07.03.19

Serially_reusable Pragma :-

In oracle, If we want to maintain state of the global variable or state of the Globalized cursor, then we are using serially_reusable pragma in packages.

Syntax :-

```
pragma serially_reusable;
```

State of the Global Variable :-

```
SQL> create or replace package pd1
     is
     g number(10) := 7;
     end;
     /
SQL> begin
     pd1.g := 80;
     end;
     /
SQL> begin
     dbms_output.put_line(pd1.g);
     end;
     /
```

80

Solution :-

```
SQL> create or replace
     package pd1 is
     g number(10) := 7;
     pragma serially_reusable;
     end;
     /
SQL> begin
     pd1.g := 80;
     end;
     /
SQL> begin
     dbms_output.put_line(pd1.g);
     end;
     /
```

7

State of the Cursor :-

```
SQL> Create or replace package pd2
     is
     cursor c1 is select * from emp;
     Pragma serially_reusable;
```

```
end;
/
SQL> begin
     Open pd2.c1;
     end;
     /
PL/SQL Procedure Successfully
Completed.
```

```
SQL> begin
     Open pd2.c1;
     end;
     /
PL/SQL procedure successfully
completed.
```

Overloading Procedures:-

Overloading refers to same name can be used for different purpose.

In oracle we are ~~or~~ implementing subprograms by using packages. In oracle overloading procedures having same ~~number~~ name with different type or different ~~num~~ no. of parameters.

```
SQL> create or replace package
     pd3 is
     Procedure p1 (a number, b number);
     Procedure p1 (x number, y number);
     end;
     /

SQL> create or replace package
     body pd3
     is
     Procedure p1 (a number, b number)
     is
     c number (10);
     begin
     c := a+b;
     dbms_output.put_line (c);
     end p1;
```

```
Procedure P1(x number, y number)
is
z number(10);
begin
z:= x-y;
dbms_output.put_line(z);
end p1;
end;
```

Execution:-

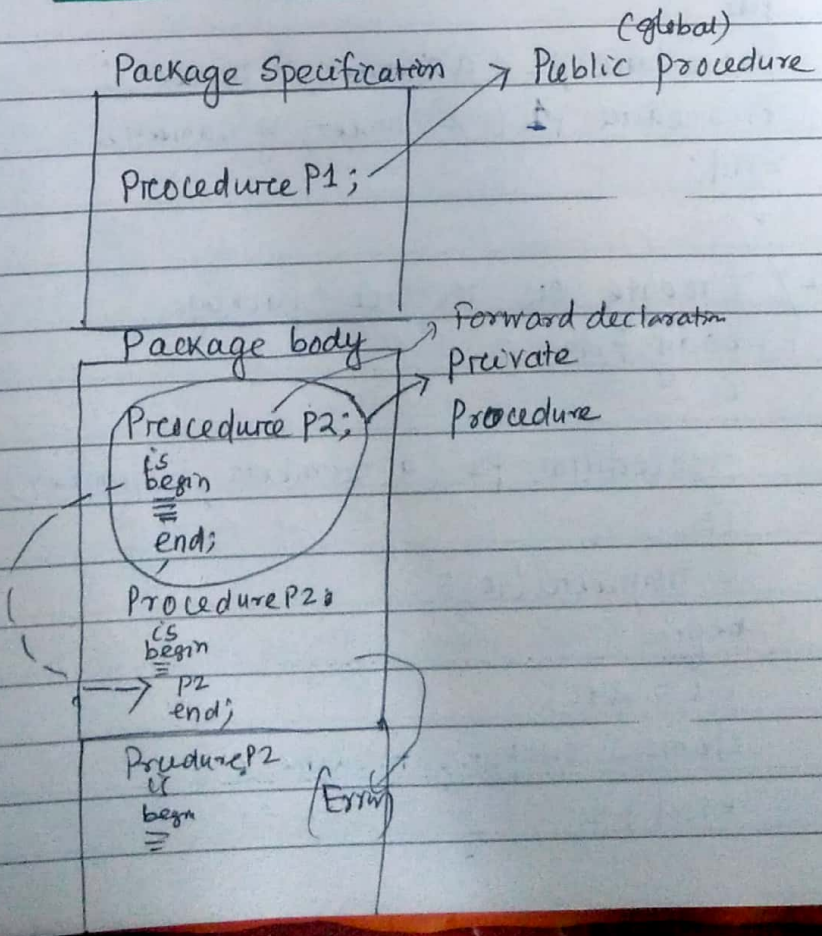
```
SQL> exec pd3.p1(8,5);
```

ERROR:- too many declarations of 'p1' match this call

NOTE:- In oracle, Whenever overloading procedures having same no. of parameters with same type, then those type of procedures are allowed to execute by using named notations only.

```
SQL> exec pd3.p1(a=>8, b=>5);
c 13
SQL> exec pd3.p1(x=>9, y=>2);
z 7
```

Forward Declaration:-



In, Oracle declaring a procedure with a packaged body is also called as forward declaration.

Generally in Oracle, before we are calling Private procedure into public procedure, then first we must implement private procedure before calling., otherwise use a forward declaration within packaged body.

ex:-

```
SQL> Create or replace package pdy
     is
     Procedure p1;
     end;
     /
SQL> create or replace package
     body pdy
     is
     procedure p2;
     procedure p1;
     is
     begin
     p2;
     end p1;
     procedure p2
     is
     begin
     dbms_output.put_line ('Private Proc');
     end p2;
     end;
     /
```

forward declaration

Execution:-

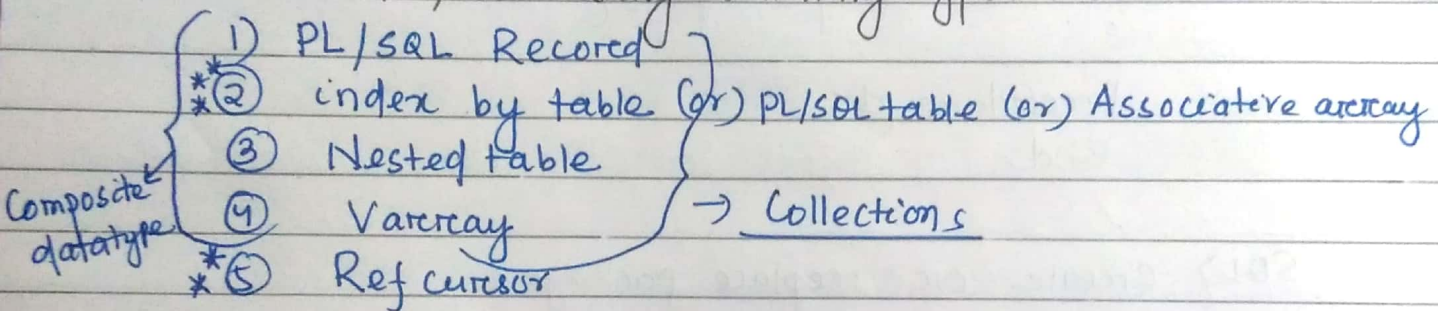
```
SQL> exec pdy.p1;
Private proc
```

★ Types are used in Packages:-

In oracle, we can also create user defined types by using type keyword.

In PL/SQL user defined types are created by using 2 step process. In this process first we are creating type by using appropriate syntax and then only we are allowed to create variable from that type name.

PL/SQL having following types. These are



04/12/19

PACKAGE

"Package" is a database object which encapsulates global variables, constants, cursors, procedures, functions, types, into single unit.

In oracle, packages doesn't have parameters and also packages can't be nested (user-defined) and also packages can't be invoked directly, but packages also used to improves performance of the application, because whenever we are calling a packaged subprog. 1st time then automatically total package is loaded into memory area.

Whenever we are calling subsequent subprograms, then oracle server call those subprogs. from memory area.

This default process automatically improves performance of the application, because this default process internally reduces this Input/output.

In oracle, every package having two parts these are, 1) Package Specification
2) Package body

In oracle, in package specification we are declaring object where as, in package body, we are implementing procedures, functions.

In oracle, by default package specification objects are public, where as package body objects are private.

Package Specification:-

Syntax:-

create or replace Package Packagename
IS/AS
→ global variable, declarations, Constant declarations;
→ Types declaration;
→ Cursores declaration;

→ Procedure declaration;
→ Function declaration;
end;
/

Package body :-

Syntax :-

Create or replace package body Packagename
IS/AS
→ Procedure implementation;
→ Function implementation;

end;
/

Calling Packaged Subprograms :-

1) Calling Packaged Procedures :-

method1 :-

Syntax :-

SQL> exec
Packagename.procedurename (actual parameters);

method2 :-
(Using Annonymous block)

Syntax :-

SQL> begin
Packagename.procedurename (actual parameters);
end;
/

2) Calling Packaged functions:-

method1:- (Using Select Statement)

Syntax:-

```
SQL> Select
     packagename.functionname(actual parameters)
     from dual;
```

method2:- (Using annonymous block):-

Syntax:-

```
SQL> begin
varname := packagename.functionname
                (actual parameters);
end;
/
```

Ex:-

```
SQL> create or replace package pj1
  is
  Procedure p1;
  Procedure p2;
  end;
  /
SQL> Create or replace package
  body pj1
  is
  Procedure p1
  is
  begin
  dbms_output.put_line('my first procedure');
  end p1;
  Procedure P2
  is
```

```
begin
dbms_output.put_line ('my second procedure');
end p2;
end;
/
```

Execution :-

```
SQL> exec pj1.p1;
  my first procedure
SQL> exec pj1.p2;
  my second procedure
```

In oracle, all packages information stored under dba_objects data dictionary.

Ex:-

```
SQL> Conn sys as sysdba;
Enter password: sys
SQL> desc dba_Objects;
SQL> select object_name from
     dba_objects
     Where object_type = 'PACKAGE';
```

In oracle, if you want to view code of the Packages then we are using text property from dba_source data dictionaries.

Example :-

```
SQL> Conn sys as sysdba;
     Enter password: sys
SQL> desc dba_source;
SQL> select text from dba_source
     Where name = 'PJ1';
```

Global Variables :-

In oracle, global variables are defined in Package specification only.

Ex:- SQL> create or replace package pj2
is
g number(10) := 1000;
procedure p1;
function f1(a number) return number;
end;
/

SQL> create or replace package body pj2
is
Procedure p1
is
z number(10);
begin
z := g/2;
dbms_output.put_line(z);
end p1;
function f1(a number) return number
is
begin
return a*g;
end f1;
end;
/

EXECUTION:-

SQL> exec pj2.p1;
500
SQL> select pj2.f1(8) from dual;
8000