**Ejercicio 1**

Crear una tabla que almacene datos de personas

Vamos a registrar los siguientes datos:

(dni, apellido, nombre, fecha de nacimiento y estado civil), siendo dni la clave primaria.

Tenemos que tener en cuenta las siguientes restricción: 1) La persona tiene que ser mayor de 18 años 2) El apellido no puede estar vacío.

# Registramos el estado civil: SOLTERO, CASADO, VIUDO, DIVORCIADO

Sabiendo que el estado civil posee las siguientes restricciones:

1. no puede pasar de SOLTERO a VIUDO, DIVORCIADO
2. No puede pasar de CASADO, VIUDO, DIVORCIADO a SOLTERO
3. No puede pasar de VIUDO a DIRVORCIADO
4. De DIVORCIADO no puede pasar a VIUDO

Todas las otras transiciones de estado civil están permitidas.

Creamos la tabla

create table persona( dni integer primary key, apellido varchar(30), nombre varchar(30), fecnac date, estadoCivil varchar(10),

constraint CH\_Persona\_EstadoCivil check (estadoCivil in

('SOLTERO','CASADO','VIUDO','DIVORCIADO'))

);

Vamos a realizar dos trigger

# Primero una función

CREATE OR REPLACE FUNCTION func\_e() RETURNS TRIGGER AS $funcemp$

DECLARE

edad smallint ; estadocivil varchar(10);

BEGIN

NEW.estadoCivil := UPPER(NEW.estadoCivil); edad := date\_part('year',age(NEW.fecnac));

IF NEW.apellido = ' ' THEN

RAISE EXCEPTION 'no puede tener apellido vacío';

END IF;

IF edad <= '18' THEN

RAISE EXCEPTION 'no puede ser menor de 18 años';

END IF;

RETURN NEW;

END; $funcemp$ LANGUAGE plpgsql;

# Y la función TRIGGER que lo llama

CREATE TRIGGER trigger\_e BEFORE INSERT OR UPDATE ON persona

FOR EACH ROW EXECUTE PROCEDURE func\_e();

Veamos que realizamos tres controles ¿cuáles son?

Los controles son, que el apellido a insertar no este vacío y que la edad sea mayor a 18 años

# Luego vamos a crear otra función para controlar las transiciones de estado civil

CREATE OR REPLACE FUNCTION func\_p() RETURNS TRIGGER AS $funcemp$

DECLARE

BEGIN

NEW.estadoCivil := UPPER(NEW.estadoCivil);

if OLD.estadoCivil = 'SOLTERO' AND (NEW.estadoCivil = 'VIUDO' or

NEW.estadoCivil='DIVORCIADO') THEN

RAISE EXCEPTION 'ERROR de transición en estado civil';

END IF;

if (OLD.estadoCivil = 'CASADO' or OLD.estadoCivil = 'DIVORCIADO' OR OLD.estadoCivil =

'VIUDO') AND (NEW.estadoCivil = 'SOLTERO') THEN

RAISE EXCEPTION 'ERROR de transición en estado civil';

END IF;

if OLD.estadoCivil = 'DIVORCIADO' AND (NEW.estadoCivil = 'VIUDO') THEN RAISE EXCEPTION 'ERROR de transición en estado civil';

END IF;

if OLD.estadoCivil = 'VIUDO' AND (NEW.estadoCivil = 'DIVORDIADO') THEN RAISE EXCEPTION 'ERROR de transición en estado civil';

END IF;

RETURN NEW;

END; $funcemp$ LANGUAGE plpgsql;

# Y la función que lo llama

CREATE TRIGGER trigger\_p BEFORE UPDATE ON persona

FOR EACH ROW EXECUTE PROCEDURE func\_p();

Realice distintas operaciones de inserción y modificación en la tabla personas para corroborar que funcione bien el disparador

**insert into persona(dni, apellido, nombre, fecnac, estadocivil)**

**VALUES**

**(42464430, 'Errandonea', 'Gonzalo', '2/3/2000', 'Soltero'),**

**(42464140, 'Errandonea', 'Gonzalo', '2/3/200', 'Soltero'),**

**(42476430, '', 'Gonzalo', '2/3/2000', 'Soltero'),**

**(42442430, 'Errandonea', 'Gonzalo', '2/3/2000', 'XDD');**

**Update into persona set estadocivil = 'Viudo' Where DNI = '42464430';**

# EJERCICIO 2

Cree un TRIGGER de auditoría, que registre en una tabla las operaciones realizadas en la tabla persona, debiendo registrarse:

1. Que operaciones se hizo (insert/delete/update)
2. Fecha-hora en que se realizó
3. Nombre de la tabla en que se originó
4. Usuario que realizó la operación
5. Valor anterior
6. Valor nuevo

Para realizar este ejercicio deberá buscar en la documentación de postgres las funciones que tienen los datos que precisa.

Guarde las operaciones en un script y describa los resultados obtenidos.

create table persona\_log(

id SERIAL primary key,

operacion varchar (6),

fecha\_hora TIMESTAMP,

usuario varchar(100),

nombre\_tabla varchar(10),

old\_dni integer,

old\_apellido varchar(30),

old\_nombre varchar(30),

old\_fecnac date,

old\_estadoCivil varchar(10),

new\_dni integer,

new\_apellido varchar(30),

new\_nombre varchar(30),

new\_fecnac date,

new\_estadoCivil varchar(10)

);

CREATE OR REPLACE FUNCTION persona\_auditoria() RETURNS TRIGGER AS $funcpersonaauditoria$

BEGIN

IF (TG\_OP = 'INSERT') THEN

INSERT INTO persona\_log (operacion, fecha\_hora, usuario, nombre\_tabla,

old\_dni, old\_apellido, old\_nombre, old\_fecnac, old\_estadoCivil,

new\_dni, new\_apellido, new\_nombre, new\_fecnac, new\_estadoCivil)

VALUES (TG\_OP, now(), USER, TG\_TABLE\_NAME,

NULL, NULL, NULL, NULL, NULL,

NEW.dni, NEW.apellido, NEW.nombre, NEW.fecnac, NEW.estadoCivil);

RETURN NEW;

ELSEIF (TG\_OP = 'UPDATE') THEN

INSERT INTO persona\_log (operacion, fecha\_hora, usuario, nombre\_tabla,

old\_dni, old\_apellido, old\_nombre, old\_fecnac, old\_estadoCivil,

new\_dni, new\_apellido, new\_nombre, new\_fecnac, new\_estadoCivil)

VALUES (TG\_OP, now(), user, TG\_TABLE\_NAME,

OLD.dni, OLD.apellido, OLD.nombre, OLD.fecnac, OLD.estadoCivil,

NEW.dni, NEW.apellido, NEW.nombre, NEW.fecnac, NEW.estadoCivil);

RETURN NEW;

ELSEIF (TG\_OP = 'DELETE') THEN

INSERT INTO persona\_log (operacion, fecha\_hora, usuario, nombre\_tabla,

old\_dni, old\_apellido, old\_nombre, old\_fecnac, old\_estadoCivil,

new\_dni, new\_apellido, new\_nombre, new\_fecnac, new\_estadoCivil)

VALUES (TG\_OP, now(), user, TG\_TABLE\_NAME,

OLD.dni, OLD.apellido, OLD.nombre, OLD.fecnac, OLD.estadoCivil,

NULL, NULL, NULL, NULL, NULL);

RETURN OLD;

END IF;

RETURN NULL;

END;

$funcpersonaauditoria$ LANGUAGE plpgsql;

CREATE TRIGGER persona\_auditoria AFTER INSERT OR UPDATE OR DELETE ON persona

FOR EACH ROW EXECUTE PROCEDURE persona\_auditoria();