PECL3

Grado en Ingeniería de Computadores
Bases de Datos – Laboratorio 8/10

Profesor: José Miguel Alonso Martínez

Carlos Eguren Esteban – 09088570M
Pablo Ruiz Vidaurre – 05962602J
Alejandro Torres Pérez de Baños – 03213701A

Contenido

1. Creación de Disparadores

La creación de los 3 triggers se ha ejecutado con el fichero llamado 'Triggers.sql'. A continuación, se explicará cada uno de ellos, además de las pruebas realizadas para su funcionamiento.

Para la creación de los triggers es necesario el comando:

SET search_path TO cine;

1.1. Trigger 1

Para la creación de este trigger se ha necesitado crear una nueva tabla en el esquema. Se ha llamado 'auditoria' y en ella se almacenan en que tabla de la base de datos de películas, el tipo de evento, el usuario y la fecha y hora en la que se ha tenido lugar dicho evento. Los tipos de eventos son inserción, actualización y borrado.

```
CREATE TABLE cine.auditoria (
    evento_id SERIAL PRIMARY KEY,
    tabla_afectada TEXT NOT NULL,
    tipo_evento TEXT NOT NULL,
    usuario TEXT NOT NULL,
    fecha_hora timestamp DEFAULT current_timestamp
);
```

El código de creación de este trigger es el siguiente:

```
CREATE OR REPLACE FUNCTION auditoria_trigger_function() RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO cine.auditoria (tabla_afectada, tipo_evento, usuario)
        VALUES (TG_TABLE_NAME, 'INSERT', current_user);
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO cine.auditoria (tabla_afectada, tipo_evento, usuario)
        VALUES (TG_TABLE_NAME, 'UPDATE', current_user);
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO cine.auditoria (tabla_afectada, tipo_evento, usuario)
        VALUES (TG_TABLE_NAME, 'DELETE', current_user);
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Además, se ha añadido este trigger a todas las tablas del esquema, aunque en la siguiente imagen sólo se muestra para 1 de ellas.

```
CREATE TRIGGER auditoria_trigger1
AFTER INSERT OR UPDATE OR DELETE ON cine.criticas_final
FOR EACH ROW EXECUTE FUNCTION auditoria_trigger_function();
```

Para la comprobación del correcto funcionamiento de este trigger se ha realizado la siguiente consulta:

```
INSERT INTO
 cine.peliculas_final (titulo, anno, duracion, calificacionMPA, idioma, nomb
re_personal_directores)
VALUES ('Titulo', 2023, '200 mins', 'PG-13', 'es', 'NombreDirector');

UPDATE cine.peliculas_final
SET duracion = '215 mins', idioma = 'en'
WHERE titulo = 'Titulo' AND anno = 2023;

DELETE FROM cine.peliculas_final
WHERE titulo = 'Titulo' AND anno = 2023;
```

Y si ahora consultamos la tabla ‘auditoria’ con un simple ‘SELECT * FROM auditoria’:

```
 evento_id | tabla_afectada  | tipo_evento | usuario  |         fecha_hora
-----------+-----------------+-------------+----------+----------------------------
      1214 | peliculas_final | INSERT      | postgres | 2024-01-04 20:09:07.204425
      1215 | peliculas_final | UPDATE      | postgres | 2024-01-04 20:09:18.451526
      1216 | peliculas_final | DELETE      | postgres | 2024-01-04 20:09:26.185789
(3 filas)
```

1.2. Trigger 2

Para la resolución de este apartado hemos utilizado 2 triggers que llaman a la misma función llamada 'insertar_pagina_web_trigger_function()':

```
-- Crear el trigger para cine.criticas_final
CREATE TRIGGER insertar_pagina_web_trigger_criticas
BEFORE INSERT
ON cine.criticas_final
FOR EACH ROW EXECUTE FUNCTION insertar_pagina_web_trigger_function();

-- Crear el trigger para cine.caratulas_final
CREATE TRIGGER insertar_pagina_web_trigger_caratulas
BEFORE INSERT
ON cine.caratulas_WEB_final
FOR EACH ROW EXECUTE FUNCTION insertar_pagina_web_trigger_function();
```

El código de la función es:

```
CREATE OR REPLACE FUNCTION insertar_pagina_web_trigger_function() RETURNS
 TRIGGER AS $$
BEGIN
    IF NEW.url_web NOT IN (SELECT url_web FROM cine.pagina_web_final) THEN
        INSERT INTO cine.pagina_web_final (url_web) VALUES (NEW.url_web);

    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Para la comprobación del correcto funcionamiento de este trigger se han realizado un par de consultas para cada uno, críticas y carátulas.

Aparecen 2 ya que queremos comprobar que todo funciona bien, por un lado, cuando la página web ya existe dentro de la tabla de páginas web, como se ven en la primera y cuarta inserción. Para la primera hemos utilizado una página que pertenecía a la película de Brave, 2012, ha sido escogida al azar para comprobar esto.

También hay que probar que si la página web no está en la tabla la cree, ya que es la finalidad del propio trigger. Esto se ve en la segunda y tercera inserción, para críticas y carátulas respectivamente.

En el caso de las críticas la página web no varía, ya que cuando se inserta por primera vez se está almacenando en la tabla de páginas web y en la segunda inserción esta misma web ya está metida.

```sql
-- Comprobacion trigger 2

INSERT INTO
 cine.caratulas_WEB_final (url_web, fecha, titulo_peliculas, anno_peliculas)
VALUES ('https.//forummovies.org/covers/brave-2012_Poster_Poster.jpg',
current_date, 'Titulo', 2023);

INSERT INTO
 cine.caratulas_WEB_final (url_web, fecha, titulo_peliculas, anno_peliculas)
VALUES ('https://url_inventada.com', current_date, 'Titulo', 2023);

INSERT INTO
 cine.criticas_final (critico, puntuacion, texto, titulo_peliculas, anno_pel
iculas, url_web)
VALUES ('Nombre', 5, 'Texto de la Critica', 'Titulo', 2023,
'https://url_inventada2.com');

INSERT INTO
 cine.criticas_final (critico, puntuacion, texto, titulo_peliculas, anno_pel
iculas, url_web)
VALUES ('Nombre', 2, 'Texto de la Critica', 'Titulo', 2023,
'https://url_inventada2.com');
```

1.3. Trigger 3

Para la creación de este trigger se ha necesitado crear una nueva tabla en el esquema. Se ha llamado 'puntuacion_media' y en ella se almacenan las películas de la base de datos junto a la puntuación media de sus críticas de la tabla 'críticas_final'.

```
CREATE TABLE cine.puntuacion_media (
    titulo_peliculas TEXT NOT NULL,
    anno_peliculas INTEGER NOT NULL,
    puntuacion_media NUMERIC(2,1) NOT NULL,

    CONSTRAINT puntuacion_media_pk PRIMARY KEY (titulo_peliculas, anno_peliculas),

    -- Clave foránea a la tabla peliculas_final
    CONSTRAINT puntuacion_media_fk1 FOREIGN KEY (titulo_peliculas, anno_peliculas)
    REFERENCES cine.peliculas_final (titulo, anno) MATCH FULL
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Además, se han importado los datos para que la tabla no esté vacía.

```
INSERT INTO cine.puntuacion_media (titulo_peliculas, anno_peliculas, puntuacion_media)
SELECT
    c.titulo_peliculas,
    c.anno_peliculas,
    AVG(c.puntuacion) AS puntuacion_media
FROM
    cine.criticas_final c
GROUP BY
    c.titulo_peliculas, c.anno_peliculas;
```

La creación del trigger es:

```
CREATE OR REPLACE FUNCTION calcular_puntuacion_media()
RETURNS TRIGGER AS $$
BEGIN
    -- Eliminar las filas existentes para la película y año en la tabla puntuacion_media
    DELETE FROM cine.puntuacion_media
    WHERE titulo_peliculas = NEW.titulo_peliculas AND anno_peliculas = NEW.anno_peliculas;

    -- Insertar la nueva puntuación media para la película y año
    INSERT INTO cine.puntuacion_media (titulo_peliculas, anno_peliculas, puntuacion_media)
    SELECT NEW.titulo_peliculas, NEW.anno_peliculas, AVG(puntuacion)
    FROM cine.criticas_final
    WHERE titulo_peliculas = NEW.titulo_peliculas AND anno_peliculas = NEW.anno_peliculas
    GROUP BY titulo_peliculas, anno_peliculas;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- Crear el trigger
CREATE TRIGGER actualiza_puntuacion_media
AFTER INSERT OR UPDATE ON cine.criticas_final
FOR EACH ROW
EXECUTE FUNCTION calcular_puntuacion_media();
```

Para la comprobación del correcto funcionamiento de este trigger se ha realizado la siguiente consulta:

```
INSERT INTO
 cine.peliculas_final (titulo, anno, duracion, calificacionMPA, idioma, nomb
re_personal_directores)
VALUES ('Titulo', 2023, '200 mins', 'PG-13', 'es', 'NombreDirector');

INSERT INTO
 cine.criticas_final (critico, puntuacion, texto, titulo_peliculas, anno_pel
iculas, url_web)
VALUES ('Nombre del Critico', 4, 'Texto de la Critica', 'Titulo', 2023,
'URL de la Critica');

INSERT INTO
 cine.criticas_final (critico, puntuacion, texto, titulo_peliculas, anno_pel
iculas, url_web)
VALUES ('Nombre del Critico', 6, 'Texto de la Critica', 'Titulo', 2023,
'URL de la Critica');

SELECT * FROM cine.puntuacion_media
WHERE titulo_peliculas = 'Titulo';
```

Y el resultado es:

```
 titulo_peliculas | anno_peliculas | puntuacion_media
------------------+----------------+------------------
 Titulo           |           2023 |              5.0
(1 fila)
```

2. Creación de usuarios

El fichero sql utilizado para la creación de los 4 usuarios se llama 'CreacionUsuarios.sql'. Además, el fichero 'ConsultasUsuarios.sql' contiene la consulta necesaria para poder ver los permisos que tiene el usuario activo en el esquema. A continuación, se detalla cada uno de los usuarios creados, así como los permisos asignados a cada uno, el código sql utilizado y el resultado de ejecutar la consulta de 'ConsultasUsuarios.sql'.

2.1. Administrador

El usuario llamado 'administrador' iniciará sesión con la contraseña 'admin'. A este usuario se le han otorgado todos los permisos en todas las tablas del esquema, por lo que puede ejecutar cualquier operación que quiera.

```
CREATE USER administrador WITH LOGIN PASSWORD 'admin';
GRANT USAGE ON SCHEMA cine TO administrador;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA cine TO administrador;
```

```
 table_schema |       table_name        |                  concatenated_privileges
--------------+-------------------------+-------------------------------------------------------------
 cine         | actores_final           | INSERT, TRIGGER, REFERENCES, TRUNCATE, DELETE, UPDATE, SELECT
 cine         | actua_final             | INSERT, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER
 cine         | auditoria               | INSERT, TRIGGER, REFERENCES, TRUNCATE, DELETE, UPDATE, SELECT
 cine         | caratulas_final         | INSERT, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER
 cine         | caratulas_web_final     | SELECT, TRIGGER, REFERENCES, TRUNCATE, DELETE, UPDATE, INSERT
 cine         | criticas_final          | TRIGGER, INSERT, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES
 cine         | directores_final        | DELETE, INSERT, SELECT, UPDATE, TRUNCATE, REFERENCES, TRIGGER
 cine         | escribe_final           | INSERT, TRIGGER, REFERENCES, TRUNCATE, DELETE, UPDATE, SELECT
 cine         | generos_peliculas_final | TRIGGER, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES, INSERT
 cine         | guionistas_final        | INSERT, TRIGGER, REFERENCES, TRUNCATE, DELETE, UPDATE, SELECT
 cine         | pagina_web_final        | TRIGGER, REFERENCES, TRUNCATE, INSERT, DELETE, UPDATE, SELECT
 cine         | peliculas_final         | TRIGGER, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES, INSERT
 cine         | personal_final          | INSERT, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER
 cine         | puntuacion_media        | TRIGGER, INSERT, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES
(14 filas)
```

Como podemos ver, este usuario tiene todos los permisos en todas las tablas del esquema.

2.2. Gestor

El usuario llamado ‘gestor’ iniciará sesión con la contraseña ‘gestor’. A este usuario se le han otorgado una serie de permisos en todas las tablas del esquema. Estos permisos son: insertar, actualizar, borrar y consultar. Con estos permisos puede consultar y modificar los datos dentro de cada tabla del esquema, pero no modificar la estructura de este.

```
CREATE USER gestor WITH LOGIN PASSWORD 'gestor';
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA cine FROM gestor;
GRANT USAGE ON SCHEMA cine TO gestor;
GRANT INSERT, UPDATE, DELETE, SELECT ON ALL TABLES IN SCHEMA cine TO gestor;
```

table_schema	table_name	concatenated_privileges
cine	actores_final	INSERT, DELETE, UPDATE, SELECT
cine	actua_final	INSERT, SELECT, UPDATE, DELETE
cine	auditoria	INSERT, DELETE, UPDATE, SELECT
cine	caratulas_final	INSERT, SELECT, UPDATE, DELETE
cine	caratulas_web_final	SELECT, DELETE, UPDATE, INSERT
cine	criticas_final	DELETE, INSERT, SELECT, UPDATE
cine	directores_final	SELECT, INSERT, UPDATE, DELETE
cine	escribe_final	INSERT, DELETE, UPDATE, SELECT
cine	generos_peliculas_final	DELETE, SELECT, UPDATE, INSERT
cine	guionistas_final	INSERT, DELETE, UPDATE, SELECT
cine	pagina_web_final	DELETE, INSERT, UPDATE, SELECT
cine	peliculas_final	INSERT, SELECT, UPDATE, DELETE
cine	personal_final	INSERT, SELECT, UPDATE, DELETE
cine	puntuacion_media	DELETE, INSERT, SELECT, UPDATE

(14 filas)

Tiene esos 4 permisos en todas las tablas.

2.3. Crítico

El usuario llamado ‘critico’ iniciará sesión con la contraseña ‘critico’. A este usuario se le ha otorgado el permiso de consultar en todas las tablas del esquema, por lo que puede visualizar todo el contenido de este. Además, podrá insertar nuevas críticas en la tabla ‘criticas_final’ dado que también tiene el permiso de insertar en esta tabla.

```
CREATE USER critico WITH LOGIN PASSWORD 'critico';
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA cine FROM critico;
GRANT USAGE ON SCHEMA cine TO critico;
GRANT SELECT ON ALL TABLES IN SCHEMA cine TO critico;
GRANT INSERT ON TABLE cine.criticas_final TO critico;
```

```
table_schema |       table_name        | concatenated_privileges
-------------+-------------------------+-------------------------
cine         | actores_final           | SELECT
cine         | actua_final             | SELECT
cine         | auditoria               | SELECT
cine         | caratulas_final         | SELECT
cine         | caratulas_web_final     | SELECT
cine         | criticas_final          | SELECT, INSERT
cine         | directores_final        | SELECT
cine         | escribe_final           | SELECT
cine         | generos_peliculas_final | SELECT
cine         | guionistas_final        | SELECT
cine         | pagina_web_final        | SELECT
cine         | peliculas_final         | SELECT
cine         | personal_final          | SELECT
cine         | puntuacion_media        | SELECT
(14 filas)
```

Como vemos, tiene asignado el permiso para poder consultar todas las tablas. Además, en la tabla ‘criticas_final’ puede también insertar.

2.4. Cliente

El usuario llamado ‘cliente’ iniciará sesión con la contraseña ‘cliente’. A este usuario sólo se le ha otorgado un permiso, el de consultar. Esto le permite visualizar todo el contenido del esquema.

```
CREATE USER cliente WITH LOGIN PASSWORD 'cliente';
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA cine FROM cliente;
GRANT USAGE ON SCHEMA cine TO cliente;
GRANT SELECT ON ALL TABLES IN SCHEMA cine TO cliente;
```

```
table_schema |       table_name        | concatenated_privileges
-------------+-------------------------+-------------------------
cine         | actores_final           | SELECT
cine         | actua_final             | SELECT
cine         | auditoria               | SELECT
cine         | caratulas_final         | SELECT
cine         | caratulas_web_final     | SELECT
cine         | criticas_final          | SELECT
cine         | directores_final        | SELECT
cine         | escribe_final           | SELECT
cine         | generos_peliculas_final | SELECT
cine         | guionistas_final        | SELECT
cine         | pagina_web_final        | SELECT
cine         | peliculas_final         | SELECT
cine         | personal_final          | SELECT
cine         | puntuacion_media        | SELECT
(14 filas)
```

Y finalmente, el usuario ‘cliente’ sólo tiene el permiso para consultar.

3. Conexión desde Python y pruebas

Para lo conexión desde Python a nuestro esquema utilizamos el archivo 'Conexión_python.py'. Para poder realizar las consultas, ejecutaremos este archivo desde el terminal de nuestra máquina. En este archivo tenemos 2 métodos, uno para iniciar sesión y otro para realizar la consulta.

- Método 1: ask_conn_parameters()

Este método le utilizamos para poder recoger por entrada de texto el usuario con el que se desea iniciar sesión como la contraseña (ambas cosas descritas anteriormente). El método devuelve los 5 datos necesarios para poder realizar la conexión.

```
def ask_conn_parameters():
    """
        pide los parámetros de conexión
    """
    print('Se va a intentar conectar a la base de datos.')
    host = 'localhost'

    port = 5432
    user = input('Usuario: ')
    password = input('Contrasenna: '
)
    database = 'postgres'

    return (host, port, user, password, database)
```

- Método 2: ask_query()

Este método le utilizamos para poder realizar la consulta que queramos.

```
def ask_query():
    return input("Query: ")
```