Triggers

BASES DE DATOS
Mercedes García Merayo

Triggers

- Un trigger es código (PL/SQL) que se ejecuta automáticamente cuando se realiza una determinada acción sobre la base de datos.
- Tipos de triggers
  - **Triggers de tabla.** Se disparan cuando ocurre una acción DML sobre una tabla.
  - **Triggers de vista.** Se lanzan cuando ocurre una acción DML sobre una vista.
  - **Triggers de sistema.** Se disparan cuando se produce un evento sobre la base de datos (conexión de un usuario, borrado de un objeto,…)

Elementos Básicos

- El evento que da lugar a la ejecución del trigger: **`INSERT, UPDATE o DELETE.`**
- Instante en el que se lanza el trigger en relación a dicho evento: **BEFORE** (antes), **AFTER** (después) o **INSTEAD OF** (en lugar de).
- Las veces que el trigger se ejecuta: **instrucción** o **fila**.
- El código que ejecuta dicho trigger.

Conceptos Básicos

- **BEFORE.** El código del trigger se ejecuta antes de ejecutar la instrucción DML que causó el lanzamiento del trigger.
- **AFTER.** El código del trigger se ejecuta después de haber ejecutado la instrucción DML que causó el lanzamiento del trigger.
- **INSTEAD OF.** El trigger sustituye a la operación DML. Se utiliza para vistas que no admiten instrucciones DML.
- **De instrucción.** El cuerpo del trigger se ejecuta una sola vez por cada evento que lance el trigger. Opción por defecto.
- **De fila.** El código se ejecuta una vez por cada fila afectada por el evento.

Triggers de instrucción

```
CREATE [OR REPLACE] TRIGGER
  nombreDeTrigger
cláusulaDeTiempo evento1 [OR
  evento2[,...]]
ON tabla
[DECLARE
declaraciones
]
BEGIN
cuerpo
[EXCEPTION captura de excepciones]
END;
```

Triggers de instrucción

- La cláusula de tiempo es una de estas palabras: **BEFORE** o **AFTER**.
- **Evento**

```
{INSERT|UPDATE
   [OF columna1[,columna2,…]]|DELETE}
```

- En el caso de la instrucción `UPDATE`, el apartado `OF` hace que el trigger se ejecute sólo cuando se modifique la columna(s) indicada(s).
- En la sintaxis del trigger, el apartado `OR` permite asociar más de un evento al trigger.

Triggers de instrucción

```
CREATE OR REPLACE TRIGGER Log_emp_salarios
AFTER UPDATE ON Empleados

BEGIN
INSERT INTO Emp_log (Log_date, Accion)
VALUES (SYSDATE, 'Empleados cambio salarios');

END;
```

Triggers de fila

```
CREATE [OR REPLACE] TRIGGER nombreDeTrigger
cláusulaDeTiempo evento1 [OR evento2[,...]]
ON tabla
[REFERENCING {OLD AS nombreViejo | NEW AS
  nombreNuevo}]
FOR EACH ROW
[WHEN condición]
[declaraciones]
Cuerpo
```

Triggers de fila

- **FOR EACH ROW** hace que el trigger se ejecute por cada fila afectada en la tabla por la instrucción DML.
- **WHEN** permite colocar una condición que deben cumplir los registros para que el trigger se ejecute.
- **REFERENCING** permite indicar un nombre para los valores antiguos y otro para los nuevos.
- Cuando se ejecutan instrucciones **UPDATE**, se modifican valores antiguos (**OLD**) por valores nuevos (**NEW**).
- En el apartado de instrucciones del trigger hay que anteponer ":" a las palabra **NEW** y **OLD**

Triggers de fila

```
CREATE TABLE PIEZAS (
tipo VARCHAR2(2),
modelo NUMBER(2),
precio_venta NUMBER(11,4) not null default 0,
PRIMARY KEY (TIPO,MODELO));

                                CREATE TABLE PIEZAS_AUDIT(
                                precio_viejo NUMBER(11,4),
                                precio_nuevo NUMBER(11,4),
                                tipo VARCHAR2(2),
                                modelo NUMBER(2),
                                fecha DATE
                                PRIMARY KEY (TIPO,MODELO,FECHA),
                                CONSTRAINT fk_pieza FOREIGN KEY (TIPO,MODELO) REFERENCES PIEZAS);

CREATE OR REPLACE TRIGGER crear_audit_piezas
 BEFORE UPDATE OF precio_venta
 ON PIEZAS
 FOR EACH ROW
  WHEN (OLD.precio_venta<NEW.precio_venta)
 BEGIN
  INSERT INTO PIEZAS_AUDIT VALUES (:OLD.precio_venta, :NEW.precio_vent, :OLD.tipo, :OLD.modelo, SYSDATE);
END;
```

IF INSERTING, IF UPDATING, IF DELETING

- Se utilizan para determinar la instrucción DML que se estaba realizando cuando se lanzó el trigger.
- Se utiliza en triggers que se lanzan para varias operaciones.

```
CREATE OR REPLACE TRIGGER trigger1
BEFORE INSERT OR DELETE OR UPDATE [OF campo1] ON tabla
FOR EACH ROW
BEGIN
IF DELETING THEN
instrucciones que se ejecutan si el trigger saltó por borrar filas
ELSIF INSERTING THEN
instrucciones que se ejecutan si el trigger saltó por insertar filas
ELSE
instrucciones que se ejecutan si el trigger saltó por modificar filas
END IF
END;
```

Instead Of

- **INSTEAD OF** triggers permiten modificar datos a través de vistas que no son directamente modificables mediante **UPDATE, INSERT,** and **DELETE**.
- **INSTEAD OF** solo se puede aplicar a triggers definidos sobre vistas.
- **BEFORE y AFTER** no son aplicables.
- **INSTEAD OF** solo puede ser de tipo fila.

```
CREATE OR REPLACE VIEW pedido_info AS
SELECT c.clienteId, c.apellido, c. nombre, o.orderId, o.fecha, o.estado
FROM clientes c, pedidos o
WHERE c.clienteId = o.clienteId;
```

Instead Of

```
CREATE OR REPLACE TRIGGER pedido_insert
INSTEAD OF INSERT ON pedido_info

BEGIN

INSERT INTO clientes(clienteId, apellido, nombre) VALUES (
:new.clienteId, :new.apellido, :new.nombre);

INSERT INTO pedidos (pdidoId, fecha, clienteId) VALUES ( :new.pedidoId,
:new.fecha, :new.clienteId);

END;
```

Orden de ejecución

- Sobre una misma tabla puede haber varios triggers. El orden de ejecución sería:

1. Disparadores de tipo BEFORE de tipo instrucción
2. Disparadores de tipo BEFORE por cada fila
3. Se ejecuta la propia orden que desencadenó al trigger.
4. Disparadores de tipo AFTER con nivel de fila.
5. Disparadores de tipo AFTER con nivel de instrucción.

Triggers

- **`DROP TRIGGER nombreTrigger;`**

  Elimina un trigger.

- **`ALTER TRIGGER nombreTrigger DISABLE;`**

  Desactiva un trigger.

- **`ALTER TRIGGER nombreTrigger ENABLE;`**

  Activa un trigger.

- **`ALTER TABLE nombreTabla {DISABLE|ENABLE} ALL TRIGGERS;`**

  Desactiva o activa todos los triggers de una tabla.