DISPARADORES

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
      { BEFORE | AFTER | INSTEAD OF }
      { INSERT | DELETE |  UPDATE [OF <column> [,<column>] ...] }
      ON {<table>|<view>}
      [ REFERENCING { OLD AS <old_name> | NEW AS <new_name> ...]
      [  FOR EACH ROW | STATEMENT]
      [WHEN (<trigger_condition>)] ]
 BEGIN
       …
 END;
```

Cabecera

Cuerpo

- Un disparador define una acción que se ejecuta cuando ocurre un cierto evento en la base de datos:
  - modificación de datos (INSERT, UPDATE o DELETE)
  - modificación del esquema
  - eventos del sistema (user logon/logoff)
- Para crear un disparadores sobre una tabla son necesarios privilegios de modificación sobre dicha tabla y de creación de disparadores.
- Los disparadores pueden referenciar tablas distintas de aquella cuyas modificaciones disparan el trigger.

TIPOS de DISPARADORES

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
  { BEFORE | AFTER | INSTEAD OF }
  { INSERT | DELETE |  UPDATE [OF <column> [,<column>] ...] }
  ON {<table>|<view>}
  [ REFERENCING { OLD AS <old_name> | NEW AS <new_name> ...]
  [  FOR EACH ROW | STATEMENT]
   [WHEN (<trigger_condition>)] ]
   <acción>;
```

- La sentencia activadora especifica el tipo de operación que despierta el disparador (DELETE, INSERT o UPDATE). Una, dos o incluso las tres operaciones pueden ser incluidas en la especificación de la sentencia activadora

```
INSERT OR DELETE OR UPDATE OF salario ON empleado
```

- Si la sentencia activadora especifica un UPDATE se puede incluir una lista de columnas en dicha sentencia.
- En la sentencia activadora se especifica la tabla asociada al disparador

TIPOS de DISPARADORES

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
  { BEFORE | AFTER | INSTEAD OF }
  { INSERT | DELETE |  UPDATE [OF <column> [,<column>] ...] }
  ON {<table>|<view>}
  [ REFERENCING { OLD AS <old_name> | NEW AS <new_name> ...]
  [  FOR EACH ROW | STATEMENT]
   [WHEN (<trigger_condition>)] ]
   <acción>;
```

Los disparadores se pueden clasificar EN FUNCIÓN DEL MOMENTO EN QUE SE EJECUTA:

- BEFORE:
  - Se ejecutan inmediatamente antes de la acción que los dispara.
- AFTER:
  - Se ejecutan inmediatamente después de la acción que los dispara.
- INSTEAD OF:
  - Se ejecutan en vez del evento que los dispara.
  - Sólo se pueden aplicar a vistas.

TIPOS de DISPARADORES

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
  { BEFORE | AFTER | INSTEAD OF }
  { INSERT | DELETE |  UPDATE [OF <column> [,<column>] ...] }
  ON {<table>|<view>}
  [ REFERENCING { OLD AS <old_name> | NEW AS <new_name> ...]
  [  FOR EACH ROW | STATEMENT]
   [WHEN (<trigger_condition>)] ]
   <acción>;
```

Los disparadores se pueden clasificar EN FUNCIÓN DEL NIVEL EN QUE SE EJECUTAN:

- ROW-LEVEL: a nivel de fila (row trigger)
  - Si está presente **FOR EACH ROW** el cuerpo del trigger se ejecuta individualmente para cada una de las filas de la tabla que haya sido afectada por la sentencia activadora.
  - La ausencia de la opción FOR EACH ROW implica que el trigger se ejecuta una sola vez, para la ejecución de una sentencia activadora
  - Opcionalmente, se pueden incluir restricciones en la definición de un row trigger. Para ello se especifica, en una cláusula WHEN, una expresión booleana de SQL que es una condición adicional que debe comprobarse antes de disparar el trigge r(no puede incluir subqueries)
- STATEMENT-LEVEL: a nivel de sentencia activadora (statement trigger)
  - El trigger se ejecuta una única vez.
  - Se utilizan para reforzar la seguridad sobre los tipos de transacciones permitidos en las tablas.

DISPARADORES. Variables Especiales

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
  { BEFORE | AFTER | INSTEAD OF }
  { INSERT | DELETE |  UPDATE [OF <column> [,<column>] ...] }
  ON {<table>|<view>}
  [ REFERENCING { OLD AS <old_name> | NEW AS <new_name> ...]
  [  FOR EACH ROW | STATEMENT]
   [WHEN (<trigger_condition>)] ]
   <acción>;
```

Las variables NEW y OLD están siempre disponibles para referirse a la nueva (después de la transacción) y vieja (previa a la transacción) tupla respectivamente.

¡¡ En el cuerpo del trigger, deben ir precedidas por dos puntos (“:”). Sin embargo, en la claúsula WHEN no es así. !!

Ejemplo DISPARADORES (I)

Prestatario(ci, nombre, dir, tel, empresa, tel_ofic)

Préstamo (num_prest, ci, tasa, monto)

Cuota (num_prest, num_cuota, f_venc, f_pago)

Prestatario almacena todos los prestatarios actuales del banco

Préstamo . almacena todos los préstamos que aún no han sido cancelados. El atributo `ci` referencia a Prestatario y es candidato a clave pues el banco no otorga simultáneamente dos préstamos a la misma persona.

Cuota. almacena todas las cuotas de los préstamos actuales (tanto las pagadas como las pendientes). El atributo `num_prest` referencia a Préstamo. Se tiene como política que toda cuota debe ser pagada antes de su fecha de vencimiento.

En la creación de las tablas se incluyen todas las restricciones, excepto aquella que dice que toda cuota debe ser pagada antes de su fecha de vencimiento. A continuación se presenta el diseño del disparador que garantiza el cumplimiento de esta restricción:

Ejemplo DISPARADORES (I)

Prestatario(ci, nombre, dir, tel, empresa, tel_ofic)

Préstamo (num_prest, ci, tasa, monto)

Cuota (num_prest, num_cuota, f_venc, f_pago)

```
CREATE TABLE Prestatario (
ci VARCHAR(8) NOT NULL PRIMARY KEY,
nombre VARCHAR(50) NOT NULL,
dir VARCHAR(100) NOT NULL,
tel VARCHAR(10) NOT NULL,
empresa VARCHAR(100) NOT NULL,
tel_ofic VARCHAR(10) NOT NULL);
```

```
 CREATE TABLE Prestamo (
num_prest NUMBER(5) NOT NULL PRIMARY KEY,
ci VARCHAR(8)UNIQUE REFERENCES Prestatario(ci),
tasa NUMBER(4,2) NOT NULL,
monto NUMBER(8)NOT NULL CHECK(monto > 0));
```

```
 CREATE TABLE Cuota (
num_prest NUMBER(5) NOT NULL,
num_cuota NUMBER(2) NOT NULL,
f_venc DATE NOT NULL,
f_pago DATE,
CONSTRAINT pk_cuota
PRIMARY KEY (num_prest, num_cuota),
CONSTRAINT fk_cuota_prest FOREIGN KEY
(num_prest) REFERENCES Prestamo(num_prest));
```

Diseño de DISPARADORES

1. **SENTENCIA DISPARADORA:** Como en la BD están todas las cuotas (pagadas o no) asociadas a los préstamos, la fecha de pago se actualiza. Por lo tanto, la sentencia disparadora es la **ACTUALIZACIÓN** de fecha de pago en la tabla Cuota.
2. ANTES O DESPUÉS: La restricción de integridad no se puede violar, por lo tanto el trigger debe ser disparado **ANTES** de realizar la actualización
3. PARA TODAS/PARA EL BLOQUE: La verificación de la restricción se hace para TODAS LAS FILAS que se actualicen al ejecutar la sentencia disparadora.
4. CONDICIÓN (WHEN): Se debe impedir la actualización, sólo cuando LA FECHA DE PAGO SEA MAYOR QUE LA FECHA DE VENCIMIENTO DE LA CUOTA
5. ACCIÓN: Dar un error por violación de la restricción

Prestatario(ci, nombre, dir, tel, empresa, tel_ofic)

Préstamo (num_prest, ci, tasa, monto)

Cuota (num_prest, num_cuota, f_venc, f_pago)

Ejemplo de DISPARADORES (I)

```
CREATE TRIGGER BUpCUOTA

BEFORE UPDATE OF f_pago ON Cuota

FOR EACH ROW

WHEN (new.f_pago > old.f_venc)

BEGIN

raise_application_error(-20000, ‘Cuota ‘ ||
TO_CHAR(:old.num_cuota) || ‘ del prestamo ‘ ||
TO_CHAR(:old.num_prest) || ‘ vencida. Por favor, dirigirse a
la gerencia.’);

END;

/
```

Prestatario(ci, nombre, dir, tel, empresa, tel_ofic)

Préstamo (num_prest, ci, tasa, monto)

Cuota (num_prest, num_cuota, f_venc, f_pago)

Ejemplo DISPARADORES (II)

Se puede usar un disparador, en combinación con una secuencia, para asegurar la unicidad de claves en la inserción de datos.

```
create table mitabla (
codigo integer not null primary key,
dato integer);

create sequence clave_mitabla;
```

```
CREATE TRIGGER TRIG_FAB
BEFORE INSERT ON  mitabla
FOR EACH ROW
BEGIN
SELECT clave_mitabla.NEXTVAL INTO :NEW.codigo FROM DUAL;
END;
/
```

```
insert into mitabla (dato) values (25);
```

MODIFICACIÓN EN DISPARADORES

Un disparador no puede ser alterado explícitamente. Debe ser reemplazado por una nueva definición. Cuando se reemplaza un disparador se debe incluir la opción OR REPLACE en la instrucción CREATE TRIGGER.

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
….
   END;
```

Otra alternativa es eliminar el *trigger y volverlo a crear.*.

```
DROP TRIGGER <nombre_trigger>;
CREATE TRIGGER <nombre_trigger> . . .
```

MODIFICACIÓN EN DISPARADORES

- Los disparadores pueden estar habilitados o deshabilitados. Los disparadores están habilitados por defecto.
- Los disparadores pueden deshabilitarse si:
  - Un objeto al que hace referencia no está disponible.
  - Se tienen que realizar cargas masivas de datos y se quiere proceder sin disparar triggers.
  - Se están volviendo a cargar datos.
  - Cualquier otro caso en el que se considere necesario.

```
ALTER TRIGGER <nombre_trigger> DISABLE;
ALTER TRIGGER <nombre_trigger> ENABLE;
```

Es posible deshabilitar todos los disparadores asociados a una tabla y volverlos a habilitar cuando se desee.

```
ALTER TABLE <nombre_tabla>
DISABLE ALL TRIGGERS;
```

```
ALTER TABLE <nombre_tabla>
ENABLE ALL TRIGGERS;
```