Database Systems Laboratory 12

STORED PROCEDURE, STORED FUNCTION & PACKAGE

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

STORED PROCEDURE, STORED FUNCTION & PACKAGE

Sub-Program

Sub-Program

- A subprogram is a named PL/SQL block that can accept parameters & can be used as per requirements
- A subprogram comprises of:
  - Declarative section
  - Executable section and
  - Exceptional handling section
- Subprograms are frequently used because of its reusability & ease of writing code in PL/SQL
- Two types of subprograms:
  - Stored Procedure
  - Stored Function

Stored Procedure

Stored Procedure

A procedure is a logically grouped set of SQL and PL/SQL statements that perform a specific task

Oracle stores both the source and compiled code of procedures in database. Therefore, procedures are called stored procedures

The syntax for procedural declaration is:

```
CREATE OR REPLACE PROCEDURE procname [(arg1, arg2)] IS
    constatnt/variable declaration
BEGIN
    executable statements
EXCEPTION
    exception handler statements
END procname;
```

Stored Procedure...

Write a procedure that accepts 2 numbers and print the sum

```
CREATE OR REPLACE PROCEDURE psum (A NUMBER,
  B NUMBER) IS
      C NUMBER;
BEGIN
      C: = A + B;
      DBMS_OUTPUT.PUT_LINE(A||'+'||B||'='||C);
END psum;
```

Executing a procedure

EXECUTE psum(10, 20); or

EXEC psum(10, 20);

Dropping a procedure

DROP PROCEDURE psum;

Parameter Passing

a. IN Mode

It is the *default* mode of parameter passing. It passes a value into the program, it acts like a constant & can't be assigned a value

```
CREATE OR REPLACE PROCEDURE psum (A IN NUMBER,
  B IN NUMBER) IS
     C NUMBER;
BEGIN
      C: = A + B;
      DBMS_OUTPUT.PUT_LINE(A||'+'||B||'='||C);
END psum;

EXECUTE psum(10, 20);
```

Parameter Passing...

b. OUT Mode

It is used to return values to the caller of a subprogram. The OUT parameter must be assigned some value in the called program

Procedures with OUT parameter can't be executed with EXECUTE statement. It must be called from other PL/SQL program

```
CREATE OR REPLACE PROCEDURE psum (A IN NUMBER,
  B IN NUMBER, C OUT NUMBER) IS
BEGIN
      C: = A + B;
END psum;
```

Parameter Passing...

b. OUT Mode...

```
DECLARE
     A NUMBER;
     B NUMBER;
     C NUMBER;
BEGIN
     A: =&A;
     B: =&B;
     psum(A, B, C);
     DBMS_OUTPUT.PUT_LINE(A||'+'||B||'='||C);
END;
```

Parameter Passing...

c. IN OUT Mode

It is used to pass initial values to the subprograms when invoked & is also returns updated values to the caller

Procedures with IN OUT parameter can't be executed using EXECUTE statements

```
CREATE OR REPLACE PROCEDURE psum (A IN OUT NUMBER,
  B NUMBER) IS
BEGIN
      A: = A + B;
END psum;
```

Parameter Passing...

c. IN OUT Mode...

```
DECLARE
    A NUMBER;
    B NUMBER;
BEGIN
    A: =&A;
    B: =&B;
    psum(A, B);
    DBMS_OUTPUT.PUT_LINE('Sum ='||A);
END;
```

Parameter Passing...

Write a procedure which takes the empid as input and displays the details of employee as the output

```
CREATE OR REPLACE PROCEDURE searchemp(eid IN
    NUMBER, nm OUT VARCHAR, sl OUT NUMBER) IS
BEGIN
    SELECT ename, sal INTO nm, sl FROM emp WHERE empid=eid;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(eid||'does't exist');
END searchemp;
DECLARE
    vname emp.ename%TYPE;
    vsal emp.sal%TYPE;
    veid emp.empid%TYPE;
BEGIN
    veid: =&veid;
    searchemp(veid, vname, vsal);
    DBMS_OUTPUT.PUT_LINE(veid||' '||vname||' '||vsal);
END;
```

Stored Function

Stored Function

These are the PL/SQL blocks that take parameters, perform some action and return a single value to the calling program

Like stored procedure, Oracle stores both the source code and compiled code in its database

The syntax for procedural declaration is:

```
CREATE OR REPLACE FUNCTION funcname [(arg1, arg2)]
RETURN datatype IS
    constatnt/variable declaration
BEGIN
    executable statements
    RETURN returnvalue
EXCEPTION
    exception handler statements
    RETURN returnvalue
END funcname;
```

Stored Function...

```
CREATE OR REPLACE FUNCTION fsum(A NUMBER,
    B NUMBER) RETURN NUMBER IS
         C NUMBER;
BEGIN
    C: =A+B;
    RETURN C;
END fsum;
```

Calling a function

```
SELECT fsum(10,15) FROM DUAL;

SELECT ename, fsum(sal,1000) newsalary FROM emp;
```

Showing Errors

```
SHOW ERRORS
```

Dropping a procedure

```
DROP FUNCTION fsum;
```

Stored Function...

Create a function which returns the deptname according to the inputted deptno

```
CREATE OR REPLACE FUNCTION get_deptname (did NUMBER)
    RETURN VARCHAR IS
         vdept VARCHAR(12);
BEGIN
    SELECT deptname INTO vdept FROM dept WHERE deptid=did;
    RETURN vdept;
END get_deptname;
```

Stored Function...

Write the PL/SQL block which takes the empid as input and displays the deptid and dept name

```
DECLARE
    vdid emp.deptno%TYPE;
    vdeptname VARCHAR(12);
    veid emp.empid%TYPE;
BEGIN
    veid: =&veid;
    SELECT deptno INTO vdid FROM emp WHERE empid=veid;
    vdeptname: =get_deptname(vdid);
    DBMS_OUTPUT.PUT_LINE(veid||' '||vdeptname);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE (veid||'not found);
END;
```

Package

Package

A package is a collection of stored procedures, functions, cursors and exceptions. A package is compiled and stored in database as an object

Packages enable to perform overloading of functions and procedures

Components of Packages

- **Package Specification**: contains the list of various functions/ procedure names which will be a part of the package
- **Package Body**: contains the actual code implementing the logics of functions and procedures declared in the specification

Package...

Package Specification

It contains information about the package elements such as definitions of functions & procedures, declarations of variables

```
CREATE OR REPLACE PACKAGE packname AS
      Declarations
BEGIN
      Executable statements
END packname;
```

Package Body

It contains actual programming code for the modules described in the specification section

```
CREATE OR REPLACE PACKAGE BODY packname AS
      Declaration
BEGIN
      Executable statements
END packname;
```

Package...

Create a package calculator which contains different functions and procedures for the different operations

```
CREATE OR REPLACE PACKAGE calculator AS
  FUNCTION fsum(A NUMBER, B NUMBER) RETURN NUMBER;
  FUNCTION fminus(A NUMBER, B NUMBER) RETURN NUMBER;
  FUNCTION fmult(A NUMBER, B NUMBER) RETURN NUMBER;
  FUNCTION fdivide(A NUMBER, B NUMBER) RETURN NUMBER;
  PROCEDURE psum(A NUMBER, B NUMBER);
  PROCEDURE pminus(A NUMBER, B NUMBER);
END calculator;
...
CREATE OR REPLACE PACKAGE BODY calculator AS
FUNCTION fsum(A NUMBER, B NUMBER) RETURN NUMBER IS
    C NUMBER;
BEGIN
    C: =A+B;
    RETURN C;
END fsum;
```

Package...

```
FUNCTION fminus(A NUMBER, B NUMBER) RETURN NUMBER IS
    C NUMBER;
BEGIN
    C: =A-B;
    RETURN C;
END fminus;
...
FUNCTION fmult(A NUMBER, B NUMBER) RETURN NUMBER IS
    C NUMBER;
BEGIN
C: =A*B;
    RETURN C;
END fmult;
```

Package...

```
FUNCTION fdivide(A NUMBER, B NUMBER) RETURN NUMBER IS
    C NUMBER;
BEGIN
    IF B<>0 THEN
     C: =A/B;
    ELSE
        C: =-1;
    END IF;
    RETURN C;
END fdivide;
```

Package...

```
PROCEDURE psum(A NUMBER, B NUMBER) IS
    C NUMBER;
BEGIN
    C: =A+B;
    DBMS_OUTPUT.PUT_LINE(A||'+'||B||'='||C);
END Psum;
...
PROCEDURE pminus(A NUMBER, B NUMBER) IS
    C NUMBER;
BEGIN
    C: =A-B;
    DBMS_OUTPUT.PUT_LINE(A||'-'||B||'=||C);
END Pminus;
END calculator;
```

Package...

Executing Package Functions & Procedures

```
EXECUTE calculator.psum(10, 20);

SELECT calculator.fsum(10, 20) FROM DUAL;

SELECT empid,calculator.fmultiply(sal, 2) FROM emp;
```