

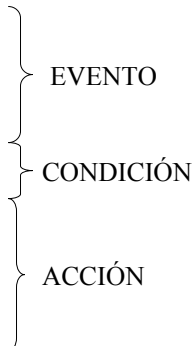
DISPARADORES EN SQL

Modelos Avanzados de Bases de
Datos

Curso 2004/2005

SINTAXIS GENERAL DE UN DISPARADOR EN SQL:2003

```
CREATE [OR REPLACE] TRIGGER nombre  
  [temporalidad del evento]  
  [granularidad del evento]  
  [WHEN condición]  
BEGIN  
  cuerpo del trigger  
END;
```



EVENTO

CONDICIÓN

ACCIÓN

SINTAXIS GENERAL DE UN DISPARADOR EN SQL:2003

- Temporalidad del evento

- BEFORE Operación
- AFTER Operación

Operación: INSERT, DELETE O UPDATE

Ej: AFTER DELETE ON nombre_tabla
AFTER DELETE OF nombre_columna ON nombre_tabla

SINTAXIS GENERAL DE UN DISPARADOR EN SQL:2003

- Granularidad del evento

- FOR EACH ROW
- FOR EACH STATEMENT

SINTAXIS GENERAL DE UN DISPARADOR EN SQL:2003

- WHEN condición (sólo para disparadores a nivel de fila)
 - Operadores relacionales:
 $<$ \leq $>$ \geq $=$ \neq
 - Operadores lógicos:
AND, OR, NOT

SINTAXIS GENERAL DE UN DISPARADOR EN SQL:2003

WHEN condición

- OLD
- NEW

Si están el en cuerpo del disparador se
referencian como :OLD ó :NEW

SINTAXIS GENERAL DE UN DISPARADOR EN SQL:2003

- Con OLD.nombre_columna referenciamos:
 - al valor que tenía la columna antes del cambio debido a una modificación (UPDATE)
 - al valor de una columna antes de una operación de borrado sobre la misma (DELETE)
 - al valor NULL para operaciones de inserción (INSERT)
- Con NEW.nombre_columna referenciamos:
 - Al valor de una nueva columna después de una operación de inserción (INSERT)
 - Al valor de una columna después de modificarla mediante una sentencia de modificación (UPDATE)
 - Al valor NULL para una operación de borrado (DELETE)

EJEMPLO

```
CREATE TRIGGER Ejemplo-fila
AFTER DELETE OF codigo ON tabla1
FOR EACH ROW
WHEN ((OLD.nombre='pepe') OR (OLD.edad > 35))
BEGIN
    DELETE FROM tabla2 WHERE tabla2.cod=:OLD.cod;
END Ejemplo-columna;
/
```

EJEMPLO

```
CREATE TRIGGER Ejemplo_sentencia
AFTER DELETE ON tabla1
REFERENCING OLD AS anterior
BEGIN
    DELETE FROM tabla2 WHERE
    tabla2.cod=anterior.cod;
END Ejemplo_sentencia;
/
```

ACTIVAR/DESACTIVAR DE DISPARADORES

- Todos los disparadores asociados a una tabla:
 - ALTER TABLE nombre_tabla ENABLE ALL TRIGGERS
 - ALTER TABLE nombre_tabla DISABLE ALL TRIGGERS
- Un disparador específico:
 - ALTER TRIGGER nombre_disparador ENABLE
 - ALTER TRIGGER nombre_disparador DISABLE

OTRAS FUNCIONES

- Eliminar un disparador
 - DROP TRIGGER nombre_disparador
- Ver todos los disparadores y su estado
 - SELECT TRIGGER_NAME , STATUS FROM USER_TRIGGERS;

OTRAS FUNCIONES

- Ver el cuerpo de un disparador
 - SELECT TRIGGER_BODY FROM USER_TRIGGERS WHERE TRIGGER_NAME='nombre_disparador';
- Ver la descripción de un disparador
 - SELECT DESCRIPTION FROM USER_TRIGGERS WHERE TRIGGER_NAME= 'nombre_disparador';

Funciones del cuerpo del disparador

- Inserting, Deleting, Updating

```
CREATE OR REPLACE TRIGGER ejemplo
BEFORE INSERT OR UPDATE OR DELETE ON tabla
BEGIN
    IF DELETING THEN
        Acciones asociadas al borrado
    ELSIF INSERTING THEN
        Acciones asociadas a la inserción
    ELSE
        Acciones asociadas a la modificación
    END IF;
END ejemplo;
/
```

Funciones del cuerpo del disparador

- RAISE_ERROR_APPLICATION

(nro_error, mensaje); [-20000 y -20999]

```
CREATE OR REPLACE TRIGGER ejemplo
BEFORE DELETE ON tabla
FOR EACH ROW
BEGIN
    IF tabla.columna= valor_no_borrable THEN
        RAISE_APPLICATION_ERROR(-20000,'La fila
        no se puede borrar');
    END IF;
    ...
END ejemplo;
```

Declaración de variables

```
CREATE...  
BEFORE...  
[FOR EACH ROW ...]  
DECLARE  
    Declaración de variables  
BEGIN  
...  
Nombre CONSTANT NUMBER:=valor;  
nombre TIPO;  
nombre nombretabla.nombrecolumna%TYPE;  
nombre nombretabla%ROWTYPE;
```

Restricciones de los disparadores

- Tabla mutante. La que está siendo modificada por una operación DML o una tabla que se verá afectada por los efectos de un DELETE CASCADE debido a la integridad referencial
- Tabla de restricción. Tabla de la que un disparador puede necesitar leer debido a una restricción de integridad referencial

Restricciones de los disparadores

- Las órdenes del cuerpo de un disparador no pueden:
 - Leer o modificar una tabla mutante
 - Leer o modificar claves primarias o ajenas de una tabla de restricción
- Para evitar estos problemas es necesaria la utilización de paquetes

Restricciones de los disparadores

```
CREATE OR REPLACE TRIGGER chequear_salario
BEFORE INSERT OR UPDATE ON empleado
FOR EACH ROW
WHEN (new.trabajo<>'presidente')
DECLARE
    v_salariomin empleado.salario%TYPE;
    v_salariomax empleado.salario%TYPE;
BEGIN
    SELECT MAX(salario), MIN(salario) FROM empleado
        INTO v_salariomin, v_salariomax
    FROM empleado
    WHERE trabajo=:new.trabajo;
    IF :new.salario<v_salariomin OR :new.salario> v_salariomax THEN
        RAISE_APPLICATION_ERROR(-20001, 'Sueldo fuera de rango');
    END IF;
END chequear_salario;
/

UPDATE empleado SET salario=1500 WHERE
nom_emp='Cortecero';
```

Restricciones de los disparadores

Dos tablas: empleado y departamento. En la tabla empleado tenemos el número de departamento como clave ajena.

```
CREATE TRIGGER ejemplo
AFTER UPDATE ON nrodep OF departamento
FOR EACH ROW
BEGIN
    UPDATE empleado
    SET empleado.dep = :NEW.nrodep
    WHERE empleado.dep= :OLD.nrodep;
END ejemplo;
/
```

```
UPDATE departamento
SET nrodep= 1
WHERE nrodep=7;
```

Disparadores de sustitución

- **INSTEAD OF.** Sólo para vistas

```
CREATE VIEW vista AS
SELECT edificio, sum(numero_asientos) FROM
habitaciones
GROUP BY edificio;
```

Es ilegal hacer una operación de borrado directamente en la vista:

```
DELETE FROM vista WHERE edificio='edificio 7';
```

Disparadores de sustitución

```
CREATE TRIGGER borra_en_vista
  INSTEAD OF DELETE ON vista
  FOR EACH ROW
  BEGIN
    DELETE FROM habitaciones
    WHERE edificio = :OLD.edificio;
  END borra_en_vista;
/
```

Paquetes y resolución del
problema de las tablas mutantes

Paquetes y tablas mutantes

- Una tabla es mutante sólo para los disparadores a nivel de fila
- No se puede usar, sin más, un disparador a nivel de orden, porque, por lo general, necesitamos acceder a valores que han sido modificados (de ahí el problema de las tablas mutantes)
- La solución consiste en crear dos disparadores, uno a nivel de fila y otro a nivel de orden

Paquetes y tablas mutantes

- En el disparador con nivel de fila almacenamos (en una estructura de datos apropiada) los datos que queremos consultar (los que provocan el error de tabla mutante)
- En el disparador con nivel de orden realizamos la consulta (pero sobre los datos almacenados en lugar de sobre la tabla)
- La mejor forma de almacenar los valores es en una tabla PL/SQL y aunar todas las operaciones descritas dentro de un paquete

Paquetes y tablas mutantes

```
CREATE OR REPLACE TRIGGER limite_especialidad(  
BEFORE INSERT OR UPDATE OF especialidad ON estudiantes  
FOR EACH ROW  
DECLARE  
maxEstudiantes CONSTANT NUMBER:=5;  
EstudiantesActuales NUMBER;  
BEGIN  
SELECT COUNT(*) INTO EstudiantesActuales  
FROM estudiantes  
WHERE especialidad = :new.especialidad;  
IF EstudiantesActuales+1>maxEstudiantes THEN  
    RAISE_APPLICATION_ERROR(-20000, 'Demasiados  
    estudiantes en la especialidad: '||  
    :new.especialidad);  
END IF;  
END limite_especialidad;
```

Paquetes y tablas mutantes

Si ejecutamos

```
UPDATE estudiantes  
SET especialidad = 'Historia'  
WHERE id=1003;
```

Nos da un error de tabla mutante

Para arreglarlo definimos el siguiente paquete:

Paquetes y tablas mutantes

Creamos el paquete:

```
CREATE OR REPLACE PACKAGE DatosEstudiantes AS
TYPE TipoEspecialidad IS TABLE OF
    estudiantes.especialidad%TYPE INDEX BY
    BINARY_INTEGER;
TYPE TipoIdentificador IS TABLE OF
    estudiantes.id%TYPE INDEX BY BINARY_INTEGER;
EspecEst TipoEspecialidad;
IdEst TipoIdentificador;
NumEntradas BINARY_INTEGER:=0;
END DatosEstudiantes;
```

Paquetes y tablas mutantes

Creamos el disparador a nivel de fila para almacenar los nuevos datos:

```
CREATE OR REPLACE TRIGGER FilaLimiteEspecialidad
BEFORE INSERT OR UPDATE OF especialidad ON estudiantes
FOR EACH ROW
BEGIN
DatosEstudiantes.NumEntradas :=
    DatosEstudiantes.NumEntradas + 1;
DatosEstudiantes.EspecEst(DatosEstudiantes.NumEntradas)
    := :new.especialidad;
DatosEstudiantes.IdEst(DatosEstudiantes.NumEntradas) :=
    :new.id;
END FilaLimiteEspecialidad;
```

Paquetes y tablas mutantes

Creamos el disparador a nivel de orden para dar la funcionalidad que queríamos:

```
CREATE OR REPLACE TRIGGER OrdenLimiteEspecialidad
AFTER INSERT OR UPDATE OF especialidad ON estudiantes
DECLARE
maxEstudiantes CONSTANT NUMBER:=5;
EstudiantesActuales NUMBER;
EstudianteId estudiantes.id%TYPE;
LaEspecialidad estudiantes.especialidad%TYPE;
BEGIN
FOR indice IN 1..DatosEstudiantes.NumEntradas LOOP
EstudianteId := DatosEstudiantes.IdEst(indice);
LaEspecialidad := DatosEstudiantes.EspecEst(indice);
SELECT COUNT(*) INTO EstudiantesActuales
FROM estudiantes
WHERE especialidad = LaEspecialidad;
IF EstudiantesActuales+1>maxEstudiantes THEN
RAISE_APPLICATION_ERROR(-20000, 'Demasiados estudiantes en la
especialidad: '|| LaEspecialidad);
END IF;
END LOOP;
DatosEstudiantes.NumEntradas := 0;
END OrdenLimiteEspecialidad;
```