TECNICAS DE ALMACENAMIENTO DE DATOS

TRIGGERS

- se conocen como disparadores o desencadenadores
- es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento sobre una determinada tabla
- los eventos pueden ser INSERT, UPDATE o DELETE (triggers DML)
- es un mecanismo adicional para conservar la integridad referencial

TRIGGERS

Diferencias con los STORED PROCEDURES:

- no pueden ser invocados directamente ya que el disparador se ejecuta automáticamente al intentar modificar los datos de su tabla asociada
- no reciben ni retornan parámetros

TRIGGERS

Privilegios necesarios para crear triggers:

- **ALTER ANY TABLE**
- **CREATE ANY TRIGGER**
- **ALTER ANY TRIGGER**
- **DROP ANY TRIGGER**

Para ver triggers creados:

```
SELECT * FROM USER_TRIGGERS;
```

TRIGGERS

Existen distintos tipos de triggers:

- Triggers DML
- Triggers INSTEAD OF
- Triggers de sistema (DATABASE y SCHEMA)

TRIGGERS - SINTAXIS

```
CREATE [ OR REPLACE ] TRIGGER TG_NAME
[ ENABLE | DISABLE ]
CUANDO → ANTES O DESPUES DE LA ACCION
ACCION → ORDEN DML QUE GENERA INVOCACIÓN
ON TABLE_NAME
NIVEL → SENTENCIA O FILA

DECLARE
     --VARIABLES
BEGIN
     --SENTENCIAS
END TG_NAME;
```

TRIGGERS - CUANDO

```
--SINTAXIS
CREATE [ OR REPLACE ] TRIGGER TG_NAME
CUANDO → ANTES O DESPUES DE LA ACCION
ACCION → ORDEN DML QUE GENERA INVOCACIÓN
ON TABLE_NAME
NIVEL → SENTENCIA O FILA
```

- BEFORE (antes de ejecutar la acción)
- AFTER (después de ejecutar la acción)

TRIGGERS - ACCION

```
--SINTAXIS
CREATE [ OR REPLACE ] TRIGGER TG_NAME
CUANDO → BEFORE | AFTER
ACCION → ORDEN DML QUE GENERA INVOCACIÓN
ON TABLE_NAME
NIVEL → SENTENCIA O FILA
```

- **INSERT**
- **UPDATE**
- **DELETE**

Pueden concatenarse acciones con el operador OR
Ejemplo:

- **INSERT OR UPDATE**
- **INSERT OR UPDATE OR DELETE**

TRIGGERS - NIVEL

```
--SINTAXIS
CREATE [ OR REPLACE ] TRIGGER TG_NAME
CUANDO → BEFORE | AFTER
ACCION → INSERT | UPDATE | DELETE
ON TABLE_NAME
NIVEL → SENTENCIA O FILA
```

- **Nivel de sentencia**
  Se ejecutan una vez para cada instrucción DML. Son el tipo predeterminado que se crea con el comando **`CREATE TRIGGER`**

- **Nivel de fila**
  Se ejecutan una vez para cada fila afectada por una instrucción DML. Los disparadores de nivel de fila se crean utilizando la cláusula **`FOR EACH ROW`**

EJEMPLO – NIVEL SENTENCIA

```
CREATE OR REPLACE TRIGGER TG_REGIONS_1
BEFORE UPDATE
ON REGIONS

DECLARE

   VMSG VARCHAR2(100);

BEGIN

   VMSG := 'TRIGGER UPDATE REGIONS - NIVEL DE SENTENCIA';
   DBMS_OUTPUT.PUT_LINE(VMSG);

END TG_REGIONS_1;
```

EJEMPLO – NIVEL SENTENCIA

```
--PRUEBA TRIGGER
UPDATE REGIONS
   SET REGION_NAME = UPPER(REGION_NAME)
 WHERE REGION_ID IN (1,2);

--DESHABILITAR TRIGGER
ALTER TRIGGER TG_REGIONS_1 DISABLE;
```

EJEMPLO – NIVEL FILA

```
CREATE OR REPLACE TRIGGER TG_REGIONS_2
BEFORE UPDATE
ON REGIONS
FOR EACH ROW

DECLARE

   VMSG VARCHAR2(100);

BEGIN

   VMSG := 'TRIGGER UPDATE REGIONS - NIVEL DE FILA';
   DBMS_OUTPUT.PUT_LINE(VMSG);

END TG_REGIONS_2;
```

EJEMPLO – NIVEL FILA

```
--PRUEBA TRIGGER
UPDATE REGIONS
   SET REGION_NAME = UPPER(REGION_NAME)
 WHERE REGION_ID IN (1,2);

--DESHABILITAR TRIGGER
ALTER TRIGGER TG_REGIONS_2 DISABLE;
```

ORDEN DE EJECUCION

- Una misma tabla puede tener varios triggers
- Los triggers se activan al ejecutarse la sentencia SQL

1. TRIGGER de tipo **BEFORE** (nivel de **sentencia**)
2. TRIGGER de tipo **BEFORE** (nivel de **fila**)
3. **SENTENCIA DML**
4. TRIGGER de tipo **AFTER** (nivel de **fila**)
5. TRIGGER de tipo **AFTER** (nivel de **sentencia**)

TRIGGER MULTIPLES ACCIONES

En triggers definidos para múltiples acciones Oracle ofrece el uso de las palabras reservadas:

- INSERTING
- UPDATING
- DELETING

Dichas variables adoptan los valores **`TRUE`** o **`FALSE`** según la acción que estamos ejecutando

EJEMPLO MULTIPLES ACCIONES

```
CREATE OR REPLACE TRIGGER TG_REGIONS_3
AFTER INSERT OR UPDATE OR DELETE
ON REGIONS

DECLARE
  VMSG VARCHAR2(100);
BEGIN

  VMSG := 'TRIGGER REGIONS - NIVEL DE SENTENCIA';
  IF INSERTING THEN
    dbms_output.put_line(VMSG || ' - INSERTING');
  ELSIF UPDATING THEN
    dbms_output.put_line(VMSG || ' - UPDATING');
  ELSIF DELETING THEN
    dbms_output.put_line(VMSG || ' - DELETING');
  END IF;

END TG_REGIONS_3;
```

EJEMPLO MULTIPLES ACCIONES

```
--PRUEBA TRIGGER
INSERT INTO REGIONS (REGION_ID,REGION_NAME)
VALUES (99,'Oceania');

UPDATE REGIONS
   SET REGION_NAME = UPPER(REGION_NAME)
 WHERE REGION_ID = 99;

DELETE
  FROM REGIONS
 WHERE REGION_ID = 99;

--DESHABILITAR TRIGGER
ALTER TRIGGER TG_REGIONS_3 DISABLE;
```

DATOS DE LA SENTENCIA DML

Para acceder a los conjuntos de datos de la sentencia DML, Oracle ofrece el uso de los registros:

- **:OLD**
- **:NEW**

Ambos tienen la estructura de los campos de la tabla que se está modificando (ídem a **%ROWTYPE**) y contienen una copia del registro antes (OLD) y después (NEW) de la acción SQL (**INSERT, UPDATE, DELETE**)

EJEMPLO OLD / NEW

```
CREATE OR REPLACE TRIGGER TG_REGIONS_4
BEFORE UPDATE
ON REGIONS
FOR EACH ROW
DECLARE
  VMSG VARCHAR2(100);
BEGIN

  VMSG := 'TRIGGER UPDATE REGIONS - REGION_NAME OLD: ';
  VMSG := VMSG || :OLD.REGION_NAME;
  DBMS_OUTPUT.PUT_LINE(VMSG);

  VMSG := 'TRIGGER UPDATE REGIONS - REGION_NAME NEW: ';
  VMSG := VMSG || :NEW.REGION_NAME;
  DBMS_OUTPUT.PUT_LINE(VMSG);

END TG_REGIONS_4;
```

EJEMPLO OLD / NEW

```
--PRUEBA TRIGGER
DELETE
  FROM REGIONS
 WHERE REGION_ID = 99;

INSERT INTO REGIONS (REGION_ID,REGION_NAME)
VALUES (99,'Oceania');

UPDATE REGIONS
   SET REGION_NAME = UPPER(REGION_NAME)
 WHERE REGION_ID = 99;

--DESHABILITAR TRIGGER
ALTER TRIGGER TG_REGIONS_4 DISABLE;
```

DATOS DE LA SENTENCIA DML

ACCION SQL	OLD	NEW
INSERT	No definido; todos los campos toman valor NULL	Valores que serán insertados cuando se complete la orden
UPDATE	Valores originales de la fila, antes de la actualización	Nuevos valores que se impactarán cuando se complete la orden
DELETE	Valores antes del borrado de la fila	No definido; todos los campos toman el valor NULL

MODIFICAR TRIGGERS

```
--DESHABILITAR TRIGGER INDIVIDUALMENTE
ALTER TRIGGER TRIGGER_NAME DISABLE;

--DESHABILITAR TRIGGERS DE UNA TABLA
ALTER TABLE TABLE_NAME DISABLE ALL TRIGGERS;

--HABILITAR TRIGGER INDIVIDUALMENTE
ALTER TRIGGER TRIGGER_NAME ENABLE;

--HABILITAR TRIGGERS DE UNA TABLA
ALTER TABLE TABLE_NAME ENABLE ALL TRIGGERS;

--RENOMBRAR TRIGGER
ALTER TRIGGER TRIGGER_NAME RENAME TO NEW_NAME;

--BORRAR TRIGGER
DROP TRIGGER TRIGGER_NAME;
```

EJERCICIO

Crear un trigger sobre la tabla REGIONS que ante un INSERT con valor de REGION_ID nulo le cargue el valor máximo existente en la tabla +1.

```
CREATE OR REPLACE TRIGGER TRIGGER_NAME
<BEFORE | AFTER> <ACTION>
ON TABLE_NAME
[FOR EACH ROW]

DECLARE
     --VARIABLES
BEGIN
     --SENTENCIAS
END TRIGGER_NAME;
```

```
--PRUEBAS
INSERT
INTO REGIONS (REGION_ID,REGION_NAME)
VALUES (NULL,'Region Prueba');

SELECT * FROM REGIONS
ORDER BY REGION_ID DESC;
```