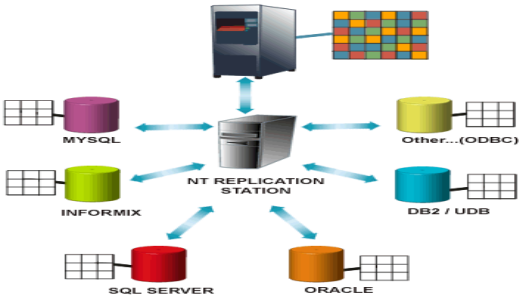
UNIVERSIDAD TECNOLOGICA DE PANAMA

FACULTAD DE INGENIERIA DE SISTEMAS

LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.

SISTEMA DE BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

TRIGGERS –PL/SQL ORACLE



CONTENIDO

Capítulo V. Implementación de Otros Objetos de servidor. (Procedimientos, almacenados, disparadores, cursares).

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Los disparadores se asemejan a los procedimientos y funciones en que son bloques PL/SQL nominados con secciones declarativas, ejecutable y de manejo de excepciones.

Al igual que los objetos de base de datos, los disparadores deben ser almacenados en la base de datos y no pueden ser locales a un bloque. Sin embargo, un procedimiento se ejecuta de manera explicita desde otro bloque mediante una llamada de procedimiento, que puede también usar argumentos.

Un disparador, por el contrario, se ejecuta de manera implícita cada vez que tiene lugar el suceso de disparo, y el disparador no admite argumentos. El acto de ejecutar disparador se conoce como disparo o trigger.

El suceso de disparo es una operación DML (INSERT, UPDATE o DELETE) sobre una tabla de la base de datos.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Los disparadores pueden ser usados para muchas cosas diferentes, incluyendo:

- El mantenimiento de restricciones de integridad compleja, que no sea posible con las restricciones declarativas definidas en el momento de crear la tabla.
- La auditoria de la información contenida en la tabla, registrando los cambios realizados y la identidad del que los llevo a cabo.
- El aviso automático a otros programas de que hay que llevar a cabo una determinada acción, cuando se realiza un cambio en la tabla.

Por ejemplo, supongamos que queremos mantener una serie de estadísticas acerca de las diferentes especialidades, incluyendo el numero de estudiantes matriculados y la cantidad total de créditos seleccionados. Se deberá almacenar estos resultados en la tabla *major_stats*:

```
CREATE TABLE major_stats (
    major           VARCHAR2(30),
    total_credits   NUMBER,
    total_students  NUMBER);
```

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Para poder mantener actualizada la información estadística actualizada en la tabla major_stats , crearemos un disparador para la tabla students que actualice mejor_stats cada vez que se modifique la tabla students.
Esto es lo que hace el disparados **UpdateMajorStats,** que mostramos a continuación:

```
CREATE OR REPLACE TRIGGER UpdateMajorStats
/* mantiene la tabla de especialidades major_stats actualizada, de acuerdo con los cambios realizados en la tabla estudiante students */
AFTER INSERT OR DELETE OR UPDATE ON  students
DECLARE c_Statistics IS
   SELECT major, count(*)  total_students, SUM(current_credits) total_credits
   FROM students
   GROUP BY major;
BEGIN
   /* Recorrer mediante un bucle cada especialidad. Intenta actualizar las estadisticas en major_stats que correspondan a dicha especialidad. Si la fila no existe , se creara. */
      FOR v_StatsRecord   IN  c_Statistics LOOP
          UPDATE  major_stats
             SET total_credits      = v_StatsRecord.total_credits,
                 total_students   = v_StatsRecord.total_students
          WHERE major = v_StatsRecord.major;
          -- Comprueba la existencia de la fila
           IF SQL%NOTFOUND THEN
              INSERT INTO major_stats (major, total_credits, total_students)
               VALUES (v_StatsRecord.major, v_StatsRecord.total_credits, v_StatsRecord.total_student);
           END IF;
      END LOOP;
END UpdateMajorStats;
```


5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La sintaxis general para crear un disparador es:

CREATE [OR REPLACE] TRIGGER *nombre_disparador*
{BEFORE| AFTER} *suceso_disparo* ON *referencia_tabla*
[FOR EACH ROW [*condicion_disparo*]]
cuerpo_dispardor;

Donde *nombre_disparador* corresponde al nombre del disparador, *suceso_disparo* especifica cuando se activa el disparador (en el caso del *trigger* anterior **UpdateMajorStats**, después de cualquier operación DML), *referencia_tabla* es la tabla para la cual se define el disparador y *cuerpo_disparador* es código principal del disparador. Antes se evalúa la condición disparo en la clausula WHEN, si es que esta presente. El cuerpo des disparador se ejecuta solo cuando dicha condición se evalúa como verdadera.

Componentes de un disparador
Los componentes requeridos en un disparador son *el nombre*, el *suceso de disparo* y el cuerpo. La clausula WHEN es opcional.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Nombre de los disparadores

El espacio de nombres para los disparadores es diferente del de los otros subprogramas. El espacio de nombre es un conjunto de identificadores validos que pueden emplearse como nombre de un objeto.

Los procedimientos, las funciones, paquetes y tablas tienen el mismo espacio de nombre, lo que quiere decir, que dentro de un esquema de base de datos, todos los objetos de cualquiera de dichos tipos deben tener nombre diferentes. Es ilegal, por ejemplo dar el mismo nombre a un procedimiento y a un paquete.

Los disparadores, sin embargo tienen un espacio de nombres separado, por lo que un disparador puede tener el mismo nombre que una tabla o que un procedimiento. Dentro de un esquema de base de datos todos los disparadores deben tener nombres diferentes entre si. Estos nombres son identificadores de base de datos y por lo tanto deben seguir las misma reglas que resto de los identificadores.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Nombre de los disparadores

Ejemplo, podemos crear un disparador llamado *major_stats* sobre la tabla *major_stats*, pero es ilegal que un procedimiento también se llame *major_stats*.

```
SQL> CREATE OR REPLACE TRIGGER major_stats
        BEFORE  INSERT ON major_stats
     BEGIN
          INSERT INTO temp_table( char_col)
              VALUES ( 'Trigger called!');
      END major_stats;
       /
    Trigger Created.

SQL> CREATE OR REPLACE  PROCEDURE major_stats AS
     BEGIN
          INSERT INTO temp_table( char_col)
              VALUES ( 'Trigger called!');
      END major_stats;
       /
    CREATE OR REPLACE PROCEDURE  major_stats AS
     *
    Error at line 1:
    ORA-00955  name is already used  by an existing object
```

Aunque es posible usar el mismo nombre para un disparador que para una tabla, no es recomendable. Es mejor dar a cada trigger un nombre diferente que identifique su función como la tabla sobre la cual se define.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Tipos de Disparadores

Los sucesos de disparos determina el tipo de disparador. Los disparadores pueden definirse para las operaciones de INSERT, UPDATE o DELETE y pueden dispararse antes o después de la operación. El nivel de los disparadores puede ser la fila o la orden.

Los valores de la orden, de la temporización y de nivel determinan el tipo de disparador. Hay un total de 12 tipos posibles: 3 ordenes por opciones de de temporización, por 2 niveles. Todos los siguientes tipos de disparadores por ejemplo, son validos:

- *Previo a la actualización y con nivel de orden*
- *Posterior a la inserción y con nivel de fila*
- *Previo al borrado con el nivel de fila.*

Puede definirse hasta 12 disparadores en una tabla, una de cada tipo, sin embargo, una tabla puede tener mas de un disparador de cada tipo. Esta capacidad permite cuantos disparadores se quiera para una tabla.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Tipos de Disparadores

Un disparador también puede activarse para mas de un tipo de orden. El disparador **UpdateMajorStats,** por ejemplo se activa con las ordenes **INSERT, UPDATE y DELETE**. El suceso de disparo especifica uno o varias operaciones DML que den activar el disparador.

Categoría	Valores	Comentario
Orden	**INSERT, UPDATE, DELETE**	Define que tipo orden DML provoca la activación del disparador.
Temporización	**BEFORE o AFTER**	Define si el disparador se activa antes o después de que se ejecute la orden (disparador previo o posterior)
Nivel	Fila u Orden	Los disparadores con nivel de fila se activan una vez por cada fila afectada por la orden que provocó el disparo. Los disparadores con nivel de orden se activan solo una vez, antes o después de la orden. Los disparadores con nivel de fila se identifican por la clausula **FOR EACH ROW** en la definición del disparador.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Restricciones de los Disparadores

El cuerpo de un disparador es bloque PL/SQL. Cualquier orden que sea legal en un bloque PL/SQL, es legal en el cuerpo de un disparador, con las siguientes restricciones:

- Un disparador no puede emitir ninguna orden de control de transacciones: COMMIT, ROLLBACK o SAVEPOINT. El disparador se activa como parte de la ejecución de la orden que provoco el disparo, y forma parte de la misma transacción que dicha orden. Cuando la orden que provoca el disparo es confirmada o cancelada, se confirma o cancela también el trabajo realizado por el disparador.
- Por razones idénticas, ningún procedimiento o función llamado por el disparador puede emitir ordenes de control de transacciones.
- El cuerpo del disparador no puede contener ninguna declaración de variables LONG o LONG RAW. Asimismo a columnas de tipo LONG o LONG RAW de la tabla sobre la que se define el disparador.
- Existen restricciones acerca de a que tablas puede acceder el cuerpo de un disparador. Dependiendo del tipo de disparador y de las restricciones que afecten a las tablas, dichas tablas pueden ser mutantes.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Los Disparadores Y el Diccionario de Datos.

De forma similar a lo que sucede con los subprogramas almacenados, algunas vistas del diccionario de datos contienen información acerca de los disparadores y de su estado. Estas vistas se actualizan cada vez que se crea o elimina un disparador.

Vista del diccionario de datos

Cuando se crea un disparador, su código fuente se almacena en la vista ***user_triggers*** del diccionario de datos. Esta vista incluye el cuerpo de disparador, la clausula WHEN, la tabla de disparo y el tipo de disparador. La consulta siguiente, por ejemplo, devuelve información acerca de ***UpdateMajorstats***:

```
SQL> SELECT trigger_type, table_name, triggering_event
       FROM user_triggers
       WHERE triggers_name = 'UPDATEMAJORSTATS';
       /
```

TRIGGER_TYPE	TABLE_NAME	TRIGGERING_EVENT
AFTER STATEMENT	STUDENTS	INSERT OR UPDATE OR DELETE

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Eliminación y deshabilitación de los Disparadores.

Al igual que los procedimientos y paquetes, también los disparadores pueden eliminarse. La sintaxis de la orden que realiza esta operación es:

DROP TRIGGER *nombre_disparador*;

Donde nombre_disparador es el nombre del disparador a eliminar. Esta orden elimina el trigger del diccionario datos de forma permanente. Al igual que los subprogramas, puede incluirse la clausula OR REPLACE en la orden CREATE de creación del disparador.

A diferencia de los procedimientos y paquetes, sin embargo se puede deshabilitar un disparador sin necesidad de eliminarlo. Cuando se deshabilita el trigger, continua existiendo en el diccionario de datos, pero no puede llegar a dispararse.

Para desactivar un disparador, se utiliza la orden **ALTER TRIGGER**,

ALTER TRIGGER nombre_disparador **{DISABLE | ENABLE};**

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Eliminación y dehabilitación de los Disparadores.

ALTER TRIGGER nombre_disparador **{DISABLE | ENABLE};**

Donde nombre_disparador es el nombre del disparador. Todos los disparadores están en principio, habilitados en el momento de la creación. ALTER TRIGGER puede deshabilitar y luego habilitar, cualquier disparador.

Ejemplos:

SQL> ALTER TRIGGER UpdateMajorStats DISABLE;
Trigger altered.

SQL> ALTER TRIGGER UpdateMajorStats ENABLE;
Trigger altered.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Eliminación y dehabilitación de los Disparadores.

También se puede habilitar o deshabilitar todos los disparadores de una tabla determinada, utilizando la orden ALTER TABLE con la clausula **ENABLE ALL TRIGGERS o DISABLE ALL TRIGGERS.**

Ejemplos:

SQL> ALTER TABLE students
ENABLE ALL TRIGGERS;
Table altered.

SQL> ALTER TABLE students
DISABLE ALL TRIGGERS;
Trigger altered.

La columna status del user_triggers contiene el valor 'ENABLE' o ''DISABLE', indicando el estado actual de cada disparador. Al deshabilitar el trigger no lo elimina del diccionario de datos, como si lo haría la operación de borrado

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

Los disparadores se activan al ejecutar la orden DML. Algoritmo de ejecucion de una orden DML es el siguiente:

1. Ejecutar, si existe, el disparador de tipo BEFORE (disparador previo) con nivel de orden.
2. Para cada fila a la que afecte la orden:
   a) Ejecutar, si existe, el disparador BEFORE con nivel de fila.
   b) Ejecutar la propia orden.
   c) Ejecutar, si existe, el disparador de tipo AFTER (disparador posterior) con nivel de fila.
3. Ejecutar, si existe, el disparador de tipo AFTER con nivel de orden.

El ejemplo a continuación…..

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

Los disparadores se activan al ejecutar la orden DML. Algoritmo de ejecucion de una orden DML es el siguiente:

1. Ejecutar, si existe, el disparador de tipo BEFORE (disparador previo) con nivel de orden.
2. Para cada fila a la que afecte la orden:
   a) Ejecutar, si existe, el disparador BEFORE con nivel de fila.
   b) Ejecutar la propia orden.
   c) Ejecutar, si existe, el disparador de tipo AFTER (disparador posterior) con nivel de fila.
3. Ejecutar, si existe, el disparador de tipo AFTER con nivel de orden.

El ejemplo a continuación…..

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.
El ejemplo a continuación…..

```
CREATE SEQUENCE trigger_seg
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER  classes_BStatement
    BEFORE UPDATE ON classes
BEGIN
      INSERT INTO temp_table (num_col, char_col)
      VALUES (trigger_seg.NEXTVAL, 'Before Statement trigger');
END classes_BStatement;

CREATE OR REPLACE TRIGGER  classes_AStatement
    AFTER UPDATE ON classes
BEGIN
      INSERT INTO temp_table (num_col, char_col)
      VALUES (trigger_seg.NEXTVAL, 'After Statement trigger');
END classes_AStatement;
```

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.
El ejemplo a continuación…..

```
CREATE OR REPLACE TRIGGER  classes_BRow
    BEFORE UPDATE ON classes
    FOR EACH ROW
BEGIN
      INSERT INTO temp_table (num_col, char_col)
      VALUES (trigger_seg.NEXTVAL, 'Before Row trigger');
END classes_BRow;

CREATE OR REPLACE TRIGGER  classes_ARow
    AFTER UPDATE ON classes
    FOR EACH ROW
BEGIN
      INSERT INTO temp_table (num_col, char_col)
      VALUES (trigger_seg.NEXTVAL, 'After Row trigger');
END classes_ARow;
```


5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

Suponga que, a continuación, ejecutamos la siguiente orden UPDATE:

```
UPDATE classes
    SET  num_credits  = 4
WHERE  department    IN  ('HIS' ,  'CS', 'CA');
```

Esta orden afecta a cuatro filas. Los disparador previo y posterior con nivel de orden se ejecutan una sola vez, y los disparadores previo y posterior con nivel de fila se ejecutan cuatro veces cada uno. Si efectuamos entonces una selección sobre temp_table, obtendremos:

```
SQL> SELECT  * FROM temp_table
          ORDER BY cum_col;
```

NUM_COL	CHAR_COL
1	Before Statement trigger
2	Before Row trigger
3	After Row trigger
4	Before Row trigger
..	.....
10	After Statement trigger

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La Cláusula WHEN

La clausula **WHEN** solo es valida para disparadores con nivel de fila. Si esta presente, el cuerpo del disparador solo se ejecutara para las filas que cumplan con la condicion especificada en la clausula. La clausula **WHEN** tiene la forma :

WHEN *condición*

Donde *condición* es la expresión booleana que será evaluada para cada fila.

```
CREATE OR REPLACE TRIGGER CheckCredits
   BEFORE INSERT OR UPDATE OF  current_credits ON students
   FOR EACH ROW
   WHEN  (new.current_credits >  20)
BEGIN
   /*  Aquí estaría  el cuerpo el disparador  */
END;
```