PRÁCTICA 2

Administración y Gestión de Bases de Datos

Escrito por

Alejandro Fernández de la Puebla Ugidos
Miguel Hermoso Mantecón
Carlos Lafuente Sanz

Universidad Politécnica de Madrid
ETSISI

11 de diciembre de 2022

LCDPM

Índice

1 Parte 1

1.1 Crear los roles y los usuarios

Se quieren crear los roles `gestor`, `compradorJuegos` y `dependiente` con las siguientes especificaciones:

El rol `gestor` podrá realizar cualquier operación LMD sobre las tablas y tendrá permiso de propagación de privilegios. Además, los usuarios con este rol podrán operar (crear, modificar, etc.) con los roles, usuarios y privilegios que se abordan en los siguientes puntos.

El rol `compradorJuegos` podrá visualizar todas las tablas de la BD y también podrá dar de alta nuevos videojuegos que ha adquirido (operando sobre la tabla juegos).

El rol `dependiente` podrá visualizar todas las tablas de la BD, dar de alta nuevos clientes y modificar clientes existentes (operando sobre la tabla Clientes), así como indicar que un cliente ha alquilado un videojuego añadiendo un nuevo registro (operando sobre la tabla clientes_juegos).

Tambíen se quieren crear los suarios `gohan` con rol `gestor`, `vegeta` con el rol `compradorJuegos`, `videl` con el rol `compradorJuegos`, `trunks` con el rol `dependiente` y `goku` con el rol `dependiente`.

Esto se realiza con los siguientes scripts:

```
#INICIADO SESIÓN COMO ROOT

CREATE ROLE 'gestor';

GRANT SELECT, INSERT, UPDATE, DELETE ON PracABD1.* TO
↪  'gestor';
GRANT CREATE ROLE, DROP ROLE, GRANT OPTION ON *.* TO
↪  'gestor';
GRANT CREATE USER, ROLE_ADMIN ON *.* TO 'gestor';

CREATE USER 'gohan' IDENTIFIED BY 'gohan';

GRANT 'gestor' TO 'gohan';

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

```
#INICIADO SESIÓN COMO GOHAN

CREATE ROLE 'compradorJuegos';

GRANT SELECT ON PracABD1.* TO 'compradorJuegos';

GRANT INSERT ON PracABD1.juegos TO 'compradorJuegos';

CREATE USER 'vegeta' IDENTIFIED BY 'vegeta';

CREATE USER 'videl' IDENTIFIED BY 'videl';

GRANT compradorJuegos TO 'vegeta', 'videl';

SET DEFAULT ROLE ALL TO 'vegeta', 'videl';
```

```
#INICIADO SESIÓN COMO GOHAN

CREATE ROLE 'dependiente';

GRANT SELECT ON PracABD1.* TO 'dependiente';

GRANT INSERT, UPDATE ON PracABD1.clientes TO 'dependiente';

GRANT INSERT ON PracABD1.clientes_juegos TO 'dependiente';

CREATE USER 'trunks' IDENTIFIED BY 'trunks';

CREATE USER 'goku' IDENTIFIED BY 'goku';

GRANT 'dependiente' TO 'trunks', 'goku';

SET DEFAULT ROLE ALL TO 'trunks', 'goku';
```

1.2 Comprobar los cambios

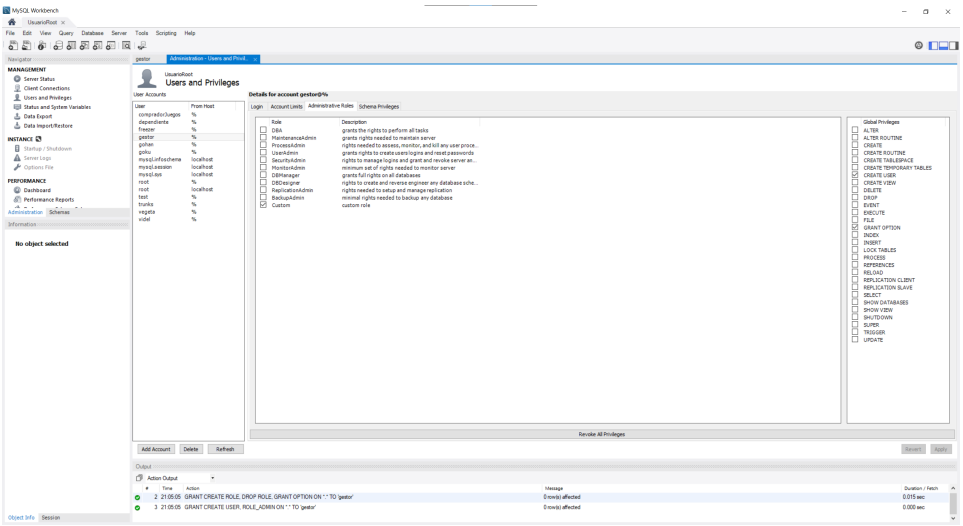
Con el usuario root se quiere comprobar que los cambios anteriores se han realizado correctamente. Eso se hizo de dos formas distintas con el siguiente script:

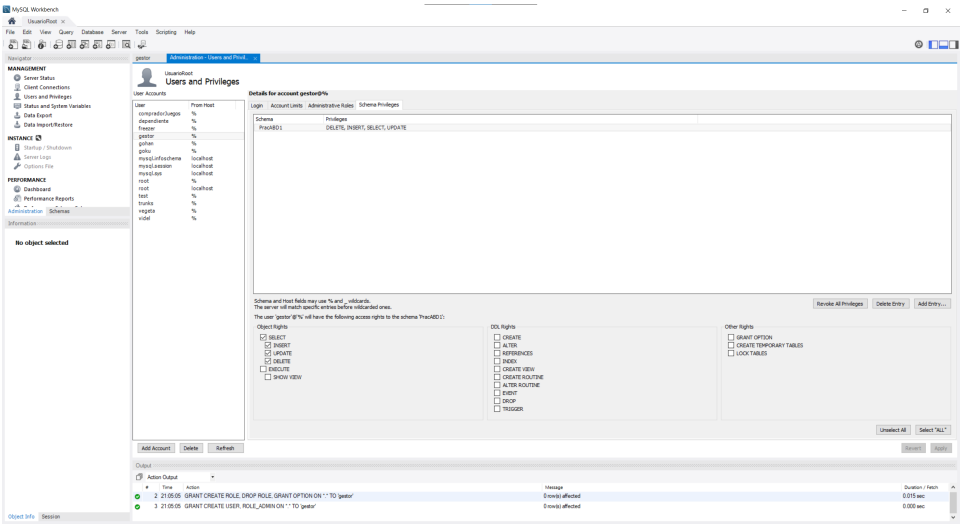
```
SELECT * FROM mysql.tables_priv;

SHOW GRANTS FOR 'gestor';
```

El primer método consulta el catálogo. El segundo directamente usa la sentencia de consulta de privilegios de MySQL.

Como tercer método se puede usar la interfaz de MySQLWorkbench:





1.3 Probar las sentencias

Cada uno de los usuarios ejecutó unas sentencias para probar sus privilegios. Estos son los scripts que se ocupan de ello y cada uno especifica que usuario debe ejecutarlo:

```
#INICIADO SESIÓN COMO GOHAN

#SENTENCIAS CON PERMISOS
SET SQL_SAFE_UPDATES = 0;

SELECT COUNT(*) AS Ventas, juegos.Titulo
        FROM clientes_juegos JOIN juegos ON
↪   clientes_juegos.juegoID = juegos.juegoID
        GROUP BY juegos.juegoID, juegos.Titulo
        HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                                                                FROM
↪   clientes_juegos
                                                                GROUP
↪   BY juegoID
                                                                    );

INSERT INTO `juegos` (`JuegoID`, `Titulo`, `Consola`,
↪   `Tamanio`, `Editor`)
VALUES (10000, "Cowabunga", "MegaDrive", 32768, "UPMGames");

SET SQL_SAFE_UPDATES = 1;
```

```
#SENTENCIAS SIN PERMISOS
DROP TABLE juegos;

CREATE TABLE juegosDeMesa (
        JuegoID INT UNIQUE NOT NULL,
        Titulo VARCHAR(256) /*UNIQUE*/ NOT NULL,
        NumeroJugadores INT,
        Empresa VARCHAR(32)
)
```

```
#INICIADO SESIÓN COMO VEGETA

#SENTENCIAS CON PERMISOS
SELECT *
FROM clientes;

INSERT INTO `juegos` (`JuegoID`, `Titulo`, `Consola`,
↪  `Tamanio`, `Editor`)
VALUES (9999, "Cowabunga 0", "Pleisteision", 32768,
↪  "UPMGames");

#SENTENCIAS SIN PERMISOS
SET SQL_SAFE_UPDATES = 0;

DELETE FROM juegos
WHERE JuegoID = 9999;

UPDATE juegos
        SET juegoId = 0
        WHERE JuegoID = 9999;

SET SQL_SAFE_UPDATES = 1;
```

```
#INICIADO SESIÓN COMO VIDEL

#SENTENCIAS CON PERMISOS
SELECT *
```

```
FROM clientes;

INSERT INTO `juegos` (`JuegoID`, `Titulo`, `Consola`,
↪  `Tamanio`, `Editor`)
VALUES (9999, "Cowabunga 0", "Pleisteision", 32768,
↪  "UPMGames");

#SENTENCIAS SIN PERMISOS
SET SQL_SAFE_UPDATES = 0;

DELETE FROM juegos
WHERE JuegoID = 9999;

UPDATE juegos
        SET juegoId = 0
        WHERE JuegoID = 9999;

SET SQL_SAFE_UPDATES = 1;
```

```
#INICIADO SESIÓN COMO TRUNKS

# VISUALIZAR
SELECT * FROM clientes;


# INDICAR ALQUILER
INSERT INTO `clientes_juegos` (`ClienteID`, `JuegoID`,
↪  `FechaAlquiler`, `Comentarios`)
VALUES (1, 12001, 'DATE: Auto CURDATE()', 'XD
↪  paraparapapappappppowwn :)')
```

```
#INICIADO SESIÓN COMO GOKU

# VISUALIZAR
SELECT * FROM clientes;
```

```
# INDICAR ALQUILER
INSERT INTO `clientes_juegos` (`ClienteID`, `JuegoID`,
↪  `FechaAlquiler`, `Comentarios`)
VALUES (1, 12001, 'DATE: Auto CURDATE()', 'XD
↪  paraparapapappappppowwn :)')
```

1.4 Revocar los permisos de Goku

Debido a un cambio organizativo, es necesario quitarle los permisos de inesrción y modificación a Goku. Para realizarlo se utiliza el siguiente script:

```
REVOKE 'dependiente' FROM 'goku';

GRANT SELECT ON PracABD1.* TO 'goku';
```

Se le quita el rol de dependiente y se le ponen privilegios personalizados para cumplir los requisitos.

2 Parte 2

2.1 Crear las vistas para freezer

Ahora se crea al usuario freezer y las vistas que podrá usar:

```
CREATE USER 'freezer' IDENTIFIED BY 'freezer';

#Conjunto A
CREATE VIEW conjuntoA
        AS SELECT CLienteID, DNI, Nombre, Apellidos,
↪  Provincia, Email
    FROM PracABD1.clientes;

GRANT SELECT, INSERT ON conjuntoA TO 'freezer';

#Conjunto B

CREATE VIEW conjuntoB
```

```
        AS SELECT JuegoID, Titulo, Consola, Tamanio, Editor
    FROM PracABD1.juegos
    WHERE Consola = "GameBoy";

GRANT SELECT, INSERT ON conjuntoB TO 'freezer';
```

2.2 Crear consultas de freezer

A continuación se crean las consultas especificadas en el enunciado para freezer sobre las vistas:

```
#INICIANDO SESIÓN COMO FREEZER

SET PROFILING = TRUE;

#A1
SELECT Nombre, Apellidos
        FROM conjuntoA
    ORDER BY provincia;

#A2
SELECT COUNT(*) AS ClientesEnSevilla
        FROM conjuntoA
    WHERE Provincia = "Sevilla";

#A3
SELECT Email
        FROM conjuntoA
    WHERE Provincia = "Barcelona";

#B1
SELECT *
        FROM conjuntoB
        ORDER BY Tamanio;

#B2
SELECT COUNT(*) AS NumeroJuegosNintendo
        FROM conjuntoB
    WHERE Editor = "Nintendo";

SHOW PROFILES;
```

```
SET PROFILING = TRUE;

#B3
INSERT INTO conjuntoB (JuegoID, Titulo, Consola, Tamanio,
↪  Editor)
        VALUES (372187, 'Final Fiesta II', 'GameBoy',
↪  '42069', 'Marvel');

#A4
INSERT INTO conjuntoA (CLienteID, DNI, Nombre, Apellidos,
↪  Provincia, Email)
        VALUES (578934798, '99999999A', 'Usopp', 'Yusuf',
↪  'Grand Line', 'ussopsenchoo@hotmale.com');

SHOW PROFILES;
```

Se añade `SHOW PROFILES` para ver los tiempos de ejecución de las consultas.

Para que lo haga root sobre las tablas directamente se crea también el siguiente script:

```
#INICIANDO SESIÓ COMO ROOT

SET PROFILING = TRUE;

#A1
SELECT Nombre, Apellidos
        FROM clientes
    ORDER BY provincia;

#A2
SELECT COUNT(*) AS ClientesEnSevilla
        FROM clientes
    WHERE Provincia = "Sevilla";

#A3
SELECT Email
        FROM clientes
    WHERE Provincia = "Barcelona";
```

```
#A4
INSERT INTO clientes (CLienteID, DNI, Nombre, Apellidos,
↪  Genero, Direccion, Localidad, Provincia, CodPostal,
↪  Telefono, Canal, FechaNacimiento, FechaContacto, Email)
        VALUES (578934798, '99999999A', 'Usopp', 'Yusuf',
↪  'H', 'La Mar', 'Alabasta', 'Grand Line', '00000',
↪  'Caracol', 'desconocido', '1990/1/1', '2002/3/3',
↪  'ussopsenchoo@hotmale.com');

#B1
SELECT *
        FROM juegos
        ORDER BY Tamanio DESC;

#B2
SELECT COUNT(*) AS NumeroJuegosNintendo
        FROM juegos
    WHERE Editor = "Nintendo" AND Consola = "GameBoy";

#B3
INSERT INTO juegos (JuegoID, Titulo, Consola, Tamanio,
↪  Editor)
        VALUES (372187, "Final Fiesta II", "GameBoy",
↪  "42069", "Marvel");

SHOW PROFILES;
```

2.3 Ejecutar las consultas con root

Los tiempos de ejecutar las consultas con el usuario root sobre las tablas son los siguientes:

- Ejecución de A1: T = 0.2714 segundos.
- Ejecución de A2: T = 0.0427 segundos.
- Ejecución de A3: T = 0.0188 segundos.
- Ejecución de A4: T = 0.0156 segundos.
- Ejecución de B1: T = 0.0299 segundos.
- Ejecución de B2: T = 0.0041 segundos.

- Ejecución de B3: T = 0.0149 segundos.

2.4 Ejecutar las consultas con freezer

Los tiempos de ejecutar las consultas con el usuario freezer sobre las vistas son los siguientes:

- Ejecución de A1 sobre A: T = 0.2769 segundos.
- Ejecución de A2 sobre A: T = 0.0299 segundos.
- Ejecución de A3 sobre A: T = 0.0223 segundos.
- Ejecución de B1 sobre B: T = 0.0027 segundos.
- Ejecución de B2 sobre B: T = 0.0016 segundos.

Las consultas A4 y B3 fallan si las vistas no tienen todos los campos que no pueden ser nulos de la tabla original. Para resolver este problema se añaden los campos nulos a las vistas (DNI al conjunto A y Consola al conjunto B). Los tiempos tras este cambio de dichas consultas son:

- Ejecución de A4 sobre A: T = 0.0084 segundos.
- Ejecución de B3 sobre B: T = 0.0153 segundos.

2.5 Comprobar las consultas de freezer sobre las tablas

Se puede comprobar que freezer no puede operar directamente sobre las tablas intentando hacer consultas desde su usuario a ellas. Fallan tanto `SELECT * FROM clientes` como `SELECT * FROM clientes_juegos`.

2.6 Comparativa de los métodos

En cuanto a la eficiencia, la consulta de datos a través de vistas puede ser más eficiente que una sobre la tabla subyacente. Esto es porque las vistas precomputan los resultados de las consultas y los almacenan en una tabla virtual, de forma que permite un acceso más rápido a los datos.

La seguridad también puede verse favorecida por el uso de las vistas. Se pueden crear vistas con privilegios propios y los datos justos para manejar el acceso de cada usuario a ellos de forma mucho más precisa.

3 Parte 3

3.1 Crear la copia de seguridad

La copia de seguridad se puede hacer a través de la consola de MySQL o de la interfaz gráfica de MySQLWorkbench.

Para realizarlo a través del terminal se haría lo siguiente:

```
mysqldump -u root PracABD1 > backup_10_12_2022.sql
```

Como en nuestro caso la MySQL está en un contendor Docker, primero hace falta acceder a su terminal con:

```
docker exec -it mysql bash
```

Para hacerlo a través de la interfaz de MySQL Workbench:

1. Se selecciona "Server > Data Export".
2. Se activa la opción "Export to Self-Contained File".
3. Se indica el fichero al que se quiere exportar.
4. Se selecciona el schema deseado.
5. Se pincha en el boton "Start Export".

3.2 Cargar la copia de seguridad

Para cargar la copia de seguridad, simplemente se crea el schema nuevo con `CREATE SCHEMA recovery_10_12_2022` y se carga a través del terminal con `mysql -u root -p recovery_10_12_2022 < backup 10_12_2022.sql` o por el mismo método que la exportación con MySQLWorkbench pero dándole al botón de importar.