**What is PL/SQL?**

**PL/SQL (Procedural Language/Structured Query Language)** is Oracle's procedural extension to SQL. It allows developers to write complex scripts combining SQL queries with procedural constructs like loops, conditions, and exception handling.

**Key Features of PL/SQL (Expanded)**

* **Portability**: PL/SQL code runs on any Oracle database, regardless of platform.
* **Efficiency**: Reduces network traffic by bundling multiple SQL statements into a single block.
* **Exception Handling**: Built-in mechanisms to catch and handle runtime errors gracefully.
* **Development Tools**: Supported by IDEs like Oracle SQL Developer, TOAD, and PL/SQL Developer.
* **Error Checking**: Compile-time and runtime error detection with detailed diagnostics.
* **Modularity**: Supports procedures, functions, packages, and triggers for reusable code.
* **Security**: Allows fine-grained access control through stored procedures and roles.

**Data Types in PL/SQL**

PL/SQL supports a wide range of data types:

|  |  |
| --- | --- |
| Category | Examples |
| Scalar | NUMBER, VARCHAR2, CHAR, DATE, BOOLEAN |
| Composite | RECORD, TABLE, VARRAY |
| Reference | REF CURSOR |
| LOBs | BLOB, CLOB, NCLOB, BFILE |

**🔹 Declaring Constants**

DECLARE

school\_name CONSTANT VARCHAR2(20) := 'DPS';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('I study in ' || school\_name);

END;

**Control Structures & Examples**

**IF-ELSE Example**

DECLARE

a1 NUMBER := 10;

b NUMBER := 100;

c1 NUMBER;

BEGIN

IF a1 > b THEN

c1 := a1;

ELSE

c1 := b;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Maximum number in 10 and 100: ' || c1);

END;

**Even/Odd Check**

DECLARE

a1 NUMBER := :a1;

BEGIN

IF MOD(a1, 2) = 0 THEN

DBMS\_OUTPUT.PUT\_LINE(a1 || ' is Even');

ELSE

DBMS\_OUTPUT.PUT\_LINE(a1 || ' is Odd');

END IF;

END;

**CASE Statement**

DECLARE

c\_grade CHAR(1) := :c\_grade;

c\_rank VARCHAR2(20);

BEGIN

CASE c\_grade

WHEN 'A' THEN c\_rank := 'Excellent';

WHEN 'B' THEN c\_rank := 'Very Good';

WHEN 'C' THEN c\_rank := 'Good';

WHEN 'D' THEN c\_rank := 'Fair';

WHEN 'F' THEN c\_rank := 'Poor';

ELSE c\_rank := 'Invalid Grade';

END CASE;

DBMS\_OUTPUT.PUT\_LINE('Grade: ' || c\_grade);

DBMS\_OUTPUT.PUT\_LINE('Rank: ' || c\_rank);

END;

**Multiplication Table**

DECLARE

i INT := :i;

j INT := 1;

BEGIN

LOOP

EXIT WHEN j > 10;

DBMS\_OUTPUT.PUT\_LINE(i || ' \* ' || j || ' = ' || (i \* j));

j := j + 1;

END LOOP;

END;

**Functions in PL/SQL**

Functions return a value and can be reused across blocks.

**Odd or Even Function**

DECLARE

a1 NUMBER := 50;

c1 NUMBER;

FUNCTION oddOrEven(x IN NUMBER) RETURN NUMBER IS

BEGIN

IF MOD(x, 2) = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('EVEN NUMBER');

ELSE

DBMS\_OUTPUT.PUT\_LINE('ODD NUMBER');

END IF;

RETURN -1; -- Placeholder return

END;

BEGIN

c1 := oddOrEven(a1);

END;

**Cursors in PL/SQL**

**🔹 Implicit Cursor**

Used automatically for single-row operations like INSERT, UPDATE, DELETE.

BEGIN

UPDATE employees SET salary = salary \* 1.1 WHERE department\_id = 10;

IF SQL%ROWCOUNT > 0 THEN

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' rows updated.');

END IF;

END;

**🔹 Explicit Cursor**

Used for multi-row queries.

DECLARE

CURSOR c\_customer IS SELECT name FROM customer;

TYPE c\_list IS TABLE OF customer.name%TYPE;

name\_list c\_list := c\_list();

counter INTEGER := 0;

BEGIN

FOR n IN c\_customer LOOP

counter := counter + 1;

name\_list.EXTEND;

name\_list(counter) := n.name;

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || n.name);

END LOOP;

END;

**Triggers in PL/SQL**

Triggers are procedures that automatically execute in response to specific events on a table or view.

**🔹 Example: AFTER INSERT Trigger**

CREATE OR REPLACE TRIGGER trg\_after\_insert

AFTER INSERT ON employees

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('New employee added: ' || :NEW.name);

END;

**Packages in PL/SQL**

Packages group related procedures, functions, variables, and cursors.

**🔹 Example Package Structure**

-- Specification

CREATE OR REPLACE PACKAGE emp\_pkg IS

PROCEDURE showEmployees;

END emp\_pkg;

-- Body

CREATE OR REPLACE PACKAGE BODY emp\_pkg IS

PROCEDURE showEmployees IS

BEGIN

FOR emp IN (SELECT name FROM employees) LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp.name);

END LOOP;

END;

END emp\_pkg;