Universitatea
Politehnica
București

Facultatea de
Automatică și
Calculatoare

Catedra de
Calculatoare

Laborator 8
Pachete PL/SQL

Autori

Conf. Dr. Ing. Alexandru Boicea
As. Dr. Ing. Ciprian-Octavian Truică

Cuprins

- Pachete PL/SQL
- Specificațiile unui pachet
- Corpul unui pachet
- Restricții în definirea unui pachet
- Informații din dicționarul de date
- Pachete standard

Pachete PL/SQL

- Un pachet (package) este o bibliotecă de obiecte stocate pe server, de tipul procedurilor stocate, funcțiilor, cursoarelor, tipurilor de date, excepțiilor, variabilelor și constantelor;
- Toate obiectele declarate în secțiunea de creare a unui pachete sunt globale (publice) și pot fi apelate din orice program PL/SQL, asemănător variabilelor globale din alte limbaje de programare;
- Un pachet este compus din două secțiuni distincte:
  - O secțiune de creare (*create package*) care conține specificațiile globale (publice) de declarare a conținutului (structura obiectelor);
  - O secțiune care cuprinde corpul pachetelor (*package body*) în care sunt descrise efectiv obiectele și subprograme, variabile, structuri locale (private).
- Un pachet este creat ca un obiect în dicționarul bazei de date.

Pachete PL/SQL

- Principalele avantaje oferite de un pachet sunt:
  - Modularitatea aplicațiilor;
  - Posibilitatea declarării de obiecte globale;
  - Îmbunătățirea performanțelor sistemului de gestiune;
  - Ușurința în proiectarea aplicațiilor;
  - Adăugarea de funcționalități noi.

Specificațiile unui pachet

- Specificațiile reprezintă partea publică a unui pachet și au următoarea sintaxă:

  CREATE [OR REPLACE] PACKAGE package_name {IS|AS}

  global (public) types and variable declarations

  global (public) subprogram specifications

  END [package_name]

- Unde :
  - **package_name** – numele pachetului
  - **global (public) types and variable declarations** – reprezintă declarațiile globale (publice) ale cursoarelor, excepțiilor, constantelor, variabilelor și tipurilor de date și descrierea acestora din punct de vedere al structurii
  - **global (public) subprogram specifications** – reprezintă numele procedurilor și funcțiilor cu parametrii formali aferenți. Acestea sunt globale (publice).

Specificațiile unui pachet

- Subprogramele, variabilele și cursoarele care vor fi apelate din exterior trebuie să fie cuprinse în secțiunea de creare a pachetului, pe când subprogramele, cursoarele, excepțiile, constantele, variabilele și tipurile de date folosite doar în pachet vor fi declarate doar în secțiunea body a acestuia.
- Referirea într-un program PL/SQL a unei componente globale (publice) a unui pachet se face folosind numele componentei, având ca prefix numele pachetului:
- **package_name.type_name**
- **package_name.item_name**
- **package_name.subprogram_name**
- Unde:
  - **package_name** – numele pachetului
  - **type_name** – numele unui tip de date
  - **item_name** – numele unui cursor, excepție, constantă sau variabilă locală
  - **subprogram_name** – numele unei proceduri sau funcții

Specificațiile unui pachet

- Dacă compilarea unui pachet se face cu succes, apare mesajul: **Package created;**
- În caz că apar erori la crearea unui pachet apare mesajul de avertizare: **Warning: Package created with compilation errors;**
- Pentru a vedea erorile de compilare se va folosi comanda: **show errors.**

Corpul unui pachet

- Corpul unui pachet (package body) conține descrierea efectivă a procedurilor și funcțiilor definite în specificații.
- Această parte poate să conțină și componente locale (private) care sunt folosite doar în interiorul pachetului.
- Sintaxa este:

CREATE [OR REPLACE] PACKAGE BODY package_name IS|AS

local type and variable declarations

subprogram bodies

[BEGIN

initialization statements

end;]

END [package_name]

Corpul unui pachet

- **package_name** – numele pachetului
- **local type and variable declarations** – reprezintă declarațiile locale ale cursoarelor, excepțiilor, constantelor, variabilelor și tipurilor de date
- **subprogram bodies** – codul sursă al procedurilor și funcțiilor definite în specificații
- **initialization statements** – codul de inițializare

Corpul unui pachet

- Cursoarele, tipurile de date, excepțiile, variabilele și constantele declarate în corpul pachetului vor avea caracter local (privat), deci vor fi accesibile numai în blocurile în care au fost definite;
- La primul apel al unui obiect dintr-un pachet, întregul pachet este inițializat;
- Inițializarea implică încărcarea unui pachet de pe disc în memorie și alocarea de spațiu în memorie pentru variabilele globale (publice);
- Dacă apelul se referă la o procedură sau funcție, inițializarea este urmată de execuția codului deja compilat;
- Fiecare sesiune de lucru are o copie proprie a variabilelor din pachet;
- Sesiuni diferite pot apela același pachet, dar fiecare apel al pachetului are rezervată o zonă diferită de memorie.

Corpul unui pachet

- PL/SQL oferă facilitatea introducerii unui cod de inițializare care este executat automat la primul apel al pachetului;
- Codul de inițializare este un bloc deschis în copul pachetului, după descrierea tuturor procedurilor și funcțiilor, și care poate face inițializări prin atribuiri directe sau apelează proceduri de inițializare.

Corpul unui pachet

- Ex. 1. Să se scrie un pachet p_angajare, care conține o funcție și o procedură, pentru a face o listă cu angajații care au venit în firmă înaintea unui director al companiei(care le este șef direct) și au primit comision. Șeful direct al unui angajat este specificat în coloana mgr. Procedura va insera rezultatele în tabela lista.
- Pasul I – crearea tabelei lista

```
create table lista
(
  den_dep varchar2(20)
  , nume_sef varchar2(20)
  , data_sef date
  , nume_sub varchar2(20)
  , data_sub date
  , com_sub number
)
```

Corpul unui pachet

- Pasul II – crearea specificațiilor

```
create or replace package angajare as
  cursor depart is select deptno, dname from dept order by deptno;
  v_dep depart%rowtype;
  function vechime(data_ang date, data_ang_sef date) return boolean;
  procedure prelucrare;
end angajare;
/
```

Corpul unui pachet

- Pasul III – crearea corpului pachetului

```
create or replace package body angajare as
  function vechime(data_ang date, data_ang_sef date)
  return boolean
  is
    verif boolean;
  begin
    if data_ang<data_ang_sef then verif := true;
    else verif := false;
    end if;
    return verif;
  end vechime;
```

Corpul unui pachet

```
procedure prelucrare is
  cursor c_ang is select * from emp;
  w_c c_ang%rowtype;
  sef number(4);
  nume_sef varchar2(20);
  data_ang_sef date;
  nume varchar2(20);
  conditie boolean;
begin
  delete from lista;
  for i in angajare.depart
  loop
    begin
      select empno, ename, hiredate into sef, nume_sef, data_ang_sef
        from emp where deptno = i.deptno and lower(job) = 'manager';
    exception
      when no_data_found then sef := 0;
    end;
```

Corpul unui pachet

```
    open c_ang;
    loop
      fetch c_ang into w_c;
      exit when c_ang%notfound;

      if w_c.mgr = sef then
        conditie := vechime(w_c.hiredate, data_ang_sef);
        if conditie in (true) and nvl(w_c.comm, 0) <> 0 then
          insert into lista values(i.dname, nume_sef, data_ang_sef,
            w_c.ename, w_c.hiredate, w_c.comm);
        end if;
      end if;
    end loop;
    close c_ang;
  end loop;
 end prelucrare;
end angajare;
/
```

Corpul unui pachet

- Pasul IV – execuția pachetului

```
begin
   angajare.prelucrare;
end;
```

- Pasul V – rezultatul execuției

```
select * from lista;
```

Restricții în definirea pachetelor

- În pachete nu se permite declararea a două proceduri sau funcții cu același nume, dacă parametrii acestora diferă numai prin nume sau mod (IN, OUT, IN OUT). Trebuie ca cel puțin un parametru să fie de un alt tip, iar tipul nu trebuie să fie din aceeași familie(de exemplul tipul CHAR este din aceeași familie cu VARCHAR2)
- Aceeași situație este și în cazul rezultatelor returnat de o funcție;
- Erorile nu apar în momentul creării, ci în momentul execuției, ceea ce creează o anumită confuzie.

Restricții în definirea pachetelor

- Exemplu de ambiguitate:

```
create or replace package p_test as
  function f_test(data_ang date, nume varchar2) return number;
  function f_test(data_ang date, nume char) return number;
end p_test;
/
create or replace package body p_test as
  function f_test(data_ang date, nume varchar2) return number is
    nr number :=0;
  begin
    if data_ang < sysdate then nr := 1;
    end if;
    return nr;
  end f_test;
  function f_test(data_ang date, nume char  ) return number is
    nr number :=2;
  begin
    if data_ang < sysdate then nr := 3;
    end if;
    return nr;
  end f_test;
end p_test;
/
```

Restricții în definirea pachetelor

- Exemplul se compilează fără erori, însă dacă executăm blocul atunci o să apară o eroare:

```
SQL> declare
  2  i number;
  3  begin
  4  i := p_test.f_test(sysdate, 'Allen');
  5  end;
  6  /
i := p_test.f_test(sysdate, 'Allen');
     *
ERROR at line 4:
ORA-06550: line 4, column 6:
PLS-00307: too many declarations of 'F_TEST' match this call
ORA-06550: line 4, column 1:
PL/SQL: Statement ignored
```

Restricții în definirea pachetelor

- Dacă schimbăm pentru una din funcții tipul parametrului al doilea, cu tipul number, atunci nu mai apare eroarea:

```
create or replace package p_test as
  function f_test(data_ang date, nume varchar2) return number;
  function f_test(data_ang date, idEmp number) return number;
end p_test;
/
create or replace package body p_test as
  function f_test(data_ang date, nume varchar2) return number is
    nr number :=0;
  begin
    if data_ang < sysdate then nr := 1;
    end if;
    return nr;
  end f_test;
  function f_test(data_ang date, idEmp number) return number is
    nr number :=2;
  begin
    if data_ang < sysdate then nr := 3;
    end if;
    return nr;
  end f_test;
end p_test;
/
```

Restricții în definirea pachetelor

- Execuția blocului:

```
SQL> declare
  2  i number;
  3  begin
  4  i := p_test.f_test(sysdate, 'Allen');
  5  end;
  6  /

PL/SQL procedure successfully completed.
```

Restricții în definirea pachetelor

- O variabilă globală (publică) poate fi folosită pentru a inițializa un parametru al subprogramelor (public/privat) cât și în interiorul acestora. De exemplu construcția următoare este permisă:

Restricții în definirea pachetelor

```
create or replace package apel as
  var number := 10;
  procedure f(x in number := var);
end apel;
/
create or replace package body apel as
  function f_test(nr number := var) return number is
  begin
    if nr = var then return 1;
    end if;
    return 2;
  end f_test;

  procedure f(x in number := var)
  is
  begin
    if x = var then
      dbms_output.put_line('Sunt pe ramura if ' || f_test());
    else
      dbms_output.put_line('Sunt pe ramura else ' || f_test(x));
    end if;
  end f;
end apel;
/
```

Restricții în definirea pachetelor

```
set serveroutput on;
begin
   apel.f();
   apel.f(4);
end;
/
```

Restricții în definirea pachetelor

- O variabilă locală (privată) poate fi folosită pentru a inițializa un parametru al unei subprogram local (privat) și în interiorul subprogramelor (private/publice). De exemplu construcția următoare este permisă:

Restricții în definirea pachetelor

```
create or replace package apel as
  procedure f(x in number);
end apel;
/

create or replace package body apel as
  var number := 10;
  function f_test(nr number := var) return number is
  begin
    if nr = var then return 1;
    end if;
    return 2;
  end f_test;

  procedure f(x in number)
  is
  begin
    if x = var then
      dbms_output.put_line('Sunt pe ramura if ' || f_test());
    else
      dbms_output.put_line('Sunt pe ramura else ' || f_test(x));
    end if;
  end f;
end apel;
/
```

Restricții în definirea pachetelor

```
set serveroutput on;
begin
   apel.f(10);
   apel.f(4);
end;
/
```

Informații din dicționarul de date

- Deoarece pachetele sunt create ca oricare alt obiect, din dicționarul bazei de date putem să aflăm informații despre ele făcând interogări pe view-urile sistemului de gestiune ORACLE.
- De exemplu dacă vrem să vedem toate pachetele create de userul curent, data când au fost create, data ultimei modificări și starea lor, putem să executăm următoarea cerere SQL:

```
SQL> col object_name for a20
SQL> select object_name, created, last_ddl_time, status
  2  from user_objects
  3  where object_type='PACKAGE';

OBJECT_NAME          CREATED   LAST_DDL_ STATUS
-------------------- --------- --------- -------
P_TEST               24-NOV-13 24-NOV-13 VALID
APEL                 24-NOV-13 24-NOV-13 VALID
ANGAJARE             24-NOV-13 24-NOV-13 VALID
```

Informații din dicționarul de date

- Pentru a vedea care pahete au specificații și nu au secțiunea *body* se execută următoarea cerere SQL:

```
SQL> select object_name, created, status
  2  from user_objects
  3  where object_type='PACKAGE'
  4    and object_name not in (select object_name from user_objects where
  5      object_type = 'PACKAGE BODY');

OBJECT_NAME          CREATED   STATUS
-------------------- --------- -------
APEL2                24-NOV-13 VALID
```

Informații din dicționarul de date

- Pentru a vedea secțiunea de specificații a pachetului p_test, executăm următoarea cerere SQL:

```
SQL> select text from user_source
  2  where lower(name) = 'p_test' and lower(type)='package'
  3  order by line;

TEXT
--------------------------------------------------------------------------------

package p_test as
  function f_test(data_ang date, nume varchar2) return number;
  function f_test(data_ang date, nume number) return number;
end p_test;
```

Informații din dicționarul de date

- Pentru a lista codul sursă al unui pachet, se execută cererea SQL:

```
SQL> set pages 100
SQL> select text from user_source
  2  where lower(name) = 'p_test' and lower(type)='package body'
  3  order by line;

TEXT
--------------------------------------------------------------------------------

package body p_test as
  function f_test(data_ang date, nume varchar2) return number is
    nr number :=0;
  begin
    if data_ang < sysdate then nr := 1;
    end if;
    return nr;
  end f_test;
  function f_test(data_ang date, nume number) return number is
    nr number :=2;
  begin
    if data_ang < sysdate then nr := 3;
    end if;
    return nr;
  end f_test;
end p_test;

16 rows selected.
```

Informații din dicționarul de date

- Specificațiile și corpul unui pachet se pot șterge din dicționar folosind comanda DDL, DROP:
- **DROP PACKAGE pakage_name;**
- Pentru a șterge numai corpul unui pachet se folosește comanda SQL:
- **DROP PACKAGE BODY package_name;**
- Pentru a da privilegii de execuție a pachetelor altor utilizatori, utilizatorul care a creat pachetul (sau administratorul) poate folosi comanda DCL, GRANT:
- **GRANT EXECUTE ON package_name TO user_name;**
- Utilizatorul grantificat poate apela un obiect din pachetul respectiv, specificând în apel userul, pachetul și obiectul. De exemplu, o procedură fără parametri poate fi apelată direct din SQL*Plus astfel:
- **EXECUTE user_name.package_nane.procedure_name;**

Informații din dicționarul de date

- O procedură poate fi apelată și dintr-un bloc SQL:

```
DECLARE
BEGIN
...
  user_name.package_name.procedure_name;
...
END;
```

- Pentru a revoca privilegiile de execuție se folosește comanda REVOKE:
- **REVOKE EXECUTE ON package_name FROM user_name;**

Pachete standard

- Serverul Oracle conține câteva pachete standard(de sistem), care sunt instalate odată cu serverul de baze de date.
- Câteva din cele mai uzuale pachete standard sunt:
  - DBMS_STANDARD – conține proceduri care ajută programatorul în interacțiunea cu Oracle. De exemplu, procedura *raise_application_error* este folosită pentru definirea propriilor mesaje;
  - DBMS_OUTPUT – conține proceduri pentru afișare, folosite în depanarea sau execuția programelor. De exemplu, procesura *put_line* este folosită pentru afișarea de mesaje în SQL*Plus;
  - DBMS_PIPE – permite comunicarea între sesiuni diferite, folosind o zonă de memorie comună (*pipe*), pentru schimbul de informații. O sesiune poate folosi două proceduri *pack_message* și *send_message*, pentru a împacheta și pune mesajul în zona comună și apoi a-l trimite către o altă sesiune. Sesiunea receptoare poate folosi două proceduri *receive_message* și *unpack_message*, care fac operațiunile inverse. De exemplu, se pot scrie rutine C++, care permit serverelor externe să capteze informații și să le trimită către proceduri stocate în baza de date Oracle.

Pachete standard

- UTL_FILE – permite programatorilor PL/SQL să scrie și să citească fișiere text gestionate de sistemul de operare. Pentru aceasta, se poate folosi funcția *fopen* pentru deschiderea fișierului și procedura *get_line* pentru citirea linie cu linie;
- UTL_HTML – permite programatorilor PL/SQL să acceseze Internetul sau Oracle Web Server, folosind protocolul HTTP; Pachetul acceptă un URL, se conectează la site-ul specificat și întoarce datele solicitate, de regulă în format HTML;
- DBMS_SQL – permite unui programator PL/SQL să execute comenzi DDL sau comenzi SQL standard, în mod dinamic;
- DBMS_ALERT – permite utilizarea triggerelor pentru alertare, în cazul în care intervin modificări în baza de date.

Pachete standard

- Informațiile din dicționar, legate de pachetele standard, se obțin folosind view-urile de sistem *DBA_OBJECTS* și *DBA_SOURCE*, acesibile userului *system* sau celor care au privilegiul DBA;
- Pentru vizualizarea mai multor informații din dicționar, legate de pachetele standard, se poate folosi cererea SQL:

  SELECT text FROM dba_source WHERE lower(name) = '*package_name*' AND lower(type)='package' ORDER BY line;

- Pachetele standard pot fi activate sau dezactivate (ENABLE/DISABLE) cu comanda SET:

  SET *standard_package* {ON|OFF}

Exercițiu

- Ex. 2. Să se scrie un pachet care conține:
  - O funcție care calculează venitul maxim pe un departament;
  - O funcție care calculează vechimea cea mai mare în companie, a unui angajat din deprtamentul din care face parte;
  - O procedură care apelează cele două funcții pentru a face o listă cu angajații care au venitul sau vechimea cea mai mare în propriul departament (calculată ca număr de luni complete);
  - Procedura face o listă care are antetul:

DEPARTAMENT NUME VENIT VECHIME

Exercițiu

```
create or replace package prima as
  function venit_maxim(nr_dep number) return number;
  function vechime_maxima(nr_dep number) return number;
  procedure calcul;
end prima;
/
create or replace package body prima as
  function venit_maxim(nr_dep number) return number is
    venit_max number;
  begin
    select max(sal+nvl(comm, 0)) into venit_max from emp where deptno=nr_dep;
    return venit_max;
  end venit_maxim;

  function vechime_maxima(nr_dep number) return number is
    vec_max number;
  begin
    select max(months_between(sysdate, hiredate)) into vec_max from emp
      where deptno = nr_dep;
    return vec_max;
  end vechime_maxima;
```

Exercițiu

```
procedure calcul is
   cursor c_dep is select distinct deptno from dept;
   w_dep c_dep%rowtype;
   den_dep dept.dname%type;
   nume emp.ename%type;
   data_ang emp.hiredate%type;
   venit_m number;
   vec_m number;
   venit_a number;
   vec_a number;
   venit_prim number;
   vec_prim number;
   ok integer := 0;
begin
   dbms_output.put_line(rpad('Departament', 15)||rpad('Nume', 10)
     ||rpad('Venit', 10)||rpad('Vechime', 10));
   dbms_output.put_line(rpad('=', 45,'='));
```

Exercițiu

```
open c_dep;
loop
  fetch c_dep into w_dep;
  exit when c_dep%notfound;
  begin
    select dname into den_dep from dept where deptno = w_dep.deptno;
    venit_m := trunc(prima.venit_maxim(w_dep.deptno));
    vec_m := trunc(prima.vechime_maxima(w_dep.deptno));
    venit_prim := 0;
    vec_prim := 0;
    for i in (select distinct empno, ename from emp
      where deptno = w_dep.deptno)
    loop
      ok := 0;
      select trunc(sal+nvl(comm,0)),
        trunc(months_between(sysdate, hiredate))
        into venit_a, vec_a from emp where empno = i.empno;
      if venit_a = venit_m and vec_a = vec_m then
        venit_prim := venit_a;
        vec_prim := vec_a;
        ok := 1;
```

Exercițiu

```
        elsif venit_a = venit_m and vec_a <> vec_m then
          venit_prim := venit_a;
          vec_prim := 0;
          ok := 1;
        elsif venit_a <> venit_m and vec_a = vec_m then
          venit_prim := 0;
          vec_prim := vec_a;
          ok := 1;
        end if;
        if ok = 1 then
          dbms_output.put_line(rpad(den_dep, 15)||rpad(i.ename, 10)
            ||rpad(venit_prim, 10)||rpad(vec_prim, 10));
        end if;
      end loop;
    exception
      when too_many_rows then
        dbms_output.put_line('Exista mai multe inregistrari');
      when no_data_found then
        dbms_output.put_line('Nu exista inregistrari');
    end;
  end loop;
  close c_dep;
 end calcul;
end prima;
/
```

Exercițiu

```
set serveroutput on;
begin
   prima.calcul;
end;
/
```