Oracle PL/SQL Programming

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

1. What is PL/SQL?

- **PL/SQL** stands for *Procedural Language/Structured Query Language*.
- It is Oracle's procedural extension to SQL.
- SQL alone can retrieve, manipulate, and manage data, but **PL/SQL adds programming features** like loops, conditions, variables, and error handling.

2. How is PL/SQL different from SQL?

SQL	PL/SQL
Declarative language for querying and manipulating data.	Procedural language that extends SQL with loops, conditions, functions, etc.
Executes one statement at a time.	Executes a block of code (multiple statements together).
Cannot handle conditional logic or loops.	Supports conditions, loops, modular programming, and exception handling.
Example: `SELECT * FROM employees;`	Example: A block that retrieves employees and applies business logic.

3. Why do we need PL/SQL?

- To **combine SQL with programming features**.
- To improve **performance** (minimizing network calls by grouping SQL statements).
- To implement **business logic** inside the database.
- To handle **exceptions (errors)** gracefully.
- To create **procedures, functions, triggers, and packages** for reusable code.

4. PL/SQL Blocks

A **PL/SQL block** is the basic unit of code. It has three sections:

```
DECLARE
   -- Declarations (variables, constants)
BEGIN
   -- Executable statements (logic, SQL queries)
EXCEPTION
   -- Error handling
END;
```

- **Anonymous blocks**: executed directly, not stored in DB.
- **Named blocks**: stored as *procedures, functions, triggers, packages*.

Output Statement

Example:

```
BEGIN
   DBMS_OUTPUT.PUT_LINE('Hello, PL/SQL!');
END;
```

5. Variables, Constants, and Attributes in PL/SQL

Variables

- Used to store data temporarily.

```
DECLARE
   v_name VARCHAR2(50);
   v_salary NUMBER(10,2);
BEGIN
   v_name := 'John';
   v_salary := 5000;
   DBMS_OUTPUT.PUT_LINE(v_name || ' earns ' || v_salary);
END;
```

Constants

- Value cannot be changed once assigned.

```
DECLARE
   c_tax_rate CONSTANT NUMBER := 0.05;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Tax Rate = ' || c_tax_rate);
END;
```

Attributes

- `%TYPE` : Use datatype of a column.

```
DECLARE
   v_empname employees.first_name%TYPE;
BEGIN
   SELECT first_name INTO v_empname FROM employees WHERE employee_id = 101;
   DBMS_OUTPUT.PUT_LINE(v_empname);
END;
```

- `%ROWTYPE` : Record with structure of a row.

```
DECLARE
   emp_record employees%ROWTYPE;
BEGIN
   SELECT * INTO emp_record FROM employees WHERE employee_id = 101;
   DBMS_OUTPUT.PUT_LINE(emp_record.first_name || ' ' || emp_record.salary);
END;
```

Working with Collections in PL/SQL

6. Collection Datatypes

Collections store multiple values.

1. **Associative Arrays (Index-by tables)**

```
DECLARE
   TYPE name_table IS TABLE OF VARCHAR2(50) INDEX BY PLS_INTEGER;
   v_names name_table;
BEGIN
   v_names(1) := 'Alice';
   v_names(2) := 'Bob';
   DBMS_OUTPUT.PUT_LINE(v_names(1));
END;
```

Associate Array

```
DECLARE
  -- Step 1: Define an associative array type
        TYPE salary_table IS TABLE OF NUMBER
         INDEX BY PLS_INTEGER;
  -- Step 2: Declare a variable of that type
        emp_salaries salary_table;
BEGIN
  -- Step 3: Assign values
        emp_salaries(100) := 5000;
         emp_salaries(101) := 6000;
        emp_salaries(102) := 5500;
  -- Step 4: Access values
        DBMS_OUTPUT.PUT_LINE('Salary of Emp 101: ' || emp_salaries(101));
END;
```

Associate Array Iteration

```
DECLARE
        -- same code as prev
      v_key PLS_INTEGER;
BEGIN
      -- same code as prev
       v_key := emp_salaries.FIRST;

       WHILE v_key IS NOT NULL LOOP
            DBMS_OUTPUT.PUT_LINE(v_key || ' → ' || emp_salaries(v_key));
             v_key := emp_salaries.NEXT(v_key);
       END LOOP;
END;
```

7. Data Type Conversion / Date Functions

- **Explicit Conversion**: `TO_CHAR`, `TO_DATE`, `TO_NUMBER`.

```
SELECT TO_DATE('2025-01-01','YYYY-MM-DD') FROM dual;
```

- **Date Functions**:
  - `SYSDATE` → current date/time
  - `ADD_MONTHS(date, n)`
  - `MONTHS_BETWEEN(date1, date2)`
  - `LAST_DAY(date)`

8. Control Structures

IF Statements

```
IF v_salary > 10000 THEN
   DBMS_OUTPUT.PUT_LINE('High salary');
ELSE
   DBMS_OUTPUT.PUT_LINE('Normal salary');
END IF;
```

LOOP

```
FOR i IN 1..5 LOOP
   DBMS_OUTPUT.PUT_LINE('Number: ' || i);
END LOOP;
```

CASE

```
CASE v_grade
   WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
   WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Good');
   ELSE DBMS_OUTPUT.PUT_LINE('Needs Improvement');
END CASE;
```