Procedimientos almacenados (PL/PgSQL)

- PL Procedural language, también conocido como procedimientos almacenados, estas nos ayuda a desarrollar código directamente en el motor de bases de datos.
- Estructura de un Pl es: Declaración + uso de variable+ código +fin + retorno de valores o no retorna valores.UN bloque de código se ejecuta con la palabra DO $$ BEGIN --insert código here END $$
- RAISE NOTICE 'message', esta sentencia es para enviar un mensaje en el log de postgres
- Retornar una tabla

  Retornar una tabla.

DO$$ -Declaración de un bloque de código SQL Estructura

```
DO $BODY$
--Asignación de variables
DECLARE rec record := NULL;
    BEGIN
        --insert código here
    END
$BODY$
```

Ejemplo de declaración de bloques de código con plpgsql

```
DO $$
    DECLARE
        rec record;
        contador integer :=0;
    BEGIN
        --recorre  tabla pasajero y lo guarda en la variabl
e rec
        FOR rec IN SELECT * FROM pasajero LOOP
```

```
            RAISE NOTICE 'id: %     ,Nombre: %      ',
                        rec.id,rec.nombre;
            contador := contador + 1;
        END LOOP;
        RAISE NOTICE 'cantidad de registros:    %', contado
r;
    END
$$
```

CREATE FUNTION - Declaración de una función SQL

```
CREATE FUNCTION  consulta_usuarios()
    RETURNS void
    LANGUAGE 'plpgsql';
AS $BODY$
    DECLARE
        rec record;
        contador integer :=0;
    BEGIN
        --recorre  tabla pasajero y lo guarda en la variabl
e rec
        FOR rec IN SELECT * FROM pasajero LOOP
            RAISE NOTICE 'id: %     ,Nombre: %      ',
                        rec.id,rec.nombre;
            contador := contador + 1;
        END LOOP;
        RAISE NOTICE 'cantidad de registros:    %', contado
r;
    END
$BODY$
-- IMPORTANTE PONER EL NOMBRE DEL LENUAJE
LANGUAGE PLPGSQL;

--PARA LLAMAR A LA FUNCION SERÍA DE LA SIGUIENTE FORMA:
SELECT consulta_usuarios();
```

OTRO Ejemplo: Retornar una tabla con plpgsql ¡¡¡¡importante!!!! es importante cual select uses para llamar la función. la función funciona de la siguiente manera en el parámetro sí se introduce NULL retorna toda la lista, si se introduce id retornará esa tupla

```
--FUNCION QUE RETORNA UNA TABLA
--Mostrar tabla con plpgsql
--https://stackoverflow.com/questions/18084936/pl-pgsql-fun
ctions-how-to-return-a-normal-table-with-multiple-columns-u
sing-an
DROP FUNCTION consulta_t_pasajero(p_pasajero_id integer);

CREATE OR REPLACE FUNCTION consulta_t_pasajero(p_pasajero_i
d integer)
RETURNS TABLE(id integer, nombre character varying, direcci
on_residencia character varying, fecha_nacimiento date)
LANGUAGE plpgsql
AS $BODY$
    BEGIN
        IF p_pasajero_id IS NULL THEN
         RETURN QUERY
            SELECT pasajero.id, pasajero.nombre, pasajero.d
ireccion_residencia, pasajero.fecha_nacimiento
            FROM public.pasajero;
        END IF;
        RETURN QUERY
            SELECT pasajero.id, pasajero.nombre, pasajero.d
ireccion_residencia, pasajero.fecha_nacimiento
            FROM public.pasajero
            WHERE pasajero.id = p_pasajero_id;
    END;
$BODY$

--Retorno en forma de fila
SELECT consulta_t_pasajero(NULL);
SELECT consulta_t_pasajero(50);
--Retorno en forma de tabla
```

```
SELECT * FROM consulta_t_pasajero(NULL);
SELECT * FROM consulta_t_pasajero(50);
```

Triggers

Los triggers, al igual que las tablas, son objetos en los cuales podemos programar unas instrucciones con el fin de que se disparen cuando haya un evento (cambio) en la tabla relacionada.

- No puede haber trigger sin ser relacionada a una tabla.
- Los cambios que puede tener una tabla pueden ser (update, insert, delete).
- ¿Para qué necesitamos los triggers? Un ejemplo claro sería, que como administrador de la base de datos, necesitamos tener un control y saber que usuarios han estado interactuando con la base de datos disparando dichos eventos.
- ¿Qué acciones puede realizar un trigger? Un trigger puede dispararse ya sea antes, después o en lugar de, de un evento.

  Por ejemplo: Un usuario al ingresar datos a una tabla por medio del comando Insert, el trigger puede ejecutarse justo antes almacenando en otra tabla información relacionada sobre. ¿quien lo hizo?, la hora, cantidad de datos agregados, etc. Y se hará de forma automática También se puede por ejemplo antes de ejecutar el comando update, hacer una copia de respaldo. Usualmente los triggers se relacionan con el mantenimiento y administración de las bases de datos.

Para la creación de triggers se debe hacer los siguiente:

- Crear la función que activará el evento. Para ello se debe tomar los siguientes aspectos:
  1. En la declaración de la función, en la sección del retorno se debe indicar que es tipo triggers es decir RETURNS TRIGGER. Luego indicar en que lenguaje está escrito es decir LANGUAE 'plpgsql'
  2. La función tipo triggers debe retornar los valores OLD acepta lo viejo o NEW acepta lo nuevo. Sí se retorna VOID en nuestra función de tipo triggers no aceptamos cambios es decir RETURN NEW;

3. Tanto NEW como OLD son un objeto de tipo record y contiene dentro de si el registro, es decir se puede acceder a los campos NEW.campo_nombre del registro

```
DROP FUNCTION IF EXISTS  count_on_insert_pasajero() CASCAD
E;

CREATE OR REPLACE FUNCTION count_on_insert_pasajero()
    RETURNS TRIGGER
    LANGUAGE 'plpgsql'
AS $BODY$
    DECLARE
        contador integer:=0;
        rec record;
    BEGIN

        FOR rec IN SELECT * FROM pasajero LOOP
            contador := contador + 1;
        END LOOP;
        RAISE NOTICE 'cantidad de registros:    %', contado
r;

        --insert record on conteo_pasajero
        INSERT INTO public.conteo_pasajero (total_pasajero,
hora_conteo)
        VALUES (contador,now());

        RETURN NEW;

    END;
$BODY$
```

Lo siguiente será crear la regla que estará a la escucha del evento para disparar el triggers, para ello se deberá tomar los siguientes aspectos.

1. CREATE TRIGGER name_trigger name_event ON name_table FOR EACH ROW EXECUTE PROCEDURE name_procedure;

2. En la primera sección cuando declaramos el trigger debemos indicar en que momento en que se debe disparar el trigger: CREATE TRIGGER name_trigger name_event ON name_table en el name_event allí puede ir alguno de estos tres parámetros para llamar la ejecución del trigger, estos son:

- BEFORE = antes,
- AFTER=luego,
- INSTEAD OF = hacer esto, en vez de lo que iba a hacer el motor de bases de datos.

1. FOR EACH ROW EXECUTE PROCEDURE name_procedure indica que es para registro o fila de nuestra tabla

```
-- CREACIÓN DE LA REGLA PARA EJECUTAR EL TRIGGER
CREATE TRIGGER trigger_on_insert_to_pasajero
AFTER INSERT ON pasajero
FOR EACH ROW EXECUTE PROCEDURE count_on_insert_pasajero();
```

Para realizar el conteo de pasajeros para cuando se elimine el usuario, nuestro Trigger quedaría de la siguiente forma:

```
CREATE TRIGGER triggerdelete
    AFTER DELETE
    ON public.pasajero
    FOR EACH ROW
    EXECUTE PROCEDURE public.importantepl();
```