# Memoria PL3

Álvaro Camacho 48083824R

Fátima C. de la Morena 71961234F

--------------------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 FUNCTION peliculas.fn\_auditoria() RETURNS TRIGGER AS $fn\_auditoria$

BEGIN

INSERT INTO auditoria (tabla, evento, usuario, fecha)

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

RETURN NULL;

END;

$fn\_auditoria$ LANGUAGE plpgsql;

* **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 FUNCTION peliculas.fn\_comprobar\_web() RETURNS TRIGGER AS $fn\_comprobar\_web$

BEGIN

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

INSERT INTO peliculas.pag\_web (url) VALUES (NEW.pag\_web\_url);

END IF;

RETURN NEW;

END;

$fn\_comprobar\_web$ LANGUAGE plpgsql;

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

CREATE TRIGGER comprobar\_web

BEFORE INSERT ON peliculas.criticas

FOR EACH ROW

EXECUTE FUNCTION 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 FUNCTION peliculas.fn\_actualizar\_media() RETURNS TRIGGER AS $fn\_actualizar\_media$

BEGIN

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

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

ELSE

UPDATE puntuacion\_media

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

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**

--CREATE ROLE administrador WITH LOGIN PASSWORD ‘administrador`

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

**Creamos rol de gestor**

--CREATE ROLE gestor WITH LOGIN PASSWORD ‘gestor`

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

**Creamos rol de cliente**

--CREATE ROLE cliente WITH LOGIN PASSWORD ‘cliente`

--GRANT SELECT ON ALL TABLES IN SCHEMA peliculas TO cliente;

**Creamos rol de critico**

--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;

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

Meter código aqui