Oracle PL/SQL – Packages

codingcircuit23@gmail.com

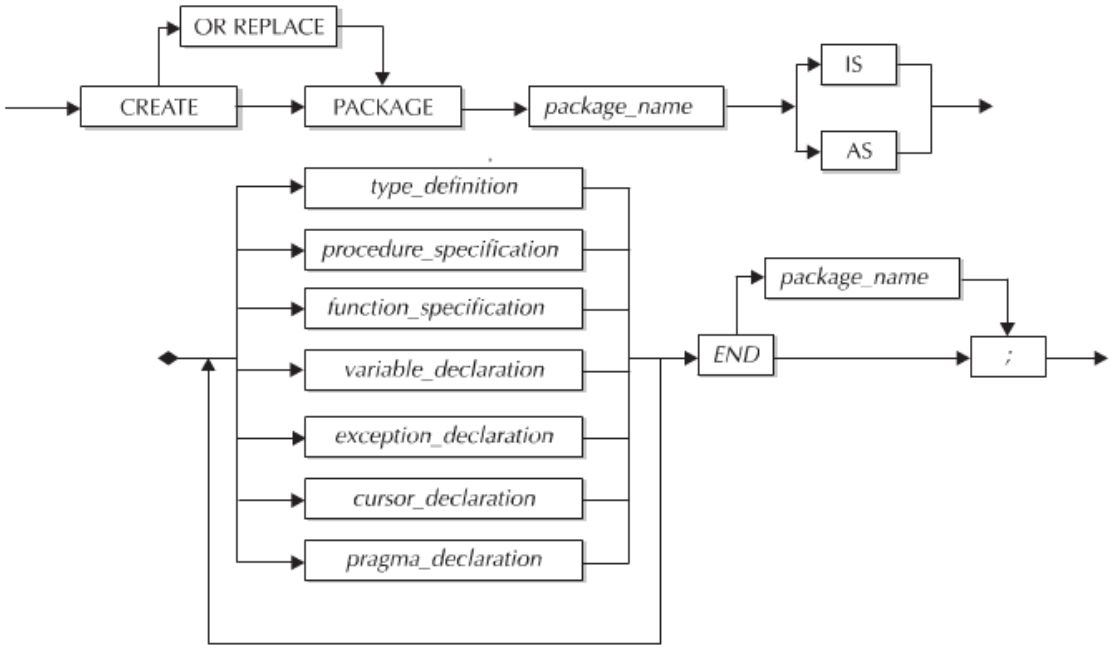
Introduction to Packages:

- A package is a PL/SQL construct or a schema object that allows related objects to be grouped and stored together.
- A package usually has two parts
    - Package Specification
    - Package Body
- The specification is the interface of the application where the different objects like cursors, exceptions, procedures, etc are declared. The body defines and implements the specification.
- One advantage of putting similar objects into a package is that they can be referenced from other PL/SQL blocks.

Package Specification:

- It declares the types, variables, constants, exceptions, cursors, and sub programs that can be referenced from outside the package.
- It contains the information about the content of the package, but excludes the code for the sub programs.
- Also known as **package header.**

Syntax for creating package specification:



Example for creating package specification:

```
CREATE OR REPLACE PACKAGE emp_actions AS
    PROCEDURE hire_employee
          (v_ename employee.emp_name%TYPE,
          v_designation employee.designation%TYPE,v_joining_date employee.join_date%TYPE);
    PROCEDURE employee_resign(v_emp_id employee.emp_id%TYPE);
    FUNCTION func_calc_comm(v_emp_id employee.emp_id%TYPE ) RETURN NUMBER;
END emp_actions;
/
```

In the above example, *emp_actions* is the package name and it includes the declarations for two procedures *hire_employee* and *employee_resign* and one function *func_calc_comm*. The package body specifies the functionalities of the mentioned procedures and function.

Package Body :

- A separate data dictionary object from package header.
- It cannot be successfully complied unless the package header is compiled successfully.
- A package body is optional if the package does not have any procedures or functions.

Example for package body:

```
CREATE OR REPLACE PACKAGE BODY emp_actions AS
    PROCEDURE hire_employee
                    (v_ename employee.emp_name%TYPE,v_designation employee.designation%TYPE,
                    v_joining_date employee.join_date%TYPE) AS
        BEGIN
                    INSERT INTO employee(emp_name,designation,join_date)
                    VALUES(v_ename,v_designation,v_joining_date);
        END hire_employee;
   PROCEDURE employee_resign(v_emp_id employee.emp_id%TYPE) AS
        BEGIN
                    DELETE FROM employee WHERE  emp_id = v_emp_id;
        END employee_resign;
   FUNCTION func_calc_comm(v_emp_id employee.emp_id%TYPE ) RETURN NUMBER AS
        v_salary NUMBER;
        v_comm NUMBER;
        BEGIN
        SELECT salary INTO v_salary FROM employee WHERE emp_id = v_emp_id;
        v_comm := 0.10 * v_salary;
        RETURN v_comm;
        END func_calc_comm;
END emp_actions;
/
```

Note : Package specification and package body are independent objects. Package body will not be existing until specification exists. But Package specification can exists without package body also.

Package Element Scope:

The procedure or function in the package can be declared as either of the following:

- Public
- Private

Public :

- The package specification holds public declarations.
- The items can be used by other packages.

Private:

- The package body holds implementation details but Specification will not have private declarations.
- The items are restricted to use within the package.

Accessing the package elements:

The package elements (variables, procedures or functions) are accessed with the following syntax:

```
package_name.element_name;
```

Example:

```
DECLARE
    s_comm NUMBER;
BEGIN
   s_comm := emp_actions.func_calc_comm(101);
   dbms_output.put_line('Commission:' || s_comm);
END;
/
```

Advantages of Packages:

a) **Modularity**

Packages allows to encapsulate logically related types, items, and sub programs 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 in application development.

b) **Easier Application Design**

When designing an application, only the interface information in the package specification is required initially. Specification can be coded and compiled without its body. Then, stored sub programs that reference the package can be compiled as well.

c) **Information Hiding**

Packages allow to classify types, items, and sub programs as public (visible and accessible) or private (hidden and inaccessible). One package can have public and private sub programs. The package hides the implementation of the private sub program so that only the package (not the entire application) is affected if the implementation changes. This simplifies maintenance and enhancement. Also, by hiding implementation details from users, integrity of the package is protected.

d) **Better Performance**

When a packaged sub program is called for the first time, the whole package is loaded into memory. So, later calls to related sub programs in the package require no disk I/O. Also, packages stop cascading dependencies and thereby avoid unnecessary recompiling. For example, if the implementation of a packaged

function is changed, Oracle need not recompile the calling sub programs because they do not depend on the package body.