SQL Dinámico

# SQL Dinámico

## Conceptos clave

* **Limitación de PL/SQL**: No permite ejecutar comandos DDL (CREATE, ALTER, DROP) o sentencias dinámicas sin SQL dinámico.
* **Técnicas**:
  + DBMS\_SQL: Método antiguo, complejo, usado antes de Oracle8i.
  + SQL Dinámico Nativo: Desde Oracle8i, usa EXECUTE IMMEDIATE, más simple e integrado.
* **EXECUTE IMMEDIATE**: Ejecuta comandos SQL o bloques PL/SQL dinámicos en tiempo de ejecución.

## Ejemplo procedimiento

DECLARE

v\_SQLString VARCHAR2(200);

BEGIN

-- Crear tabla dinámicamente

EXECUTE IMMEDIATE 'CREATE TABLE execute\_table (col1 VARCHAR(10))';

-- Insertar filas dinámicamente

FOR v\_Counter IN 1..10 LOOP

v\_SQLString := 'INSERT INTO execute\_table VALUES (''Row ' || v\_Counter || ''')';

EXECUTE IMMEDIATE v\_SQLString;

END LOOP;

-- Mostrar contenido

EXECUTE IMMEDIATE '

BEGIN

FOR v\_Rec IN (SELECT \* FROM execute\_table) LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_Rec.col1);

END LOOP;

END;

';

-- Borrar tabla

EXECUTE IMMEDIATE 'DROP TABLE execute\_table';

END;

### Anotaciones

* No se usa ; en comandos individuales dentro de EXECUTE IMMEDIATE.
* Sí se usa ; en bloques PL/SQL completos.

### Usos

* Crear usuarios, modificar esquemas, ejecutar DML dinámico.

## Ejemplo procedimiento

CREATE OR REPLACE PROCEDURE CREARUSUARIO(P\_NOMBRE VARCHAR2, P\_CLAVE VARCHAR2) IS

V\_CREARUSUARIO VARCHAR2(250);

V\_ROLE VARCHAR2(150);

BEGIN

V\_CREARUSUARIO := 'CREATE USER ' || P\_NOMBRE || ' IDENTIFIED BY ' || P\_CLAVE ||

' DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP';

EXECUTE IMMEDIATE V\_CREARUSUARIO;

V\_ROLE := 'GRANT CONNECT, RESOURCE TO ' || P\_NOMBRE;

EXECUTE IMMEDIATE V\_ROLE;

END;

### Uso

* Automatiza creación de usuarios (ej.: VS1DAW1, VS1DAW2, etc.).

### Privilegios

* El propietario necesita CREATE USER, CREATE TABLE y roles CONNECT, RESOURCE con WITH ADMIN OPTION.

# Disparadores de base de datos (triggers)

## Conceptos clave

### Definición

* Bloques PL/SQL que se ejecutan automáticamente ante eventos (DML: INSERT, UPDATE, DELETE; DDL: CREATE, DROP; o eventos del sistema: LOGON, STARTUP).

### Tipos

* **DML**: Antes (BEFORE) o después (AFTER) de INSERT, UPDATE, DELETE. Pueden ser por fila (FOR EACH ROW) o por orden.
* **Sustitución**: Para vistas no actualizables directamente (INSTEAD OF).
* **Sistema**: Para eventos DDL o de base de datos (desde Oracle8i).

### Usos

* Restricciones complejas.
* Auditorías de cambios.
* Notificaciones automáticas.

### Restricciones

* No permite COMMIT, ROLLBACK ni tipos LONG/LONG RAW.
* Tablas mutantes: No se puede leer/modificar la tabla que se está modificando en un disparador a nivel de fila.

## Ejemplo disparador de sustitución

CREATE OR REPLACE TRIGGER TRVEMP

INSTEAD OF INSERT OR UPDATE OR DELETE ON VEMP

FOR EACH ROW

DECLARE

NALOJ EMPLEADO.ALOJAMIENTO%TYPE;

BEGIN

IF INSERTING OR UPDATING THEN

SELECT NUMALOJ INTO NALOJ

FROM ALOJAMIENTO

WHERE ALOJAMIENTO = :NEW.ALOJAMIENTO;

END IF;

IF INSERTING THEN

INSERT INTO EMPLEADO (NUMEMP, NOMBRE, ALOJAMIENTO)

VALUES (:NEW.NUMEMP, :NEW.NOMBRE, NALOJ);

ELSIF UPDATING THEN

UPDATE EMPLEADO

SET NOMBRE = :NEW.NOMBRE, ALOJAMIENTO = NALOJ

WHERE NUMEMP = :OLD.NUMEMP;

ELSE

DELETE FROM EMPLEADO

WHERE NUMEMP = :OLD.NUMEMP;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'ALOJAMIENTO NO VALIDO');

END;

### Explicación

* Permite insertar/actualizar/borrar en una vista (VEMP) que combina tablas EMPLEADO y ALOJAMIENTO.

## Tablas mutantes y solución

### Problema

* No se puede leer la tabla que se modifica en un disparador a nivel de fila (error ORA-04091).

### Solución tradicional

* Usar un paquete para almacenar datos temporalmente.
* Combinar disparadores a nivel de fila y orden.

### Ejemplo

CREATE OR REPLACE PACKAGE PACKCONTROLCUENTA AS

TYPE T\_CUENTA IS TABLE OF TITULAR.NUMCUENTA%TYPE INDEX BY BINARY\_INTEGER;

V\_CUENTA T\_CUENTA;

V\_FILAS NUMBER := 0;

END;

CREATE OR REPLACE TRIGGER ANOTARCUENTA

BEFORE INSERT OR UPDATE ON TITULAR

FOR EACH ROW

BEGIN

PACKCONTROLCUENTA.V\_FILAS := PACKCONTROLCUENTA.V\_FILAS + 1;

PACKCONTROLCUENTA.V\_CUENTA(PACKCONTROLCUENTA.V\_FILAS) := :NEW.NUMCUENTA;

END;

CREATE OR REPLACE TRIGGER CONTROLTITULARES

AFTER INSERT OR UPDATE ON TITULAR

DECLARE

V\_NUMTITULARES NUMBER;

V\_MAXTITULARES CONSTANT NUMBER := 3;

BEGIN

FOR I IN 1..PACKCONTROLCUENTA.V\_FILAS LOOP

SELECT COUNT(NUMCLI) INTO V\_NUMTITULARES

FROM TITULAR

WHERE NUMCUENTA = PACKCONTROLCUENTA.V\_CUENTA(I);

IF V\_NUMTITULARES > V\_MAXTITULARES THEN

PACKCONTROLCUENTA.V\_FILAS := 0;

RAISE\_APPLICATION\_ERROR(-20000, 'PUEDE HABER UN MAXIMO DE 3 TITULARES POR CUENTA');

END IF;

END LOOP;

PACKCONTROLCUENTA.V\_FILAS := 0;

END;

#### Explicación

* Controla que no haya más de 3 titulares por cuenta. Usa un paquete para evitar problemas con tablas mutantes.

## Disparadores Compound (Solución moderna, Oracle 11g+)

* Combina BEFORE STATEMENT, AFTER STATEMENT, BEFORE EACH ROW, AFTER EACH ROW en un solo disparador.
* No necesita paquetes auxiliares; las variables son accesibles en todas las secciones.

# Ejemplo general

## Código

-- 1. Crear las tablas necesarias

CREATE TABLE PRODUCTOS (

IDPRODUCTO NUMBER PRIMARY KEY,

STOCK NUMBER

);

CREATE TABLE AUDITORIA\_PRODUCTOS (

ID\_AUDIT NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

TIPO\_CAMBIO CHAR(1),

USUARIO\_MODIFICACION VARCHAR2(100),

FECHA\_CAMBIO DATE,

IDPRODUCTO\_ANTERIOR NUMBER,

STOCK\_ANTERIOR NUMBER,

IDPRODUCTO\_NUEVO NUMBER,

STOCK\_NUEVO NUMBER

);

-- 2. Crear el paquete PKG\_CONTROL\_STOCK

CREATE OR REPLACE PACKAGE PKG\_CONTROL\_STOCK AS

TYPE T\_PRODUCTO\_IDS IS TABLE OF PRODUCTOS.IDPRODUCTO%TYPE

INDEX BY BINARY\_INTEGER;

V\_PRODUCTO\_IDS T\_PRODUCTO\_IDS;

V\_FILAS NUMBER := 0;

END;

/

-- 3. Crear el trigger que anota los productos afectados

CREATE OR REPLACE TRIGGER trg\_registrar\_producto

BEFORE INSERT OR UPDATE OF STOCK ON PRODUCTOS

FOR EACH ROW

BEGIN

-- Solo registrar si el ID es válido

IF :NEW.IDPRODUCTO IS NOT NULL THEN

pkg\_control\_stock.v\_filas := pkg\_control\_stock.v\_filas + 1;

pkg\_control\_stock.v\_producto\_ids(pkg\_control\_stock.v\_filas) := :NEW.IDPRODUCTO;

END IF;

END;

/

-- 4. Crear el trigger COMPOUND que valida el stock

CREATE OR REPLACE TRIGGER trg\_validar\_stock

FOR INSERT OR UPDATE OF STOCK ON PRODUCTOS

COMPOUND TRIGGER

v\_stock NUMBER;

AFTER STATEMENT IS

BEGIN

FOR i IN 1 .. pkg\_control\_stock.v\_filas LOOP

-- SQL dinámico para validar que ningún producto quede con stock negativo

BEGIN

IF pkg\_control\_stock.v\_producto\_ids.EXISTS(i) AND pkg\_control\_stock.v\_producto\_ids(i) IS NOT NULL THEN

EXECUTE IMMEDIATE 'SELECT NVL(STOCK, 0) FROM PRODUCTOS WHERE IDPRODUCTO = :1'

INTO v\_stock

USING pkg\_control\_stock.v\_producto\_ids(i);

IF v\_stock < 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Producto con stock negativo: ' || pkg\_control\_stock.v\_producto\_ids(i));

END IF;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL; -- Ignorar si el producto no existe

WHEN OTHERS THEN

NULL; -- Ignorar cualquier otro error para evitar ORA-04088

END;

END LOOP;

-- Resetear variables del paquete

pkg\_control\_stock.v\_filas := 0;

pkg\_control\_stock.v\_producto\_ids.DELETE; -- Limpiar la colección

END AFTER STATEMENT;

END;

/

-- 5. Crear el trigger de auditoría

CREATE OR REPLACE TRIGGER trg\_auditar\_cambios\_productos

BEFORE INSERT OR UPDATE OR DELETE ON PRODUCTOS

FOR EACH ROW

WHEN (NEW.STOCK IS NULL OR NEW.STOCK > 50) -- Solo si el nuevo stock es NULL o > 50

DECLARE

v\_tipo\_cambio CHAR(1);

BEGIN

IF INSERTING THEN

v\_tipo\_cambio := 'I';

ELSIF UPDATING THEN

v\_tipo\_cambio := 'U';

ELSIF DELETING THEN

v\_tipo\_cambio := 'D';

END IF;

INSERT INTO AUDITORIA\_PRODUCTOS

(TIPO\_CAMBIO, USUARIO\_MODIFICACION, FECHA\_CAMBIO,

IDPRODUCTO\_ANTERIOR, STOCK\_ANTERIOR,

IDPRODUCTO\_NUEVO, STOCK\_NUEVO)

VALUES

(v\_tipo\_cambio, USER, SYSDATE,

:OLD.IDPRODUCTO, :OLD.STOCK,

:NEW.IDPRODUCTO, :NEW.STOCK);

END;

/

-- 6. Bloque PL/SQL para pruebas y mostrar resultados

BEGIN

-- Operaciones DML de prueba

DBMS\_OUTPUT.PUT\_LINE('Iniciando operaciones DML de prueba...');

-- Insertar un producto con stock > 50 (debería auditar)

INSERT INTO PRODUCTOS (IDPRODUCTO, STOCK) VALUES (1, 100);

DBMS\_OUTPUT.PUT\_LINE('Producto 1 insertado con stock 100.');

-- Insertar un producto con stock < 50 (no debería auditar)

INSERT INTO PRODUCTOS (IDPRODUCTO, STOCK) VALUES (2, 30);

DBMS\_OUTPUT.PUT\_LINE('Producto 2 insertado con stock 30.');

-- Actualizar producto a stock negativo (debería fallar)

BEGIN

UPDATE PRODUCTOS SET STOCK = -10 WHERE IDPRODUCTO = 1;

DBMS\_OUTPUT.PUT\_LINE('Producto 1 actualizado a stock -10.'); -- No se ejecuta si falla

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error al actualizar producto 1: ' || SQLERRM);

END;

-- Actualizar producto a stock > 50 (debería auditar)

UPDATE PRODUCTOS SET STOCK = 60 WHERE IDPRODUCTO = 2;

DBMS\_OUTPUT.PUT\_LINE('Producto 2 actualizado a stock 60.');

-- Eliminar un producto (debería auditar porque NEW.STOCK es NULL)

DELETE FROM PRODUCTOS WHERE IDPRODUCTO = 1;

DBMS\_OUTPUT.PUT\_LINE('Producto 1 eliminado.');

-- Mostrar los registros de auditoría

DBMS\_OUTPUT.PUT\_LINE('--- Registros en AUDITORIA\_PRODUCTOS ---');

FOR rec IN (SELECT ID\_AUDIT, TIPO\_CAMBIO, USUARIO\_MODIFICACION,

TO\_CHAR(FECHA\_CAMBIO, 'YYYY-MM-DD HH24:MI:SS') AS FECHA\_CAMBIO,

IDPRODUCTO\_ANTERIOR, STOCK\_ANTERIOR,

IDPRODUCTO\_NUEVO, STOCK\_NUEVO

FROM AUDITORIA\_PRODUCTOS)

LOOP

DBMS\_OUTPUT.PUT\_LINE('ID\_AUDIT: ' || rec.ID\_AUDIT ||

', TIPO: ' || rec.TIPO\_CAMBIO ||

', USUARIO: ' || rec.USUARIO\_MODIFICACION ||

', FECHA: ' || rec.FECHA\_CAMBIO ||

', ID\_ANTERIOR: ' || NVL(TO\_CHAR(rec.IDPRODUCTO\_ANTERIOR), 'NULL') ||

', STOCK\_ANTERIOR: ' || NVL(TO\_CHAR(rec.STOCK\_ANTERIOR), 'NULL') ||

', ID\_NUEVO: ' || NVL(TO\_CHAR(rec.IDPRODUCTO\_NUEVO), 'NULL') ||

', STOCK\_NUEVO: ' || NVL(TO\_CHAR(rec.STOCK\_NUEVO), 'NULL'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- Fin de registros ---');

END;

/

-- 7. Bloque PL/SQL para limpieza consolidada de objetos creados

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Iniciando limpieza de objetos creados...');

-- Eliminar disparadores

FOR obj IN (SELECT 'DROP TRIGGER ' || object\_name AS stmt, object\_name

FROM user\_objects

WHERE object\_type = 'TRIGGER'

AND object\_name IN ('TRG\_AUDITAR\_CAMBIOS\_PRODUCTOS', 'TRG\_VALIDAR\_STOCK', 'TRG\_REGISTRAR\_PRODUCTO'))

LOOP

BEGIN

EXECUTE IMMEDIATE obj.stmt;

DBMS\_OUTPUT.PUT\_LINE('Trigger ' || obj.object\_name || ' eliminado.');

EXCEPTION

WHEN OTHERS THEN

NULL; -- Ignorar errores

END;

END LOOP;

-- Eliminar paquete

BEGIN

EXECUTE IMMEDIATE 'DROP PACKAGE PKG\_CONTROL\_STOCK';

DBMS\_OUTPUT.PUT\_LINE('Paquete PKG\_CONTROL\_STOCK eliminado.');

EXCEPTION

WHEN OTHERS THEN

NULL; -- Ignorar errores

END;

-- Eliminar tablas

FOR obj IN (SELECT 'DROP TABLE ' || object\_name AS stmt, object\_name

FROM user\_objects

WHERE object\_type = 'TABLE'

AND object\_name IN ('AUDITORIA\_PRODUCTOS', 'PRODUCTOS'))

LOOP

BEGIN

EXECUTE IMMEDIATE obj.stmt;

DBMS\_OUTPUT.PUT\_LINE('Tabla ' || obj.object\_name || ' eliminada.');

EXCEPTION

WHEN OTHERS THEN

NULL; -- Ignorar errores

END;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Limpieza completada.');

END;

/