Lab 12: Stored Procedures

Objective(s):

To learn the Stored Procedures

Subprograms

Subprograms are named PL/SQL blocks that can accept parameters and be invoked from a calling environment.

Types of Subprograms

PL/SQL has two types of subprograms, ***procedures*** and ***functions***

Subprogram Specification

- The header is relevant for named blocks only and determines the way that the program unit is called or invoked.
- The header determines:
- The PL/SQL subprogram type, that is, ***either a procedure or a function***
- The ***name*** of the subprogram
- The ***parameter list***, if one exists
- The ***RETURN clause***, which applies ***only to functions***
- The **IS** or **AS** keyword is mandatory.

Subprogram Body

- The declaration section of the block between **IS|AS** and **BEGIN.**
- The keyword **DECLARE** that is used to indicate the start of the declaration section in anonymous blocks is not used here.
- The executable section between the **BEGIN** and **END** keywords is mandatory, enclosing the body of actions to be performed.
- There must be at least one statement existing in the executable section. There should be at least one **NULL**; statement, which is considered an executable statement.
- The exception section between **EXCEPTION** and **END** is optional.

Example:

```
CREATE OR REPLACE PROCEDURE display
IS
        cursor employee is
                select ename,deptno,sal
                from emp
                where deptno = 10;
                emp_rec employee%rowtype;  --cursor base record
BEGIN
        open employee;
LOOP
        fetch employee into emp_rec;
        exit when employee%notfound;
        dbms_output.put_line(emp_rec.ename || ' '|| emp_rec.deptno || ' ' || emp_rec.sal);
END LOOP;
END;
```

You can execute the SUBPROGRAM through command EXECUTE subprogram_name in sql*plus.

Example:

EXECUTE display;

The user can view the validity of the procedure by using the select statement as:

SELECT object_name, object_type , status

FROM user_objects

WHERE object_type = 'PROCEDURE';

User can see the errors of the SUBPROGRAM through PL/SQL command ***SHOW***

ERRORS.

You can use the DROP command to drop the procedure.

Example:

DROP display.

Procedures with Parameters

EXAMPLE (CREATING PROCEDURE with PARAMETER):

```
CREATE OR REPLACE PROCEDURE display
(p_empno in emp.empno%type,
p_ename out emp.ename%type,
p_deptno out emp.deptno%type,
p_sal out emp.sal%type)
IS
BEGIN
        select ename,deptno,sal
        into
        p_ename,p_deptno,p_sal
        from emp
        where empno = p_empno;
END;
```

(Calling of Procedure with Parameters)

```
DECLARE
        emp_rec emp%rowtype;
BEGIN
        display(7788,emp_rec.ename,emp_rec.deptno,emp_rec.sal);
        dbms_output.put_line(emp_rec.ename || ' ' || emp_rec.deptno || ' ' || emp_rec.sal);
END;
```

Exercise

1. Create a stored procedure **DISPLAY** without parameters. The procedure must display empno, ename and salary of all the employees of DEPTNO = 10.

   DISPLAY procedure must be called from the PL/SQL anonymous block.

2. Create a stored procedure **DISPLAY2** with parameters. It must take DEPTNO as an input and must return the DNAME and TOTAL SALARY of the input department number. DISPLAY2 procedure must be called from the PL/SQL anonymous block.

3. Create a stored procedure **DISPLAY3** with parameters. It must take DEPTNO as an input and must return the DNAME, SMALLEST and HIGHEST SALARIES of the input department number. DISPLAY3 must also display empno,ename,total salary (sal+comm) of all the employees of the input department number.

   DISPLAY3 procedure must be called from the PL/SQL anonymous block.