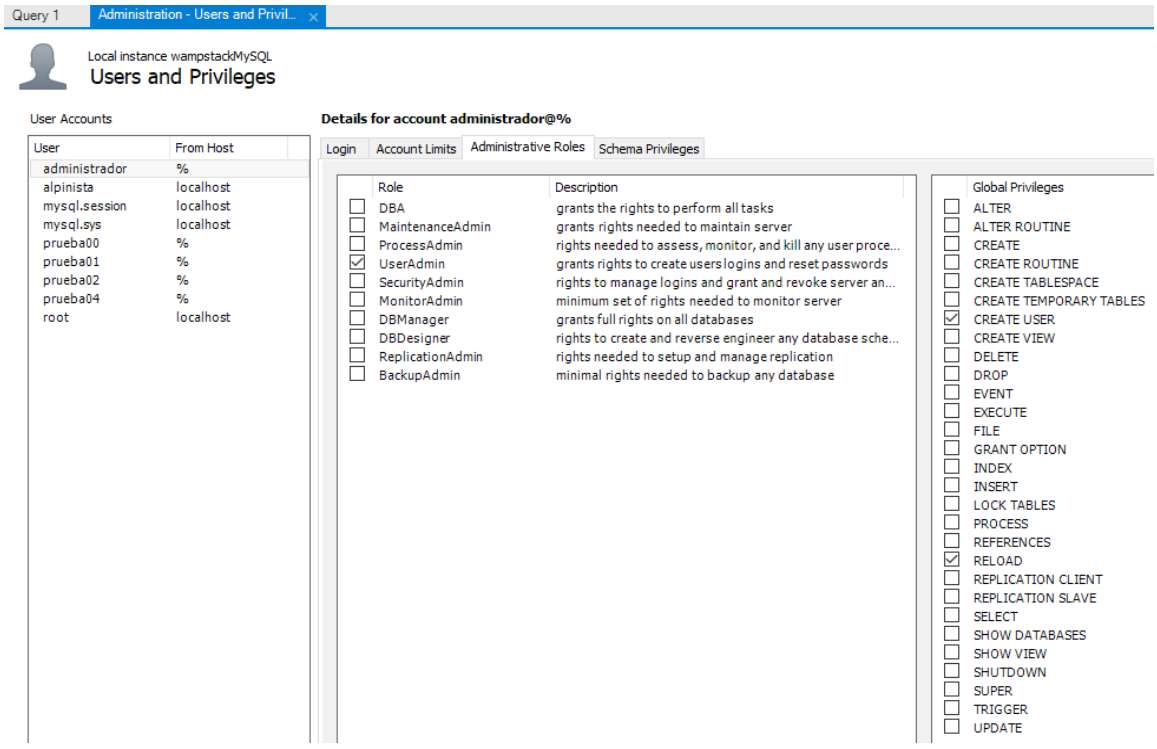
```
mysql> select * from mysql.db where User = 'administrador';
+------+------+---------------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+-------------+-----------------+----------+-----------+-------------+--------------+-----------+-----------+-------------+-----------------+
| Host | Db   | User          | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Grant_priv | References_priv | Index_priv | Alter_priv | Create_tmp_table_priv | Lock_tables_priv | Create_view_priv | Sho
w_view_priv | Create_routine_priv | Alter_routine_priv | Execute_priv | Event_priv | Trigger_priv |
+------+------+---------------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+-------------+-----------------+----------+-----------+-------------+--------------+-----------+-----------+-------------+-----------------+
| %    | users | administrador | Y           | Y           | Y           | Y           | N           | N         | N           | N             | N           | N               | N        | N         | N           | N            | N         | N         | N           | N
           | N                   | N                  | N            | N          | N            |
+------+------+---------------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+-------------+-----------------+----------+-----------+-------------+--------------+-----------+-----------+-------------+-----------------+
```

Podemos ver que están los privilegios SELECT (Select_priv), INSERT (Insert_priv), UPDATE (Update_priv) y DELETE (Delete_priv) con valor “Y” y corresponden a la Base de Datos (Db) users.

Para revocar los privilegios de sistema, la sentencia a ejecutar es:

```
mysql> REVOKE CREATE USER ON *.* FROM administrador;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Mediante MySQL Workbench se le asigna el Rol predefinido *UserAdmin* al usuario *administrador*.



Versión MySQL 8.0.3 rc

Para ver los privilegios:

```
mysql> SHOW GRANTS FOR administrador;
+-----------------------------------------------------------------------+
| Grants for administrador@%                                            |
+-----------------------------------------------------------------------+
| GRANT CREATE USER ON *.* TO 'administrador'@'%'                       |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `users`.* TO 'administrador'@'%' |
+-----------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Para revocar los privilegios:

```
mysql> REVOKE CREATE USER ON *.* FROM administrador;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Para la asignación del ROLE “ADMIN”:

```
mysql> GRANT 'ADMIN' TO 'administrador';
Query OK, 0 rows affected (0.09 sec)

mysql>
```

Tras la concesión del *ROLE* “ADMIN” a la cuenta de usuario debemos activarlo para se apliquen los privilegios del *ROLE* en la cuenta de usuario. Este paso no se realiza en la BD de Oracle, ya que en Oracle los privilegios de los roles aplican en las cuentas de usuario en el mismo momento en que son concedidos.

```
mysql> SET DEFAULT ROLE ALL TO 'administrador';
Query OK, 0 rows affected (0.09 sec)
```

9.14 Ejercicio 14

Crear, conectado como SYSTEM, un usuario llamado “prueba01” autenticado por prueba01 al que no se le asigne "tablespace" por defecto ni temporal.

Tal y como está planteado el enunciado, la declaración para esta sentencia es la misma en MySQL como en Oracle, con la salvedad que en MySQL la cadena de texto donde indicamos la contraseña en *IDENTIFIED BY* tiene que estar entre comillas simples.

```
mysql> CREATE USER prueba01
    -> IDENTIFIED BY 'prueba01';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

9.15 Ejercicio 15

Crear un usuario llamado “prueba02” autenticado por prueba02, asignando como "tablespace" por defecto NOMINA y como "tablespace" temporal TEMP_NOMINA (no se le asignara cuota en NOMINA).

Como ya hemos indicado en ejercicios anteriores, MySQL utiliza la declaración TABLESPACE para la creación de usuarios, por lo que la sentencia será la siguiente:

1. Creamos DB *nomina*.

```
mysql> CREATE DATABASE nomina CHARACTER SET utf8 COLLATE utf8_spanish_ci;
Query OK, 1 row affected (0.00 sec)

mysql>
```

2. Concedemos los privilegios SELECT, INSERT, UPDATE, DELETE, CREATE, DROP sobre la BD nomina al mismo tiempo que creamos el usuario *prueba02*.

```
mysql> GRANT CREATE, DROP, SELECT, INSERT, UPDATE, DELETE ON nomina.* TO prueba02 IDENTIFIED BY 'prueba02';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>
```

Esta era la forma en la que se creaban los usuarios en versiones de MySQL anteriores a la 5.0.2 donde aparece por primera vez la declaración *CREATE USER*. Mediante esta forma, en una misma sentencia podemos crear una cuenta de usuario y asignarle los privilegios que sean oportunos.

9.16 Ejercicio 16

Asignar al usuario “prueba01” los "tablespace" ACADEMO y TEMP_ACADEMO como "tablespace" de trabajo y temporal respectivamente (sin especificar cuota).

Como ya hemos comentado anteriormente, en MySQL no se puede asignar ningún TABLESPACE a una cuenta de usuario, por lo que crearemos una BD llamada *academo* y le otorgaremos al dicho usuario los privilegios SELECT, INSERT, UPDATE, DELETE, CREATE, DROP sobre la BD.

```
mysql> CREATE DATABASE academo CHARACTER SET utf8 COLLATE utf8_spanish_ci;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT CREATE, DROP, SELECT, INSERT, UPDATE, DELETE ON academo.* TO prueba01;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

9.17 Ejercicio 17

Consultar en las vistas correspondientes los "tablespace" y la cuota en cada uno de ellos que tiene los usuarios “prueba01” y “prueba02”.

A diferencia de la BD de Oracle, MySQL no soporta el uso de *TABLESPACE* del mismo modo que en Oracle y por ende tampoco utiliza la declaración *QUOTA*, por lo que no está disponible las vista a las que se refiere el enunciado: *DBA_TS_QUOTAS*. En MySQL se podría consultar en que BD tienen concedidos privilegios los usuarios *prueba01* y *prueba02* de la siguiente forma:

```
mysql> SELECT Db, User FROM mysql.db WHERE User IN('prueba01', 'prueba02');
+---------+----------+
| Db      | User     |
+---------+----------+
| academo | prueba01 |
| nomina  | prueba02 |
+---------+----------+
2 rows in set (0.00 sec)
```

9.18 Ejercicio 18

Crear un rol llamado “CONEXION” y asignarle el permiso “CREATE SESSION”.

Versión MySQL 7.21

MySQL no soporta ni la creación ni el uso de *ROLE*, tampoco existe el privilegio *CREATE SESSION* ya que por defecto MySQL si permite crear una sesión a un usuario mediante el privilegio *USAGE*, que es asignado por defecto al crear una cuenta de usuario.

Versión 8.0.3 rc

Para crear el ROLE “CONEXION”:

```
mysql> CREATE ROLE 'CONEXION';
Query OK, 0 rows affected (0.08 sec)
```

Para la asignación de privilegios:

```
mysql> GRANT USAGE ON *.* TO 'CONEXION';
Query OK, 0 rows affected (0.06 sec)

mysql>
```

9.19 Ejercicio 19

Asignar el rol “CONEXION” a los usuarios “prueba00”, “prueba01” y “prueba02”.

Versión MySQL 7.21

Toda cuenta de usuario creada en MySQL, por defecto, tiene asignado el privilegio *USAGE*, que es el equivalente al privilegio *CREATE SESSION* de Oracle, por ello no es necesario y tampoco posible la asignación de dicho rol a los usuarios indicados.

Versión MySQL 8.0.3 rc

```
mysql> GRANT 'CONEXION' TO 'prueba01', 'prueba02', 'prueba00';
Query OK, 0 rows affected (0.13 sec)

mysql>
```

Tras la concesión del *ROLE* “CONEXION” a las cuentas de usuario debemos activarlo para se apliquen los privilegios del *ROLE* en la cuenta de usuario. Este paso no se realiza en la BD de Oracle, ya que en Oracle los privilegios de los roles aplican en las cuentas de usuario en el mismo momento en que son concedidos.

```
mysql> SET DEFAULT ROLE ALL TO 'prueba00', 'prueba01', 'prueba02';
Query OK, 0 rows affected (0.14 sec)
```

9.20 Ejercicio 20

Comprobar en la vista correspondiente cuales son los roles asignados a los usuarios “prueba00”, “prueba01” y “prueba02”.

Versión MySQL 7.21

A diferencia de la BD de Oracle, MySQL no soporta ni la creación ni el uso de *ROLE*, por lo que no se puede comprobar al no existir vista equivalente en MySQL a la vista indicada en el enunciado: *DBA_ROLE_PRIVS* de Oracle.

Versión 8.0.3 rc

En la versión 8.0.3 rc existe la tabla role_edges de la BD mysql donde se guarda la información referente a cuentas de usuario y ROLE.

```
mysql> SELECT * FROM mysql.role_edges WHERE TO_USER IN('prueba00', 'prueba01', 'prueba02');
+-----------+-----------+---------+----------+-------------------+
| FROM_HOST | FROM_USER | TO_HOST | TO_USER  | WITH_ADMIN_OPTION |
+-----------+-----------+---------+----------+-------------------+
| %         | CONEXION  | %       | prueba00 | N                 |
| %         | CONEXION  | %       | prueba01 | N                 |
| %         | CONEXION  | %       | prueba02 | N                 |
```

*9.21 **Ejercicio 21***

Conectarse como usuario “prueba01” y crear la tabla siguiente en el "tablespace" ACADEMO:

CREATE TABLE CODIGOS
(CODIGO varchar(3),
DESCRIPCION varchar(20))
TABLESPACE ACADEMO
STORAGE (INITIAL 64K
NEXT 64K
MINEXTENTS 5
MAXEXTENTS 10);

¿Es posible hacerlo?, ¿falta algún permiso?

Lo primero que voy hacer en este ejercicio es revocar los privilegios *CREATE* y *DROP* por que entiendo que no los debí incluir cuando creé los usuarios, ya que actualmente pueden crear tablas y el objetivo de este ejercicio es mostrar que no pueden hacerlo. Para revocar los privilegios, la sentencia será la siguiente:

```
mysql> REVOKE CREATE, DROP ON academo.* FROM prueba01;
Query OK, 0 rows affected (0.00 sec)
```

La creación de la tabla, tal cual está planteada en el enunciado no es posible ejecutarla en MySQL debido al uso de la delcaraciones *TABLESPACE*, *STORAGE*, etc. en la sentencia, la cual generará un error:

```
mysql> CREATE TABLE CODIGOS
    -> (CODIGO varchar(3),
    -> DESCRIPCION varchar(20))
    -> TABLESPACE ACADEMO
    -> STORAGE (INITIAL 64K
    -> NEXT 64K
    -> MINEXTENTS 5
    -> MAXEXTENTS 10);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(INITIAL 64K
NEXT 64K
MINEXTENTS 5
MAXEXTENTS 10)' at line 5
mysql>
```

Para crear la tabla en MySQL la sentencia a ejecutar sería la siguiente:

```
mysql> CREATE TABLE Codigos(Codigo VARCHAR(3),
    -> Descripcion VARCHAR(20))ENGINE=InnoDB;
ERROR 1142 (42000): CREATE command denied to user 'prueba01'@'localhost' for table 'codigos'
mysql>
```

En este caso, nos genera un error debido a que el usuario no tiene los privilegios para poder crear la tabla. El privilegio para poder hacerlo es *CREATE* a diferencia de la BD de Oracle que es *CREATE TABLE*.

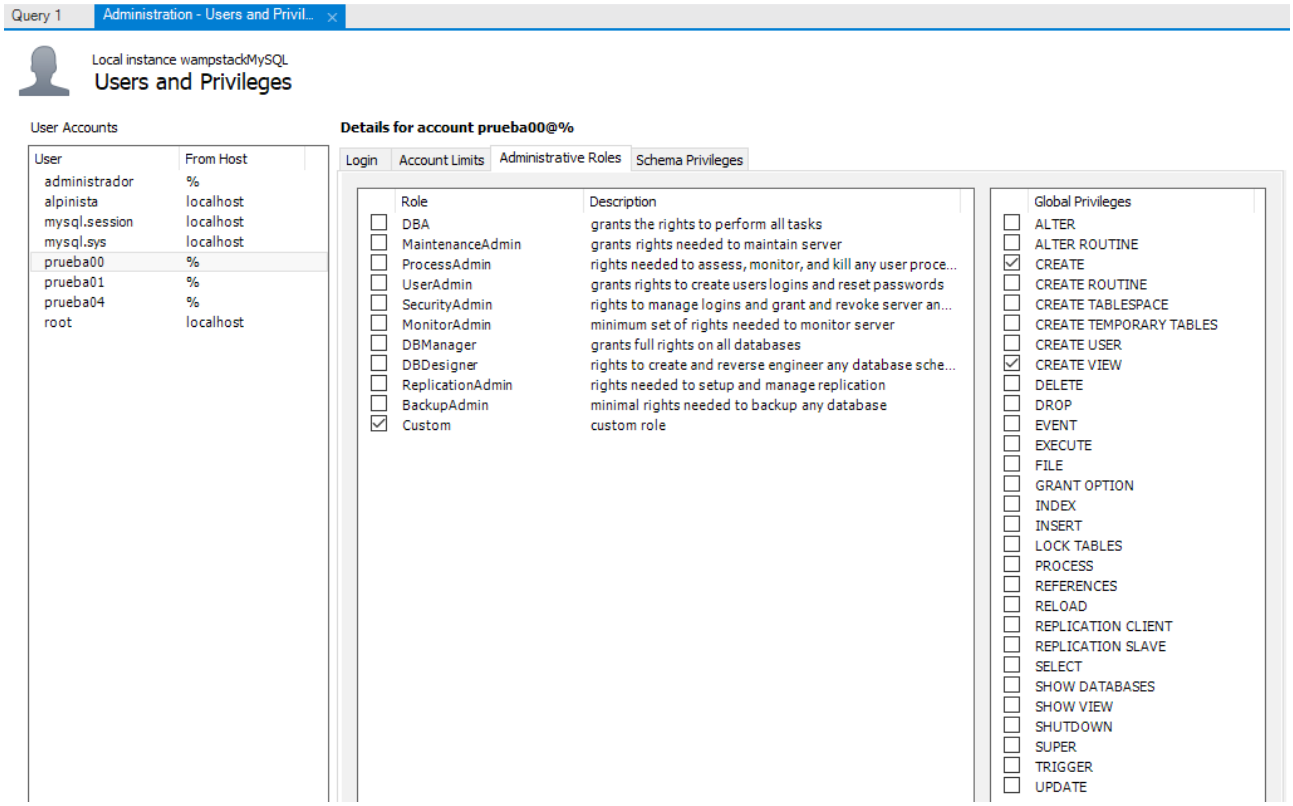
9.22 Ejercicio 22

Crear un rol llamado “DESARROLLO” y asignarle los permisos "CREATE SEQUENCE", "CREATE SESSION", "CREATE SYNONYM", "CREATE TABLE" y "CREATE VIEW". Asignar el rol “DESARROLLO” a los usuarios “prueba00”, “prueba01” y “prueba02”.

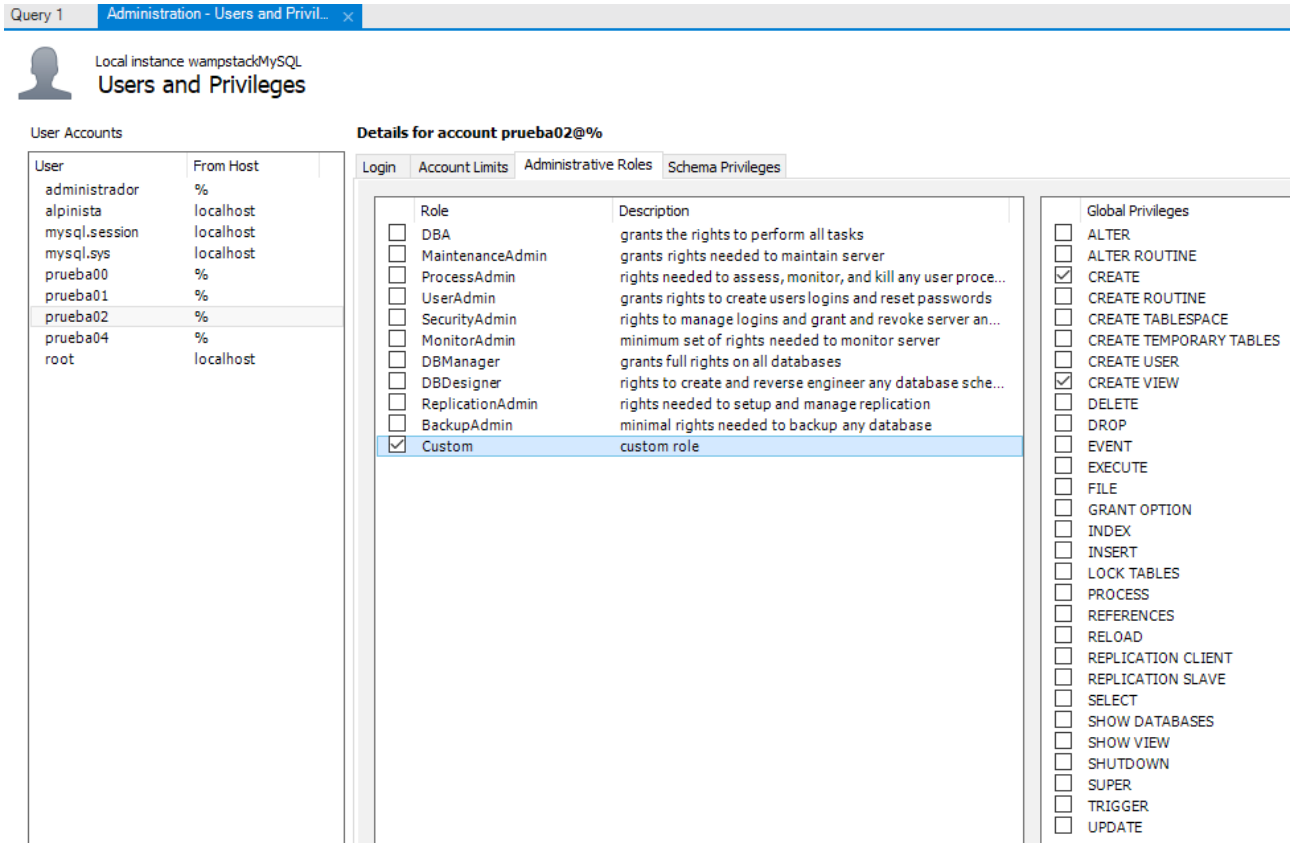
Versión MySQL 7.21

Desde MySQL Workbench no hay un Rol predefinido equivalente al planteado en el enunciado, por lo que los privilegios equivalentes en MySQL a los del enunciado de Oracle son: CREATE (CREATE TABLE), CREATE VIEW (CREATE VIEW), estos privilegios se añadirían al Rol llamado *CUSTOM*.

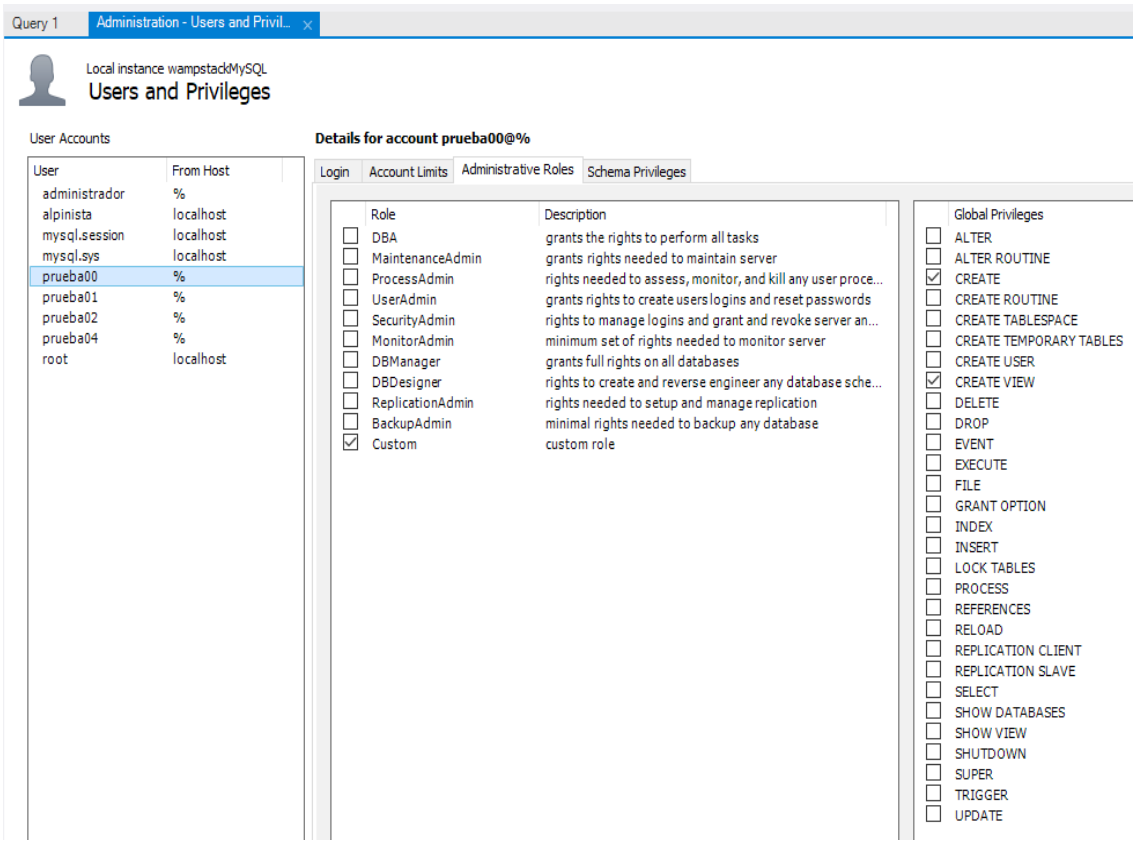
Para el usuario *prueba01*:



Para el usuario *prueba02*:



Para el usuario *prueba00*:



Versión MySQL 8.0.3 rc

La sentencia para crear el ROLE “DESARROLLO” es:

```
mysql> CREATE ROLE 'DESARROLLO';
Query OK, 0 rows affected (0.15 sec)

mysql>
```

Para agregar privilegios al ROLE “DESARROLLO” la sentencia es:

```
mysql> GRANT CREATE, CREATE VIEW ON *.* TO 'DESARROLLO';
Query OK, 0 rows affected (0.14 sec)

mysql>
```

La asignación de dicho ROLE a los usuarios es:

```
mysql> GRANT 'DESARROLLO' TO 'prueba00', 'prueba01', 'prueba02';
Query OK, 0 rows affected (0.10 sec)

mysql>
```

Tras la concesión del *ROLE* “DESARROLLO” a las cuentas de usuario debemos activarlo para se apliquen los privilegios del *ROLE* en la cuenta de usuario. Este paso no se realiza en la BD de Oracle, ya que en Oracle los privilegios de los roles aplican en las cuentas de usuario en el mismo momento en que son concedidos.

```
mysql> SET DEFAULT ROLE ALL TO 'prueba00', 'prueba01', 'prueba02';
Query OK, 0 rows affected (0.14 sec)
```

9.23 Ejercicio 23

Volver a conectarse como usuario “prueba01” y crear la tabla anterior (del ejercicio 21) en el "*tablespace*" ACADEMO que es su *tablespace* por defecto.

Como ya hemos comentado en el ejercicio 21, tal y como está planteada la creación de la tabla en MySQL no se puede ejecutar. La forma de poder crear la misma tabla en MySQL sería:

```
mysql> CREATE TABLE Codigos(Codigo VARCHAR(3),
    -> Descripcion VARCHAR(20))ENGINE=InnoDB;
Query OK, 0 rows affected (0.33 sec)

mysql> select user();
+--------------------+
| user()             |
+--------------------+
| prueba01@localhost |
+--------------------+
1 row in set (0.00 sec)
```

La selección del usuario conectado la expongo para demostrar que el usuario que crea la tabla es el que se especifica en el enunciado.

9.24 Ejercicio 24

Asignar cuota ilimitada al usuario “prueba01” en el "tablespace" ACADEMO. Volver a repetir el ejercicio 23.

A diferencia de la BD Oracle, MySQL no soporta el uso de la declaración *TABLESPACE* y por ende tampoco soporta la declaración *QUOTA*.

9.25 Ejercicio 25

Asignar cuota ilimitada al usuario “prueba02” en el "tablespace" NOMINA.

A diferencia de la BD de Oracle, MySQL no soporta el uso de la declaración *TABLESPACE* y por ende tampoco soporta la declaración *QUOTA*.

9.26 Ejercicio 26

Obtener información sobre roles, privilegios de sistema, "tablespace" y cuotas para los usuarios “prueba00”, “prueba01” y “prueba02”.

Versión MySQL 7.21

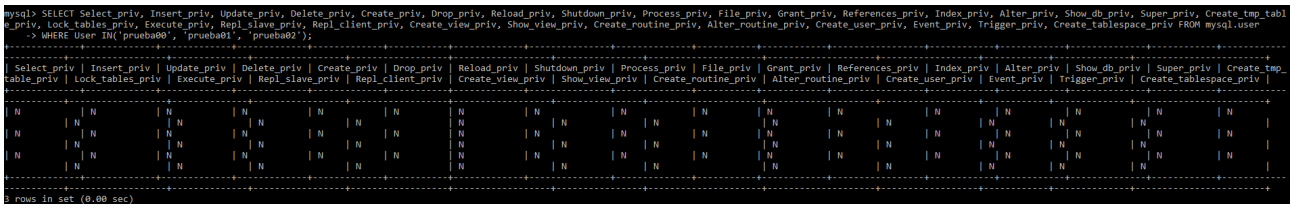
A diferencia que en la BD de Oracle. En MySQL no se puede obtener información de *TABLESPACE*, *QUOTAS* y *ROLE*, pero si se puede obtener información de los privilegios concedidos a las cuentas de usuario en sus distintos niveles:

De forma general (Muetra privilegios administrativos y de BD):

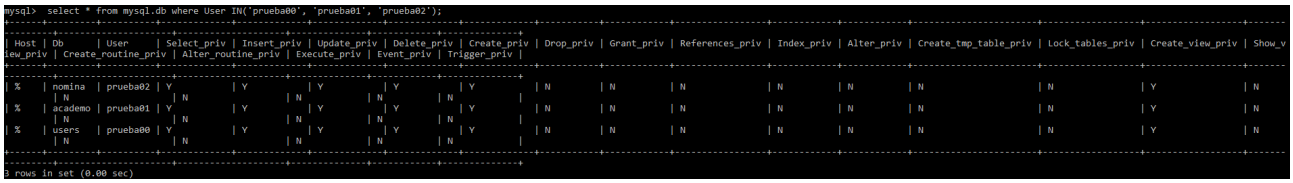
```
mysql> SHOW GRANTS FOR prueba01;
+-----------------------------------------------------------------------------------------------+
| Grants for prueba01@%                                                                         |
+-----------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'prueba01'@'%'                                                          |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE VIEW ON `academo`.* TO 'prueba01'@'%' |
+-----------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

```
mysql> SHOW GRANTS FOR prueba02;
+-----------------------------------------------------------------------------------------------+
| Grants for prueba02@%                                                                         |
+-----------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'prueba02'@'%'                                                          |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE VIEW ON `nomina`.* TO 'prueba02'@'%' |
+-----------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Privilegios administrativos:



Privilegios de BD:



Versión MySQL 8.0.3 rc

De forma general:

```
mysql> show grants for 'prueba00' using 'CONEXION', 'DESARROLLO';
+-----------------------------------------------------------------------+
| Grants for prueba00@%                                                 |
+-----------------------------------------------------------------------+
| GRANT CREATE, CREATE VIEW ON *.* TO `prueba00`@`%`                    |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `users`.* TO `prueba00`@`%`   |
| GRANT `CONEXION`@`%`,`DESARROLLO`@`%` TO `prueba00`@`%`               |
+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

```
mysql> show grants for 'prueba01' using 'CONEXION', 'DESARROLLO';
+-----------------------------------------------------------------------+
| Grants for prueba01@%                                                 |
+-----------------------------------------------------------------------+
| GRANT CREATE, CREATE VIEW ON *.* TO `prueba01`@`%`                    |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `academo`.* TO `prueba01`@`%` |
| GRANT `CONEXION`@`%`,`DESARROLLO`@`%` TO `prueba01`@`%`               |
+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

```
mysql> show grants for 'prueba02' using 'CONEXION', 'DESARROLLO';
+-----------------------------------------------------------------------+
| Grants for prueba02@%                                                 |
+-----------------------------------------------------------------------+
| GRANT CREATE, CREATE VIEW ON *.* TO `prueba02`@`%`                    |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `nomina`.* TO `prueba02`@`%`  |
| GRANT `CONEXION`@`%`,`DESARROLLO`@`%` TO `prueba02`@`%`               |
+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

Mostramos información guardada en las *TABLAS GRANT*:

En la versión 8.0.3 incluimos la tabla mysql.role_edges a las anteriores. En esta sentencia he acotado los privilegios administrativos, solicitando solo los más importantes o que más aplican con los conceptos que estamos manejando. Como se genera una tabla con muchos campos, hacemos que la presentación de los datos se presente tabulada verticalmente usando “\G” al final de la sentencia. Esta es:

```
mysql> SELECT a.TO_USER AS USUARIO, a.FROM_USER AS 'ROLE', b.Select_priv, b.Insert_priv, b.Update_priv, b.Delete_priv, b.Create_priv, b.Drop_priv, b.Alter_priv, b.Creat
e_view_priv, b.Create_user_priv, b.Create_role_priv, b.Drop_role_priv, c.DB AS 'BD CON ACCESO' FROM mysql.role_edges a, mysql.user b, mysql.db c
    -> WHERE a.TO_USER = b.User AND a.TO_USER = c.User AND a.TO_USER IN('prueba00', 'prueba01', 'prueba02')\G;
*************************** 1. row ***************************
        USUARIO: prueba00
           ROLE: CONEXION
    Select_priv: N
    Insert_priv: N
    Update_priv: N
    Delete_priv: N
    Create_priv: N
      Drop_priv: N
     Alter_priv: N
Create_view_priv: N
Create_user_priv: N
Create_role_priv: N
  Drop_role_priv: N
   BD CON ACCESO: users
*************************** 2. row ***************************
        USUARIO: prueba00
           ROLE: DESARROLLO
    Select_priv: N
    Insert_priv: N
    Update_priv: N
    Delete_priv: N
    Create_priv: N
      Drop_priv: N
     Alter_priv: N
Create_view_priv: N
Create_user_priv: N
Create_role_priv: N
  Drop_role_priv: N
   BD CON ACCESO: users
```

```
*************************** 3. row ***************************
        USUARIO: prueba01
           ROLE: CONEXION
    Select_priv: N
    Insert_priv: N
    Update_priv: N
    Delete_priv: N
    Create_priv: N
      Drop_priv: N
     Alter_priv: N
Create_view_priv: N
Create_user_priv: N
Create_role_priv: N
  Drop_role_priv: N
   BD CON ACCESO: academo
*************************** 4. row ***************************
        USUARIO: prueba01
           ROLE: DESARROLLO
    Select_priv: N
    Insert_priv: N
    Update_priv: N
    Delete_priv: N
    Create_priv: N
      Drop_priv: N
     Alter_priv: N
Create_view_priv: N
Create_user_priv: N
Create_role_priv: N
  Drop_role_priv: N
   BD CON ACCESO: academo
```

```
*************************** 5. row ***************************
        USUARIO: prueba02
            ROLE: CONEXION
    Select_priv: N
    Insert_priv: N
    Update_priv: N
    Delete_priv: N
    Create_priv: N
      Drop_priv: N
     Alter_priv: N
Create_view_priv: N
Create_user_priv: N
Create_role_priv: N
  Drop_role_priv: N
   BD CON ACCESO: nomina
*************************** 6. row ***************************
        USUARIO: prueba02
            ROLE: DESARROLLO
    Select_priv: N
    Insert_priv: N
    Update_priv: N
    Delete_priv: N
    Create_priv: N
      Drop_priv: N
     Alter_priv: N
Create_view_priv: N
Create_user_priv: N
Create_role_priv: N
  Drop_role_priv: N
   BD CON ACCESO: nomina
6 rows in set (0.00 sec)
```

*9.27 **Ejercicio 27***

Conectarse como usuario "prueba01" y modificar su clave, ¿es posible?.

Al igual que en la BD de Oracle, un usuario puede modificar su propia clave de acceso. La sentencia es en ambos casos la misma:

```
mysql> ALTER USER prueba01 IDENTIFIED BY 'prueba1';
Query OK, 0 rows affected (0.00 sec)

mysql> select user();
+--------------------+
| user()             |
+--------------------+
| prueba01@localhost |
+--------------------+
1 row in set (0.00 sec)
```

La selección del usuario conectado la expongo para demostrar que el usuario que modifica la contraseña es el que se especifica en el enunciado.

*9.28 **Ejercicio 28***

Averiguar qué usuarios o roles de base de datos tienen asignado el privilegio ALTER USER.

En MySQL los usuario que pueden ejecutar la declaración *ALTER USER* son los que tienen concedido el privilegio *CREATE USER*. Para mostrar los usuario que tienen dicho privilegio mostraremos las columnas User y Create_user_priv de la tabla *user* de la BD *mysql*, con la condición de que Create_user_priv tiene que tener valor "Y".

La sentencia sería la siguiente:

```
mysql> SELECT User, Create_user_priv FROM mysql.user WHERE Create_user_priv = 'Y';
+---------------+------------------+
| User          | Create_user_priv |
+---------------+------------------+
| root          | Y                |
| alpinista     | Y                |
| administrador | Y                |
+---------------+------------------+
3 rows in set (0.00 sec)

mysql>
```

9.29 **Ejercicio 29**

Abrir una sesión con el usuario “administrador” y otra con el usuario “prueba02”. Siendo el usuario “administrador”, intentar borrar el usuario “prueba02”.

A diferencia de la BD de Oracle, en MySQL un usuario con privilegios para poder crear o borrar una cuenta de usuario, puede borrar una cuenta de un usuario que está con una sesión abierta. Esto es debido a que la eliminación del usuario no implica perdida alguna de información de la BD o la perdida de la propia BD, como si ocurre en Oracle. Una vez se desconecte usuario de la cuenta borrada, ya no podrá volver a conectarse.

```
mysql> select user();
+-------------------------+
| user()                  |
+-------------------------+
| administrador@localhost |
+-------------------------+
1 row in set (0.00 sec)

mysql> DROP USER prueba02;
Query OK, 0 rows affected (0.00 sec)
```

9.30 Ejercicio 30

Consultar qué perfiles tiene asignados cada usuario de la base de datos.

MySQL no soporta el uso de *PROFILE*, por lo que no dispone de una vista como *DBA_PROFILES* de la BD de Oracle.

9.31 Ejercicio 31

Asignar el permiso “CREATE PROFILE” al rol ADMIN.

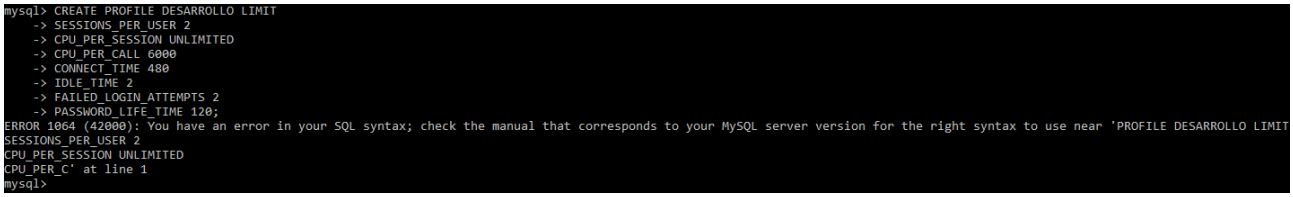
MySQL no soporta el uso de *PROFILE* y *ROLE* además de no existir el privilegio *CREATE PROFILE*, por lo que no se puede asignar dicho privilegio.

*9.32 **Ejercicio 32***

Crear un perfil llamado “DESARROLLO” con las siguientes especificaciones:

Sessions_per_user	2
Cpu_per_session	unlimited
Cpu_per_call	6000
Connect_time	480
Idle_time	2
Failed_login_attempts	2
Password_life_time	120

MySQL no soporta el uso de *PROFILE*, por lo al intentar crearlo no generará un error.



*9.33 **Ejercicio 33***

Asignar el perfil anterior a los usuarios “prueba00”, “prueba01”y “prueba02”.

A diferencia de la BD de Oracle, MySQL no soporta el uso de *PROFILE*, por lo que no se puede asignar a ninguna cuenta de usuario. MySQL permite la configuración de recursos del sistema y establecer límites utilizando cuatro variables de sistema:

MAX_QUERIES_PER_HOUR que establece el número de consultas que puede realizar un usuario por hora.
MAX_UPDATES_PER_HOUR que establece el número de actalizaciones que puede realizar un usuario por hora.
MAX_CONNECTIONS_PER_HOUR que establece el número de conexiones que puede realizar un usuario por hora.
MAX_USER_CONNECTIONS que establece el número de conexiones simultáneas que puede realizar un usuario.

También permite la configuración para la expedición de las contraseñas de usuario.

Para que los usuario solicitados tengan establecidos los límites de recursos del sistema la sentencia sería:

```
mysql> ALTER USER prueba00 WITH MAX_QUERIES_PER_HOUR 0
    -> MAX_UPDATES_PER_HOUR 0
    -> MAX_CONNECTIONS_PER_HOUR 0
    -> MAX_USER_CONNECTIONS 2
    -> PASSWORD EXPIRE INTERVAL 180 DAY;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER USER prueba01 WITH MAX_QUERIES_PER_HOUR 0
    -> MAX_UPDATES_PER_HOUR 0
    -> MAX_CONNECTIONS_PER_HOUR 0
    -> MAX_USER_CONNECTIONS 2
    -> PASSWORD EXPIRE INTERVAL 180 DAY;
Query OK, 0 rows affected (0.00 sec)
```

9.34 Ejercicio 34

Intentar la conexión dos veces como usuario "prueba01" fallando la contraseña, ¿qué sucede?. Comprobar si la cuenta ha sido bloqueada en la vista de base de datos correspondiente.

Tras varias conexiones fallidas con el usuario *prueba01*

```
C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u prueba01 -p
Enter password: *******
ERROR 1045 (28000): Access denied for user 'prueba01'@'localhost' (using password: YES)

C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u prueba01 -p
Enter password: *************
ERROR 1045 (28000): Access denied for user 'prueba01'@'localhost' (using password: YES)

C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u prueba01 -p
Enter password: ***********
ERROR 1045 (28000): Access denied for user 'prueba01'@'localhost' (using password: YES)

C:\Bitnami\wampstack-7.1.15-0>
```

La cuenta sigue estando desbloqueada debido a que no se puede configurar el bloqueo de la cuenta por el número de intentos fallidos como en la BD de Oracle. Para comprobar el estado de la cuenta de usuario, ejecutamos la siguiente sentencia:

```
mysql> SELECT User, account_locked FROM mysql.user WHERE User = 'prueba01';
+----------+----------------+
| User     | account_locked |
+----------+----------------+
| prueba01 | N              |
+----------+----------------+
1 row in set (0.00 sec)
```

9.35 Ejercicio 35

Crear un usuario "prueba04" con el parámetro "password expire", sus "tablespace" por defecto y temporal serán USERS (cuota 0k) y TEMP. Asignar los roles CONEXION y DESARROLLO. Conectarse como usuario "prueba04", ¿qué sucede?.

A diferencia de la BD de Oracle, MySQL no soporta el uso de la declaración *TABLESPACE* y *ROLE* por lo que la sentencia para la creación del usuario solicitado será:

Creamos el usuario otorgándole los privilegios necesarios y posteriormente modificamos dicho usuario haciendo que su contraseña expire.

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE VIEW ON users.* TO prueba04 IDENTIFIED BY 'password';
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> ALTER USER prueba04 PASSWORD EXPIRE;
Query OK, 0 rows affected (0.00 sec)
```

Cuando nos conectamos con el usuario con su contraseña inicial, el sistema no nos solicita el cambio de contraseña tras la conexión como si ocurre con la BD de oracle Oracle.

```
C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u prueba04 -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, 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>
```

Pero cuando intentamos ejecutar una sentencia nos genera el un error indicando que debemos resetear la contraseña usando la declaración *ALTER USER*.

```
mysql> SELECT 1;
ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.
```

Tras el cambio de contraseña utilizando la sentencia *ALTER USER*, ya nos deja realizar consultas sobre la BD.

```
mysql> ALTER USER prueba04 IDENTIFIED BY 'prueba04';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye

C:\Bitnami\wampstack-7.1.15-0>mysql -h localhost -u prueba04 -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, 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> SELECT 1;
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.00 sec)

mysql>
```

9.36 Ejercicio 36

Bloquear la cuenta del usuario “prueba04”, ¿qué sucede al conectarse de nuevo?

Del mismo modo que en la BD de Oracle, para bloquear la cuenta de usuario manualmente ejecutamos la siguiente sentencia:

```
mysql> ALTER USER prueba04 ACCOUNT LOCK;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Una vez bloqueada la cuenta, al intentar conectar de nuevo con ella nos aparece el siguiente error:

```
C:\Bitnami\wampstack-7.1.11-0>mysql -h localhost -u prueba04 -p
Enter password: ********
ERROR 3118 (HY000): Access denied for user 'prueba04'@'localhost'. Account is locked.
```

9.37 Ejercicio 37

Desbloquear la cuenta del usuario “prueba04”.

Del mismo modo que en la BD de Oracle, para desbloquear la cuenta de usuario manualmente ejecutamos la siguiente sentencia:

```
mysql> ALTER USER prueba04 ACCOUNT UNLOCK;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

10 DIFERENCIAS MYSQL vs ORACLE.

Existen muchas similitudes entre los objetos de esquema y privilegios en Oracle y MySQL. Sin embargo, algunos de estos objetos de esquema y privilegios difieren entre estas BD.

A continuación se muestra unas tablas de equivalencias entre los tipos de objetos de esquema y entre los Privilegios de MySQL y Oracle, de los cuales están marcados las principales diferencias que se han aparecido y comentado a lo largo de este documento.

- **OBJETOS DE ESQUEMA:**

Oracle	MySQL
AFTER trigger	trigger
BEFORE trigger	trigger
Check constraint	Check constraint
Column default	Column default
Database	Database
Foreign key	Foreign key
Index	Index
Package	N/A
PL/SQL function	Routine
PL/SQL procedure	Routine
Primary key	Primary key
Role	N/A (Vers 7.21)
Schema	Schema
Sequence	AUTO_INCREMENT for a column
Snapshot	N/A
Synonym	N/A
Table	Table
Tablespace	N/A
Temporary table	Temporary table
Trigger for each row	Trigger for each row
Unique key	Unique key
User	User
View	View

- **PRIVILEGIOS:**

Nivel	Privilegios MySQL	Privilegios de Sistema en Oracle
Global	ALTER	ALTER ANY TABLE, ALTER ANY SEQUENCE, ALTER ANY CUSTER, COMMENT ANY TABLE
Global	ALTER ROUTINE	ALTER ANY PROCEDURE, DROP ANY PROCEDURE
Global	CREATE	CREATE ANY TABLE, CREATE ANY SEQUENCE, CREATE ANY CLUSTER, CREATE DATABASE LINK, COMMENT ANY TABLE
Global	CREATE ROUTINE	CREATE ANY PROCEDURE
Global	CREATE USER	CREATE USER, GRANT ANY PRIVILEGE
Global	CREATE VIEW	CREATE ANY VIEW
Global	DELETE	ALTER ANY TABLE, DROP USER, DELETE ANY TABLE
Global	DROP	DROP ANT TABLE, DROP ANY SEQUENCE, DROP ANY

Nivel	Privilegios MySQL	Privilegios de Sistema en Oracle
		CLUSTER, DROP ANY VIEW
Global	EXECUTE	EXECUTE ANY PROCEDURE
Global	INDEX	CREATE ANY INDEX, ALTER ANY INDEX, DROP ANY INDEX
Global	INSERT	INSERT ANY TABLE
Global	LOCK TABLES	LOCK ANY TABLE
Global	SELECT	SELECT ANY TABLE
Global	SUPER	CREATE ANY TRIGGER, DROP ANY TRIGGER
Global	UPDATE	UPDATE ANY TABLE
Global	USAGE	CREATE SESSION, ALTER SESSION, UNLIMITED TABLESPACE
BD	CREATE	CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE TABLE
BD	CREATE ROUTINE	CREATE PROCEDURE
BD	CREATE VIEW	CREATE VIEW
Tabla	CREATE	CREATE TABLE
Tabla	CREATE VIEW	CREATE VIEW

11 FUENTES DE INFORMACIÓN.

- Documentación BD MySQL: https://dev.mysql.com/doc/refman/5.7/en/
- dasini.net – Journal dún expert MySQL: http://dasini.net/blog/en
- Curso MySQL: http://mysql.conclase.net
- Comparación entre Oracle vs MySQL: http://arbo.com.ve/oracle-vs-mysql/