Oracle (PL/SQL)

Procedures and Functions

Capgemini

Lesson Objectives

- To understand the following topics:
  - Subprograms in PL/SQL
  - Anonymous blocks versus Stored Subprograms
  - Procedure
    - Subprogram Parameter modes
  - Functions

Introduction

- A subprogram is a named block of PL/SQL.
- There are two types of subprograms in PL/SQL, namely: Procedures and Functions.
- Each subprogram has:
  - A declarative part
  - An executable part or body, and
  - An exception handling part (which is optional)
- A function is used to perform an action and return a single value.

Comparison

- Anonymous Blocks & Stored Subprograms Comparison

Anonymous Blocks	Stored Subprograms/Named Blocks
1. Anonymous Blocks do not have names.	1. Stored subprograms are named PL/SQL blocks.
2. They are interactively executed. The block needs to be compiled every time it is run.	2. They are compiled at the time of creation and stored in the database itself. Source code is also stored in the database.
3. Only the user who created the block can use the block.	3. Necessary privileges are required to execute the block.

Procedures

- A procedure is used to perform an action.
- It is illegal to constrain datatypes.
- Syntax:

```
CREATE PROCEDURE  Proc_Name
   (Parameter {IN | OUT | IN OUT} datatype := value,...)
AS
   Variable_Declaration ;
   Cursor_Declaration ;
   Exception_Declaration ;
BEGIN
    PL/SQL_Statements ;
EXCEPTION
    Exception_Definition ;
END Proc_Name ;
```

Subprogram Parameter Modes

IN	OUT	IN OUT
The default	Must be specified	Must be specified
Used to pass values to the procedure.	Used to return values to the caller.	Used to pass initial values to the procedure and return updated values to the caller.
Formal parameter acts like a constant.	Formal parameter acts like an uninitialized variable.	Formal parameter acts like an uninitialized variable.
Formal parameter cannot be assigned a value.	Formal parameter cannot be used in an expression, but should be assigned a value.	Formal parameter should be assigned a value.
Actual parameter can be a constant, literal, initialized variable, or expression.	Actual parameter must be a variable.	Actual parameter must be a variable.
Actual parameter is passed by reference (a pointer to the value is passed in).	Actual parameter is passed by value (a copy of the value is passed out) unless NOCOPY is specified.	Actual parameter is passed by value (a copy of the value is passed in and out) unless NOCOPY is specified.

Examples

- Example 1:

```
CREATE OR REPLACE PROCEDURE raise_salary
   ( s_no IN number, raise_sal IN number) IS
   v_cur_salary   number ;
    missing_salary exception;
BEGIN
      SELECT staff_sal INTO v_cur_salary FROM staff_master
      WHERE staff_code=s_no;
   IF  v_cur_salary IS NULL THEN
      RAISE  missing_salary;
   END IF ;
         UPDATE staff_master SET staff_sal = v_cur_salary + raise_sal
         WHERE staff_code =  s_no ;
EXCEPTION
            WHEN  missing_salary THEN
            INSERT into emp_audit VALUES( sno, 'salary is missing');
 END raise_salary;
```

Examples

- Example 2:

```
CREATE OR REPLACE PROCEDURE
    get_details(s_code IN number,
    s_name OUT varchar2,s_sal OUT number ) IS
BEGIN
  SELECT staff_name, staff_sal INTO s_name, s_sal
  FROM staff_master WHERE staff_code=s_code;
EXCEPTION
            WHEN no_data_found  THEN
            INSERT into auditstaff
            VALUES( 'No employee with id ' || s_code);
            s_name := null;
            s_sal := null;
    END get_details ;
```

Executing a Procedure

- Executing the Procedure from SQL*PLUS environment,
  - Create a bind variables salary and name SQLPLUS by using VARIABLE command as follows:

```
variable salary number
variable name varchar2(20)
```

  - Execute the procedure with EXECUTE command

```
EXECUTE get_details(100003,:Salary, :Name)
```

  - After execution, use SQL *PLUS PRINT command to view results.

```
print salary
print name
```

Functions

- A function is similar to a procedure.
- A function is used to compute a value.
  - A function accepts one or more parameters, and returns a single value by using a return value.
  - A function can return multiple values by using OUT parameters.
  - A function is used as part of an expression, and can be called as Lvalue = Function_Name(Param1, Param2, …….)
  - Functions returning a single value for a row can be used with SQL statements.

Functions

- Syntax :

```
CREATE  FUNCTION Func_Name(Param datatype :=
  value,..) RETURN datatype1 AS
      Variable_Declaration ;
      Cursor_Declaration ;
      Exception_Declaration ;
  BEGIN
       PL/SQL_Statements ;
       RETURN Variable_Or_Value_Of_Type_Datatype1 ;
  EXCEPTION
       Exception_Definition ;
  END Func_Name ;
```

Examples

- Example 1:

```
CREATE FUNCTION crt_dept(dno number,
  dname varchar2)  RETURN number AS
BEGIN
        INSERT into department_master
        VALUES (dno,dname) ;
                return 1 ;
EXCEPTION
        WHEN others THEN
                return 0 ;
END crt_dept ;
```

Executing a Function

- Executing functions from SQL*PLUS:
  - Create a bind variable Avg salary in SQLPLUS by using VARIABLE command as follows:

```
variable flag number
```

  - Execute the Function with EXECUTE command:

```
EXECUTE :flag:=crt_dept(60,‘Production');
```

  - After execution, use SQL*PLUS PRINT command to view results.

```
PRINT  flag;
```

Exceptions handling in Procedures and Functions

- If procedure has no exception handler for any error, the control immediately passes out of the procedure to the calling environment.
- Values of OUT and IN OUT formal parameters are not returned to actual parameters.
- Actual parameters will retain their old values.

Summary

- In this lesson, you have learnt:
  - Subprograms in PL/SQL are named PL/SQL blocks.
    - There are two types of subprograms, namely: Procedures and Functions.
  - Procedure is used to perform an action.
    - Procedures have three subprogram parameter modes, namely: IN, OUT, and INOUT.

Summary

- Functions are used to compute a value.
  - A function accepts one or more parameters, and returns a single value by using a return value.
  - A function can return multiple values by using OUT parameters.

Review Question

- Question 1: Anonymous Blocks do not have names.
  - True / False
- Question 2: A function can return multiple values by using OUT parameters
  - True / False

Review Question

- Question 4: An ____ parameter returns a value to the caller of a subprogram.
- Question 5: A procedure contains two parts: ____ and ____.
- Question 6: In ____ notation, the order of the parameters is not significant.