Tech Mahindra

Packages

What is A Package?

- A **package** is a schema object that groups logically related PL/SQL types, items, and subprograms.
- Packages usually have two parts, a specification and a body, although sometimes the body is unnecessary.
- The **specification** (**spec** for short) is the interface to your applications; it declares the types, variables, constants, exceptions, cursors, and subprograms available for use.
- The **body** fully defines cursors and subprograms, and so implements the spec.
- When we create different procedures and functions, they are stored as independent objects. To group these objects together, we are creating packages.

Package

- Each package contains the following parts:
  - Package specification
  - Package body

Package Specification

- Contains all declarations for variable, cursor, procedures and functions that are to be made public. All public objects of package are visible outside the package.
  - **Syntax**: CREATE OR REPLACE PACKAGE package_name
    IS
    /* declare public objects of package */
    END;

Package Body

Defines all the objects of the package. Objects declared in the specification are called as public objects and the objects directly defined in the body without declaring in the specification, are called as PRIVATE members.

Syntax: CREATE or REPLACE PACKAGE BODY package_name

```
        IS
                [ declarations of variables and types ]
                [ specification and SELECT statement of cursors ]
                [ specification and body of modules ]
        [ BEGIN
                executable statements ]
        [ EXCEPTION
                exception handlers ]
        END [ package_name ];
```

In the body you can declare other variables, but you do not repeat the declarations in the specification.

Package Body

The body contains the full implementation of cursors and modules. In the case of a cursor, the package body contains both specification and SQL statement for the cursor. In the case of a module the package body contains both the specification and body of the module.

The body of a package can contain

- Procedures declared in the package specification
- Functions declared in the package specification
- Definitions of cursors declared in the package specification
- Local procedures and functions, not declared in the package specification.
- Local variables

Advantages of PL/SQL Packages

- Packages offer several advantages: modularity, easier application design, information hiding, added functionality, and better performance.

- **Modularity**

  - Packages let you encapsulate logically related types, items, and subprograms in a named PL/SQL module. Each package is easy to understand, and the interfaces between packages are simple, clear, and well defined. This aids application development.

- **Easier Application Design**
  - When designing an application, all you need initially is the interface information in the package specs. You can code and compile a spec without its body. Then, stored subprograms that reference the package can be compiled as well.

- **Information Hiding**
  - We can hide the functionality by defining them as private members.

- **Better Performance**
  - When you call a packaged subprogram for the first time, the whole package is loaded into memory. So, later calls to related subprograms in the package require no disk I/O. Also, packages stop cascading dependencies and thereby avoid unnecessary recompiling.

Program

```
CREATE OR REPLACE PACKAGE samplepack
IS
PROCEDURE proc1(p_n  NUMBER, p_n1 OUT NUMBER);
FUNCTION fun1(p_n NUMBER) RETURN NUMBER;
END;
/
CREATE OR REPLACE PACKAGE BODY samplepack
IS
PROCEDURE proc1(p_n NUMBER, p_n1 OUT NUMBER)
IS
BEGIN
p_n1 := p_n * 5;
END proc1;
FUNCTION fun1(p_n NUMBER) RETURN NUMBER
IS
v_n1 NUMBER;
BEGIN
v_n1  := p_n * 2;
RETURN v_n1;
END fun1;
END;
/
```

```
VARIABLE N NUMBER
EXECUTE SAMPLEPAK.PROC1(5,:N)
 PRINT N
EXECUTE :N := SAMPLEPAK.FUN1(4)
PRINT N
```

Program to define private member

```
CREATE OR REPLACE PACKAGE samplepack
IS
   PROCEDURE proc1(p_n  NUMBER, p_n1 OUT NUMBER);
   FUNCTION  fun1(p_n NUMBER) RETURN NUMBER;
 END;
/
CREATE OR REPLACE PACKAGE BODY samplepack
IS
PROCEDURE test     -- private member definition
IS
BEGIN
     DBMS_OUTPUT.PUT_LINE('I AM A PRIVATE MEMBER');
END test;
 PROCEDURE proc1(p_n NUMBER, p_n1 OUT NUMBER)
 IS
 BEGIN
      TEST;     -- private member called
       p_n1 := p_n * 5;
 END proc1;
FUNCTION fun1(p_n NUMBER) RETURN NUMBER
IS
    v_n1 NUMBER;
BEGIN
    v_n1  := p_n * 2;
    RETURN v_n1;
END fun1;
END samplepack;
/
```

```
VARIABLE N NUMBER
EXECUTE samplepack.proc1(5,:N)
PRINT N
EXECUTE :N := samplepack.fun1(4)
PRINT N
```

Note:

A private member cannot be accessed by referring package object. They are called only through public members of the package object.

Program to test Polymorphism

```
CREATE OR REPLACE PACKAGE polypack
IS
PROCEDURE proc1( v_n  NUMBER, v_n1 OUT NUMBER);
PROCEDURE proc1(v_x VARCHAR2,v_y VARCHAR2,v_z OUT VARCHAR2);
END;
/
CREATE OR REPLACE PACKAGE BODY polypack
IS
PROCEDURE proc1(v_n NUMBER, v_n1 OUT NUMBER)
IS
BEGIN
v_n1 := v_n * 5;
END proc1;
PROCEDURE proc1(v_x VARCHAR2,v_y VARCHAR2,v_z OUT VARCHAR2)
IS
BEGIN
Z :=  CONCAT(v_x,v_y);
END proc1;
END polypack;
/
```

Data Dictionary Views

- Package objects are stored in the Data Dictionary view :
  - USER_OBJECTS : Shows current user objects
  - ALL_OBJECTS :Shows you all objects of current user and those object which you have rights to access.
  - DBA_OBJECTS : Shows all objects of all the users
  - USER_SOURCE : Describes the text source of the stored objects owned by user
  - ALL_SOURCE
  - DBA_SOURCE

Exercise

- Define a package to maintain passbook
  - Steps to maintain passbook
    - 1. Generate transaction number
      - System generates number and no input is given by user
      - Returns only one value. Define Function
    - 2. Getting the opening balance
      - Based on transaction number generated by Step 1 function, get the opening balance. So, there is an input and one output.
      - Define a function
    - 3. Input trntype and amount, handle suitable exceptions
      Calculate closing balance and store the data
      - There are 2 inputs, not getting any output.
      - Define a procedure.
- In the above 3 steps, when we run 3rd procedure, by calling function1 and function2 , it completes our transaction.
- So, define a package, with one public member(procedure) and 2 private functions.