## Práctica realizada por:

U6P01: Usuarios y permisos

Una vez estudiadas dos de las tres caras de SQL (DML, manipulación de datos, y DDL, definición de datos), nos queda por explorar la tercera: DCL o lenguaje de control de datos. Vamos a explorar el sistema de privilegios de MySQL a través de una práctica guiada.

Sobre las capturas:

* Incluye capturas donde se indica.
* Ten en cuenta que la práctica es guiada, pero en algunos apartados se te pide escribir tú el código SQL para realizar algunas acciones: la captura te ayudará a conservar la solución.
* Si lo deseas, incluye capturas adicionales.

## Primera parte: creación de usuarios con CREATE USER

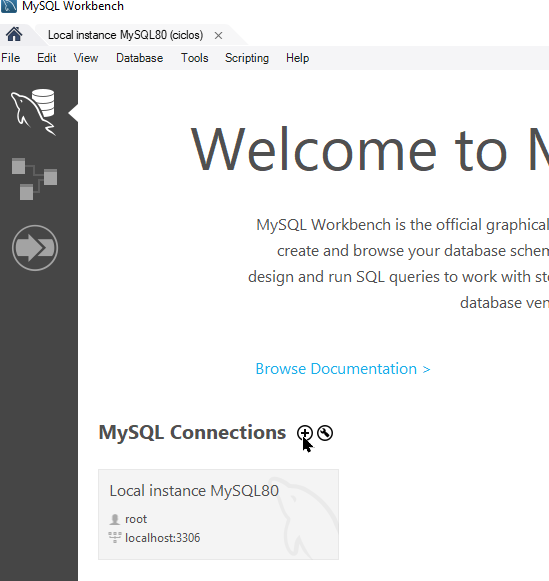
1-Crea un usuario con tu nombre en el servidor MySQL, por ejemplo:

CREATE USER alberto@localhost IDENTIFIED BY "secreto";

Ten en cuenta que ponemos “localhost” porque nuestro servidor se ejecuta en nuestra propia máquina: en otros casos se podría poner la dirección IP del servidor, por ejemplo [alberto@45.22.212.10](mailto:alberto@45.22.212.10)

2- Crea otros dos usuarios de prueba llamados billgates y stevejobs. Ponles la misma contraseña “secreto” para que no se te olvide.

3-En la pantalla principal de MySQL Workbench, crea una nueva conexión para el nuevo usuario. Observa que tienes un recuadro con el acceso para “root”, que es el que usamos siempre: ahora tendrás que crear una nueva conexión con el símbolo +.



4- En el recuadro que aparece, introduce un nombre de conexión (“Conexión de Alberto”), un Username (“alberto”) y una clave (“secreto”). En esta práctica tendrás que conectarte con varios usuarios, por tanto añadirás varios recuadros de acceso directo a conexión de usuario en la portada de MySQL Workbench.

Como ves, un usuario recién creado no tiene permiso para ver ninguna de las bases de datos.

## Segunda parte: concesión de permisos con GRANT

La gestión de permisos es muy sencilla, y utiliza la siguiente sintaxis:

GRANT permiso ON basededatos.tabla TO usuario@servidor;

5 – Con “alberto”: Prueba el siguiente ejemplo:

GRANT SELECT ON nba.jugadores TO alberto@localhost;

Te ha dado un error porque “alberto” (cuando mencione este usuario, entiende que me refiero al que lleva tu nombre) no tiene permisos para hacer nada, y mucho menos para conceder permisos a otros usuarios.

Cuando hablemos de sentencias GRANT y REVOKE en los próximos apartados, entiende que **debes ejecutarlos desde tu pestaña de conexión con el usuario admin**. Para evitar dudas se indicará antes de cada apartado el usuario con el que debes ejecutarlo. Puedes tener abiertas ambas conexiones para actuar con distintos usuarios cómodamente.

Ahora ejecuta la sentencia con “admin” para que tenga efecto.

6 – Con “alberto”: comprueba que ahora puedes visualizar la base de datos “nba”. Despliega en el panel izquierdo las tablas, y añade captura en la que se vea que sólo puedes ver la tabla “jugadores” y no el resto.

7 – Con “alberto”: Muestra los datos de todos los jugadores del equipo “Bobcats” (tienes permiso para hacer SELECT).

8 – Con “alberto”: Mediante SQL o el entorno gráfico de MySQL Workbench, intenta modificar datos de un jugador o añadir uno nuevo: captura del error (no tienes permiso para INSERT o UPDATE).

9 – Con “admin”: prueba esta sentencia errónea e incluye la captura del error que obtienes:

GRANT SELECT ON nba.equipos TO alberto;

El motivo es que alberto no es lo mismo que alberto@localhost. Debes poner siempre el servidor. El fallo te da a entender que no conoce el usuario “alberto”. Es muy probable que se te olvide ponerlo alguna vez, y ahora ya puedes entender el error que te dará.

10 – Con “admin”: Prueba ahora el siguiente ejemplo:

GRANT SELECT, INSERT, UPDATE, DELETE ON nba.\* to alberto@localhost, billgates@localhost

Con este ejemplo has comprobado que en una sola orden…

* Puedes dar varios permisos, separando los comandos por comas
* Puedes incluir a varios usuarios, separándolos por comas
* Puedes incluir todas las tablas de una base de datos, utilizando el comodín \*

11- Con “billgates”: comprueba que puedes ver todas las tablas de NBA en el panel izquierdo, y puedes añadir o modificar jugadores. Recuerda que para tu comodidad puedes hacer estas comprobaciones con el entorno gráfico de MySQL Workbench, no hace falta que escribas sentencias SQL. Incluye captura de alguna prueba.

12 – Con “admin”: El caso más generoso sería el siguiente (pruébalo):

GRANT **ALL** ON \*.\* TO stevejobs@localhost

Esto significa “concede todos los privilegios en todas las tablas de todas las bases de datos al usuario”.

Ponlo a prueba con un par de ejemplos en alguna base de datos e incluye captura de alguna prueba

## Tercera parte: averiguar los permisos que tiene un usuario con SHOW GRANTS

13 – Con “billgates”: ejecuta esta sentencia SQL e incluye captura del resultado:

SHOW GRANTS;

USAGE indica que el usuario no tiene ningún privilegio, es decir, no puede ejecutar ninguna sentencia: por tanto tener permiso de USAGE no es algo positivo.

Comprueba que con “admin” puedes averiguar los permisos de Bill Gates usando:

SHOW GRANTS FOR billgates@localhost;

¿Para qué comandos puedes dar permisos? Para todos los que hemos estudiado (SELECT, INSERT, UPDATE, DELETE, CREATE, DROP) pero para otros que no hemos estudiado:

14 – Con “admin”: muestra tus permisos para ver todas las cosas que se pueden hacer con SQL. Para ver bien el resultado, cópialo al portapapeles y pégalo en un editor de texto. Incluye captura:

Algunos comentarios sobre el resultado:

* Como ves, el lenguaje SQL es muy amplio y solo hemos estudiado los commandos más importantes. Aquí puedes encontrarlos todos: <https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html>
* Observa que en MySQL es posible conceder permisos de PROXY. Esto no lo vamos a probar, pero implicaría que un usuario pueda “suplantar” a otro, es decir, actuar en nombre del otro, y por tanto disponer de sus permisos.
* Observa que las sentencias terminan con “WITH GRANT OPTION”. El permiso GRANT OPTION indica que este usuario no solo puede disfrutar de esos permisos, sino que puede concederlos a otros usuarios. Vamos a explorar esto en el siguiente apartado.

15 – Con “billgates”: utiliza SHOW GRANTS para recordar lo que podía hacer el usuario. Incluye captura.

16 – Con “admin”: permite que billgates pueda propagar sus permisos a otros usuarios:

GRANT **GRANT OPTION** ON nba.\* TO billgates@localhost

17 – Con “billgates”: utiliza SHOW GRANTS de nuevo y observa que aparece “WITH GRANT OPTIONS” tras sus permisos. Incluye captura.

18 –Con “admin”: Crea un usuario llamado “nuevo” (recuerda: “nuevo@localhost”).

19 - Con “billgates”: prueba tu nuevo permiso concediendo al usuario “nuevo” el permiso de hacer SELECT e INSERT en la tabla nba.equipos.

20 – Con “billgates”: comprueba que no puedes conceder al usuario “nuevo” un permiso que tú mismo no tienes, por ejemplo para hacer DROP (captura del error).

## Cuarta parte: revocación de permisos y borrado de usuarios con REVOKE y DROP USER

21 – Con “stevejobs”: comprueba que tienes los mismos permisos que “admin”, debido al GRANT ALL que hicimos en el apartado 11.

22- Con “admin”: vamos a eliminar todos sus permisos usando la siguiente sentencia:

REVOKE ALL ON \*.\* FROM stevejobs@localhost

Es posible que MySQL Workbench te marque la sentencia errónea por usar ALL, pero funciona. Observa que cambiamos de TO a FROM:

* GRANT ... TO (conceder ... **a**)
* REVOKE ... FROM (retirar ... **de**)

Comprueba que “stevejobs” se ha quedado sin permisos.

23 – Con “admin”: quítale a “billgates” el permiso para hacer INSERT o DELETE en la base de datos NBA:

REVOKE INSERT, DELETE ON nba.\* FROM billgates@localhost;

Comprueba con SHOW GRANTS que billgates mantiene sus permisos para hacer SELECT y UPDATE (captura).

24 – Con “billgates”: aprovecha que tienes GRANT OPTION para revocar al usuario “nuevo” el permiso de SELECT que tú le concediste en el apartado 19. Comprobamos así que el permiso GRANT OPTION permite también revocar permisos

25 – Con “billgates”: comprueba que no puedes revocar permisos que tú no tienes: intenta revocar al usuario “nuevo” el permiso de INSERT que le concediste en el apartado 19. Recuerda que en el apartado 23 el administrador te quitó ese permiso, y por eso tú no puedes concederlo ni revocarlo. Captura del error.

26 – Con “admin”: Elimina el usuario “nuevo”:

DROP USER nuevo@localhost;

Comprueba que ya no puedes acceder con dicho usuario.

27 – Con “admin”: concede al usuario “alberto” el permiso de crear usuarios, y además con la opción de que él conceda el privilegio a otros usuarios:

GRANT CREATE USER ON \*.\* TO alberto@localhost WITH GRANT OPTION;

28 – Con “alberto”: crea de nuevo el usuario nuevo@localhost

29 – Con “alberto”: borra este usuario. Es decir: que con el permiso CREATE USER viene incluido el permiso para borrarlo.

30 – Con “alberto”: bórrate a ti mismo. Comprueba que ya no puedes acceder. Captura del error.

31 – Borra todos los usuarios que hemos usado hasta ahora (alberto, billgates, stevejobs, nuevo). Así nos aseguraremos de empezar de cero para el resto de apartados y no arrastrar diferencias entre el enunciado y tu situación actual. En la portada de MySQL Workbench, borra las conexiones para todos ellos con el botón secundario del ratón.

## Quinta parte: modificación de usuarios con ALTER USER

32 – Con “admin”: crea un usuario “alberto@localhost” sin indicar su contraseña:

CREATE USER alberto@localhost;

33 – Comprueba que puedes iniciar sesión con MySQL Workbench y no te pide contraseña.

34 – Con “admin”: Añade contraseña al usuario alberto:

ALTER USER alberto@localhost IDENTIFIED BY "secreto";

35 – Comprueba que ahora puedes continuar con tu sesión iniciada con alberto, pero si la cierras y vuelves a conectar, te pedirá la contraseña.

36 – Con “admin”: haz que la contraseña expire y se le solicite una nueva:

ALTER USER alberto@localhost PASSWORD EXPIRE;

37 – Con “alberto”: inicia una nueva sesión y comprueba que se te solicita cambiar la contraseña. Captura de la ventana en que se te pide.

El comando ALTER USER permite personalizar muchas opciones de seguridad. Puedes encontrarlas todas aquí: <https://dev.mysql.com/doc/refman/8.0/en/alter-user.html>

Por ejemplo podrías hacer ACCOUNT LOCK para bloquear una cuenta, o PASSWORD HISTORY 2 para evitar que el usuario ponga la misma contraseña que puso en sus últimos 2 cambios.

## Sexta parte: gestión de roles con CREATE ROLE y DROP ROLE

Si quieres aplicar los mismos permisos a 50 usuarios, sería muy pesado hacerlo de forma individual. Además, si luego quisieras cambiar alguno de esos permisos, tendrías que hacer de nuevo 50 cambios. Para solucionar este problema se ha creado el concepto de rol, similar a la pertenencia a grupos en administración de sistemas. La idea es que varios usuarios puedan tener un determinado rol, y que sea ese rol el que tenga permisos asignados.

38- Con “admin”: en este punto sólo tenemos el usuario “alberto”, creado en la quinta parte de la práctica. Añade los siguientes usuarios: julia, sergio, ana y javier.

39 – Con “admin”: crea el rol ADMIN\_CICLOS:

CREATE ROLE ADMIN\_CICLOS;

40 – Con “admin”: concede al rol ADMIN\_CICLOS todos los permisos sobre la base de datos de ciclos:

GRANT ALL ON ciclos.\*TO ADMIN\_CICLOS;

41 – Con “admin”: comprueba los permisos que tiene el rol:

SHOW GRANTS FOR ADMIN\_CICLOS;

42 – Con “admin”: crea dos nuevos roles, SECRETARÍA (con tilde en la Í) y DOCENTE. Debes dar a estos roles los siguientes permisos:

* SECRETARÍA tiene todos los permisos en la tabla “pertenece”
* DOCENTE tiene todos los permisos en la tabla “ciclo”

Para asignar roles a usuarios se usa GRANT:

GRANT ROL1, ROL2 TO usuario1@servidor, usuario2@servidor;

43 – El Con “admin”: asigna el rol ADMIN\_CICLOS al usuario julia:

GRANT ADMIN\_CICLOS TO julia@localhost;

44 – Con “admin”: comprueba los permisos de julia (ver apartado 41): observa que no aparecen los permisos que obtiene por su rol. Para ello hay que poner:

SHOW GRANTS FOR julia@localhost **USING ADMIN\_CICLOS**;

Incluye captura del resultado.

45 – Con “julia”: intenta acceder a la base de datos… No podrás. Captura del error.

¿Qué ha pasado si todo iba bien, tiene su rol aplicado y en el apartado anterior comprobamos los permisos que tenía? El problema es que en MySQL los roles no se activan por defecto para las sesiones que abran los usuarios. Compruébalo ejecutando con “julia”:

SELECT CURRENT\_ROLE();

46 – Con “admin”: activa el rol de julia:

SET DEFAULT ROLE **ALL** TO julia@localhost;

Podríamos haber puesto ADMIN\_CICLOS en lugar de ALL. Entonces, ¿porqué ponemos ALL? “ALL” representa “todos los roles que se hayan asignado al usuario”. De esta forma no tenemos que pensar o recordar qué roles tenía ese usuario concreto.

47 – Con “julia”: inicia una nueva sesión y comprueba que ya tienes el rol activo (ver código en paso 45). Haz alguna prueba para comprobar que efectivamente puedes manipular la base de datos ciclos (pero no las demás).

48 – Basándote en lo anterior, debes establecer los siguientes roles (incluye el código de cada sentencia o captura en que se vea):

* Alberto y Ana tienen el rol DOCENTE (hazlo en una única sentencia usando comas)
* Sergio tiene el rol SECRETARÍA
* Javier tiene los roles DOCENTE y SECRETARÍA (hazlo en una única sentencia usando comas)

Cuando termines recuerda ejecutar:

SET DEFAULT ROLE ALL TO alberto@localhost, ana@localhost, sergio@localhost, javier@localhost

49 – Comprueba que todo ha salido bien iniciando sesión con cada usuario y viendo lo que aparece en el panel izquierdo, así como con la orden SELECT CURRENT\_ROLE();

50 – Con “admin”: quita a javier el rol de SECRETARÍA a javier:

REVOKE SECRETARÍA FROM javier@localhost;

Ahora javier sólo tiene el rol DOCENTE. Compruébalo.

51 – Elimina el rol DOCENTE:

DROP ROLE DOCENTE;

52 – Con “alberto” y “ana”: inicia una nueva sesión y comprueba que ya no tienen el rol, puesto que ya no existe.

53 – Limpieza: elimina los usuarios julia, alberto, ana, sergio y javier . Elimina también sus accesos directos de conexión de la portada de MySQL Workbench.

*¡Enhorabuena por llegar hasta el final! Ha sido una práctica larga, pero era importante estudiar algo sobre administración del gestor de bases de datos, y por las limitaciones de tiempo de este curso no hemos podido estudiarlo en más profundidad. Ahora ya conoces los fundamentos de control de acceso a las distintas bases de datos con SQL.*

*No olvides que todo lo estudiado sirve también, con pequeñas diferencias puntuales de sintaxis, para otros SGBD como Oracle.*