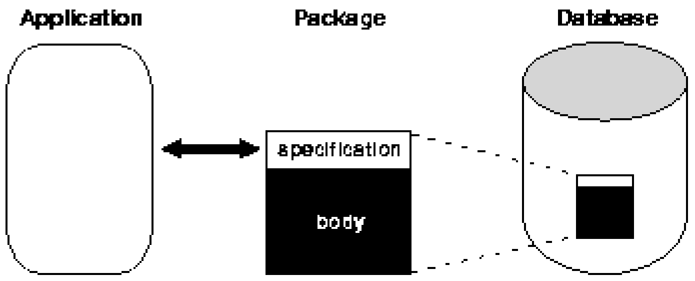
# PACKAGES

- A **package** is a schema object that groups logically related PL/SQL types, items, and subprograms.
- Packages usually have two parts
    - Specification
    - Body (optional)
- The **specification** is the interface to your applications; it declares the types, variables, constants, exceptions, cursors, and subprograms available for use.

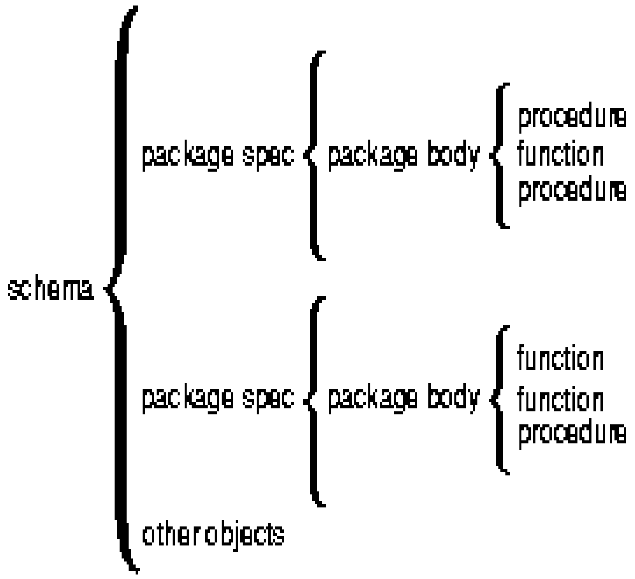
- The **body** fully defines cursors and subprograms, and so implements the spec.

## *Package Interface*



- The specifications holds **public declarations**, which are visible to your application. You must declare subprograms at the end of the specification after all other items
- The body holds implementation details and **private declarations**, which are hidden from your application.

***Package Scope***



**<u>Advantage of Package</u>**

- All related code in a single object
- All related code loaded into memory simultaneously
- Session global variables and types.
- Single object compilation.
- Variables persist for term of session.
- Initialization section.
- Overloading.
- Fewer objects to manage and grant/revoke privileges.

## Package Definition:

**CREATE OR REPLACE PACKAGE Package_name IS**

**Variables;**
**Ref Cursors;**
**Procedures procedure_name();**
**Functions function_name();**

**END Package_name;**

## Accessing Package variables:

The package variables can be accesed by using **•(dot)** operator.

// Package with variable definition

```
create or replace package package_1 is

lname varchar2(20) := 'Sivaji';
dept constant varchar2(10) := 'Oracle';
salary int := 20000;
end;
```

**Executing:** Using anonymous block

```
declare
  fname varchar2(20);
begin
            dbms_output.put_line('hello');
            fname := 'Dinesh Kumar' || package_1.lname;
            dbms_output.put_line(' Name : ' || fname);
            dbms_output.put_line(' Dept : ' || package_1.dept);
            dbms_output.put_line(' Salary : ' || package_1.salary);
end;
```

**Output:**

```
hello
Name : Dinesh KumarSivaji
Dept : Oracle
Salary : 20000
```

## Simple Packages:

By Dinesh

## // Package with one Procedure

```
-- package spec

create or replace package package_2 as

procedure p99;

end;



--package body

create or replace package body package_2 as

         procedure p99 is
         begin
         dbms_output.put_line('welcome to this world');
         end;
end;
```

## Accessing Package procedure inside anonymous block

```
SQL> exec package_2.p99();
```

**(OR)**

```
begin

package_2.p99;

end;
```

## Output:

```
welcome to this world
```

By Dinesh

## // Package with one function

```
-- package spec

create or replace package package_3 as

function f1 return number;

end;


-- package body

create or replace package body package_3 as

         function f1 return number as

            r number;
            pi float default 3.14;
            aoc float;
         begin
            r:= 10;
            aoc := (pi * r * r);

            return aoc;
         end;

end;
```

**Accessing Package function inside anonymous block**

SQL> select package_3.f1() from dual;

**(or)**

```
declare
res number;
begin

res:=package_3.f1;
dbms_output.put_line('result :' || res);

end;
```

**Output:**

```
result :314
```

## Complex Packages:

### // Packages with multiple Procedures & Functions

*--Package spec*

```
create or replace package package_4 as

procedure pp1;
procedure factorial_1(n number);
procedure using_cursor;
function ff1 return number;
function search_in_table return user_1.name%TYPE;
procedure calling_fn;
end;
```

*-- Package body*

```
create or replace package body package_4 as

         -- Procedure for User menu choice
         procedure pp1 is
         ch int;
         begin
            dbms_output.put_line('1.add 2.Sub 3.Div');
            dbms_output.put_line('choice ???');
            ch :=&ch;

               case
               when ch = 1 then
                  dbms_output.put_line(' addition selected');
               when ch = 2 then
                  dbms_output.put_line(' Subtract selected');
                  when ch = 3 then
                     dbms_output.put_line(' Division selected');
               end case;
         end;

         -- Procedure to calculate factorial

         procedure factorial_1(n number) is
            res int;
            fact number(10):=1;
        begin
          for i in 1..n loop
                     fact:=fact*i;
           end loop;
                res:=fact;
                  dbms_output.put_line(res);
      end;

       -- Procedure using cursor
         procedure using_cursor as
                        a1 num.a%TYPE;
                        b1 num.b%TYPE;

                        cursor c1 is
                        select a,b from num;
                        begin
```

```
                              open c1;
                              loop
                              fetch c1 into a1,b1;
                              exit when c1%NOTFOUND;
                              dbms_output.put_line(a1||'     ' || b1);
                              end loop;
                       end;


          -- function for area of circle
          function ff1 return number as
           r number;
             pi float default 3.14;
             aoc float;
          begin
           r:= 10;
         aoc := (pi * r * r);

             return aoc;
          end;


          -- Function is used to find a name in table user_1

          function search_in_table return user_1.name%TYPE as
          uname user_1.name%TYPE;
          begin
          select name into uname from user_1 where id = 113;
          return uname;
          end;


        -- Procedure to call function search_in_table
       procedure calling_fn is
          res user_1.name%TYPE;
          begin
            res := search_in_table();
              dbms_output.put_line('result name : ' || res);
          end;

end;
```

## Package with local Function or Procedure:

// Package With Local Function and Global Procedure

*--Package spec*

```
create or replace package package_5 as
procedure ppx;
end;
```

*-- Package body*

```
create or replace package body package_5 as
      -- local function

         function local_1(str varchar2) return varchar2 as
         begin
             return upper(str);
         end;

      -- global procedure

         procedure ppx as
         call varchar2(100);
         begin

            call := local_1('dinesh');
            dbms_output.put_line(' executing local function & global
      procedure');
            dbms_output.put_line(' Text received' || call);
         end;
end;
```

**NOTE: Before calling the function it should be declared in package body.**

## Package Overloading:

// Simple package to implement overloading

```
-- Package spec


create or replace package package_7 as

procedure over1(x int);
procedure over1(x varchar2);

end;


-- Package body

create or replace package body package_7 as

         procedure over1(x int) as
         res int;
         begin
            res := x*x;
              dbms_output.put_line(' Result of x*x = '|| res);
         end;

         procedure over1(x varchar2) as
         begin
               dbms_output.put_line(' text received : '|| x);
         end;
end;
```