


DISPARADORES DE LA BASE DE DATOS (TRIGGER).

- Un disparador o trigger es un procedimiento PL/SQL asociado a una tabla que Oracle ejecuta automáticamente cuando se realiza una determinada operación sobre la tabla.
- Los trigger se asemejan a procedimientos y funciones en que son bloque pl/sql nominados con sección declarativa, ejecutable y de control de errores
- Sin embargo, un trigger es un conjunto de acciones que se ejecutan implícitamente cuando ocurre un evento.
- Con un disparador, por ejemplo, se puede controlar que a determinadas tablas no se acceda en determinados días, o bien que no se realicen determinadas operaciones por determinados usuarios.
- El acto de ejecutar un trigger se conoce como disparar y el evento que lo provoca generalmente son DML sobre tablas.
- Con los disparadores en general podemos:
 - Implementar restricciones complejas de integridad y seguridad
 - Prevenir transacciones erróneas
 - Implementar reglas administrativas complejas
 - Generar automáticamente valores derivados
- No se deben definir trigger para duplicar o reemplazar la funcionalidad ya construida dentro de la bbdd. Por ejemplo, no definir triggers para implementar las reglas de integridad
- El uso excesivo de triggers puede dar lugar a interdependencias complejas o podría dificultar el mantenimiento en aplicaciones largas.
- Cuidado con los efectos recursivos y en cascada
- Son tres los eventos que hacen que se dispare un trigger: insert, delete, update, pudiendo elegir si se dispara antes o después de producirse uno o varios de los eventos anteriores.
- Para poder crear, modificar o borrar disparadores es necesario tener los privilegios necesarios:

CREATE TRIGGER  Crear un disparador en el esquema propio.

CREATE ANY TRIGGER  Crear un disparador en cualquier esquema.

ALTER ANY TRIGGER  Modificar un disparador en cualquier esquema.

DROP ANY TRIGGER  Eliminar cualquier disparador en cualquier esquema.

- La sintaxis para crear disparadores de la base de datos es la siguiente:

```
CREATE [OR REPLACE] TRIGGER
nombre {BEFORE| AFTER }
{DELETE| INSERT|UPDATE [OF
columnas...]} [OR {DELETE|
INSERT|UPDATE [OF columnas...]} ON
tabla
[REFERENCING OLD as alias NEW as alias]
[FOR EACH {STATEMENT|ROW[WHEN
CONDICIONN}}] /*Aqui comienza el bloque
pl/sql*/ [DECLARE
```

Declaraciones; Es opcional]

BEGIN

Sentencias;

END;

- Con las opciones BEFORE o AFTER se le indica si se dispara antes o después de producirse el evento.
- Los eventos pueden ser: la inserción de nuevas filas, la actualización de todas o algunas de las columnas y el borrado de todas o algunas filas. Podemos especificar más de un evento a la

vez, separándolos con operadores de relación OR. En este caso se lanza un mismo trigger por distintos eventos.

- Se puede especificar la columna o columnas que deben modificarse para que se lance el disparador
- El nivel del disparador puede ser:
 - A nivel de orden (por defecto): Se activa una sola vez por cada orden, independientemente del número de filas afectadas. Se puede incluir FOR EACH STATEMENT aunque no hace falta
 - A nivel de fila: Se activa un vez por cada fila afectada. Se debe incluir FOR EACH ROW
- La cláusula WHEN sirve para especificar la restricción del trigger, es decir, se dispara solo cuando se cumple la condición. Esta condición tiene las siguientes restricciones:
 - Solo se puede utilizar con triggers a nivel de filas
 - Debe ser una condición SQL no PL/SQL
 - No puede incluir una consulta a ninguna tabla

NEW y OLD

- Con ellos podemos referenciar a los valores antes y después de la actualización a nivel de fila
- Solo se puede utilizar a nivel de fila
- Se utiliza :new.columna :old.columna
- Ejemplo: if :new.columna<:old.columna
- Cuando hacemos DELETE :new.columna es NULL
- Cuando hacemos INSERT :old.columna es NULL
- Cuando hacemos UPDATE ambas tienen valor
- Cuando utilicemos new y old con la cláusula WHEN del trigger no hace falta :

Ejemplo: when new.salario<old.salario

- Cuando queramos poner un alias a old o new podremos hacerlo con REFERENCING

Ejemplo: Vamos a controlar qué usuarios modifican la información de la tabla emp (insertan, borran o actualizan) y cuándo lo hacen. El ejemplo ejecutarlo a nivel de fila y de orden y ver las diferencias

1º. Creamos una tabla donde almacenaremos el usuario y el momento de utilización de la tabla y la operación realizada sobre ella.

```
CREATE TABLE controla_usuarios (  
    usuario VARCHAR2(17),  
    operacion VARCHAR2(6),  
    fecha DATE);
```

2º. Creamos los disparadores, para lo que es necesario tener el privilegio CREATE TRIGGER, si lo creamos en nuestro esquema, o el de CREATE ANY TRIGGER, si se crea en cualquier otro esquema.

```
CREATE TRIGGER actualiza_emple  
    BEFORE UPDATE ON EMP  
DECLARE  
    nom_usuario VARCHAR2(30);  
BEGIN  
    nom_usuario := USER;  
    INSERT INTO controla_vecinos  
        VALUES (nom_usuario, 'UPDATE', sysdate);  
END;  
/
```

```
CREATE TRIGGER inserta_emp  
    BEFORE INSERT ON emp  
DECLARE  
    nom_usuario VARCHAR2(30);  
BEGIN  
    nom_usuario := USER;
```

```

INSERT INTO controla_emp
VALUES (nom_usuario, 'INSERT', sysdate);
END;
/
CREATE OR REPLACE TRIGGER
borra_emp BEFORE DELETE ON emp
DECLARE
nom_usuario VARCHAR2(30);
BEGIN
nom_usuario := USER;
INSERT INTO controla_emp
VALUES (nom_usuario, 'delete',
sysdate); END;

/

```

Una vez creados los disparadores, cada vez que un usuario realiza cualquier tipo de modificación del contenido de la tabla EMP, se genera una fila en la tabla CONTROLA_EMP, que nos permite conocer que usuario realizó una determinada operación y en que momento.

TIPOS DE DISPARADORES Y ORDEN DE EJECUCIÓN

- Cuando existen varios disparadores se lanzaran en el siguiente orden:
 - 1º . Los asociados a BEFORE ... FOR EACH STATEMENT
 - 2º Para cada fila los BEFORE .. FOR EACH ROW
 - 3º Se realiza INSERT, DELETE o UPDATE bloqueando las filas hasta que la transacción se confirme
 - 4º Se ejecutan los AFTER .. FOR EACH ROW
 - 7º AFTER FOR EACH STATEMENT
- Cuando se dispara un trigger este forma parte de la operación de actualización que lo lanzó. Si el trigger falla, lo hace toda la actualización y Oracle hará un Rollback de toda la actualización
- A veces en lugar de varios trigger se hace uno solo que se puede lanzar por múltiples eventos

Multiples eventos de disparo

- Cuando un mismo trigger es disparado por múltiples eventos esto se hace: Before delete or update ON emp
- Para distinguir posteriormente utilizamos los siguientes predicados:
 - INSERTING: Devuelve TRUE Si el evento que disparó el trigger fue INSERT
 - DELETING: Devuelve TRUE Si el evento que disparó el trigger fue DELETE
 - UPDATING: Devuelve TRUE Si el evento que disparó el trigger fue UPDATE
 - UPDATING(nombrecolumna): Devuelve TRUE Si el evento que disparó el trigger fue update y la columna especificada ha sido modificada

Begin

If INSERTING then

ELSEIF DELETING then

ELSEIF UPDATING('SALARIO') THEN

ENDIF

Restricciones sobre los trigger

- En el bloque del trigger debemos tener las siguientes consideraciones:
 - No podemos escribir ninguna orden de control de transacciones(COMMIT,ROLLBACK). Tampoco podrán ser utilizadas estas sentencias en funciones o procedimientos que sean llamados desde el trigger

Triggers Instead of

- Son un tipo especial de trigger que solo se puede definir sobre vistas de la bbdd
- Se activan en lugar de la orden DML funcionando de manera invisible
- No deja especificar los of en el update, se debe hacer con código updating('columna');

```
CREATE OR REPLACE TRIGGER nombre
INSTEAD OF {DELETE| INSERT|UPDATE
} [OR {DELETE| INSERT|UPDATE }
ON nombrevista
[referencing old as alias new as alias]
[for each row]
bloque pl/sql
```

- Todos los trigger INSTEAD OF son triggers a nivel de fila por lo que esta cláusula no es obligatoria, ya que está implícita
- En estos trigger no se puede usar When

Ejemplo: Crear una vista con los empleados del departamento 20. (EMP20)

A continuación, si intentamos actualizar o borrar algo en esta vista no da problemas. Sin embargo, cuando intentamos insertar si da problemas, ya que no introduzco el número de departamento y es NOT NULL en la tabla (EMP)

Solución: Crear un disparador INSTEAD OF sobre DEPT20

```
.....
INSTEAD OF INSERT ON emp_20
BEGIN
Insert into emp (empno,ename,job,mgr,hiredate,sal,comm.,deptno)
values (:new.empno,:new.ename,:new.job,
7766,sysdate,:new.sal,null,20);
End;
/
```

```
sql> Insert into emp_20 values (7000,'PETER', 'CLERK',2000);
```

Otros triggers en la bbdd

- Estos trigger están disponibles a partir de la versión 8i
 - Triggers sobre sentencias DDL
 - Triggers sobre eventos del sistema
- Estos trigger se definen a nivel de bbdd o a nivel de esquema. Los definidos a nivel de bbdd se lanzan para todos los usuarios y los triggers a nivel de esquema se lanzan para el usuario del esquema del trigger.

Trigger sobre sentencias DDL

```
CREATE OR REPLACE TRIGGER nombre {BEFORE|AFTER}  
{CREATE|ALTER|DROP}[OR {CREATE|ALTER|DROP}] ON  
{DATABASE|nombreusuario.SCHEMA} Pl/sql block;
```

- Cuando no se especifica nombreusuario de toma por defecto el esquema del usuario que está creando el trigger

Trigger sobre eventos del sistema

Estos trigger se disparan cuando se arranca o para la base de datos, entra o sale un usuario, cuando se crea, modifica o elimina un objeto, etc.

En Oracle para crear este tipo de trigger tenemos que tener privilegios de Administer database trigger.

La sintaxis de este trigger seria 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.

Al asociar un disparador a un evento debemos indicar el momento en que se dispare. A continuación os dejo una tabla de evento, momento y cuando se dispararía para dejarlo todo mas o menos claro.

Evento	Momento	Se disparan:
STARTUP	AFTER	Después de arrancar la instancia
SHUTDOWN	BEFORE	Antes de apagar la istancia
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
DLL	BEFORE AFTER	Antes o después de ejecutar cualquier comando de definición de datos

db_event1. Eventos de la bdd	Descripción
AFTER SERVERERROR	Se dispara después de que se produzca un error
AFTER LOGON	Se dispara después de que el usuario se conecte a la bdd
BEFORE LOGOFF	Se dispara antes de que un usuario se desconecte
AFTER STARTUP	Se disparan después del arranque de la bdd. Este evento solo se puede aplicar a nivel de Database
BEFORE SHUTDOWN	Se dispara antes de la parada de la bdd. Este evento solo se puede aplicar a nivel de Database

Información sobre triggers

DESC USER_TRIGGERS

TRIGGER_TYPE: Puede tener Before, After o Instead of

TRIGGERING_EVENT: DML que dispara el Trigger

REFERENCING_NAMES: Alias para old y New

TRIGGER_BODY: El cuerpo del trigger

Operaciones sobre triggers

DROP TRIGGER nombre

Cuando eliminamos una tabla se eliminan todos los trigger asociados a ella

ALTER TRIGGER nombre_trigger ENABLE|DISABLE

Para activar o desactivar el trigger

ALTER TABLE nombre_trigger ENABLE|DISABLE ALL
TRIGGERS Activa o desactiva todos los triggers de la tabla