# Memoria PL3

Álvaro Camacho 48083824R

Fátima C. de la Morena 71961234F

Índice

Pág. 2-5: Ejercicio 1 Triggers

Pág. 5-6: Ejercicio 2 Usuarios y permisos

Pág. 6-7: Ejercicio 3 Conexión Python

--------------------CREACION DE TRIGGERS (Ejercicio 1)---------------------------------------------------------------

* **Creación de la tabla auditoría**

CREATE TABLE auditoria (

tabla varchar,

evento varchar,

usuario varchar,

fecha varchar

);

* **Creación de función para el trigger global de modificaciones. Inserta en la tabla auditoria (la que almacena la información de cada modificación) el nombre de la tabla modificada (*TG\_TABLE\_NAME*), el tipo de modificación (*TG\_OP*), el usuario que la ha realizado (*USER()*), y la fecha y hora (*NOW()*). Meto directamente TG\_OP en la columna del tipo de evento ya que es más sencillo y devuelve el tipo igualmente: ‘INSERT’, ‘UPDATE’ o ‘DELETE’.**

CREATE OR REPLACE FUNCTION peliculas.fn\_auditoria() RETURNS TRIGGER AS $fn\_auditoria$

BEGIN

INSERT INTO peliculas.auditoria (tabla, tipo, usuario, fecha)

VALUES (TG\_TABLE\_NAME, TG\_OP, USER, NOW());

RETURN NULL;

END;

$fn\_auditoria$ LANGUAGE plpgsql;

**Aquí pongo un ejemplo de cómo quedaría una inserción hecha desde el administrador en la tabla de auditoria:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* **Triggers de modificaciones (tiene que haber uno por cada tabla). Después de cada inserción, actualización, o borrado en cada tabla se ejecuta la función anterior. Son todos iguales, solo cambia sobre qué tabla actúan.**

**Creación del trigger para la tabla de actor**

CREATE TRIGGER tr\_auditoria\_actor

AFTER INSERT OR UPDATE OR DELETE ON peliculas.actor

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de actua**

CREATE TRIGGER tr\_auditoria\_actua

AFTER INSERT OR UPDATE OR DELETE ON peliculas.actua

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de caratulas**

CREATE TRIGGER tr\_auditoria\_caratulas

AFTER INSERT OR UPDATE OR DELETE ON peliculas.caratulas

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de criticas**

CREATE TRIGGER tr\_auditoria\_criticas

AFTER INSERT OR UPDATE OR DELETE ON peliculas.criticas

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de director**

CREATE TRIGGER tr\_auditoria\_director

AFTER INSERT OR UPDATE OR DELETE ON peliculas.director

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de generos**

CREATE TRIGGER tr\_auditoria\_generos

AFTER INSERT OR UPDATE OR DELETE ON peliculas.generos

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de pag\_web**

CREATE TRIGGER tr\_auditoria\_pag\_web

AFTER INSERT OR UPDATE OR DELETE ON peliculas.pag\_web

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de peliculas**

CREATE TRIGGER tr\_auditoria\_peliculas

AFTER INSERT OR UPDATE OR DELETE ON peliculas.peliculas

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

**Creación del trigger para la tabla de personal**

CREATE TRIGGER tr\_auditoria\_personal

AFTER INSERT OR UPDATE OR DELETE ON peliculas.personal

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_auditoria();

* **Creación de la función para el trigger que comprueba las pag.web. Si no existe la página web en la tabla de pags, la añade.**

CREATE OR REPLACE FUNCTION peliculas.fn\_comprobar\_web() RETURNS TRIGGER AS $fn\_comprobar\_web$

BEGIN

IF NOT EXISTS (SELECT 1 FROM peliculas.pag\_web WHERE nombre = NEW.nombre\_pag\_web) THEN

INSERT INTO peliculas.pag\_web (nombre) VALUES (NEW.nombre\_pag\_web);

END IF;

RETURN NEW;

END;

$fn\_comprobar\_web$ LANGUAGE plpgsql;

* **Creación del trigger que comprueba las pag.web usando la función.**

CREATE OR REPLACE TRIGGER comprobar\_web

BEFORE INSERT ON peliculas.criticas

FOR EACH ROW

EXECUTE FUNCTION peliculas.fn\_comprobar\_web();

* **Creación de la tabla para las puntuaciones medias con columnas título y su calificación media para poder ir insertándolas.**

CREATE TABLE puntuacion\_media (

titulo varchar,

calificacion varchar

);

* **Crear la función para el trigger. Si la película ya está metida en la tabla de puntuación media, simplemente se actualiza la puntuación. En caso de que no esté metida, se inserta.**

CREATE OR REPLACE FUNCTION peliculas.fn\_actualizar\_media() RETURNS TRIGGER AS $fn\_actualizar\_media$

BEGIN

IF NOT EXISTS (SELECT 1 FROM peliculas.puntuacion\_media WHERE titulo = NEW.titulo\_peliculas) THEN

INSERT INTO peliculas.puntuacion\_media (titulo, calificacion) VALUES (NEW.titulo\_peliculas, NEW.calificacion);

ELSE

UPDATE peliculas.puntuacion\_media

SET calificacion = (SELECT AVG(calificacion) FROM peliculas.criticas WHERE titulo = NEW.titulo\_peliculas)

WHERE titulo = NEW.titulo;

END IF;

RETURN NULL;

END;

$fn\_actualizar\_media$ LANGUAGE plpgsql;

* **Creación del trigger para actualizar las puntuaciones que usa la función de actualizar la media**

CREATE TRIGGER actualizar\_media

AFTER INSERT ON peliculas.criticas

FOR EACH ROW

EXECUTE FUNCTION actualizar\_media();

--------------------CREACION DE USUARIOS Y ROLES (Ejercicio 2)-------------------------------------------------

**Creamos rol de administrador. No quitamos permisos a ningún rol porque al crearlos no tienen por defecto. Administrador tiene todos los privilegios en todas las tablas del esquema peliculas.**

**En todos los roles la contraseña será la misma que el nombre de su usuario**

CREATE ROLE administrador WITH LOGIN PASSWORD ‘administrador`

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA peliculas TO administrador;

**Creamos rol de gestor. Tiene privilegios de inserción, actualización, selección (consultar) y borrar en todas las tablas del esquema peliculas.**

CREATE ROLE gestor WITH LOGIN PASSWORD ‘gestor`

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA peliculas TO gestor;

**Creamos rol de cliente. Tiene privilegios de hacer consultas en todas las tablas del esquema películas.**

CREATE ROLE cliente WITH LOGIN PASSWORD ‘cliente`

GRANT SELECT ON ALL TABLES IN SCHEMA peliculas TO cliente;

Prueba de inserción siendo cliente:

Texto

Descripción generada automáticamente

**Creamos rol de critico. Tiene privilegios de consulta en todas las tablas de peliculas y permiso de inserción para la tabla de criticas únicamente.**

CREATE ROLE critico WITH LOGIN PASSWORD ‘critico`

GRANT SELECT ON ALL TABLES IN SCHEMA peliculas to critico;

GRANT INSERT ON TABLE peliculas.criticas TO critico;

Prueba de insercion en una tabla que no sea criticas:

Texto

Descripción generada automáticamente

--------------------CONEXIÓN CON PYTHON (Ejercicio 3)-------------------------------------------------

**Creamos un archivo .py (python)para conectar nuestra base de datos peliculas con la posibilidad de que cualquier usuario creado pueda acceder.**

**En este programa solicitamos el usuario, la contraseña y la consulta que quiere hacer**

**Para realizar todo esto tenemos que tener la libreria psycopg2 descargada.**

**PROGRAMA:**

import psycopg2

def main():

connstr = 'host=localhost

port=5432

user=postgres

password=postgres

dbname=peliculas'

conn = psycopg2.connect(connstr)

#

# Se realiza la conexión a la base de datos de pelicula con el usuario postgres

#

cur = conn.cursor()

cur.execute('SELECT \* FROM personal.peliculas;')

#

# Se crea un cursor que se carga con los valores del resultado de la consulta

#

for nombre, nacionalidad, f\_nacimiento in cur.fetchall(): print(nombre,nacionalidad,f\_nacimiento)

#

# Se recorre el cursor y se imprimen por pantalla los valores del resultado de la consulta

#

cur.close

conn.close

#

# Se cierra el cursor y la conexión a la base de datos

#

main()

**En el programa que hemos adjuntado esta conectado como usuario postgres,pero se podria hacer como cualquier tipo de usuario.**

**Con este programa conseguimos imprimir por pantalla los valores que se consultan.**

**import psycopg2**

def main():

connstr = 'host=localhost

port=5432

user=gestor

password=gestor

dbname=peliculas'

conn = psycopg2.connect(connstr)

#

# Se realiza la conexión a la base de datos de pelicula con el usuario gestor

#

cur = conn.cursor()

cur.execute('SELECT \* FROM personal.peliculas;')

#

# Se crea un cursor que se carga con los valores del resultado de la consulta

#

for nombre, nacionalidad, f\_nacimiento in cur.fetchall(): print(nombre,nacionalidad,f\_nacimiento)

#

# Se recorre el cursor y se imprimen por pantalla los valores del resultado de la consulta

#

cur.close

conn.close

#

# Se cierra el cursor y la conexión a la base de datos

#

**En este ejemplo esta el usuario gestor conectado**