UA4B. PL/SQL

4.17. Disparadores

Un disparador no es más que un procedimiento que es ejecutado cuando se realiza alguna sentencia de manipulación de datos sobre una tabla dada y bajo unas circunstancias establecidas a la hora de definirlo.

Por lo que un disparador puede ser usado para:

- Llevar a cabo auditorías sobre la historia de los datos en nuestra base de datos.
- Garantizar complejas reglas de integridad.
- Automatizar la generación de valores derivados de columnas.
- Etc.

Cuando diseñamos un disparador debemos tener en cuenta que:

- No debemos definir disparadores que dupliquen la funcionalidad que ya incorpora Oracle.
- Debemos limitar el tamaño de nuestros disparadores, y si estos son muy grandes codificarlos por medio de subprogramas que sean llamados desde el disparador.
- Cuidar la creación de disparadores recursivos.

Un disparador puede ser lanzado antes o después de realizar la operación que lo lanza. Por lo que tendremos disparadores **BEFORE** y disparadores **AFTER**.

Un disparador puede ser lanzado una vez por sentencia o una vez por cada fila a la que afecta. Por lo que tendremos disparadores de sentencia y disparadores de fila.

Un disparador puede ser lanzado al insertar, al actualizar o al borrar de una tabla, por lo que tendremos disparadores **INSERT**, **UPDATE** o **DELETE** (o mezclados).

Definición

Por lo visto anteriormente, para definir un disparador deberemos indicar si será lanzado antes o después de la ejecución de la sentencia que lo lanza, si se lanzará una vez por sentencia o una vez por fila a la que afecta, y si será lanzado al insertar y/o al actualizar y/o al borrar. La sintaxis que seguiremos para definir un disparador será la siguiente:

```
CREATE [OR REPLACE] TRIGGER nombre
momento acontecimiento ON tabla
[[REFERENCING (old AS alias_old|new AS alias_new)
FOR EACH ROW
[WHEN condicion]]
bloque_PL/SQL;
```

Donde nombre nos indica el nombre que le damos al disparador, momento nos dice cuando será lanzado el disparador (**BEFORE** o **AFTER**), acontecimiento será la acción que provoca el lanzamiento del disparador (**INSERT** y/o **DELETE** y/o **UPDATE**). **REFERENCING** y **WHEN** sólo podrán ser utilizados con disparadores para filas. REFERENCING nos permite asignar un alias a los valores **NEW** o/y **OLD** de las filas afectadas por la operación, y WHEN nos permite indicar al disparador que sólo sea lanzado cuando sea **TRUE** una cierta condición evaluada para cada fila afectada.

En un disparador de fila, podemos acceder a los valores antiguos y nuevos de la fila afectada por la operación, referenciados como **:old** y **:new** (de ahí que podamos utilizar la opción **REFERENCING** para asignar un alias). Si el disparador es lanzado al insertar, el valor antiguo no tendrá sentido y el valor nuevo será la fila que estamos insertando. Para un disparador lanzado al actualizar el valor antiguo contendrá la fila antes de actualizar y el valor nuevo contendrá la fila que vamos actualizar. Para un disparador lanzado al borrar sólo tendrá sentido el valor antiguo.

En el cuerpo de un disparador también podemos acceder a unos predicados que nos dicen qué tipo de operación se está llevando a cabo, que son: **INSERTING**, **UPDATING** y **DELETING**.

Un disparador de fila no puede acceder a la tabla asociada. Se dice que esa tabla está mutando. Si un disparador es lanzado en cascada por otro disparador, éste no podrá acceder a ninguna de las tablas asociadas, y así recursivamente.

```
CREATE TRIGGER prueba BEFORE UPDATE ON agentes
FOR EACH ROW
BEGIN
...
SELECT identificador FROM agentes WHERE ...
/*devolvería el error ORA-04091: table AGENTES is mutating, trig
...
END;
/
```

Si tenemos varios tipos de disparadores sobre una misma tabla, el orden de ejecución será:

- Triggers before de sentencia.
- Triggers before de fila.
- Triggers after de fila.
- Triggers after de sentencia.

Existe una vista del diccionario de datos con información sobre los disparadores:

```

USER_TRIGGERS;


SQL>DESC USER_TRIGGERS;
Name Null? Type
------------------------------ -------- ----
TRIGGER_NAME NOT NULL VARCHAR2(30)
TRIGGER_TYPE VARCHAR2(16)
TRIGGERING_EVENT VARCHAR2(26)
TABLE_OWNER NOT NULL VARCHAR2(30)
TABLE_NAME NOT NULL VARCHAR2(30)
REFERENCING_NAMES VARCHAR2(87)

```

```
WHEN_CLAUSE VARCHAR2(4000)
STATUS VARCHAR2(8)
DESCRIPTION VARCHAR2(4000)
TRIGGER_BODY LONG
```

Ejemplos

Ejemplo 1

Como un agente debe pertenecer a una familia o una oficina pero no puede pertenecer a una familia y a una oficina a la vez, deberemos implementar un disparador para llevar a cabo esta restricción que Oracle no nos permite definir.

Para este cometido definiremos un disparador de fila que saltará antes de que insertemos o actualicemos una fila en la tabla agentes, cuyo código podría ser el siguiente:

```
CREATE OR REPLACE TRIGGER integridad_agentes
BEFORE INSERT OR UPDATE ON agentes
FOR EACH ROW
BEGIN
     IF (:new.familia IS NULL and :new.oficina IS NULL) TH
          RAISE_APPLICATION_ERROR(-20201, 'Un agente no pu
     ELSIF (:new.familia IS NOT NULL and :new.oficina IS N
          RAISE_APPLICATION_ERROR(-20202, 'Un agente no pu
     END IF;
END;
/
```

Ejemplo 2

Supongamos que tenemos una tabla de históricos para agentes que nos permita auditar las familias y oficinas por la que ha ido pasando un agente. La tabla tiene la fecha de inicio y la fecha de finalización del agente en esa familia u oficina, el identificador del agente, el nombre del agente, el nombre de la familia y el nombre de la oficina. Queremos hacer un disparador que inserte en esa tabla.

Para llevar a cabo esta tarea definiremos un disparador de fila que saltará después de insertar, actualizar o borrar una fila en la tabla agentes, cuyo código podría ser el siguiente:

```
1  CREATE OR REPLACE TRIGGER historico_agentes
2  AFTER INSERT OR UPDATE OR DELETE ON agentes
3  FOR EACH ROW
4  DECLARE
5       oficina VARCHAR2(40);
6       familia VARCHAR2(40);
7       ahora DATE := sysdate;
8  BEGIN
9       IF INSERTING THEN
10           IF (:new.familia IS NOT NULL) THEN
11                SELECT nombre INTO familia FROM familias WH
12                oficina := NULL;
13           ELSIF
14                SELECT nombre INTO oficina FROM oficinas WH
15                familia := NULL;
16           END IF;
17           INSERT INTO histagentes VALUES (ahora, NULL, :ne
18           COMMIT;
19      ELSIF UPDATING THEN
20           UPDATE histagentes SET fecha_hasta = ahora WHERE
21           IF (:new.familia IS NOT NULL) THEN
22                SELECT nombre INTO familia FROM familias WH
23                oficina := NULL;
             ELSE
```

```
24                SELECT nombre INTO oficina FROM oficinas WH
25                familia := NULL;
26           END IF;
27           INSERT INTO histagentes VALUES (ahora, NULL, :ne
28           COMMIT;
29      ELSE
30           UPDATE histagentes SET fecha_hasta = ahora WHERE
31           COMMIT;
32      END IF;
33 END;
34 /
```

Ejemplo 3

Queremos realizar un disparador que no nos permita llevar a cabo operaciones con familias si no estamos en la jornada laboral.

```
1  CREATE OR REPLACE TRIGGER jornada_familias
2  BEFORE INSERT OR DELETE OR UPDATE ON familias
3  DECLARE
4       ahora DATE := sysdate;
5  BEGIN
6       IF (TO_CHAR(ahora, 'DY') = 'SAT' OR TO_CHAR(ahora, 'D
7            RAISE_APPLICATION_ERROR(-20301, 'No podemos mani
8       END IF;
9       IF (TO_CHAR(ahora, 'HH24') < 8 OR TO_CHAR(ahora, 'HH2
10           RAISE_APPLICATION_ERROR(-20302, 'No podemos mani
11      END IF;
12 END;
13 /
```