Parameters in Procedure and Functions

How to pass parameters to Procedures and Functions in PL/SQL?

In PL/SQL, we can pass parameters to procedures and functions in three ways.

1) IN type parameter: These types of parameters are used to send values to stored procedures.
2) OUT type parameter: These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.
3) IN OUT parameter: These types of parameters are used to send values and get values from stored procedures.

NOTE: If a parameter is not explicitly defined a parameter type, then by default it is an IN type parameter.

1) IN parameter:

This is similar to passing parameters in programming languages. We can pass values to the stored procedure through these parameters or variables. This type of parameter is a read only parameter. We can assign the value of IN type parameter to a variable or use it in a query, but we cannot change its value inside the procedure.

General syntax to pass a IN parameter is

CREATE [OR REPLACE] PROCEDURE procedure_name (
param_name1 IN datatype, param_name12 IN datatype ...)

- param_name1, ☐param_name2... are unique parameter names.
- datatype - defines the datatype of the variable.
- IN - is optional, by default it is a IN type parameter.

2) <u>OUT Parameter</u>:

The OUT parameters are used to send the OUTPUT from a procedure or a function. This is a write-only parameter i.e, we cannot pass values to OUT paramters while executing the stored procedure, but we can assign values to OUT parameter inside the stored procedure and the calling program can recieve this output value.

The General syntax to create an OUT parameter is

CREATE [OR REPLACE] PROCEDURE proc2 (param_name OUT datatype)

The parameter should be explicity declared as OUT parameter.

3) <u>IN OUT Parameter</u>:

The IN OUT parameter allows us to pass values into a procedure and get output values from the procedure. This parameter is used if the value of the IN parameter can be changed in the calling program.

By using IN OUT parameter we can pass values into a parameter and return a value to the calling program using the same parameter. But this is possible only if the value passed to the procedure and output value have a same datatype. This parameter is used if the value of the parameter will be changed in the procedure.

The General syntax to create an IN OUT parameter is

CREATE [OR REPLACE] PROCEDURE proc3 (param_name IN OUT datatype)

The below examples show how to create stored procedures using the above three types of parameters.

Example1:

Using IN and OUT parameter:

Let's create a procedure which gets the name of the employee when the employee id is passed.

```
CREATE OR REPLACE PROCEDURE emp_name (id IN NUMBER, emp_name OUT NUMBER)
IS
BEGIN
   SELECT first_name INTO emp_name
   FROM emp_tbl WHERE empID = id;
END;
/
```

We can call the procedure 'emp_name' in this way from a PL/SQL Block.

```
DECLARE
 empName varchar(20);
 CURSOR id_cur SELECT id FROM emp_ids;
BEGIN
FOR emp_rec in id_cur
LOOP
  emp_name(emp_rec.id, empName);
  dbms_output.putline('The employee ' || empName || ' has id ' || emp-rec.id);
END LOOP;
END;
/
```

In the above PL/SQL Block

In line no 3; we are creating a cursor ‘id_cur’ which contains the employee id.

In line no 7; we are calling the procedure ‘emp_name’, we are passing the ‘id’ as IN parameter and ‘empName’ as OUT parameter.

In line no 8; we are displaying the id and the employee name which we got from the procedure ‘emp_name’.

Example 2:

Using IN OUT parameter in procedures:

```
1> CREATE OR REPLACE PROCEDURE emp_salary_increase
2> (emp_id IN emptbl.empID%type, salary_inc IN OUT emptbl.salary%type)
3> IS
4>    tmp_sal number;
5> BEGIN
6>    SELECT salary
7>    INTO tmp_sal
8>    FROM emp_tbl
9>    WHERE empID = emp_id;
10>   IF tmp_sal between 10000 and 20000 THEN
11>      salary_inout := tmp_sal * 1.2;
12>   ELSIF tmp_sal between 20000 and 30000 THEN
13>      salary_inout := tmp_sal * 1.3;
14>   ELSIF tmp_sal > 30000 THEN
15>      salary_inout := tmp_sal * 1.4;
16>   END IF;
17> END;
18> /
```

The below PL/SQL block shows how to execute the above 'emp_salary_increase' procedure.

```
1> DECLARE
2>    CURSOR updated_sal is
3>    SELECT empID,salary
4>    FROM emp_tbl;
5>    pre_sal number;
6> BEGIN
7>   FOR emp_rec IN updated_sal LOOP
8>       pre_sal := emp_rec.salary;
9>       emp_salary_increase(emp_rec.empID, emp_rec.salary);
10>       dbms_output.put_line('The salary of ' || emp_rec.empID ||
11>                ' increased from '|| pre_sal || ' to '||emp_rec.salary);
12>   END LOOP;
13> END;
14> /
```