

Laborator 5 PL/SQL

Pachete

- Pachetele sunt unități de program care pot cuprinde proceduri, funcții, cursoare, tipuri de date, constante, variabile și excepții.
- Pachetele nu pot fi apelate, nu pot transmite parametri și nu pot fi încuibărite.
- Un pachet are două părți, fiecare fiind stocată separat în dicționarul datelor:

- **specificația pachetului;**

```
CREATE [OR REPLACE] PACKAGE [schema.]nume_pachet
{IS | AS}
    declarații;
END [nume_pachet];
```

- **corpul pachetului.**

```
CREATE [OR REPLACE] PACKAGE BODY [schema.]nume_pachet
{IS | AS}
[BEGIN]
    instrucțiuni;
END [nume_pachet];
```

- **Recompilarea pachetului**

```
ALTER PACKAGE [schema.]nume_pachet
COMPILE [ {PACKAGE | BODY} ];
```

- **Eliminarea pachetului**

```
DROP PACKAGE [schema.]nume_pachet
[ {PACKAGE | BODY} ];
```

I. Pachete definite de utilizator

1. Definiți un pachet care permite prin intermediul a două funcții calculul numărului de angajați și suma ce trebuie alocată pentru plata salariilor și a comisioanelor pentru un departament al cărui cod este dat ca parametru.

```
CREATE OR REPLACE PACKAGE pachet1_*** AS
    FUNCTION f_numar(v_dept departments.department_id%TYPE)
        RETURN NUMBER;
    FUNCTION f_suma(v_dept departments.department_id%TYPE)
        RETURN NUMBER;
END pachet1_***;
/
CREATE OR REPLACE PACKAGE BODY pachet1_*** AS
    FUNCTION f_numar(v_dept departments.department_id%TYPE)
        RETURN NUMBER IS numar NUMBER;
    BEGIN
        SELECT COUNT(*) INTO numar
        FROM employees
        WHERE department_id =v_dept;
    RETURN numar;
END f_numar;
```

```

FUNCTION f_suma (v_dept departments.department_id%TYPE)
RETURN NUMBER IS
suma NUMBER;
BEGIN
SELECT SUM(salary+salary*NVL(commission_pct,0))
INTO suma
FROM employees
WHERE department_id =v_dept;
RETURN suma;
END f_suma;
END pachet1_***;
/

```

Apelare:

În SQL:

```

SELECT pachet1_***.f_numar(80)
FROM DUAL;
SELECT pachet1_***.f_suma(80)
FROM DUAL;

```

În PL/SQL:

```

BEGIN
DBMS_OUTPUT.PUT_LINE('numarul de salariatii este '||
pachet1_***.f_numar(80));
DBMS_OUTPUT.PUT_LINE('suma alocata este '||
pachet1_***.f_suma(80));
END;
/

```

2. Creați un pachet ce include acțiuni pentru adăugarea unui nou departament în tabelul *dept_**** și a unui nou angajat (ce va lucra în acest departament) în tabelul *emp_****. Procedurile pachetului vor fi apelate din SQL, respectiv din PL/SQL. Se va verifica dacă managerul departamentului există înregistrat ca salariat. De asemenea, se va verifica dacă locația departamentului există. Pentru inserarea codului salariatului se va utiliza o secvență.

```

CREATE OR REPLACE PACKAGE pachet2_*** AS
PROCEDURE p_dept (v_codd dept_***.department_id%TYPE,
v_nume dept_***.department_name%TYPE,
v_manager dept_***.manager_id%TYPE,
v_loc dept_***.location_id%TYPE);
PROCEDURE p_emp (v_first_name emp_***.first_name%TYPE,
v_last_name emp_***.last_name%TYPE,
v_email emp_***.email%TYPE,
v_phone_number emp_***.phone_number%TYPE:=NULL,
v_hire_date emp_***.hire_date%TYPE :=SYSDATE,
v_job_id emp_***.job_id%TYPE,
v_salary emp_***.salary%TYPE :=0,
v_commission_pct emp_***.commission_pct%TYPE:=0,
v_manager_id emp_***.manager_id%TYPE,
v_department_id emp_***.department_id%TYPE);

```

```
FUNCTION exista (cod_loc dept_***.location_id%TYPE,
                manager dept_***.manager_id%TYPE)
RETURN NUMBER;
END pachet2_***;
/

CREATE OR REPLACE PACKAGE BODY pachet2_*** AS

FUNCTION exista(cod_loc dept_***.location_id%TYPE,
                manager dept_***.manager_id%TYPE)
RETURN NUMBER IS
    rezultat NUMBER:=1;
    rez_cod_loc NUMBER;
    rez_manager NUMBER;
BEGIN
    SELECT count(*) INTO    rez_cod_loc
    FROM    locations
    WHERE   location_id = cod_loc;

    SELECT count(*) INTO    rez_manager
    FROM    emp_***
    WHERE   employee_id = manager;

    IF rez_cod_loc=0 OR rez_manager=0 THEN
        rezultat:=0;
    END IF;
RETURN rezultat;
END;

PROCEDURE p_dept(v_codd dept_***.department_id%TYPE,
                v_numa dept_***.department_name%TYPE,
                v_manager dept_***.manager_id%TYPE,
                v_loc dept_***.location_id%TYPE) IS
BEGIN
    IF exista(v_loc, v_manager)=0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu s-au introdus date coerente pentru
                                tabelul dept_***');
    ELSE
        INSERT INTO dept_***
            (department_id,department_name,manager_id,location_id)
        VALUES (v_codd, v_numa, v_manager, v_loc);
    END IF;
END p_dept;

PROCEDURE p_emp
(v_first_name emp_***.first_name%TYPE,
v_last_name emp_***.last_name%TYPE,
v_email emp_***.email%TYPE,
v_phone_number emp_***.phone_number%TYPE:=null,
v_hire_date emp_***.hire_date%TYPE :=SYSDATE,
v_job_id emp_***.job_id%TYPE,
v_salary emp_***.salary %TYPE :=0,
```

```

        v_commission_pct emp_***.commission_pct%TYPE:=0,
        v_manager_id emp_***.manager_id%TYPE,
        v_department_id emp_***.department_id%TYPE)
AS
BEGIN
    INSERT INTO emp_***
    VALUES (sec_***.NEXTVAL, v_first_name, v_last_name, v_email,
            v_phone_number, v_hire_date, v_job_id, v_salary,
            v_commission_pct, v_manager_id, v_department_id);
END p_emp;
END pachet2_***;
/

```

Apelare:

În SQL:

```

EXECUTE pachet2_***.p_dept(50, 'Economic', 200, 2000);

SELECT * FROM dept_*** WHERE department_id=50;

EXECUTE pachet2_***.p_emp('f', 'l', 'e', v_job_id=>'j',
                        v_manager_id=>200, v_department_id=>50);

SELECT * FROM emp_*** WHERE job_id='j';

ROLLBACK;

```

În PL/SQL:

```

BEGIN
    pachet2_***.p_dept(50, 'Economic', 99, 2000);
    pachet2_***.p_emp('f', 'l', 'e', v_job_id=>'j', v_manager_id=>200,
                    v_department_id=>50);
END;
/

SELECT * FROM emp_*** WHERE job_id='j';
ROLLBACK;

```

3. Definiți un pachet cu ajutorul căruia 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 acel maxim. Pachetul va conține un cursor și un subprogram funcție.

```

CREATE OR REPLACE PACKAGE pachet3_*** AS
    CURSOR c_emp(nr NUMBER) RETURN employees%ROWTYPE;
    FUNCTION f_max (v_oras locations.city%TYPE) RETURN NUMBER;
END pachet3_***;
/

```

```

CREATE OR REPLACE PACKAGE BODY pachet3_*** AS

CURSOR c_emp(nr NUMBER) RETURN employees%ROWTYPE
IS
    SELECT *
    FROM employees
    WHERE salary >= nr;

FUNCTION f_max (v_oras locations.city%TYPE) RETURN NUMBER IS
    maxim NUMBER;
BEGIN
    SELECT MAX(salary)
    INTO maxim
    FROM employees e, departments d, locations l
    WHERE e.department_id=d.department_id
          AND d.location_id=l.location_id
          AND UPPER(city)=UPPER(v_oras);
    RETURN maxim;
END f_max;
END pachet3_***;
/

DECLARE
    oras locations.city%TYPE:= 'Toronto';
    val_max NUMBER;
    lista employees%ROWTYPE;
BEGIN
    val_max:= pachet3_***.f_max(oras);
    FOR v_cursor IN pachet3_***.c_emp(val_max) LOOP
        DBMS_OUTPUT.PUT_LINE(v_cursor.last_name||' '||
                               v_cursor.salary);
    END LOOP;
END;
/

```

4. Definiți un pachet care să conțină o procedură prin care se verifică dacă o combinație specificată dintre câmpurile *employee_id* și *job_id* este o combinație care există în tabelul *employees*.

```

CREATE OR REPLACE PACKAGE pachet4_*** IS
    PROCEDURE p_verific
        (v_cod employees.employee_id%TYPE,
         v_job employees.job_id%TYPE);
    CURSOR c_emp RETURN employees%ROWTYPE;
END pachet4_***;
/

```

```

CREATE OR REPLACE PACKAGE BODY pachet4_*** IS

CURSOR c_emp RETURN employees%ROWTYPE IS
    SELECT *
    FROM employees;

```

```

PROCEDURE p_verific(v_cod    employees.employee_id%TYPE,
                   v_job     employees.job_id%TYPE)
IS
    gasit BOOLEAN:=FALSE;
    lista employees%ROWTYPE;
BEGIN
    OPEN c_emp;
    LOOP
        FETCH c_emp INTO lista;
        EXIT WHEN c_emp%NOTFOUND;
        IF lista.employee_id=v_cod AND lista.job_id=v_job
            THEN gasit:=TRUE;
        END IF;
    END LOOP;
    CLOSE c_emp;
    IF gasit=TRUE THEN
        DBMS_OUTPUT.PUT_LINE('combinatia data exista');
    ELSE
        DBMS_OUTPUT.PUT_LINE('combinatia data nu exista');
    END IF;
END p_verific;
END pachet4_***;
/

EXECUTE pachet4_***.p_verific(200,'AD_ASST');

```

II. Pachete predefinite

1. Pachetul DBMS_OUTPUT permite afișarea de informații. Procedurile pachetului sunt:

- PUT – depune (scrie) în buffer informație;
- PUT_LINE – depune în buffer informația, împreună cu un marcaj de sfârșit de linie;
- NEW_LINE – depune în buffer un marcaj de 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.

Exemplul 1:

```

DECLARE
    -- paramentrii de tip OUT pt procedura GET_LINE
    linie1 VARCHAR2(255);
    stare1 INTEGER;
    linie2 VARCHAR2(255);
    stare2 INTEGER;
    linie3 VARCHAR2(255);
    stare3 INTEGER;

    v_emp employees.employee_id%TYPE;
    v_job employees.job_id%TYPE;
    v_dept employees.department_id%TYPE;

```

```

BEGIN
    SELECT employee_id, job_id, department_id
    INTO    v_emp,v_job,v_dept
    FROM    employees
    WHERE   last_name='Lorentz';

    -- se introduce o linie in buffer fara caracter
    -- de terminare linie
    DBMS_OUTPUT.PUT(' 1 '||v_emp|| ' ');

    -- se incearca extragerea liniei introdusa
    -- in buffer si starea acesteia
    DBMS_OUTPUT.GET_LINE(linie1,stare1);

    -- se depunde informatie pe aceeaasi linie in buffer
    DBMS_OUTPUT.PUT(' 2 '||v_job|| ' ');

    -- se inchide linia depusa in buffer si se extrage
    -- linia din buffer
    DBMS_OUTPUT.NEW_LINE;
    DBMS_OUTPUT.GET_LINE(linie2,stare2);

    -- se introduc informatii pe aceeaasi linie
    -- si se afiseaza informatia
    DBMS_OUTPUT.PUT_LINE(' 3 ' ||v_emp|| ' '|| v_job);
    DBMS_OUTPUT.GET_LINE(linie3,stare3);

    -- se afiseaza ceea ce s-a extras
    DBMS_OUTPUT.PUT_LINE('linie1 = '|| linie1||
                          ' ; stare1 = '||stare1);
    DBMS_OUTPUT.PUT_LINE('linie2 = '|| linie2||
                          ' ; stare2 = '||stare2);
    DBMS_OUTPUT.PUT_LINE('linie3 = '|| linie3||
                          ' ; stare3 = '||stare3);

END;
/

```

Exemplul 2:

```

DECLARE
    -- parametru de tip OUT pentru NEW_LINES
    -- tablou de siruri de caractere
    linii DBMS_OUTPUT.CHARARR;
    -- parametru de tip IN OUT pentru NEW_LINES
    nr_linii INTEGER;

    v_emp employees.employee_id%TYPE;
    v_job employees.job_id%TYPE;
    v_dept employees.department_id%TYPE;

```

```

BEGIN
  SELECT employee_id, job_id, department_id
  INTO    v_emp,v_job,v_dept
  FROM    employees
  WHERE   last_name='Lorentz';

  -- se mareste dimensiunea bufferului
  DBMS_OUTPUT.ENABLE(1000000);
  DBMS_OUTPUT.PUT(' 1 '||v_emp|| ' ');
  DBMS_OUTPUT.PUT(' 2 '||v_job|| ' ');
  DBMS_OUTPUT.NEW_LINE;
  DBMS_OUTPUT.PUT_LINE(' 3 ' ||v_emp|| ' '|| v_job);
  DBMS_OUTPUT.PUT_LINE(' 4 ' ||v_emp|| ' '||
                        v_job||' ' ||v_dept);
  -- se afiseaza ceea ce s-a extras
  nr_linii := 4;
  DBMS_OUTPUT.GET_LINES(linii,nr_linii);
  DBMS_OUTPUT.put_line('In buffer sunt '||
                        nr_linii ||' linii');
  FOR i IN 1..nr_linii LOOP
    DBMS_OUTPUT.put_line(linii(i));
  END LOOP;

  -- nr_linii := 4;
  -- DBMS_OUTPUT.GET_LINES(linii,nr_linii);
  -- DBMS_OUTPUT.put_line('Acum in buffer sunt '||
                          nr_linii ||' linii');
  -- FOR i IN 1..nr_linii LOOP
  --   DBMS_OUTPUT.put_line(linii(i));
  -- END LOOP;
  --
  ---- DBMS_OUTPUT.disable;
  ---- DBMS_OUTPUT.enable;
  ----
  ---- nr_linii := 4;
  ---- DBMS_OUTPUT.GET_LINES(linii,nr_linii);
  ---- DBMS_OUTPUT.put_line('Acum in buffer sunt '||
                          nr_linii ||' linii');

END;
/

```

2. Pachetul DBMS_JOB este utilizat pentru planificarea execuției programelor PL/SQL

SUBMIT – adaugă un nou job în coada de așteptare a job-urilor;

REMOVE – șterge un job din coada de așteptare;

RUN – execută imediat un job specificat.

Exemplu:

```

CREATE OR REPLACE PROCEDURE marire_salariu_***
    (id_angajat emp_***.employee_id%type,
     valoare     number)
IS
BEGIN
    UPDATE emp_***
    SET     salary = salary + valoare
    WHERE  employee_id = id_angajat;
END;
/

```

- INTERVAL este de tip VARCHAR2 DEFAULT 'NULL'
- se verifică trimiterea spre execuție a procedurii (în Enterprise Manager Console → baza de date → Instance → Configuration → All Initialization parameters se setează parametrul JOB_QUEUE_PROCESSES la o valoare mai mare decât 0)

Varianta 1

```

VARIABLE nr_job NUMBER

BEGIN
    DBMS_JOB.SUBMIT(
        -- întoarce numărul jobului, printr-o variabilă de legătură
        JOB => :nr_job,

        -- codul PL/SQL care trebuie executat
        WHAT => 'marire_salariu_***(100, 1000);',

        -- data de start a execuției (dupa 30 secunde)
        NEXT_DATE => SYSDATE+30/86400,

        -- intervalul de timp la care se repetă execuția
        INTERVAL => 'SYSDATE+1');

    COMMIT;
END;
/

```

```

SELECT salary FROM emp_*** WHERE employee_id = 100;
-- asteptati 30 de secunde
SELECT salary FROM emp_*** WHERE employee_id = 100;

-- numarul jobului
PRINT nr_job;

-- informatii despre joburi
SELECT JOB, NEXT_DATE, WHAT
FROM    USER_JOBS;

```

```
-- lansarea jobului la momentul dorit
SELECT salary FROM emp_*** WHERE employee_id = 100;
BEGIN
    -- presupunand ca jobul are codul 1 atunci:
    DBMS_JOB.RUN(job => 1);
END;
/
SELECT salary FROM emp_*** WHERE employee_id = 100;

-- stergerea unui job
BEGIN
    DBMS_JOB.REMOVE(job=>1);
END;
/

SELECT JOB, NEXT_DATE, WHAT
FROM    USER_JOBS;

UPDATE emp_***
SET     salary = 24000
WHERE   employee_id = 100;

COMMIT;
```

Varianta 2

```
CREATE OR REPLACE PACKAGE pachet_job_***
IS
    nr_job NUMBER;
    FUNCTION obtine_job RETURN NUMBER;
END;
/

CREATE OR REPLACE PACKAGE body pachet_job_***
IS
    FUNCTION obtine_job RETURN NUMBER IS
    BEGIN
        RETURN nr_job;
    END;
END;
/
```

```
BEGIN
    DBMS_JOB.SUBMIT(
        -- întoarce numărul jobului, printr-o variabilă de legătură
        JOB => pachet_job_***.nr_job,

        -- codul PL/SQL care trebuie executat
        WHAT => 'marire_salariu_***(100, 1000);',
```

```

-- data de start a execuției (dupa 30 secunde)
NEXT_DATE => SYSDATE+30/86400,

-- intervalul de timp la care se repetă execuția
INTERVAL => 'SYSDATE+1');

COMMIT;
END;
/

-- informatii despre joburi
SELECT JOB, NEXT_DATE, WHAT
FROM   USER_JOBS
WHERE  JOB = pachet_job_***.obține_job;

-- lansarea jobului la momentul dorit
SELECT salary FROM emp_*** WHERE employee_id = 100;
BEGIN
    DBMS_JOB.RUN(JOB => pachet_job_***.obține_job);
END;
/
SELECT salary FROM emp_*** WHERE employee_id = 100;

-- stergerea unui job
BEGIN
    DBMS_JOB.REMOVE(JOB=>pachet_job_***.obține_job);
END;
/
SELECT JOB, NEXT_DATE, WHAT
FROM   USER_JOBS
WHERE  JOB = pachet_job_***.obține_job;

UPDATE emp_***
SET    salary = 24000
WHERE  employee_id = 100;

COMMIT;

```

3. Pachetul UTL_FILE extinde operațiile I/O la fișiere. Se apelează funcția FOPEN pentru a deschide un fișier; acesta este folosit pentru operațiile de citire sau scriere. După ce s-au încheiat operațiile I/O se închide fișierul (FCLOSE).

Observație: în Enterprise Manager Console → baza de date → Instance → Configuration → All Initialization parameters se setează parametrul UTL_FILE_DIR la o valoare care reprezintă directorul unde se face citirea/scrierea (de exemplu F:). Aceasta operație va cere oprirea bazei de date și repornirea ei.

Exemplu:

Menținem rezultatele unei comenzi SELECT într-un fișier.

```

CREATE OR REPLACE PROCEDURE scriu_fisier_***
(director VARCHAR2,
 fisier VARCHAR2)
IS
  v_file UTL_FILE.FILE_TYPE;
  CURSOR cursor_rez IS
    SELECT department_id departament, SUM(salary) suma
    FROM employees
    GROUP BY department_id
    ORDER BY SUM(salary);
  v_rez cursor_rez%ROWTYPE;
BEGIN
  v_file:=UTL_FILE.FOPEN(director, fisier, 'w');
  UTL_FILE.PUTF(v_file, 'Suma salariilor pe departamente \n Raport
    generat pe data ');
  UTL_FILE.PUT(v_file, SYSDATE);
  UTL_FILE.NEW_LINE(v_file);
  OPEN cursor_rez;
  LOOP
    FETCH cursor_rez INTO v_rez;
    EXIT WHEN cursor_rez%NOTFOUND;
    UTL_FILE.NEW_LINE(v_file);
    UTL_FILE.PUT(v_file, v_rez.departament);
    UTL_FILE.PUT(v_file, ' ');
    UTL_FILE.PUT(v_file, v_rez.suma);
  END LOOP;
  CLOSE cursor_rez;
  UTL_FILE.FCLOSE(v_file);
END;
/

SQL> EXECUTE scriu_fisier('F:\', 'test.txt');

```

EXERCITII

E1. Definiți un pachet care să permită gestiunea angajaților companiei. Pachetul va conține:

- a. o procedură care determină adăugarea unui angajat, dându-se informații complete despre acesta:
 - codul angajatului va fi generat automat utilizându-se o secvență;
 - informațiile personale vor fi date ca parametrii (nume, prenume, telefon, email);
 - data angajării va fi data curentă;
 - salariul va fi cel mai mic salariu din departamentul respectiv, pentru jobul respectiv (se vor obține cu ajutorul unei funcții stocate în pachet);
 - nu va avea comision;
 - codul managerului se va obține cu ajutorul unei funcții stocate în pachet care va avea ca parametrii numele și prenumele managerului;
 - codul departamentului va fi obținut cu ajutorul unei funcții stocate în pachet, dându-se ca parametru numele acestuia;
 - codul jobului va fi obținut cu ajutorul unei funcții stocate în pachet, dându-se ca parametru numele acesteia.

Observație: Tratați toate excepțiile.

- b.** o procedură care determină mutarea în alt departament a unui angajat (se dau ca parametrii numele și prenumele angajatului, respectiv numele departamentului, numele jobului și numele și prenumele managerului acestuia):
- se vor actualiza informațiile angajatului:
 - codul de departament (se va obține cu ajutorul funcției corespunzătoare definită la punctul a);
 - codul jobului (se va obține cu ajutorul funcției corespunzătoare definită la punctul a);
 - codul managerului (se va obține cu ajutorul funcției corespunzătoare definită la punctul a);
 - salariul va fi cel mai mic salariu din noul departament, pentru noul job dacă acesta este mai mare decât salariul curent; altfel se va păstra salariul curent;
 - comisionul va fi cel mai mic comision din acel departament, pentru acel job;
 - data angajării va fi data curentă;
 - se vor înregistra informații corespunzătoare în istoricul joburilor.

Observație: Tratați toate excepțiile.

- c.** o funcție care întoarce numărul de subalterni direcți sau indirecti ai unui angajat al cărui nume și prenume sunt date ca parametri;

Observație: Tratați toate excepțiile.

- d.** o procedură care determină promovarea unui angajat pe o treaptă imediat superioară în departamentul său; propuneți o variantă de restructurare a arborelui care implementează ierarhia subaltern – șef din companie;

Observație: Tratați toate excepțiile.

- e.** o procedură prin care se actualizează cu o valoare dată ca parametru salariul unui angajat al cărui nume este dat ca parametru:
- se va verifica dacă valoarea dată pentru salariu respectă limitele impuse pentru acel job;
 - dacă sunt mai mulți angajați care au același nume, atunci se va afișa un mesaj corespunzător și de asemenea se va afișa lista acestora;
 - dacă nu există angajați cu numele dat, atunci se va afișa un mesaj corespunzător;
- f.** un cursor care obține lista angajaților care lucrează pe un job al cărui cod este dat ca parametru;
- g.** un cursor care obține lista tuturor joburilor din companie;
- h.** o procedură care utilizează cele două cursoare definite anterior și obține pentru fiecare job numele acestuia și lista angajaților care lucrează în prezent pe acel job; în plus, pentru fiecare angajat să se specifice dacă în trecut a mai avut sau nu jobul respectiv.

E2. Adaptați cerința exercițiului 3 din partea I (pachete definite de utilizator) pentru diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați acest exercițiu în PL/SQL, folosind baza de date proprie.