Introduction to PL/SQL

SUB PROGRAMS

- ❑ A subprogram is a named block that can be invoked repeatedly.
- ❑ A subprogram is either a **procedure** or a **function**.
- ❑ Typically, you use a procedure to perform an action and a function to compute and return a value.

Need of Sub Programs :

1. **Reusability** : Can be invoked several times to perform the same operations with different values.
2. **Modularity :** Subprograms let you break a program into manageable, well-defined modules.
3. **Better Performance** : Each subprogram is compiled and stored in executable form, which can be invoked repeatedly.

   Because stored subprograms run in the database server, a single invocation over the network can start a large job.

   This division of work reduces network traffic and improves response times. Stored subprograms are cached and shared among users, which lowers memory requirements and invocation overhead.

Introduction to PL/SQL

SUB PROGRAMS

Need of Sub Programs :

Earlier :

```
DECLARE
    num1 NUMBER:=1;
    num2 NUMBER:=2;
 BEGIN
    DBMS_OUTPUT.PUT_LINE('SUM IS = '|| (num1+num2) );
END;
```

We can pass different values for num1 and num2 to operate using sub programs.

This whole block can be reused using sub programs

Note : **Here (||) OR symbol is used for string concatenation.**

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

Example : **How to create stored procedures.**

```
CREATE  PROCEDURE  ProcedureDemo(
    num1 NUMBER,
    num2 NUMBER
) AS
BEGIN
    DBMS_OUTPUT.PUT_LINE('SUM IS = '||(num1+num2));
END ProcedureDemo;
```

Invocation/Calling :

```
BEGIN
            ProcedureDemo(2,4);
END;
```

OR

```
DECLARE
    x NUMBER:=2;
    y NUMBER:=3;
BEGIN
            ProcedureDemo(x,y);
END;
```

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

```
CREATE  PROCEDURE  ProcedureDemo(
    num1 NUMBER,
    num2 NUMBER
) AS
BEGIN
    DBMS_OUTPUT.PUT_LINE('SUM IS = '||(num1+num2));
END ProcedureDemo;
```

The variables/values which are specified as parameter at the time of function calling aka formal arguments.

Actual Parameters

Formal Parameter

The variables which are declared as parameter of the function definition are known as formal arguments.

```
DECLARE
    x NUMBER:=2;
    y NUMBER:=3;
BEGIN
        ProcedureDemo( x, y );
END;
```

Parameter Modes in Subprograms

There are 3 parameter mode for subprograms :

1.	**IN**	(Use to **get** value of the **actual parameter**)
2.	**OUT**	(Use to **set** value of the **actual parameter**)
3.	**IN OUT**	(Use to **get** and **set** the value of **actual parameter**)

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

❑ **IN mode**

- Use to **get** value of actual arguments.
- You **can't** change the value of parameter having IN mode.
- If you change the value of IN parameter, it will give compilation error.
- The **default mode** for the parameter is **IN** mode.

Example : Procedure with IN parameter mode

```
CREATE  PROCEDURE  ProcedureDemo(
    num1  IN  NUMBER,
    num2  NUMBER
) AS
BEGIN
    DBMS_OUTPUT.PUT_LINE('The Value of First
                              IN parameter : ' ||num1);
    DBMS_OUTPUT.PUT_LINE('The Value of
                         Second IN parameter : ' ||num2);
END ProcedureDemo;
```

IN parameter mode specified using IN clause

Default parameter mode is IN mode

Invocation :

```
DECLARE
    x NUMBER:=2;
    y NUMBER:=3;
BEGIN
        ProcedureDemo( x, y );
END;
```

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

❑ **IN mode**

Example : Procedure with IN parameter mode

```
CREATE  PROCEDURE  ProcedureDemo(
    num1  IN  NUMBER,
    num2  NUMBER
) AS
BEGIN
            num1:=10;
            num2:=20;
END ProcedureDemo;
```

Invocation :

```
DECLARE
     x NUMBER:=2;
     y NUMBER:=3;
BEGIN
            ProcedureDemo( x, y );
END;
```

Compilation Error : The values of IN parameter can't be updated.

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

❑ **OUT mode**

- Use to **set** value of actual arguments.
- It is recommended to change the value of parameter having OUT mode.
- If you change the value of OUT parameter, it will reflect the change to the actual parameter also.
- If use access OUT parameter value w/o updating, it will give it’s default value i.e. **NULL**.

Example : Procedure with OUT parameter mode

```
CREATE  PROCEDURE  ProcedureDemo(
    num1  OUT  NUMBER,
    num2  OUT NUMBER
) AS
BEGIN
    num1:=10;
    num2:=20;
END ProcedureDemo;
```

Invocation :

```
DECLARE
     x NUMBER;
     y NUMBER;
BEGIN
 ProcedureDemo( x, y );
 DBMS_OUTPUT.PUT_LINE(‘After Invoking Procedure’);
 DBMS_OUTPUT.PUT_LINE(x);
 DBMS_OUTPUT.PUT_LINE(y);
END;
```

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

- ❑ **OUT mode**

Example : Procedure with OUT parameter mode

```
CREATE  PROCEDURE  ProcedureDemo(
    num1  OUT  NUMBER,
    num2  OUT NUMBER
) AS
BEGIN
            DBMS_OUTPUT.PUT_LINE( num1 );
            DBMS_OUTPUT.PUT_LINE( num2 );
END ProcedureDemo;
```

Invocation :

```
DECLARE
    x NUMBER:=10;
    y NUMBER:=20;
BEGIN
  ProcedureDemo( x, y );
END;
```

NOTE : With OUT parameter we can't access the value of actual parameter.
If do, it will print the default value i.e. NULL (represented by blank space.)

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

❑ IN OUT mode

- Use to **get** and **set** value of actual arguments.
- It is recommended to change the value of parameter having IN OUT mode.
- If use access IN OUT parameter value w/o updating, it will give the original value of actual parameter.
- Using IN OUT parameter mode we can access the original values of actual parameter as well as update them.

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

- ❑ **IN OUT mode**

Example : Procedure with OUT parameter mode

```
CREATE  PROCEDURE  ProcedureDemo(
    num1  IN OUT  NUMBER,
    num2  IN OUT NUMBER
) AS
BEGIN
            DBMS_OUTPUT.PUT_LINE(‘Before Updating the values, Values are : ’);
            DBMS_OUTPUT.PUT_LINE( num1 );
            DBMS_OUTPUT.PUT_LINE( num2 );
               num1:=10;
               num2:=20;
            DBMS_OUTPUT.PUT_LINE(‘After Updating the values, Values are : ’);
            DBMS_OUTPUT.PUT_LINE( num1 );
            DBMS_OUTPUT.PUT_LINE( num2 );
END ProcedureDemo;
```

Continue…

Introduction to PL/SQL

SUB PROGRAMS (PROCEDURES)

Parameter Modes in Subprograms

- **IN OUT mode**

Invoking :

```
DECLARE
    x NUMBER=1;
    y NUMBER=2;
BEGIN
       DBMS_OUTPUT.PUT_LINE('Before Invoking Procedure');
       DBMS_OUTPUT.PUT_LINE( x );
       DBMS_OUTPUT.PUT_LINE( y );
            ProcedureDemo( x, y );
       DBMS_OUTPUT.PUT_LINE('After Invoking Procedure');
       DBMS_OUTPUT.PUT_LINE( x );
       DBMS_OUTPUT.PUT_LINE( y );
END;
```