**PL/SQL Packages**

PL/SQL Packages is schema object and collection of related data type (variables, constants), cursors, procedures, functions are defining within a single context. Package are device into two part,

1. Package Specification
2. Package Body

Package specification block you can define variables, constants, exceptions and package body you can create procedure, function, subprogram.

PL/SQL Package Advantages

1. You can create package to store all related functions and procedures are grouped together into single unit called packages.
2. Package are reliable to granting a privileges
3. All function and procedure within a package can share variable among them.
4. Package are support overloading to overload functions and procedures.
5. Package are improve the performance to loading the multiple object into memory at once, therefore, subsequent calls to related program doesn't required to calling physically I/O.
6. Package are reduce the traffic because all block execute all at once.

PL/SQL Package Syntax

PL/SQL Specification : This contain the list of variables, constants, functions, procedure names which are the part of the package. PL/SQL specification are public declaration and visible to a program.

Defining Package Specification Syntax

CREATE [OR REPLACE] PACKAGE package\_name

IS | AS

[variable\_declaration ...]

[constant\_declaration ...]

[exception\_declaration ...]

[cursor\_specification ...]

[PROCEDURE [Schema..] procedure\_name

[ (parameter {IN,OUT,IN OUT} datatype [,parameter]) ]

]

[FUNCTION [Schema..] function\_name

[ (parameter {IN,OUT,IN OUT} datatype [,parameter]) ]

RETURN return\_datatype

]

END [package\_name];

PL/SQL Body : This contains the actual PL/SQL statement code implementing the logics of functions, procedures which are you already before declare in "Package specification".

Creating Package Body Syntax

CREATE [OR REPLACE] PACKAGE BODY package\_name

IS | AS

[private\_variable\_declaration ...]

[private\_constant\_declaration ...]

BEGIN

[initialization\_statement]

[PROCEDURE [Schema..] procedure\_name

[ (parameter [,parameter]) ]

IS | AS

variable declarations;

constant declarations;

BEGIN

statement(s);

EXCEPTION

WHEN ...

END

]

[FUNCTION [Schema..] function\_name

[ (parameter [,parameter]) ]

RETURN return\_datatype

IS | AS

variable declarations;

constant declarations;

BEGIN

statement(s);

EXCEPTION

WHEN ...

END

]

[EXCEPTION

WHEN built-in\_exception\_name\_1 THEN

User defined statement (action) will be taken;

]

END;

/

PL/SQL Package Example

PL/SQL Package example step by step explain to you, you are create your own package using this reference example. We have emp1 table having employee information,

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | EMP\_NAME | EMP\_DEPT | EMP\_SALARY |
| 1 | Forbs ross | Web Developer | 45k |
| 2 | marks jems | Program Developer | 38k |
| 3 | Saulin | Program Developer | 34k |
| 4 | Zenia Sroll | Web Developer | 42k |

Package Specification Code

Create Package specification code for defining procedure, function IN or OUT parameter and execute package specification program.

CREATE or REPLACE PACKAGE pkg1

IS | AS

PROCEDURE pro1

(no in number, info out varchar2);

FUNCTION fun1

(no in number)

RETURN varchar2;

END;

/

Package Body Code

Create Package body code for implementing procedure or function that are defined package specification. Once you implement execute this program.

CREATE or REPLACE PACKAGE BODY pkg1

IS

PROCEDURE pro1(no in number,info out varchar2)

IS

BEGIN

SELECT first\_name || ’ ‘ || last\_name INTO info FROM emp1 WHERE eno = no;

END;

FUNCTION fun1(no in number) return varchar2

IS

name varchar2(20);

BEGIN

SELECT ename INTO name FROM emp1 WHERE eno = no;

RETURN name;

END;

END;

/

Pl/SQL Program calling Package

Now we have a one package pkg1, to call package defined function, procedures also pass the parameter and get the return result.

DECLARE

no number := &no;

name varchar2(20);

BEGIN

pkg1.pro1(no,name);

dbms\_output.put\_line('Procedure Result');

dbms\_output.put\_line(name);

dbms\_output.put\_line('Function Result');

name := pkg1.fun1(no);

dbms\_output.put\_line(name);

/\*dbms\_output.put\_line(info.eno||' '||

info.ename||' '||

info.edept||' '||

info.esalary||' '||);\*/

END;

/

PL/SQL Program Result

Now execute the above created pkg\_prg.sql program to asking which user information you want to get, you put user id and give information.

**SQL>@pkg\_prg**  
no number &n=2  
Procedure Result  
2    marks jems    Program Developer    38K  
Function Result  
marks jems  
  
PL/SQL procedure successfully completed.

PL/SQL Package Alter

You can update package code you just recompile the package body,

Package Alter Syntax :

ALTER PACKAGE package\_name COMPILE BODY;

Recompile the already created/executed package code,

Package Alter Code :

**SQL>ALTER PACKAGE pkg1 COMPILE BODY;**  
  
Package body Altered.

PL/SQL Package Drop

You can drop package using package DROP statement,

Package Drop Syntax :

DROP PACKAGE package\_name;

Drop the pkg1 program that was we created,

Package Drop Code :

**SQL>DROP PACKAGE pkg1;**  
  
Package dropped.