Triggers

...

PBD - Prof. Santiago Neira

¿Qué son los Triggers?

- Son programas almacenados especializados
- Este tipo de programas no son llamados, sino que son DESENCADENADOS por eventos
- Los Triggers son desencadenados por 3 tipos de eventos:
  - Eventos DML: INSERT, DELETE o UPDATE
  - Eventos DDL: CREATE, ALTER o DROP
  - Eventos de BD (LOGON, LOGOFF, STARTUP, SHUTDOWN, etc)

Sintaxis Trigger Eventos DML

```
1  CREATE OR REPLACE TRIGGER nombre_trigger
2  {BEFORE | AFTER}
3  {INSERT | DELETE | UPDATE | UPDATE OF column1[, column2[, column(n+1)]]}
4  ON nombre_tabla
5  [FOR EACH ROW]
6  [WHEN (logical_expresion)]
7  [DECLARE]
8    declaration_statements;
9  BEGIN
10   execution_statements;
11 END nombre_trigger;
12
```

Tipos de Triggers

1. Row-Level Triggers
    a. Usa la cláusula FOR EACH ROW
    b. Permite el uso de NEW y OLD pseudo-records
    c. Se desencadena por cada fila
    d. NO se desencadena si no hay filas afectadas
2. Statement-Level Triggers
    a. Se desencade aunque no hayan filas afectadas
    b. Se ejecuta una sola vez, aunque la consulta modifique por ej. 0 ó 100 filas
    c. NO se puede utilizar NEW y OLD

OLD y NEW pseudo-record

- NEW nos permite acceder a los valores que serán actualizados o modificados
- OLD nos permite acceder a los valores que serán eliminados o modificados
- El pseudo-record :NEW se puede modificar directamente cuando utilizamos un TRIGGER que utiliza la cláusula BEFORE (ej. :NEW.salary := 2000;)
- Si utilizamos AFTER, :NEW es de sólo lectura

Uso de OLD y NEW

OPERACIÓN	VALOR OLD	VALOR NEW
Insert	NULL	Valores insertados o que se insertarán
Update	Valores anteriores a la actualización	Valores nuevos
Delete	Valores existentes en la tabla antes del DELETE	NULL

Ejemplo Row-Level Trigger

```
1  CREATE OR REPLACE TRIGGER t_empleado
2  AFTER UPDATE OF salario
3    ON empleado
4  FOR EACH ROW
5  BEGIN
6    IF :NEW.salario > 900000 THEN
7      RAISE_APPLICATION_ERROR(-20000, 'Salario muy elevado.');
8    END IF;
9
10   INSERT INTO
11     historia_empleado(empleado_id, salario_nuevo, salario_antiguo, fecha)
12     VALUES(:NEW.empleado_id, :NEW.salario, :OLD.salario, sysdate);
13 END t_empleado;
```

Ejemplo Statement-Level Trigger

```
1  CREATE OR REPLACE TRIGGER t_seguridad_horaria
2  BEFORE INSERT OR UPDATE OR DELETE
3    ON empleado
4  BEGIN
5    IF TO_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00' THEN
6      Raise_application_error(-20326,'Might not change '
7            ||'emp table during Nonworking hours');
8    END IF;
9  END t seguridad horaria;
```

Una Excepción en el Trigger causa el rollback

```
 1  CREATE OR REPLACE TRIGGER t_empleado
 2  BEFORE UPDATE
 3    ON empleado
 4  FOR EACH ROW
 5  BEGIN
 6    IF :NEW.salario > 900000 THEN
 7      RAISE_APPLICATION_ERROR(-20000, 'Salario muy elevado.');
 8    END IF;
 9
10    INSERT INTO historia_empleado(empleado_id, salario, fecha)
11      VALUES(:NEW.empleado_id, :NEW.salario, sysdate);
12  END t_empleado;
```

Trigger Predicates

```
1  CREATE OR REPLACE TRIGGER t_empleado_predicados
2    AFTER INSERT OR UPDATE OR DELETE ON empleado
3  FOR EACH ROW
4  BEGIN
5    -- predicado para saber cuando se inserta
6    IF INSERTING THEN
7      RAISE_APPLICATION_ERROR(-20001, 'No se puede insertar.');
8    -- predicado para saber cuando se actualiza
9    ELSIF UPDATING THEN
10     RAISE_APPLICATION_ERROR(-20002, 'No se puede actualizar.');
11   -- predicado para saber cuando se elimina
12   ELSIF DELETING THEN
13     RAISE_APPLICATION_ERROR(-20003, 'No se puede borrar.');
14   END IF;
15 END t_empleado_predicados;
```

Gestión de Triggers

```
1  -- Activa el trigger
2  ALTER TRIGGER t_empleado ENABLE;
3
4  -- Desactiva el trigger
5  ALTER TRIGGER t_empleado DISABLE;
6
7  -- Activa todos los TRIGGERS de la tabla EMPLEADO
8  ALTER TABLE empleado ENABLE ALL TRIGGERS;
9
10 -- Desactiva todos los TRIGGERS de la tabla EMPLEADO
11 ALTER TABLE empleado DISABLE ALL TRIGGERS;
12
13 -- Elimina el trigger t_empleado2
14 DROP TRIGGER t_empleado2;
```

FIN