

## Laborator 10: PACHETE in PL/SQL

### 1. Definirea pachetelor

- Pachetul (*package*) permite încapsularea într-o unitate logică în baza de date a procedurilor, funcțiilor, cursorilor, tipurilor, constantelor, variabilelor și excepțiilor.
- Spre deosebire de subprograme, pachetele nu pot:
  - fi apelate
  - transmite parametri
  - fi încuibărite
- Un pachet are două părți, fiecare fiind stocată separat în dicționarul datelor.
  - Specificarea pachetului (*package specification*) – partea „vizibilă”, adică interfața cu aplicației sau cu alte unități program. Se declară tipuri, constante, variabile, excepții, cursori și subprograme folosite de utilizatorul.
  - Corpul pachetului (*package body*) – partea „acunsă”, mascată de restul aplicației, adică realizarea specificației. Corpul definește cursori și subprograme, implementând specificația. Obiectele conținute în corpul pachetului sunt fie private, fie publice.
- Un pachet are următoarea formă generală:

```
CREATE PACKAGE nume_pachet {IS | AS} -- specificația  
/* interfața utilizator, care conține: declarații de tipuri și obiecte  
   publice, specificații de subprograme */  
END [nume_pachet];
```

```
CREATE PACKAGE BODY nume_pachet {IS | AS} -- corpul  
/* implementarea, care conține: declarații de obiecte și tipuri private,  
   corpuri de subprograme specificate în partea de interfață */  
[BEGIN]  
/* instrucțiuni de inițializare, executate o singură dată când  
   pachetul este invocat prima oară de către sesiunea  
   utilizatorului */  
END [nume_pachet];
```

### 2. Pachete predefinite

- *DBMS\_OUTPUT* permite afișarea de informații. *DBMS\_OUTPUT* lucrează cu un *buffer* (conținut în *SGA*) în care poate fi scrisă sau regăsită informație. Procedurile pachetului sunt:
  - PUT – depune (scrie) în buffer informație
  - PUT\_LINE – depune în buffer informația, împreună cu un marcaj - sfârșit de linie
  - NEW\_LINE – depune în buffer un marcaj - sfârșit de linie
  - GET\_LINE – regăsește o singură linie de informație;
  - GET\_LINES – regăsește mai multe linii de informație;
  - ENABLE/DISABLE – activează/dezactivează procedurile pachetului.

- *DBMS\_SQL* permite folosirea dinamică a comenzilor *SQL* în proceduri stocate sau în blocuri anonime și analiza gramaticală a comenzilor *LDD*.
  - *OPEN\_CURSOR* (deschide un nou cursor, adică se stabilește o zonă de memorie în care este procesată comanda *SQL*);
  - *PARSE* (stabilește validitatea comenzii *SQL*, adică se verifică sintaxa instrucțiunii și se asociază cursorului deschis);
  - *BIND\_VARIABLE* (leagă valoarea data de variabila corespunzătoare din comanda *SQL* analizată)
  - *EXECUTE* (execută comanda *SQL* și returnează numărul de linii procesate);
  - *FETCH\_ROWS* (regăsește o linie pentru un cursor specificat, iar pentru mai multe linii folosește un *LOOP*);
  - *CLOSE\_CURSOR* (închide cursorul specificat).
- *DBMS\_JOB* este utilizat pentru planificarea execuției programelor *PL/SQL*. Dintre subprogramele acestui pachet menționăm:
  - *SUBMIT* – adaugă un nou *job* în coada de așteptare a *job*-urilor;
  - *REMOVE* – șterge un *job* specificat din coada de așteptare a *job*-urilor;
  - *RUN* – execută imediat un *job* specificat;
  - *NEXT\_DATE* – modifică momentul următoarei execuții a unui *job*;
  - *INTERVAL* – modifică intervalul între diferite execuții ale unui *job*.
- *UTL\_FILE* permite programului *PL/SQL* citirea din fișierele sistemului de operare, respectiv scrierea în aceste fișiere. El este utilizat pentru exploatarea fișierelor text. Scrierea și regăsirea informațiilor se face cu ajutorul unor proceduri asemănătoare celor din pachetul *DBMS\_OUTPUT*.  
Procedura *FCLOSE* permite închiderea unui fișier.

## Probleme rezolvate

1.

- a) Creați specificația și corpul unui pachet numit *DEPT\_PKG\_PNU* care conține:
- procedurile *ADD\_DEPT*, *UPD\_DEPT* și *DEL\_DEPT*, corespunzătoare operațiilor de adăugare, actualizare (a numelui) și ștergere a unui departament din tabelul *DEPT\_PNU*;
  - funcția *GET\_DEPT*, care determină denumirea unui departament, pe baza codului acestuia.
- b) Invocați procedurile și funcția din cadrul pachetului atât prin blocuri *PL/SQL* cât și prin comenzi *SQL*.

*Soluție:*

Specificația pachetului:

```
create or replace PACKAGE dept_pkg_pnu IS
    PROCEDURE add_dept (p_deptid dept.deptno%TYPE, p_deptname
dept.dname%TYPE);
    PROCEDURE del_dept (p_deptid dept.deptno%TYPE);
    FUNCTION get_dept (p_deptid dept.deptno%TYPE) RETURN dept.dname%TYPE;
    PROCEDURE upd_dept (p_deptid dept.deptno%TYPE, p_deptname
dept.dname%TYPE);
END dept_pkg_pnu;
```

*Obs:* Dacă apare un *warning* care anunță că există erori de compilare, atunci este utilă comanda:

**SHOW ERRORS**

Corpul pachetului:

```
create or replace PACKAGE BODY dept_pkg_pnu IS
```

```

PROCEDURE add_dept (p_deptid dept.deptno%TYPE, p_deptname
dept.dname%TYPE)
IS
BEGIN
    INSERT INTO dept_pnu(deptno, dname)
    VALUES (p_deptid, p_deptname);
    COMMIT;
END add_dept;

PROCEDURE del_dept (p_deptid dept.deptno%TYPE) IS
BEGIN
    DELETE FROM dept_pnu
    WHERE deptno = p_deptid;
    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20203, 'Nici un departament sters');
    END IF;
END del_dept;

FUNCTION get_dept (p_deptid dept.deptno%TYPE)
RETURN dept.dname%TYPE IS
    v_ume dept.dname%TYPE;
BEGIN
    SELECT dname
    INTO v_ume
    FROM dept_pnu
    WHERE deptno = p_deptid;
    RETURN v_ume;
END get_dept;

PROCEDURE upd_dept (p_deptid dept.deptno%TYPE, p_deptname
dept.dname%TYPE) IS
BEGIN
    UPDATE dept_pnu
    SET dname = p_deptname
    WHERE deptno = p_deptid;
    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20204, 'Nici un departament
actualizat');
    END IF;
END upd_dept;
END dept_pkg_pnu;

```

**Obs:** Pentru invocarea procedurii:

```

BEGIN
    dept_pkg_pnu.add_dept(12, 'IT');
    upd_dept(12, 'Information technology');
END;

```

Pentru invocarea funcției:

```

BEGIN
    DBMS_OUTPUT.PUT_LINE('Departamentul cautat este: ' ||
dept_pkg_pnu.get_dept(12) );
END;

```

2. Creați specificația și corpul unui pachet numit EMP\_PKG\_PNU care conține:

- procedura publică ADD\_EMP - adaugă o înregistrare în tabelul EMP\_PNU; utilizează o secvență pentru generarea cheilor primare; vor fi prevăzute valori implicite pentru parametrii nespecificați;

- procedura publică GET\_EMP - pe baza unui cod de angajat transmis ca parametru, întoarce în doi parametri de ieșire salariul și job-ul corespunzător;

- funcția privată VALID\_JOB\_ID - rezultatul acestei funcții indică dacă job-ul unui angajat corespunde unei valori existente în tabelul JOBS. Funcția va fi utilizată în cadrul procedurii ADD\_EMP, făcând posibilă doar introducerea de înregistrări având coduri de job valide.

Tratați eventualele excepții.

*Soluție:*

```
create or replace PACKAGE emp_pkg_pnu IS
  Procedure add_emp (
    p_name emp.ename%TYPE,
    p_hire_date emp.hiredate%TYPE DEFAULT SYSDATE,
    p_mgr emp.mgr%TYPE DEFAULT 145,
    p_sal emp.sal%TYPE DEFAULT 1000,
    p_job emp.job%TYPE DEFAULT 'SA_REP',
    p_comm emp.comm%TYPE DEFAULT 0,
    p_deptid emp.deptno%TYPE DEFAULT 30);
  Procedure get_emp (p_empid IN emp.empno%TYPE, p_sal OUT emp.sal%TYPE,
    p_job OUT emp.job%TYPE);
END emp_pkg_pnu;
```

```
create or replace PACKAGE BODY emp_pkg_pnu IS
  PROCEDURE add_emp (
    p_name emp.ename%TYPE, --implicit de tip IN
    p_hire_date emp.hiredate%TYPE DEFAULT SYSDATE,
    p_mgr emp.mgr%TYPE DEFAULT 145,
    p_sal emp.sal%TYPE DEFAULT 1000,
    p_job emp.job%TYPE DEFAULT 'SA_REP',
    p_comm emp.comm%TYPE DEFAULT 0,
    p_deptid emp.deptno%TYPE DEFAULT 30)
  IS
  BEGIN
    INSERT INTO emp_pnu(empno, ename, mgr, sal, job, comm, deptno, hiredate)
    VALUES(123, p_name, p_mgr, p_sal, p_job, p_comm, p_deptid, p_hire_date);
  END add_emp;

  PROCEDURE get_emp (p_empid IN emp.empno%TYPE,
    p_sal OUT emp.sal%TYPE,
    p_job OUT emp.job%TYPE)
  IS
  BEGIN
    SELECT sal, job
    INTO p_sal, p_job
    FROM emp_pnu
    WHERE empno = p_empid;
  END get_emp;
END emp_pkg_pnu;
```

Exemplu de apel:

Declare

```
v1 emp.sal%TYPE;  
v2 emp.job%TYPE;
```

begin

```
emp_pkg_pnu.add_emp('Popescu', p_job => 'SA_MAN');  
emp_pkg_pnu.get_emp(123,v1,v2);  
DBMS_OUTPUT.PUT_LINE(v1||' '||v2);
```

End;

3. Să se creeze un pachet cu ajutorul căruia, utilizând un cursor și un subprogram funcție, să se obțină salariul maxim înregistrat pentru salariații care lucrează într-un anumit oraș și lista salariaților care au salariul mai mare sau egal decât maximum salariilor din orașul respectiv.

LOCATIONS										
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL
Add Column	Modify Column	Rename Column	Drop Column	Rename	Copy	Drop	Truncate	Create Lookup Table		
Column Name	Data Type	Nullable	Default	Primary Key						
CITY	VARCHAR2(20)	Yes	-	-						
LOCATION_ID	VARCHAR2(20)	Yes	-	-						
COUNTRY_ID	VARCHAR2(20)	Yes	-	-						
POSTAL_CODE	NUMBER(10,0)	Yes	-	-						
STREET_ADDRESS	NUMBER(10,0)	Yes	-	-						

```
CREATE OR REPLACE PACKAGE problema_3 AS  
  CURSOR c_emp(nr NUMBER) RETURN emp%ROWTYPE;  
  FUNCTION sal_max (p_oras locations.city%TYPE) RETURN NUMBER;  
END problema_3;
```

```
create or replace PACKAGE BODY problema_3 AS  
  CURSOR c_emp(nr NUMBER) RETURN emp%ROWTYPE IS  
  SELECT * FROM emp WHERE sal >= nr;  
  FUNCTION sal_max (p_oras locations.city%TYPE) RETURN NUMBER IS  
  maxim NUMBER;  
  BEGIN  
    SELECT MAX(sal)  
    INTO maxim  
    FROM emp e, dept d, locations l  
    WHERE e.deptno=d.deptno AND d.loc=l.location_id  
    AND UPPER(city)=UPPER(p_oras);  
    RETURN maxim;  
  END sal_max;  
END problema_3;
```

```
DECLARE
v_oras locations.city%TYPE:= 'Montreal';
v_max NUMBER;
v_emp emp%ROWTYPE;
BEGIN
v_max:=problema_3.sal_max(v_oras);
OPEN problema_3.c_emp(v_max);
LOOP
  FETCH problema_3.c_emp INTO v_emp;
  EXIT WHEN problema_3.c_emp%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(v_emp.ename||' '||v_emp.sal);
END LOOP;
CLOSE problema_3.c_emp;
END;
```

### Probleme propuse spre rezolvare

1. Să se creeze un pachet prin care să se calculeze, utilizând subprograme supraîncărcate, suma salariilor, salariul maxim, cel mediu și cel minim, suma salariilor pe departamente, respectiv pe job-uri. Codul departamentului sau al job-ului vor fi furnizate ca parametri. Apelați funcțiile utilizând un bloc. Codurile de departament și job vor fi initializate pe linia de declarare.
2. Să se creeze un pachet care conține o funcție ce returnează locația unui angajat al cărui cod este transmis ca parametru și un cursor prin care se obține lista angajaților care lucrează într-o locație al cărei cod este transmis ca parametru. Apelați funcția și cursorul prin intermediul unui bloc.

#### **Bibliografie web:**

<https://www.w3resource.com/>  
<https://www.bullraider.com/database/pl-sql/pl-sql-examples>  
<https://www.oracletutorial.com/plsql-tutorial/plsql-package-specification/>  
<https://www.oracletutorial.com/plsql-tutorial/plsql-package-body/>  
<https://www.oracletutorial.com/plsql-tutorial/plsql-drop-package/>