UT6. TRIGGERS

CURSO BBDD

TRIGGERS

Los triggers o disparadores son bloques de PL/SQL almacenados que se ejecutan automáticamente cuando se producen ciertos eventos.

Hay 3 tipos de triggers de BD:

- De **tablas**: asociados a una tabla. Se disparan cuando se produce una manipulación (insert, delete o update) en la tabla.
- De **sustitución**: Asociados a vistas. Se disparan cuando se produce una manipulación (insert, delete o update) en la tabla.
- De **sistema**: Se disparan cuando hay un evento de sistema (arranque o parada de BBDD, entrada o salida de un usuario,...) o una instrucción de definición de datos (creación, modificación o eliminación de una tabla u objeto)

TRIGGERS

Se suelen utilizar para:

- Implementar restricciones complejas de seguridad o integridad.
- Posibilitar la realización de operaciones de manipulación sobre vistas.
- Prevenir transacciones erróneas
- Implementar reglas administrativas complejas
- Generar automáticamente valores derivados.
- Auditar actualizaciones o enviar alertas.

Los permisos necesarios para creación de triggers (CREATE TRIGGER) son específicos (distintos de los de ejecución de SQL por ejemplo).

TRIGGERS DE TABLAS

Para crear un disparador utilizaremos la siguiente instrucción:

```
CREATE {OR REPLACE} TRIGGER nombre_disp
  [BEFORE|AFTER]
  [DELETE|INSERT|UPDATE {OF columnas}] [ OR [DELETE|INSERT|UPDATE {OF columnas}]...]
  ON tabla
  [FOR EACH ROW [WHEN condicion disparo]]
[DECLARE]
  -- Declaración de variables locales
BEGIN
  -- Instrucciones de ejecución
[EXCEPTION]
  -- Instrucciones de excepción
END;
```

TIPOS DE TRIGGERS DE TABLAS

- Los triggers pueden definirse para las operaciones **INSERT**, **DELETE** o **UPDATE**.
- Pueden dispararse antes o después de la operación: El modificador **BEFORE** o **AFTER** indica que el trigger se ejecutará antes o después de ejecutarse la sentencia SQL definida por DELETE, INSERT o UPDATE.
- Finalmente, el nivel de los disparadores puede ser la fila o registro o la orden.
  - Si incluimos el modificador OF el trigger solo se ejecutará cuando la sentencia SQL afecte a los campos incluidos en la lista.
  - El alcance de los disparadores puede ser la fila o de orden. El modificador **FOR EACH ROW** indica que el trigger se disparará cada vez que se realizan operaciones sobre cada fila de la tabla. Si se acompaña del modificador **WHEN**, se establece una restricción; el trigger solo actuará, sobre las filas que satisfagan la restricción.

Si queremos conocer los valores del registro de la tabla sobre la que se ha disparado el trigger, este último debe estar declarado de forma FOR EACH ROW y acceder a sus valores mediante las pseudocolumnas :NEW y :OLD.

TRIGGERS DE TABLAS

En principio, dentro del cuerpo de programa del trigger podemos hacer uso de cualquier orden de consulta o manipulación de la BD, y llamadas a funciones o procedimientos siempre que:

- No se utilicen comandos DDL (creación de tablas, etc,...)
- No se acceda a las tablas que están siendo modificadas con DELETE, INSERT o UPDATE en la misma sesión
  - Cualquier acceso sobre esta tabla que no sea el que permite el propio trigger (lo veremos más adelante) nos devolverá un error de “Tabla mutando”.
- No se violen las reglas de integridad, es decir no se pueden modificar llaves primarias, ni actualizar llaves externas
- No se utilicen sentencias de control de transacciones (Commit, Rollback o Savepoint)
- No se llamen a procedimientos que transgredan la restricción anterior
- No se llame a procedimientos que utilicen sentencias de control de transacciones

Un trigger forma parte de la operación de actualización que lo lanza. Si el trigger falla, falla por tanto toda la operación y Oracle hace Rollback completo.

TRIGGERS DE TABLAS

Un disparador con nivel de fila se ejecuta una vez por cada fila procesada por la orden que provoca el disparo. Dentro del disparador, puede accederse a la fila que está siendo actualmente procesada utilizando, para ello, dos pseudo-registros, **:old** y **:new**.

En principio tanto :old como :new son del tipo **tabla_disparo%ROWTYPE**;

Orden de Disparo	:old	:new
INSERT	No definido; todos los campos toman el valor NULL.	Valores que serán insertados cuandos e complete la orden
UPDATE	Valores originales de la fila, antes dela actualización.	Nuevos valores que serán escritos cuando se complete la orden.
DELETE	Valores originales, antes del borrado de la fila.	No definido; todos los campos toman el valor NULL.

TRIGGERS DE TABLAS

```
Create or replace trigger EJEMPLO
    BEFORE insert or update of salary on employees
for each row
Declare
    aumento_no_valido   EXCEPTION;
Begin
    if :new.job_id = 'AD_PRES' then
        If :new.salary > :old.salary then
            :new.salary := :old.salary;
            Raise aumento_no_valido;
        End If;
    End if;
Exception
    When aumento_no_valido then
        dbms_output.put_line('No se puede subir el sueldo al presi');
        return;
    When Others then
        dbms_output.put_line(SQLERRM);
        return;
End;
```

TRIGGERS DE TABLAS

¿Qué hace este trigger? Vamos a analizar:

- Es un trigger que se ejecutará ANTES
- De una inserción o modificación del SALARIO en la tabla EMPLOYEES
- Por cada fila

Dentro del trigger se evalúa una condición y es únicamente para el presidente y es que su salario no puede ser mayor que el anterior. En ese caso “deshace” el cambio (asignando el valor :old.salary al :new.salario y nos saca un aviso (gestionado como una excepción)

```
update employees set
salary = 25000
where employee_id = 100;
```

```
No se puede subir el sueldo al presi


1 fila actualizadas.
```

Escribimos el mismo trigger pero indicando que solo se dispare cuando estemos hablando del presidente.

TRIGGERS DE TABLAS

```
Create or replace trigger EJEMPLO
    BEFORE insert or update of salary on employees
for each row when (:new.jod_id = 'AD_PRES')
Declare
    aumento_no_valido   EXCEPTION;
Begin
    If :new.salary > :old.salary then
        :new.salary := :old.salary;
        Raise aumento_no_valido;
    End If;
Exception
    When aumento_no_valido then
        dbms_output.put_line('No se puede subir el sueldo al presi');
        return;
    When Others then
        dbms_output.put_line(SQLERRM);
        return;
End;
```

ORDEN DE EJECUCIÓN DE LOS TRIGGERS

Una misma tabla puede tener varios triggers asociados. En tal caso es necesario conocer el orden en el que se van a ejecutar.

Los disparadores se activan al ejecutarse la sentencia SQL.

1. Si existe, se ejecuta el disparador de tipo BEFORE (disparador previo) con nivel de tabla.
2. Se ejecuta si existe, el disparador de tipo BEFORE con nivel de fila.
3. Se ejecuta la propia orden.
4. Se ejecuta si existe, el disparador de tipo AFTER (disparador posterior) con nivel de fila.
5. Se ejecuta, si existe, el disparador de tipo AFTER con nivel de tabla.

TRIGGER DE SUSTITUCIÓN

Podemos crear triggers que no se ejecutan antes ni después de una instrucción sino en lugar de (instead of).

Solo podemos utilizar estos triggers si están asociados a vistas, además actúan siempre a nivel de fila. La sintaxis de este tipo de trigger es la siguiente:

TRIGGER DE SUSTITUCIÓN

```
create [or replace] trigger nombre_trigger
instead of { insert | delete | update [of columnas]}
[ or { insert | delete | update}]
on nombre vista
[ for each row]
[declare]
    declaraciones
begin
    instrucciones
[execption]
    excepciones
end;
```

TRIGGER DE SISTEMA

Desde el punto de vista de este curso, estos son posiblemente los triggers menos interesantes, ya que si bien los triggers a nivel de tabla normalmente suelen tener lógica de negocio y por tanto los suele hacer el equipo de desarrollo de la aplicación, los triggers de sistema están asociados a eventos de sistema por lo que lo normal es que la responsabilidad recaiga sobre el equipo de administradores de la BBDD.

La sintaxis de los triggers sería la siguiente:

create [or replace] trigger nombre_trigger

{ before | after } { <lista eventos de definición> | <lista eventos del sistema>}

on { database | schema} [when (condición)]

<cuerpo del trigger (bloque PL/SQL)>

Donde la lista de eventos de definición puede tener uno o más eventos DDL separados por or y la lista de eventos del sistema igualmente separados por or.

TRIGGER DE SISTEMA

Al asociar un disparador a un evento debemos indicar el momento en que se dispare. Os dejo una tabla de evento, momento y cuando se dispararía:

Evento	Momento	Se disparan:
STARTUP	AFTER	Después de arrancar la instancia
SHUTDOWN	BEFORE	Antes de apagar la instancia
LOGON	AFTER	Después de que el usuario se conecte a la base de datos.
LOGOFF	BEFORE	Antes de la desconexión de un usuario
SERVERERROR	AFTER	Cuando ocurre un error en el servidor
CREATE	BEFORE | AFTER	Antes o después de crear un objeto en el esquema
DROP	BEFORE | AFTER	Antes o después de borrar un objeto en el esquema
ALTER	BEFORE | AFTER	Antes o después de cambiar un objeto en el esquema
TRUNCATE	BEFORE | AFTER	Antes o después de ejecutar un comando truncate
GRANT	BEFORE | AFTER	Antes o después de ejecutar un comando grant
REVOKE	BEFORE | AFTER	Antes o después de ejecutar un comando revoke
DDL	BEFORE | AFTER	Antes o después de ejecutar cualquier comando de definición de datos

TRIGGER DE SISTEMA

El evento SERVERERROR se activa ante cualquier error en la BBDD que no sea uno de los siguientes:

- ORA-01034: ORACLE not available
- ORA-01403: NO_DATA_FOUND
- ORA-01422: TOO_MANY_ROWS
- ORA-01423: error encountered while checking for extra rows in exact fetch
- ORA-04030: out of process memory when trying to allocate “X”bytes

Oracle tiene algunas funciones que permiten acceder a los atributos del evento del disparo USER o ORA_SYSEVENT, etc. Estas funciones pueden usarse en la cláusula WHEN o en el cuerpo del disparador. En el manual de Oracle podéis encontrar el listado de todas estas funciones.

Un ejemplo sería un trigger que nos guarda los datos de un usuario al hacer login en la base de datos:

TRIGGER DE SISTEMA

```
CREATE TABLE AUDITORIA_EVENTO_BD(
USUARIO VARCHAR2(50),
EVENTO VARCHAR2(50),
FECHA    DATE,
PRIMARY KEY(USUARIO,EVENTO,FECHA ));
```

```
CREATE OR REPLACE TRIGGER AUDITORIA_LOGIN
AFTER LOGON ON DATABASE
  BEGIN
     INSERT INTO auditoria_evento_bd VALUES (
     USER,
     ORA_SYSEVENT,
     SYSDATE );
 END;
```

ACTIVAR/DESACTIVAR TRIGGERS

Para activar o desactivar un trigger se utilizan las siguientes expresiones:

- alter trigger nombredeltrigger disable;
- alter trigger nombredeltrigger enable;

Para volver a compilar un trigger se utiliza la siguiente expresión:

- alter trigger nombredeltrigger compile;

Para eliminar un trigger se utiliza la siguiente expresión:

- drop trigger nombredeltrigger;