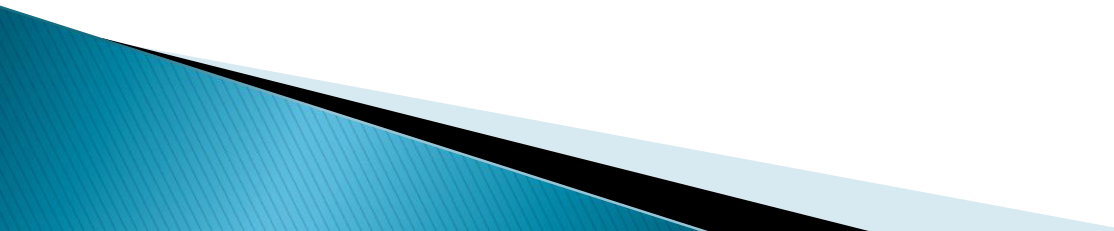


UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE COMPUTACIÓN

Tema V.
Programación con SQL
Disparadores

Ing. Lucila Patricia
Arellano Mendoza
2017

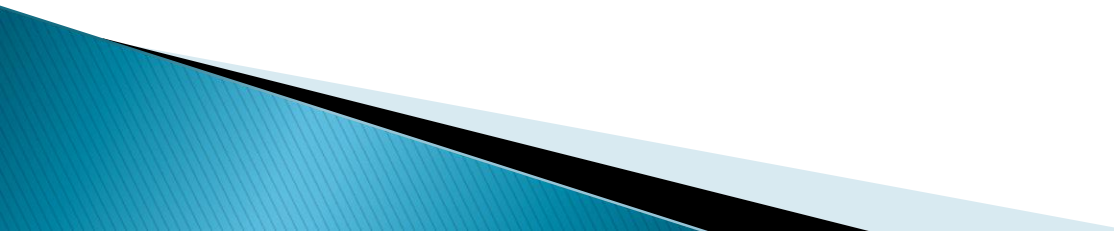


Disparadores

- ▶ Un disparador es un procedimiento almacenado que se asocia con una tabla específica y que es llamado cuando un evento particular ocurre como una inserción, borrado o actualización.

Los disparadores suelen utilizarse para restricciones de integridad complejas, auditoría de la información de una tabla o aviso a otros programas para ejecutar una acción.

Tienen la restricción de que no pueden manejar parámetros ni ser llamados directamente.



Tipos de disparadores

- ▶ DML
 - Fila
 - Sentencia
- ▶ Vistas
- ▶ Sistema

Estructura general

```
CREATE [OR REPLACE] TRIGGER <nombreDisparador>
{BEFORE|AFTER}
{INSERT|DELETE|UPDATE [OF columna [,columna]...]}
ON <tabla>
[FOR EACH ROW | STATEMENT]
[WHEN condición]
[DECLARE
... ]
BEGIN
... cuerpo del trigger...
[EXCEPTION
... ]
END <nombreDisparador>;
```

/

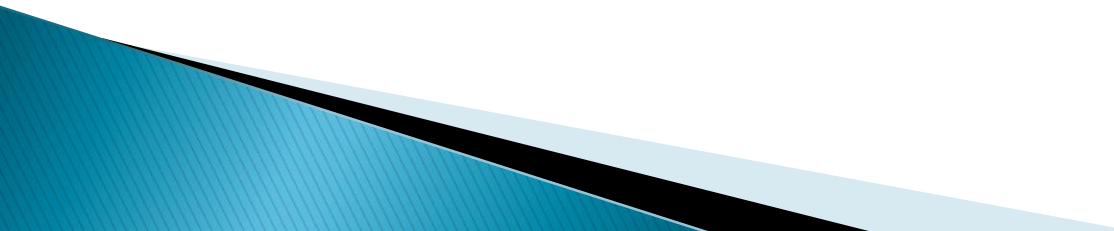
Temporalidad del Evento

- ▶ BEFORE. Ejecuta la acción asociada antes de que la sentencia sea llevada a cabo.
 - Decide si la acción debe realizarse
 - Utiliza valores alternativos para la sentencia
- ▶ AFTER. Ejecuta la acción asociada después de que se haya llevado a cabo la sentencia.

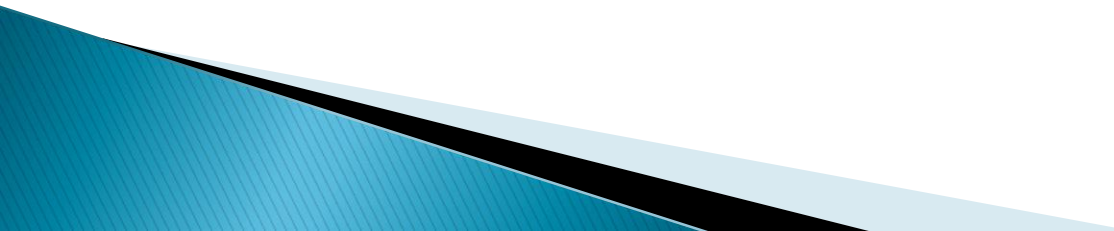
Granularidad del evento

- ▶ A nivel de fila: FOR EACH ROW
 - Ejecuta la acción tantas veces como filas se vean afectadas por la sentencia que lo dispara.
 - Si ninguna fila se ve afectada, no se dispara
- ▶ A nivel de sentencia: STATEMENT
 - Ejecuta una única vez la acción asociada, independientemente del número de filas afectadas por la sentencia, incluso si no hay filas afectadas.
 - Es una opción por default.

Condición

- ▶ WHEN condición. Expresa una condición que debe cumplirse en el momento de producirse el evento, para que la acción sea ejecutada.
 - ▶ Se evalúa para cada fila.
 - ▶ Debe ser una expresión booleana y no puede contener subconsultas.
 - ▶ Se pueden utilizar cualquier combinación de operadores lógicos(AND,OR,NOT) y relacionales (<, <=, >, >=, =, <>).
 - ▶ No se puede especificar una condición para los disparadores a nivel de sentencia.
- 

Restricciones en el cuerpo de los disparadores

- ▶ No se permiten ordenes de control de transacciones
 - COMMIT
 - ROLLBACK
 - SAVEPOINT
 - ▶ No se permiten ordenes de definición de datos
 - CREATE
 - ALTER
 - DROP
- 

Identificadores correlacionales

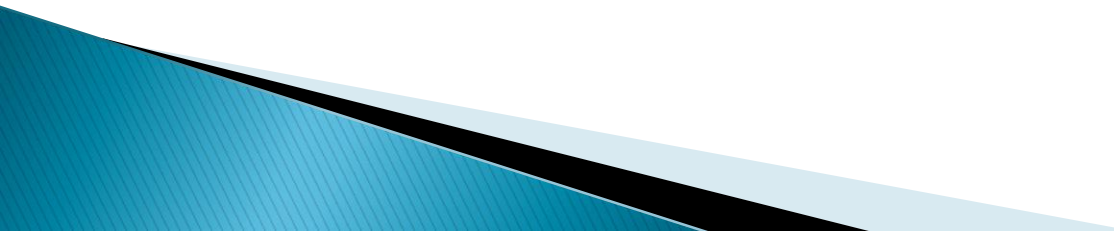
- ▶ Es un tipo especial de variable PL/SQL tratada como un registro de tipo tabla %ROWTYPE
- ▶ Sintaxis:
 - ▶ Condición (WHEN) OLD, NEW
 - ▶ En el cuerpo del disparador :OLD, :NEW

Dentro del disparador puede accederse a la fila que está siendo actualmente procesada utilizando para ello los identificadores :OLD o :NEW.

Orden de disparo	:OLD	:NEW
INSERT	No definido, valor nulo	Valores que se insertaran cuando se complete la orden
UPDATE	Valores originales de fila, antes de la actualización	Nuevos valores que serán escritos cuando se complete la orden
DELETE	Valores originales de fila, antes del borrado	No definido valor nulo

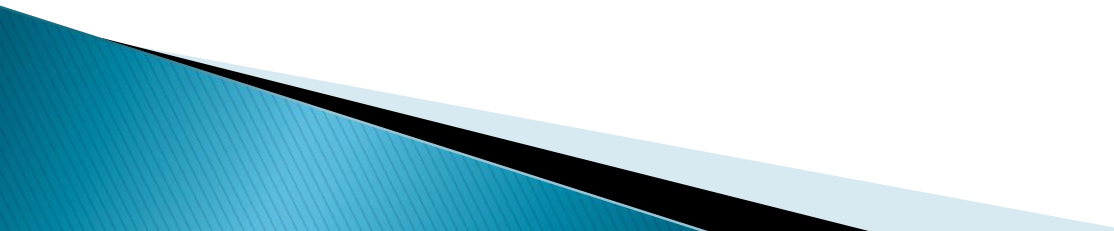
Ejemplo: A nivel de fila

```
CREATE OR REPLACE TRIGGER borraPersona
AFTER DELETE
ON persona
FOR EACH ROW
WHEN OLD.Edad >= 60
BEGIN
    INSERT INTO pension VALUES(:OLD.idPersona,
    :OLD.nombre);
END borraPersona;
/
```



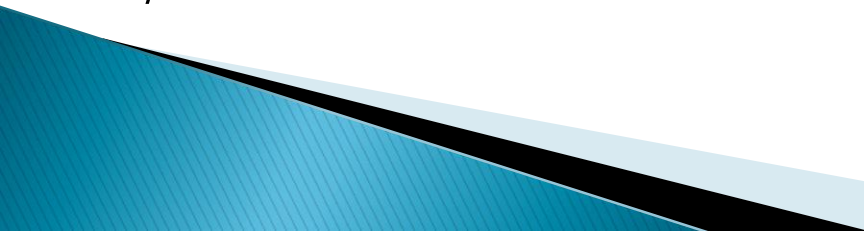
Ejemplo: A nivel de sentencia

```
CREATE OR REPLACE TRIGGER actualizaEstadisticas
AFTER INSERT OR DELETE OR UPDATE
ON Libros
DECLARE
    ...
BEGIN
    CASE ...
        UPDATE Estadisticas SET ...
    ...
END actualizaEstadisticas;
/
```



Ejemplo: Excepciones en el cuerpo del disparador

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

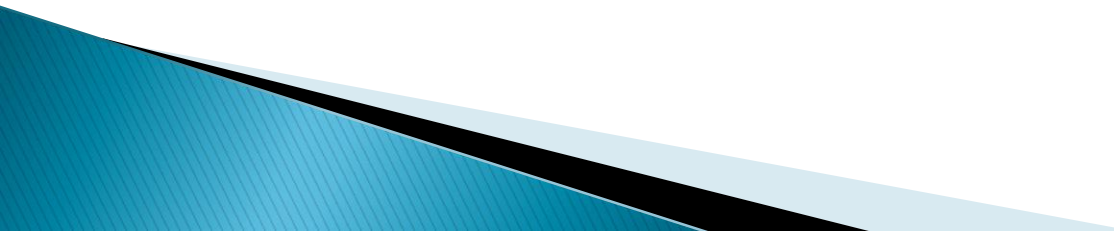


Funciones del cuerpo del disparador

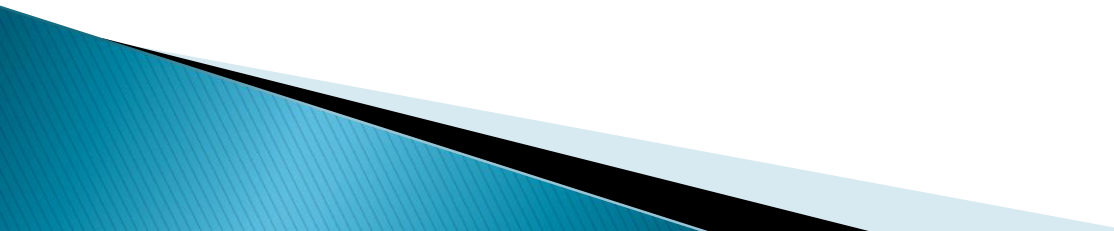
Funciones booleanas que se emplean para determinar de qué operación se trata.

- ▶ INSERTING
- ▶ UPDATING
- ▶ DELETING

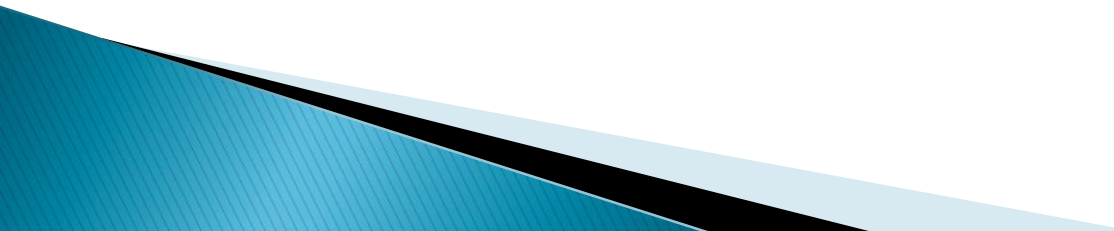
Cada uno de estos predicados devuelve TRUE si la orden de disparo es alguno de ellos, respectivamente.



Tablas mutantes

- ▶ Es una tabla que está siendo modificada por un INSERT, UPDATE o DELETE, o por efecto de integridad referencial como es el ON DELETE CASCADE.
 - ▶ Una tabla esta restringida si una sentencia activadora necesita leerla directamente, por una sentencia SQL, o indirectamente por una restricción de integridad referencial.
- 

Restricciones de los disparadores

- ▶ Un ROW TRIGGER no puede leer, ni modificar una tabla que esta mutando.
 - ▶ Un ROW TRIGGER no puede cambiar la llave primaria, ni foránea ni atributos únicos de tablas que estén restringidas. Excepto un BEFORE ROW TRIGGER disparado por un INSERT de una sola fila de una tabla con una clave foránea, siempre que no se violen las restricciones de integridad referencial
- 

Restricciones de los disparadores (tablas mutantes)

EVENTO	GRANULARIDAD	TEMPORALIDAD	ERROR
INSERT	ROW	BEFORE	NO*
		AFTER	SI
	STATEMENT	BEFORE	NO
		AFTER	NO
UPDATE	ROW	BEFORE	SI
		AFTER	SI
	STATEMENT	BEFORE	NO
		AFTER	NO
DELETE	ROW	BEFORE	SI
		AFTER	SI
	STATEMENT	BEFORE	NO**
		AFTER	NO**

* Siempre que la inserción que provoque la activación del disparador sea una inserción simple (inserte una única fila).

** Siempre que el disparador no se active como consecuencia de un borrado en cascada.

Los disparadores se pueden deshabilitar o habilitar

Sintaxis

```
ALTER TRIGGER <nombreDisparador >[DISABLE | ENABLE];
```



Para deshabilitar o habilitar todos los disparadores de una tabla

```
ALTER TABLE <nombreTabla> ENABLE  
ALL TRIGGERS | DISABLE ALL TRIGGERS
```

Eliminación de un disparador

Sintaxis

```
DROP TRIGGER <nombreDisparador>;
```

Consultar información de los disparadores

- ▶ Ver todos los disparadores y su estado

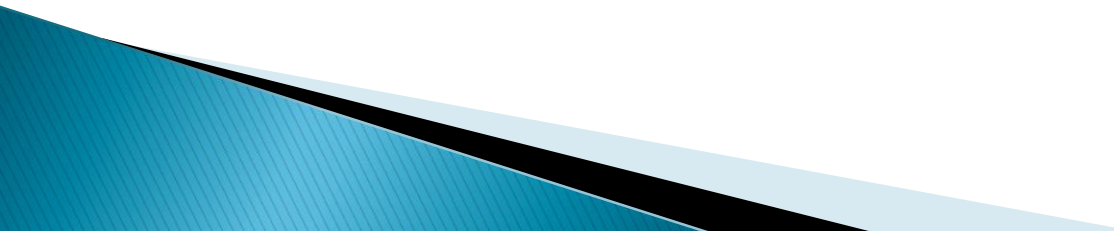
```
SELECT TRIGGER_NAME, STATUS  
FROM USER_TRIGGERS;
```

- ▶ Ver el cuerpo de un disparador

```
SELECT TRIGGER_BODY  
FROM USER_TRIGGERS  
WHERE TRIGGER_NAME = 'nombreDisparador';
```

- ▶ Ver la descripción de un disparador

```
SELECT DESCRIPTION  
FROM USER_TRIGGERS  
WHERE TRIGGER_NAME='nombreDisparador';
```



Ejemplo 11. Disparadores

Abra un archivo spool y realice lo siguiente:

Vamos a crear un disparador que indique que sentencias se realizaron en la tabla proyecto.

1. Revisamos la estructura de la tabla proyecto

DESCRIBE proyecto;

Nombre	¿Nulo?	Tipo
-----	-----	-----
CLAVEPROY	NOT NULL	CHAR(6)
DESCRIPPROY	NOT NULL	VARCHAR2(20)
CLAVEDEPTO	NOT NULL	CHAR(4)
PRIORIDAD	NOT NULL	CHAR(1)

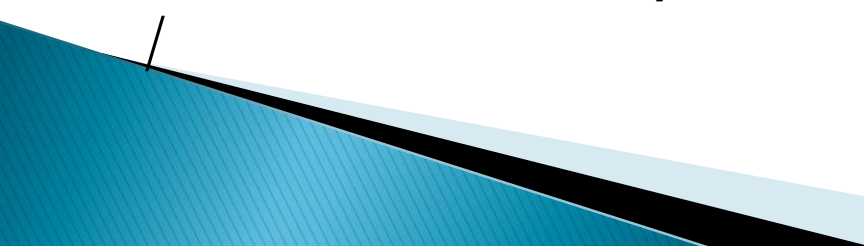
2. Creamos la tabla donde tendrá efecto el disparador.

```
CREATE TABLE bitacoraProy(  
fecha VARCHAR2(60),  
operacion CHAR(1),  
usuario VARCHAR2(10)  
);
```

3. Ahora creamos el disparador

```
CREATE OR REPLACE TRIGGER cambiosProyecto  
AFTER INSERT OR UPDATE OR DELETE  
ON proyecto  
DECLARE  
    sentencia CHAR(1);  
    tiempo VARCHAR2(60);
```

```
BEGIN
  IF INSERTING THEN
    sentencia:= 'I';
  ELSIF UPDATING THEN
    sentencia:= 'U';
  ELSE
    sentencia:= 'D';
  END IF;
  tiempo:=TO_CHAR(SYSDATE, 'DAY", "DD" de“
    MONTH" del "YY", a las "HH24:MI:SS');
  INSERT INTO bitacoraProy
  VALUES (tiempo, sentencia, USER);
  DBMS_OUTPUT.PUT_LINE ('Transaccion realizada: ‘
    ||tiempo);
END cambiosProyecto;
```



Para probarlo realice lo siguiente:

```
INSERT INTO proyecto  
VALUES ('Proy8','nuevo', 'D5', 'B');
```

```
UPDATE proyecto SET descripProy='otro'  
WHERE claveProy='Proy8';
```

```
DELETE FROM proyecto  
WHERE claveProy='Proy8';
```

Al terminar revisamos la tabla bitacoraProy para observar lo emitido por el disparador.

```
SELECT * FROM bitacoraProy;
```

FECHA	O	USUARIO
-----	-	-----
miércoles, 16 de noviembre del 2016, a las 16:50:26	I	SYSTEM
miércoles, 16 de noviembre del 2016, a las 16:51:09	U	SYSTEM
miércoles, 16 de noviembre del 2016, a las 16:51:29	D	SYSTEM

Registrar en forma detallada que sentencias DML se realizaron en la tabla proyecto.

1.Creamos la tabla donde se guardará el resultado del disparador

```
CREATE TABLE bitacoraDetallesProy(  
fecha DATE,  
usuario VARCHAR2(10),  
claveProyNuevo CHAR(6),  
descripProyNuevo VARCHAR2(20),  
claveProyAnterior CHAR(6),  
descripProyAnterior VARCHAR2(20)  
);
```

2. Creamos el disparador

```
CREATE OR REPLACE TRIGGER cambioDetallesProyecto
BEFORE INSERT OR UPDATE OR DELETE
ON proyecto
FOR EACH ROW
BEGIN
    INSERT INTO bitacoraDetallesProy
    VALUES (SYSDATE, USER, :NEW.claveProy,
        :NEW.descripProy, :OLD.claveProy, :OLD.descripProy);
END cambioDetallesProyecto;
/
```

Para observar su utilización realizamos lo siguiente

```
INSERT INTO proyecto  
VALUES ('Proy8','nuevo', 'D5', 'B');
```

```
UPDATE proyecto SET descripProy='otro'  
WHERE claveProy='Proy8';
```

```
DELETE proyecto  
WHERE claveProy='Proy8';
```


Revisamos lo que realizó el disparador

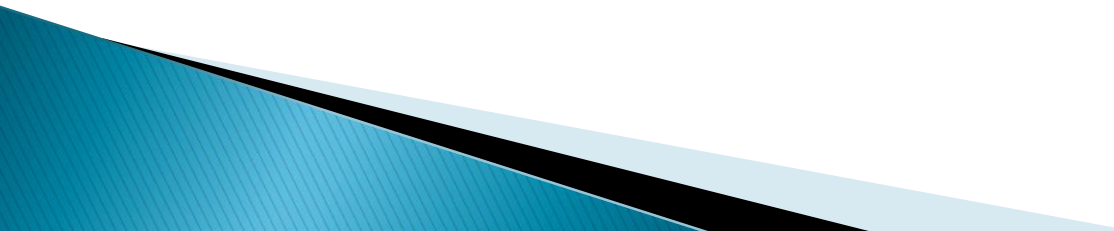
```
SELECT * FROM bitacoraDetallesProy;
```

FECHA	USUARIO	NUMPROYNUEVO	NOMPROYNUEVO	NUMPROYANTERIOR	NOMPROYANTERIOR
-----	-----	-----	-----	-----	-----
16/11/16	GRUPO04	Proy8	nuevo		
16/11/16	GRUPO04	Proy8	otro	proy8	nuevo
16/11/16	GRUPO04			proy8	otro

Ejemplo de tablas mutantes

Vamos a crear un disparador que impida que se actualice el salario en la tabla puesto, por debajo del salario mínimo o por encima del salario máximo, sin considerar al puesto de presidente

```
CREATE OR REPLACE TRIGGER verificaSalario
  BEFORE UPDATE ON puesto
  FOR EACH ROW
  WHEN (new.descripPsto <> 'presidente')
DECLARE
  vSalarioMin NUMBER(12,2);
  vsalarioMax NUMBER(12,2);
```



BEGIN

SELECT MAX(salario), MIN(salario)
INTO vSalarioMin, vSalarioMax
FROM puesto;

IF :new.salario < vSalarioMin
OR :new.salario > vSalarioMax THEN
RAISE_APPLICATION_ERROR(-20001, 'Salario fuera de rango');
END IF;

END verificaSalario;

/



Realizamos una actualización en la tabla puesto para probar el disparador.

```
UPDATE puesto SET salario= 2000  
WHERE descripPsto = 'Secretaria';
```

Disparadores de sustitución

Podemos crear disparadores que no se ejecutan **antes** ni **después** de una instrucción sino en lugar de (INSTEAD OF).

Sólo podemos utilizar estos disparadores si están asociados a **vistas**, además actúan siempre a nivel de fila.

Ejemplo

Vamos a crear un disparador que al eliminar una factura éste no lo elimine, sino modifique su estatus a cancelado.

1. Modificamos la tabla factura para agregarle el campo status.

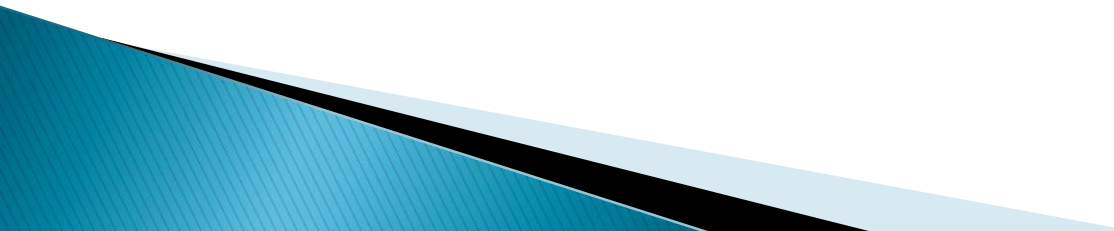
```
ALTER TABLE factura  
ADD status CHAR(1) NULL;
```

```
UPDATE factura SET status='A'  
WHERE status IS NULL;
```



2. Creamos la vista que contendrá los datos de las facturas.

```
CREATE OR REPLACE VIEW statusFactura  
AS  
SELECT numFact, claveCte, claveProy, total, fecha  
FROM factura  
WHERE status = 'A';
```



2. Creamos el disparador

```
CREATE OR REPLACE TRIGGER cancelaFactura  
INSTEAD OF DELETE ON statusFactura  
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE factura SET status='C'
```

```
    WHERE numFact=:OLD.numFact;
```

```
END cancelaFactura;
```

```
/
```



3. Probamos el disparador

```
DELETE FROM statusFactura  
WHERE numFact='0001';
```

Verifique la tabla factura y la vista asociada

```
SELECT * FROM factura;
```

```
SELECT * FROM statusFactura;
```

Al terminar cierre su archivo spool.

