Disparadores (triggers) en PostgreSQL

Jesús Reyes Carvajal

¿Qué es un disparador?

Una de las funcionalidades disponibles en PostgreSQL son los denominados disparadores (triggers).

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

INSERT, UPDATE ó un DELETE.

Un disparador se puede definir para que ocurra:

ANTES de cualquier INSERT,UPDATE ó DELETE
DESPUES de cualquier INSERT,UPDATE ó DELETE
Una sola vez por comando SQL (statement-level trigger)
Una vez por cada linea afectada por un comando SQL (row-level trigger)

Estructura de un disparador

CREATE TRIGGER nombre { BEFORE | AFTER } { INSERT | UPDATE | DELETE [OR ...] }

ON tabla [FOR [EACH] { ROW | STATEMENT }]

EXECUTE PROCEDURE nombre de funcion (argumentos)

ELIMINAR UN DISPARADOR

drop trigger nombre_disparador ON tabla;

Características de un disparador

A continuación veremos algunas características y reglas a tener en cuenta cuando definamos un disparador.

Un procedimiento almacenado se debe de definirse antes de utilizarlo en un disparador.

Un procedimiento en un disparador no puede tener argumentos y su retorno debe ser de tipo "trigger".

Un procedimiento almacenado se puede utilizar por múltiples disparadores en diferentes tablas.

Un procedimiento almacenado se puede se aplicar una sola vez, o por cada una de las tuplas en la tabla.

Si se aplica una sola vez se coloca FOR EACH STATEMENT.
Si se aplica en todas las tuplas de la tabla se utiliza FOR EACH ROW.

Si una tabla tiene más de un disparador definido para un mismo evento (INSERT,UPDATE,DELETE), estos se ejecutarán en orden alfabético por el nombre.

Variables especiales en PL/pgSQL

Cuando una función escrita en PL/pgSQL es llamada por un disparador tenemos ciertas variable disponibles:

NEW
Tipo de dato RECORD; Variable que contiene la nueva fila de la tabla para las operaciones INSERT/UPDATE en disparadores del tipo row-level. Esta variable es NULL en disparadores del tipo statement-level.

OLD
Tipo de dato RECORD; Variable que contiene la antigua fila de la tabla para las operaciones UPDATE/DELETE en disparadores del tipo row-level. Esta variable es NULL en disparadores del tipo statement-level.

TG_NAME
Tipo de dato name; variable que contiene el nombre del disparador que está usando la función actualmente.

TG_WHEN
Tipo de dato text; retorna el valor BEFORE o AFTER que esta definido en el disparador activado.

Variables especiales en PL/pgSQL

TG_LEVEL
Tipo de dato text; retorna el valor ROW o STATEMENT que esta definido en el disparador activado.

TG_OP
Tipo de dato text; retorna la operación INSERT, UPDATE o DELETE que esta definido en el disparador activado.

Existen otras funciones que son muy importantes

TG_TABLE_NAME
Tipo de dato name; nombre de la tabla sobre la que se desencadena la función.

TG_TABLE_SCHEMA
Tipo de dato name; nombre del scheme sobre la que se desencadena la función.

TG_NARGS
Tipo de dato integer; número de argumentos dados al procedimiento en la sentencia CREATE TRIGGER.

TG_ARGV[]: Tipo de dato text array; los argumentos de la sentencia CREATE TRIGGER.

Ejemplo de un disparador en PL/pgSQL

```
CREATE TABLE empleados(
  id integer NOT NULL,
  nombre varchar(50),
  fecha_nac date,
  edad int,
  PRIMARY KEY (id)
);

CREATE OR REPLACE FUNCTION actualizar_edad()
RETURNS  trigger AS $BODY$
DECLARE
edad integer;
BEGIN
  select DATE_PART('years',CURRENT_DATE::date)-DATE_PART('years',fecha_nac::date) into
edad from empleados;
  NEW.edad:=edad;
  RETURN  NEW;
end;
$BODY$
LANGUAGE plpgsql;


CREATE TRIGGER update_edad BEFORE INSERT OR UPDATE ON empleados
FOR EACH ROW EXECUTE PROCEDURE actualizar_edad();
```

Ejemplo de un disparador en PL/pgSQL

Insertando datos:

```
insert into empleados(id,nombre,fecha_nac) values(3,'Claudia Torres', '1979-05-08');
```

```
 select * from empleados;
 id |       nombre                    | fecha_nac   | edad
----+-------------------+-----------+------
  2 | Yolanda Colmenares  | 1999-05-08 |
  3 | Claudia Torres           | 1979-05-08 |   40
```

Si actualizamos:

```
update empleados set nombre='Yolanda Colmenares Sanchez' where id=2;
```

```
 select * from empleados;
 id |       nombre                                          | fecha_nac   | edad
----+-------------------+-----------+------
  2 | Yolanda Colmenares Sanchez     | 1999-05-08 |  20
  3 | Claudia Torres                                | 1979-05-08 |   40
```

Disparador de tipo statement-level en PL/pgSQL

Disparador de tipo **statement-level** que se ejecute después de INSERT, UPDATE y DELETE. La función ejecutada por este disparador grabará datos de la ejecución en la tabla eventos.

```
CREATE TABLE eventos (
  timestamp_ TIMESTAMP WITH TIME ZONE default NOW(),
  nomb_disp text,
  tipo_disp text,
  nivel_disp text,
  comando text
);

CREATE OR REPLACE FUNCTION grabar_eventos() RETURNS TRIGGER AS $$
  DECLARE
  BEGIN

    INSERT INTO eventos (
                nomb_disp,
                tipo_disp,
                nivel_disp,
                comando)
        VALUES (
                TG_NAME,
                TG_WHEN,
                TG_LEVEL,
                TG_OP
               );

    RETURN NULL;
  END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER grabar_eventos AFTER INSERT OR UPDATE OR DELETE
    ON empleados FOR EACH STATEMENT
    EXECUTE PROCEDURE grabar_eventos();
```

Disparador de tipo statement-level en PL/pgSQL

Actualizar el inventario de la tienda, cuando se venden productos

```
CREATE OR REPLACE FUNCTION actualizar_stock()
RETURNS  trigger AS $BODY$
DECLARE
stock1 integer;
cantidad integer;
BEGIN
  SELECT stock into stock1 FROM productos WHERE cod_prod=NEW.cod_prod;
  cantidad:=stock1 - NEW.cant;
  IF cantidad >= 0 THEN
    UPDATE productos SET stock=cantidad WHERE cod_prod=NEW.cod_prod;
  END IF;
  RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_stock BEFORE INSERT OR UPDATE ON detalle
FOR EACH ROW EXECUTE PROCEDURE actualizar_stock();
```