1. Disparadors o triggers

NF4. Disparadors i paquets

UF3 - Llenguatges SQL: DCL i extensió procedimental

Desenvolupament d'Aplicacions Multiplataforma
M02 – Bases de dades. Versió 1.0

Excepcions

1.1. PL/SQL avanzado

1.2. Disparadores o triggers

1.2.1. Declaración de un disparador o trigger

1.2.2. Elementos de un disparador o trigger

1.2.3. Creación de un disparador o trigger

1.2.4. Tratamiento de los disparador es o triggers

1.2.5. Uso de los disparadores o triggers

1.2.6. Uso de los pseudo-registros :OLD y :NEW en los disparadores a nivel de fila

1.2.7. Ejemplo

1.2.8. La cláusula WHEN

1.2.9. Uso de los predicados INSERTING, UPDATING y DELETING en los disparadores o triggers

1.3. Tablas mutantes

1.1. PL/SQL avanzado

Tras estudiar la introducción al PL/SQL, estudiando cómo se pueden programar procedimientos y funciones introduciendo sentencias SQL; vamos a dar un paso más y vamos a profundizar en el PL/SQL estudiando los paquetes y los triggers (disparadores).

Paquete:

es una sección declarativa nominada y cualquier cosa que se pueda incluir en esta sección, se puede incluir en un paquete: procedimientos, funciones, cursores, tipos y variables.

Triggers o **disparadores** de la base de datos:

son bloques PL/SQL almacenados asociados a una tabla que se ejecutan o se disparan automáticamente cuando se producen ciertos eventos o sucesos (inserción, borrado o modificación de filas) que afectan a la tabla. También podemos programar triggers de la aplicación.

1.2. Disparadores o triggers

Los triggers o disparadores de la base de datos son bloques PL/SQL almacenados asociados a una tabla que se ejecutan o se disparan automáticamente cuando se producen ciertos eventos o sucesos (inserción, borrado o modificación de filas) que afectan a la tabla. También podemos programar triggers de la aplicación.

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

Estos disparadores se han de almacenar en la base de datos y no pueden ser locales a un bloque. Además, se ejecutan de manera implícita cada vez que tiene lugar el suceso de disparo.

A diferencia de un procedimiento o una función (que admite argumentos o parámetros), un disparador no admite parámetros.

1.2.1. Declaración de un disparador o trigger

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
{BEFORE|AFTER} {DELETE|INSERT|UPDATE OF lista_campos}
[OR {BEFORE|AFTER} {DELETE|INSERT|UPDATE OF lista_campos]
ON nombre_tabla
[FOR EACH {STATEMENT | ROW [WHEN condición]}]

//BLOQUE PL/SQL
  acciones
[EXCEPTION
  gestión de excepciones]
END;
/
```

1.2.2. Elementos de un disparador o trigger

Los componentes que se pueden distinguir en los triggers, son los siguientes:

- **Nombre del trigger.**

  Un identificador de un trigger puede recibir el mismo nombre que una tabla o un procedimiento, ya que, el espacio donde almacenan los identificadores de los disparadores está separado de la zona donde se almacena los identificadores de la tabla, procedimientos, etc.

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
```

- **Momento del trigger**:

  Indica cuándo se ejecuta el trigger con relación al evento. Es decir, si se ejecuta antes del evento DML (BEFORE) o después del evento (AFTER).

  Además, también contamos con la cláusula `INSTEAD OF`; en este caso el código del cuerpo del trigger se ejecutará en vez de la sentencia. Se usa para modificar VISTAS.

```
{BEFORE|AFTER} {DELETE|INSERT|UPDATE OF lista_campos}
[OR {BEFORE|AFTER} {DELETE|INSERT|UPDATE OF lista_campos]
```

- **Evento del trigger**:

  Es el suceso que producirá la ejecución del trigger y siempre es una sentencia DML (*Manipulación de datos*): INSERT, DELETE o UPDATE.

  Si la sentencia es de modificación (UPDATE) se puede especificar opcionalmente las columnas cuya modificación producirá el disparo.

```
{BEFORE|AFTER} {DELETE|INSERT|UPDATE OF lista_campos}
[OR {BEFORE|AFTER} {DELETE|INSERT|UPDATE OF lista_campos]
```

- **Tipo del trigger**:

  Indica cuántas veces se ejecuta el cuerpo del trigger, si se ejecuta por filas (ROW) o se ejecuta por sentencias (STATEMENT).

  Por defecto, el tipo de comportamiento de un trigger es statement. Los disparadores con nivel de filas se identifican por la cláusula FOR EACH NOW en la definición del disparador.

```
ON nombre_tabla
[FOR EACH {STATEMENT | ROW [WHEN condición]}]
```

- **Cuerpo del trigger**:

  Representa la acción que realizará el trigger, y es simplemente un bloque PL/SQL anónimo.

```
//BLOQUE PL/SQL
  acciones
[EXCEPTION
  gestión de excepciones]
END;
/
```

1.2.3. Creación de un disparador o trigger

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
tiempo evento1 [OR evento2 OR evento3]
ON nombre_tabla
<bloque PL/SQL>;
```

donde:

`nombre_disparador` ⇨ nombre que recibe el disparador o trigger.

`tiempo` ⇨ indica el momento en el que se ejecuta el disparador en relación al evento del mismo (`BEFORE` o `AFTER`).

`evento` ⇨ indica la operación de manipulación de datos que hace que el disparador se ejecute (`INSERT`, `DELETE` o `UPDATE`).

`nombre_tabla` ⇨ nombre de la tabla a la que se asocia el trigger.

`bloque PL/SQL` ⇨ cuerpo del disparador que define la acción a realizar.

1.2.4. Tratamiento de los disparadores o triggers

- Activar / desactivar un trigger de la base de datos, es el siguiente:

```
ALTER TRIGGER nombre_disparador DISABLE | ENABLE
```

- Activar / desactivar todos los triggers de una tabla, es el siguiente:

```
ALTER TRIGGER nombre_disparador DISABLE | ENABLE ALL TRIGGERS
```

- Recompilar explícitamente un trigger para una tabla:

```
ALTER TRIGGER nombre_disparador COMPILE
```

- Borrar un trigger de una base de datos:

```
DROP TRIGGER nombre_disparador
```

1.2.5. Uso de los disparadores o triggers

Los triggers o disparadores se pueden emplear para:

- Mantener restricciones de integridad complejas, que no son posible definirlas en el momento de crear la tabla.
- Auditar la información contenida en una tabla, registrando los cambios realizados y la identidad que los llevó a cabo.
- Avisar automáticamente a otros programas que hay que llevar acabo una acción, a la hora de realizar un cambio en la tabla.

1.2.6. Uso de los pseudo-registros :OLD y :NEW en los disparadores a nivel de fila

Recordemos que un disparador a nivel de fila, se ejecuta una vez por cada fila procesada por la orden que provoca el disparo.

Para acceder a la fila que se está procesando hemos de usar los siguientes pseudo-registros: `:OLD` y `:NEW`.

Orden del trigger	:OLD	:NEW
INSERT	No está definido. Todos los campos toman el valor NULL.	Valores que serán insertados cuando se complete la orden.
UPDATE	Valores originales de la fila, antes de actualizar.	Nuevos valores que serán escritos cuando se complete la orden.
DELETE	Valores originales, antes del borrado de la fila.	No está definido. Todos los campos toman el valor NULL.

1.2.7. Ejemplo

Teniendo en cuenta la tabla de DEPARTAMENTO, con los siguientes campos:

```
deptno  NUMBER(2)
dname   VARCHAR2(14)
loc     VARCHAR2(13)
```

Vamos a crear un disparador que nos controle a nivel de fila que se introduzca forzosamente el lugar al que pertenece el departamento, es decir, el campo lugar (LOC) no podrá ser nulo.

```
Rem BORRADO DEL DISPARADOR
DROP TRIGGER lugar_no_nulo;

Rem CREAR EL DISPARADOR A NIVEL DE FILA (lugar no nulo)
CREATE TRIGGER lugar_no_nulo
 BEFORE INSERT ON dept
 FOR EACH ROW
 BEGIN
 IF :NEW.loc IS NULL THEN
   RAISE_APPLICATION_ERROR(-20001,'ERROR: LUGAR DEL DEPT. NULO');
 END IF;
 END;
/
```

Para comprobar el disparador del ejemplo1: Inserción de dos registros en la tabla DEPARTAMENTO, donde el primero introducimos todos los datos y el segundo, el lugar es nulo.

```
Rem INSERTAR UN DEPARTAMENTO SIN PROVOCAR NINGUN ERROR
INSERT INTO dept (deptno, dname, loc)
 VALUES (60,'ECONOMISTA','BCN');

Rem INSERTAR UN DEPARTAMENTO DONDE LUGAR SERÁ NULO
INSERT INTO dept (deptno, dname)
 VALUES (60,'PROGRAMACION');
```

1.2.7. Creación de un trigger a nivel de registro

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
tiempo evento1 [OR evento2 OR evento3]
ON nombre_tabla
[REFERENCING OLD as old | NEW AS new]
FOR EACH ROW
[WHEN condición]
 <bloque PL/SQL>;
```

1.2.8. La cláusula WHEN

La cláusula WHEN sólo es válida para trigger a nivel de fila y por tanto, se ejecutará para las filas que cumplan la condición especificada en dicha cláusula. La sintaxis de la cláusula es la siguiente:

```
WHEN condición
```

donde,

condición ⇒ es una expresión booleana que se evaluará para cada fila. En esta condición también se puede hacer referencia a los registros `:OLD` y `:NEW`, pero eliminando los dos puntos. Los dos puntos sólo serán válidos en el cuerpo del trigger.

```
Rem BORRADO DEL DISPARADOR
DROP TRIGGER lugar_no_nulo_WHEN;

Rem CREAR EL DISPARADOR A NIVEL DE FILA (lugar no nulo)
CREATE TRIGGER lugar_no_nulo_WHEN
 BEFORE INSERT ON dept
 FOR EACH ROW
 WHEN (new.dname IS NULL)
 BEGIN
 IF :NEW.loc IS NULL THEN
   RAISE_APPLICATION_ERROR(-20001,'ERROR: LUGAR DEL DEPT. NULO');
 END IF;
 END;
/
```

Para comprobar el anterior disparador:

```
Rem INSERTAR UN DEPARTAMENTO SIN PROVOCAR NINGUN ERROR
INSERT INTO dept (deptno, dname, loc)
 VALUES (80,UPPER('Analista'),UPPER('madrid'));

Rem INSERTAR UN DEPARTAMENTO DONDE LUGAR SERÁ NULO
INSERT INTO dept (deptno, dname)
 VALUES (90,'PROGRAMACION');

Rem INSERTAR UN DEPARTAMENTO DONDE NOMBRE Y LUGAR SERÁ NULO
INSERT INTO dept (deptno)
 VALUES (99);
```

1.2.9. Uso de los predicados INSERTING, UPDATING y DELETING en los disparadores o triggers

En un trigger podemos controlar las diferentes órdenes DML y contamos con tres funciones booleanas que pueden emplearse para determinar de qué operaciones se trata. Estos predicados son:

`INSERTING` ⇒ devuelve CIERTO si la orden de disparo es INSERT y FALSO en caso contrario.

`UPDATING` ⇒ devuelve CIERTO si la orden de disparo es UPDATE y FALSO en caso contrario.

`DELETING` ⇒ devuelve CIERTO si la orden de disparo es DELETE y FALSO en caso contrario.

Ejemplo los predicados INSERTING, UPDATING y DELETING :

La idea es crear un trigger que nos dé el mensaje correcto, teniendo en cuenta que:

• Cuando se inserta un registro (`INSERTING` es cierto), ha de decir:

"SE HA INTRODUCIDO UN REGISTRO"

• Cuando se modifique un registros (`UPDATING` es cierto), ha de decir:

"SE HA MODIFICADO UN REGISTRO"

• Cuando se elimine un registro y el nombre del departamento es diseñador (`DELETING` es cierto y se cumple la condición), ha de decir:

"BORRADO EL DEPARTAMENTO DISEÑADOR"

En caso contrario:

"SE HA ELIMINADO UN REGISTRO"

Y luego crearemos un script para comprobar que funciona el script, de la siguiente manera:

- Insertar el siguiente registro:

Deptno	Dname	Loc
90	DISEÑADOR	BCN

- Modificar el campo LOC y ponerlo en mayúsculas.
- Eliminar el departamento número 90 y el departamento número 70.

```
Rem BORRADO DEL DISPARADOR
DROP TRIGGER departamento;

Rem CREAR EL DISPARADOR
CREATE TRIGGER departamento
 BEFORE INSERT OR DELETE OR UPDATE OF loc ON dept
 FOR EACH ROW
 BEGIN
 IF INSERTING THEN
  DBMS_OUTPUT.PUT_LINE ('SE HA INTRODUCIDO UN REGISTRO');
 END IF;
 IF UPDATING THEN
  DBMS_OUTPUT.PUT_LINE ('SE HA MODIFICADO UN REGISTRO');
 END IF;
 IF DELETING THEN
  IF :old.dname = UPPER('DISEÑADOR') THEN
   DBMS_OUTPUT.PUT_LINE ('SE HA BORRADOR EL DEPT. DISEÑADOR');
  ELSE
   DBMS_OUTPUT.PUT_LINE ('SE HA ELIMINADO UN REGISTRO');
 END IF;
 END;
/
```

Para comprobar el anterior disparador:

```
Rem INSERTAR UN DEPARTAMENTO INSERT INTO dept (deptno, dname,
loc)
 VALUES (90,UPPER('Diseñador'),UPPER('madrid'));

Rem MODIFICAR
UPDATE dept
 SET loc = UPPER(loc);

Rem ELIMINAR
DELETE FROM dept
WHERE deptno = 90;

DELETE FROM dept
WHERE deptno = 70;
```

1.3. Tablas mutantes

Una tabla mutante (*mutating table*) es una tabla que está modificándose actualmente por una orden DML (*Lenguaje de Manipulación de Datos*).

Una tabla de restricción (constraing table) es una tabla de la que puede ser necesario consultar para una restricción de integridad referencial.

Preguntes!!!!!