# **Sentencias SQL Permitidas en PL/SQL**:

* + Solo se permiten sentencias de **Manipulación de Datos (DML)**: SELECT, INSERT, UPDATE, DELETE.
  + También se permiten sentencias de **Control de Transacciones**: COMMIT, ROLLBACK, SAVEPOINT.
    - Estas sentencias gestionan transacciones, asegurando que los cambios en la base de datos sean consistentes (es decir, que se apliquen completamente o no se apliquen en absoluto).
    - Ejemplo: Después de insertar una factura, un COMMIT hace permanente el cambio; un ROLLBACK lo deshace si ocurre un error.
  + Las sentencias de **Definición de Datos (DDL)** (como CREATE TABLE) no están permitidas debido al **acoplamiento temprano** (resolución de variables durante la compilación), que impide cambios en el esquema en tiempo de ejecución.
    - La razón es el acoplamiento temprano: durante la compilación, PL/SQL necesita conocer la estructura de los objetos (tablas, columnas) referenciados. Si una sentencia DDL crea o modifica un objeto en tiempo de ejecución, PL/SQL no puede verificar su existencia o permisos en la fase de compilación, lo que provocaría errores.

## **SELECT en PL/SQL**:

* + Recupera datos y los almacena en variables PL/SQL o registros.
  + Debe devolver exactamente una fila para evitar el error ORA-1427 (demasiadas filas).
  + Se puede usar %ROWTYPE para definir registros que coincidan con la estructura de una tabla.
  + Las cláusulas como WHERE, ORDER BY, GROUP BY son válidas.

## **INSERT, UPDATE, DELETE**:

* + **INSERT**:
    - Añade nuevas filas a una tabla.
    - Puede usar valores literales, variables PL/SQL o resultados de subconsultas.
    - INSERT INTO students (id, first\_name, last\_name) VALUES (v\_StudentID, 'Max', 'Verstappen'); inserta un nuevo estudiante usando una variable para el ID.
  + **UPDATE**:
    - Modifica valores en columnas existentes.
    - La cláusula SET específica los nuevos valores, y WHERE determina qué filas se actualizan.
    - UPDATE students SET current\_credits = current\_credits + v\_CreditIncrease WHERE major = v\_Major; aumenta los créditos para estudiantes de una especialidad.
  + **DELETE**:
    - Elimina filas de una tabla según la condición en WHERE.
    - Sin WHERE, elimina todas las filas, lo que debe usarse con precaución.
    - DELETE FROM classes WHERE current\_students < v\_StudentCutoff; elimina clases con pocos estudiantes.
  + Usan sintaxis SQL estándar, pero permiten incluir variables PL/SQL en cláusulas como WHERE o SET.
  + La cláusula WHERE define el conjunto de filas afectadas (activo).

# **Precaución con Nombres de Variables**:

* + Evitar nombrar variables PL/SQL igual que columnas de tablas, ya que PL/SQL prioriza columnas sobre variables en cláusulas WHERE. Esto puede causar errores lógicos, como eliminar todas las filas
    - DELETE FROM classes WHERE department = Department; , si Department es una variable PL/SQL pero también una columna, PL/SQL compara la columna department consigo misma (department = department), lo que es siempre verdadero y elimina todas las filas.
  + Solución:
    - Usar nombres de variables distintos a los de las columnas (por ejemplo, v\_Department en lugar de Department).

# **Semántica de Comparación de Caracteres**:

* + **Con relleno de espacios** (para CHAR): Las cadenas más cortas se rellenan con espacios para compararlas.
  + **Sin relleno de espacios** (para VARCHAR2): La longitud de la cadena afecta la comparación.
  + Problemas Comunes:
    - Si una variable PL/SQL es VARCHAR2 pero la columna es CHAR, las comparaciones pueden fallar inesperadamente.
  + Solución:
    - Usar %TYPE asegura que las variables coincidan con el tipo de datos de las columnas, evitando problemas de comparación.

# **Control de Transacciones**:

* + Una transacción es un conjunto de sentencias SQL que se ejecutan o fallan como una unidad.
  + Transacción:
    - Comienza con la primera sentencia SQL después de un COMMIT, ROLLBACK o la conexión a la base de datos.
    - Termina con COMMIT o ROLLBACK.
  + COMMIT:
    - Hace permanentes los cambios de la transacción.
    - Permite que otras sesiones vean los cambios.
    - Libera bloqueos establecidos por la transacción.
  + ROLLBACK
    - deshace los cambios
    - libera bloqueos.
  + SAVEPOINT
    - Marca un punto en la transacción al que se puede volver con ROLLBACK TO SAVEPOINT.
    - No finaliza la transacción, permitiendo deshacer solo una parte.

# EJEMPLOS

Ejemplo 1: Uso de SELECT con %ROWTYPE

DECLARE

v\_EmpRecord Empleado%ROWTYPE; -- Registro basado en la estructura de la tabla Empleado

v\_EmpID Empleado.numemp%TYPE := 4; -- Nombre diferente a 'numemp' para evitar confusión

BEGIN

-- Selecciona los detalles del empleado en el registro

SELECT \*

INTO v\_EmpRecord

FROM Empleado

WHERE numemp = v\_EmpID;

-- Muestra los detalles

DBMS\_OUTPUT.PUT\_LINE('Nombre: ' || v\_EmpRecord.nombre);

DBMS\_OUTPUT.PUT\_LINE('Edad: ' || v\_EmpRecord.edad);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No se encontró empleado con ID ' || v\_EmpID);

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Múltiples empleados encontrados');

END;

/

Ejemplo 2: inserta nuevo empleado, actualiza su edad, y confirma el cambio

DECLARE

v\_EmpRecord Empleado%ROWTYPE;

v\_NewID Empleado.numemp%TYPE;

BEGIN

-- Punto de restauración por si algo falla

SAVEPOINT inicio\_operacion;

-- Obtener el siguiente ID disponible

SELECT NVL(MAX(numemp), 0) + 1 INTO v\_NewID FROM Empleado;

-- Insertar nuevo empleado

INSERT INTO Empleado (numemp, nombre, edad)

VALUES (v\_NewID, 'Ana Torres', 30);

-- Seleccionar datos insertados

SELECT \*

INTO v\_EmpRecord

FROM Empleado

WHERE numemp = v\_NewID;

DBMS\_OUTPUT.PUT\_LINE('Empleado insertado:');

DBMS\_OUTPUT.PUT\_LINE('Nombre: ' || v\_EmpRecord.nombre);

DBMS\_OUTPUT.PUT\_LINE('Edad: ' || v\_EmpRecord.edad);

-- Actualizar la edad

UPDATE Empleado

SET edad = edad + 1

WHERE numemp = v\_NewID;

-- Volver a leer el empleado actualizado

SELECT \*

INTO v\_EmpRecord

FROM Empleado

WHERE numemp = v\_NewID;

DBMS\_OUTPUT.PUT\_LINE('Empleado actualizado:');

DBMS\_OUTPUT.PUT\_LINE('Nombre: ' || v\_EmpRecord.nombre);

DBMS\_OUTPUT.PUT\_LINE('Edad: ' || v\_EmpRecord.edad);

-- Confirmar los cambios

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No se encontró el empleado con ID ' || v\_NewID);

ROLLBACK TO inicio\_operacion;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK TO inicio\_operacion;

END;

/

Ejemplo 3: Comparación de Caracteres con %TYPE y UPDATE

DECLARE

v\_EmpNombre Empleado.nombre%TYPE := 'WILLIAM SWING'; -- Usa %TYPE para coincidir con la columna

v\_AumentoSueldo Empleado.sueldo%TYPE := 100.50;

BEGIN

-- Actualiza el sueldo del empleado

UPDATE Empleado

SET sueldo = sueldo + v\_AumentoSueldo

WHERE nombre = v\_EmpNombre;

-- Verifica si se actualizó alguna fila

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No se encontró empleado con nombre ' || v\_EmpNombre);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Sueldo actualizado para ' || v\_EmpNombre || '. Filas afectadas: ' || SQL%ROWCOUNT);

END IF;

-- Confirma los cambios

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;

/