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();