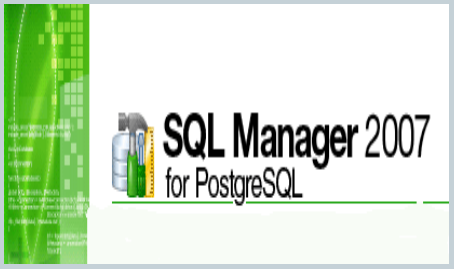
CURSO DE FUNDAMENTOS Y ADMINISTRACION DE DBMS

UBUNTU 10.4

vmware

Java

PostgreSQL



Windows

php

The Apache Software Foundation

SYBASE

INSTRUCTOR: RUDY SALVATIERRA RODRIGUEZ

TRIGGERS EN MYSQL

DEFINICION

- Un trigger es comúnmente llamado disparador.
- Estos disparadores reaccionan a un cierto tipo de evento como INSERT, UPDATE o DELETE.
- Cuando ocurre el evento estos triggers saltan o reaccionan realizando las acciones que tiene en su cuerpo
- Son útiles a la hora de validar los datos y para crear BITACORAS
- Usan dos variables:
  - **NEW** -> hace referencia a los datos que estoy insertando, esta variable puede ser referenciada desde el INSERT y UPDATE
  - **OLD** -> hace referencia a los datos que estoy borrando, esta variable es referenciada por UPDATE y DELETE

- Un ejemplo seria:
  - Cuando hago el insert puedo agarrar los datos a insertar, manipularlos y luego insertarlos

TRIGGERS EN MYSQL

- **Sintaxis**

```
CREATE TRIGGER [Nombre_trigger] [BEFORE/AFTER]
    [INSERT/UPDATE/DELETE] ON [tabla]
    FOR EACH ROW
    Begin
        [declaraciones]
        [sentencias]
    End
```

TRIGGERS EN MYSQL

- **Ejemplo de trigger**

```
Create trigger InsertCanal before insert on canal
For  each row
begin
   insert into bitcora(usuario,tabla,datosNuevos,fecha)
   VALUES(user(),'canal',CONCAT(new.cod_canal,new.c
   od_gen,new.nombre_canal,new.rating,NEW.numero_ca
   nal,new.logo),now());
end
```

TRIGGERS EN MYSQL

```
Create trigger UpdateCanal before update on canal
For  each row
begin
    insert into bitcora
    (usuario,tabla,datosNuevos,datosAntiguos,fecha)
    VALUES(user(),'canal',CONCAT_sw('-',new.cod_canal,
    new.cod_gen,new.nombre_canal,new.rating,NEW.numero_canal,
    new.logo), CONCAT_sw('-', old.cod_canal, old.cod_gen,
    old.nombre_canal, old.rating, old.numero_canal,
    old.logo),now());
end
```

TRIGGERS EN PL/PGSQL

DEFINICIÓN:

Un disparador es una acción definida en una tabla de una base de datos y ejecutada automáticamente por una función programada por nosotros. Esta acción se activará, según la definamos, cuando realicemos un INSERT, un UPDATE ó un DELETE.

Un disparador se puede definir de las siguientes maneras:

- Para que ocurra ANTES de cualquier INSERT,UPDATE ó DELETE.
- Para que ocurra DESPUES de cualquier INSERT,UPDATE ó DELETE.
- Para que se ejecute una sola vez por comando SQL (statement-level trigger).
- Para que se ejecute por cada linea afectada por un comando SQL (row-level trigger).

TRIGGERS EN PL/PGSQL

TIPOS DE TRIGGERS:

Existen 3 tipos de triggers.

- Triggers pl/pgsql
- Triggers para Auditoria
- Triggers para Mantener una tabla resumen.

TRIGGERS EN PL/PGSQL

- Variables predefinidas en un trigger

NAME	TIPO	DESCRIPCION
NEW	**%ROWTYPE**	Valores nuevos para insert o update
OLD	**%ROWTYPE**	Valores antiguos para delete o update
TG_NAME	**Name**	Nombre del trigger
TG_WHEN	**Text**	Antes(before) o despues(after)
TG_LEVEL	**Text**	Aplicado a cada registro(row) o solo una sentencia
TG_OP	**Text**	Tipo de evento: insert, update o delete
TG_RELID	**Oid**	Identificador OID de la tabla trigger
TG_RELNAME	**Name**	Nombre de la tabla del trigger
TG_NARGS	**Integer**	Indica la cantidad de argumentos
TG_ARGV[]	**Text[]**	Argumentos opcionales

TRIGGERS EN PL/PGSQL

CREANDO TRIGGERS

Para crear un trigger primero debemos definir una función Trigger.

Sintaxis para un Trigger:

CREATE TRIGGER nombre_trigger [BEFORE | AFTER]

[INSERT | DELETE | UPDATE [OR ...]] ON nombre_tabla FOR EACH ROW EXECUTE PROCEDURE funcion_trigger_ejecutarse;

nombre_trigger = Cualquier nombre para el Trigger

nombre_tabla = nombre de la tabla sobre el cual se esta ocurriendo el evento(insert, update, delete).

funcion_trigger_ejecutarse = función que se ejecutara cuando ocurra este evento.

TRIGGERS EN PL/PGSQL

- **REQUISITOS PARA CREAR UNA FUNCIÓN TRIGGER.**

  1. La función **trigger** no debe recibir ningún parámetro
  2. El tipo de retorno de la función **Trigger** debe ser de tipo **trigger** y no un tipo de dato.

  El trigger del **Bit_docente** se ejecuta cada ves que eliminamos (evento **delete**) un docente de la tabla, al activarse el trigger ejecuta la función **bitacora_docente(),** esta función a la ves inserta en otra tabla llamada **docente_respaldo(…)** el registro eliminado mas la fecha en la que se esta eliminando.

TRIGGERS EN PL/PGSQL

DESCRIPCIÓN DEL EJEMPLO

CREANDO LA TABLA RESPALDO

CREATE TABLE docente _respaldo (usuario VARCHAR(20),accion VARCHAR(20), fecha date, codigo_docente INTEGER, nombre_docente VARCHAR(50) , apellido_pat VARCHAR(50), apellido_mat VARCHAR(50), profesion VARCHAR(50));

CREANDO LA FUNCION BITACORA DOCENTE

```
CREATE OR REPLACE FUNCTION bitacora_docente() RETURNS trigger AS ‘ BEGIN
  INSERT INTO docente_respaldo VALUES(user,TG_OP, now(), OLD.codigo_docente, OLD.nombre_docente, OLD.apellido_pat, OLD.apellido_mat, OLD.profesion);
RETURN NULL;
END;
’ LANGUAGE 'plpgsql’;
```

TRIGGERS EN PL/PGSQL

- **DESCRIPCIÓN DE LA FUNCIÓN TRIGGER**

```
CREATE TRIGGER Bit_docente
                    AFTER DELETE ON docente
                    FOR EACH ROW
                    EXECUTE PROCEDURE bitacora_docente();
```

TRIGGERS EN PL/PGSQL
EJEMPLO

```
CREATE FUNCTION public.respaldo_empleado (
)
RETURNS trigger
AS
$body$
DECLARE
   usuario varchar(50);
   accion varchar(50);
   datos_fecha varchar(200);
BEGIN
   usuario:=USER;
   accion:= TG_OP;
   datos_fecha:= now();
      if accion='INSERT' THEN
      insert into bit_empleado values(usuario, accion, datos_fecha,new.ci
                                    ||'  '||new.id_cuenta||'  '||new.nombre||'  '||
                                    new.apellido||'  '||new.cargo||'  '||
                                    new.genero);
      raise notice 'Se inserto correctamente';
   END IF;
```

TRIGGERS EN PL/PGSQL
EJEMPLO

```
else
   if accion='UPDATE' THEN
      insert into bit_empleado values(usuario, accion, datos_fecha,old.ci ||'  '||old.id_cuenta||'
     '||old.nombre||'  '|| old.apellido ||'  '||old.cargo ||'  '|| old.genero );
      raise notice 'Se modifico correctamente';
   END IF;
   else
   if accion='DELETE' THEN
      insert into bit_empleado values(usuario, accion, datos_fecha,old.ci ||'  '||old.id_cuenta||'
     '||old.nombre||'  '|| old.apellido ||'  '||old.cargo ||'  '|| old.genero );
      raise notice 'Se elimino correctamente';
   END IF;
  END IF;
 END IF;
 RETURN NULL;
end;
$body$
    LANGUAGE plpgsql;
```