

Laborator 4 PL/SQL

[Subprograme PL/SQL \(funcții și proceduri\)](#)

[Exerciții:](#)

[I. \[Proceduri locale\]](#)

[II. \[Proceduri stocate\]](#)

[III. \[Funcții locale\]](#)

[IV. \[Funcții stocate\]](#)

Subprograme PL/SQL (funcții și proceduri)

Un subprogram este un bloc PL/SQL cu nume (spre deosebire de blocurile anonime) care poate primi parametri și poate fi invocat dintr-un anumit mediu (de exemplu, SQL*Plus, Oracle Forms, Oracle Reports etc.)

Subprogramele sunt bazate pe structura de bloc PL/SQL. Similar, ele conțin o parte declarativă facultativă, o parte executabilă obligatorie și o parte de tratare de excepții facultativă.

- Exista 2 tipuri de subprograme:
 - proceduri;
 - funcții (trebuie să conțină cel puțin o comandă RETURN);
- Subprogramele pot fi:
 - locale (în cadrul altui bloc PL/SQL sau subprogram)
 - stocate (create cu comanda CREATE) - odată create, procedurile și funcțiile sunt stocate în baza de date de aceea ele se numesc subprograme stocate.

Sintaxa simplificată pentru crearea unei proceduri este următoarea:

```
[CREATE [OR REPLACE] ] PROCEDURE nume_procedură [ (lista_parametri) ]
    {IS | AS}
    [declarații locale]
BEGIN
    partea_executabilă
[EXCEPTION
    partea_de_tratare_a_excepțiilor]
END [nume_procedură];
```

Sintaxa simplificată pentru crearea unei funcții este următoarea:

```
[CREATE [OR REPLACE] ] FUNCTION nume_funcție
    RETURN tip_de_date
    {IS | AS}
    [declarații locale]
BEGIN
    partea_executabilă
[EXCEPTION
```

Laborator 4 PL/SQL

```
    partea de tratare a excepiilor]  
END [nume_funcție];
```

- Lista de parametri conține specificații de parametri separate prin virgulă de forma:
nume_parametru mod_parametru tip_parametru;
 - mod_parametru specifică dacă parametrul este:
 - de intrare (IN) – singurul care poate avea o valoare inițială
 - de intrare/ieșire (IN OUT)
 - de ieșire (OUT)
 - mod_parametru are valoarea implicită IN.
- O funcție îndeplinește următoarele condiții:
 - Acceptă numai parametri de tip IN
 - Acceptă numai tipuri de date SQL, nu și tipuri specifice PL/SQL
 - Returnează valori de tipuri de date SQL
 - Nu modifică tabelul care este blocat pentru comanda respectivă (mutating tables)
 - Poate fi folosită în lista de expresii a comenzii SELECT, clauza WHERE și HAVING, CONNECT BY, START WITH, ORDER BY, GROUP BY, clauza VALUES a comenzii INSERT, clauza SET a comenzii UPDATE.
- În cazul în care se modifică un obiect (vizualizare, tabel etc) de care depinde un subprogram, acesta este invalidat. Revalidarea se face fie prin recrearea subprogramului fie prin comanda:
ALTER PROCEDURE nume_proc COMPILE;
ALTER FUNCTION nume_functie COMPILE;
- Ștergerea unei funcții sau proceduri se realizează prin comenzile:
DROP PROCEDURE nume_proc;
DROP FUNCTION nume_functie;
Informații despre procedurile și funcțiile deținute de utilizatorul curent se pot obține interogând vizualizarea USER_OBJECTS din dicționarul datelor.
SELECT OBJECT_NAME, OBJECT_TYPE, STATUS
FROM USER_OBJECTS
WHERE OBJECT_TYPE IN ('PROCEDURE', 'FUNCTION');
- **Obs:** STATUS – starea subprogramului (valid sau invalid).
- Codul complet al unui subprogram poate fi vizualizat folosind următoarea sintaxă:
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = 'nume_subprogram'
ORDER BY LINE;
- Eroarea apărută la compilarea unui subprogram poate fi vizualizată folosind următoarea sintaxă:
SELECT LINE, POSITION, TEXT
FROM USER_ERRORS
WHERE NAME = 'nume';
- Erorile pot fi vizualizate și prin intermediul comenzii SHOW ERRORS.
- Descrierea specificației unui subprogram se face prin comanda DESCRIBE.

Laborator 4 PL/SQL

Când este apelată o procedură PL/SQL, sistemul Oracle furnizează două metode pentru definirea parametrilor actuali:

- specificarea explicită prin nume;
- specificarea prin poziție.

Exemplu: subprog(a tip_a, b_tip_b, c tip_c, d tip_d)

- specificare prin poziție: subprog(var_a,var_b,var_c,var_d);
- specificare prin nume subprog(b=>var_b,c=>var_c,d=>var_d,a=>var_a);
- specificare prin nume și poziție subprog(var_a,var_b,d=>var_d,c=>var_c);

Exerciții:

I. [Proceduri locale]

1. Să se declare o procedură locală într-un bloc PL/SQL anonim prin care să se introducă în tabelul DEPT_pnu o nouă înregistrare precizând, prin intermediul parametrilor, valori pentru toate câmpurile. Invocați procedura în cadrul blocului. Interogați tabelul DEPT_pnu și apoi anulați modificările (ROLLBACK).

```
ALTER TABLE dept_pnu
DROP COLUMN info;

DECLARE
    PROCEDURE add_dept
        (p_cod dept_pnu.department_id %TYPE,
         p_nume dept_pnu.department_name %TYPE,
         p_manager dept_pnu.manager_id %TYPE,
         p_location dept_pnu.location_id%TYPE)
    IS
    BEGIN
        INSERT INTO dept_pnu
        VALUES (p_cod, p_nume, p_manager,p_location);
    END;
BEGIN
    add_dept(45, 'DB Administration' , 100, 2700);
END;
/

SELECT *
FROM dept_pnu;

ROLLBACK;
```

2. Să se declare o procedură locală care are următorii parametri:

- p_rezultat (parametru de tip IN OUT) de tipul coloanei last_name din tabelul employees;
- p_comision (parametru de tip OUT) de tipul coloanei commission_pct din employees, inițializat cu NULL;
- p_cod (parametru de tip IN) de tipul coloanei employee_id din employees, initializat cu NULL.

Laborator 4 PL/SQL

Dacă p_comision nu este NULL atunci în p_rezultat se va memora numele salariatului care are salariul maxim printre salariații având comisionul respectiv. În caz contrar, în p_rezultat se va memora numele salariatului al cărui cod are valoarea dată la apelarea procedurii.

```
SET SERVEROUTPUT ON
DECLARE
    v_ume employees.last_name%TYPE;
    PROCEDURE p214_pnu (p_rezultat IN OUT employees.last_name% TYPE,
                        p_comision employees.commission_pct
%TYPE:=NULL,
                        p_cod IN employees.employee_id
%TYPE:=NULL)
    IS
    BEGIN
        IF (p_comision IS NOT NULL) THEN
            SELECT last_name
            INTO p_rezultat
            FROM employees
            WHERE commission_pct = p_comision
            AND salary = (SELECT MAX(salary)
                        FROM employees
                        WHERE commission_pct = p_comision);
            DBMS_OUTPUT.PUT_LINE('Numele salariatului care are
comisionul '||p_comision||' este '||p_rezultat);
        ELSE
            SELECT last_name
            INTO p_rezultat
            FROM employees
            WHERE employee_id = p_cod;
            DBMS_OUTPUT.PUT_LINE('numele salariatului avand codul
'||p_cod||' este '||p_rezultat);
        END IF;
    END;
BEGIN
    p214_pnu (v_ume,0.4);
    p214_pnu (v_ume,p_cod=>205);
END;
/
```

II. [Proceduri stocate]

3. Să se creeze o procedură stocată fără parametri care afișează un mesaj 'Programare PL/SQL', ziua de astăzi în formatul DD-MONTH-YYYY și ora curentă, precum și ziua de ieri în formatul DD-MON-YYYY.

```
CREATE PROCEDURE first_pnu
IS
    azi DATE := SYSDATE;
    ieri azi%TYPE;
BEGIN
```

Laborator 4 PL/SQL

```
DBMS_OUTPUT.PUT_LINE('Programare PL/SQL') ;
DBMS_OUTPUT.PUT_LINE(TO_CHAR(azi, 'dd-month-yyyy hh24:mi:ss'));
ieri := azi -1;
DBMS_OUTPUT.PUT_LINE(TO_CHAR(ieri, 'dd-mon-yyyy'));
END;
/
EXECUTE first_pnu; --apelare procedura
```

4. Să se șteargă procedura precedentă și să se re-creeze, astfel încât să accepte un parametru IN de tip VARCHAR2, numit p_nume. Mesajul afișat de procedură va avea forma « <p_nume> învață PL/SQL». Invocați procedura cu numele utilizatorului curent furnizat ca parametru.

```
DROP PROCEDURE first_pnu;
CREATE PROCEDURE first_pnu(p_nume VARCHAR2)
IS
...
END;
/
EXECUTE first_pnu(USER);
```

5. a) Creați o copie JOBS_pnu a tabelului JOBS. Implementați constrângerea de cheie primară asupra lui JOBS_pnu.

```
CREATE TABLE jobs_pnu AS SELECT * FROM jobs ;
ALTER TABLE jobs_pnu ADD CONSTRAINT pk_jobs_pnu PRIMARY KEY(job_id);
```

b) Creați o procedură ADD_JOB_pnu care inserează un nou job în tabelul JOBS_pnu. Procedura va avea 2 parametri IN p_id și p_title corespunzători codului și denumirii noului job.

```
CREATE OR REPLACE PROCEDURE add_job_pnu
(p_job_id IN jobs.job_id%TYPE, p_job_title IN jobs.job_title%TYPE)
IS BEGIN
    INSERT INTO jobs_pnu (job_id, job_title)
    VALUES (p_job_id, p_job_title);
    COMMIT;
END add_job_pnu;
/
```

c) Testați procedura, invocând-o astfel:

```
EXECUTE ADD_JOB_pnu('IT_DBA', 'Database Administrator');
SELECT * FROM JOBS_pnu;
EXECUTE ADD_JOB_pnu('ST_MAN', 'Stock Manager');
SELECT * FROM JOBS_pnu;
```

Laborator 4 PL/SQL

6. a) Creați o procedură stocată numită UPD_JOB_pnu pentru modificarea unui job existent în tabelul JOBS_pnu. Procedura va avea ca parametri codul job-ului și noua sa denumire (parametri IN). Se va trata cazul în care nu are loc nici o actualizare.

```
CREATE OR REPLACE PROCEDURE upd_job_pnu
(p_job_id IN jobs.job_id%TYPE, p_job_title IN jobs.job_title%TYPE)
IS
BEGIN
    UPDATE jobs_pnu
    SET job_title = p_job_title
    WHERE job_id = p_job_id;
    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20202, 'Nici o actualizare');
        -- sau doar cu afisare mesaj
        -- DBMS_OUTPUT.PUT_LINE('Nici o actualizare');
    END IF;
END upd_job_pnu;
/
```

b) Testați procedura, invocând-o astfel:

```
EXECUTE upd_job_pnu('IT_DBA', 'Data Administrator');
SELECT * FROM jobs_pnu
WHERE UPPER(job_id) = 'IT_DBA';
EXECUTE upd_job_pnu('IT_WEB', 'Web master');
```

Obs: A doua invocare va conduce la apariția excepției. Analizați ce s-ar fi întâmplat dacă nu prevedeam această excepție, punând între comentarii liniile aferente din procedură și recreând-o cu CREATE OR REPLACE PROCEDURE...

7. a) Creați o procedură stocată numită DEL_JOB_pnu care șterge un job din tabelul JOBS_pnu. Procedura va avea ca parametru (IN) codul job-ului. Includeți o excepție corespunzătoare situației în care nici un job nu este șters.

b) Testați procedura, invocând-o astfel:

```
EXECUTE DEL_JOB_pnu('IT_DBA');
EXECUTE DEL_JOB_pnu('IT_WEB');
```

8. a) Să se creeze o procedură stocată care calculează salariul mediu al angajaților, returnându-l prin intermediul unui parametru de tip OUT.

b) Să se apeleze procedura regăsind valoarea medie a salariilor într-o variabilă gazdă. Afișați valoarea variabilei.

```
CREATE OR REPLACE PROCEDURE p814_pnu (p_salAvg OUT employees.salary%TYPE)
AS
BEGIN
```

Laborator 4 PL/SQL

```
SELECT AVG(salary)
  INTO p_salAvg
  FROM employees;
END;
/

VARIABLE g_medie NUMBER
EXECUTE p8l4_pnu (:g_medie)
PRINT g_medie
```

9. a) Să se creeze o procedură stocată care primește printr-un parametru salariul unui angajat și returnează prin intermediul aceluiași parametru salariul actualizat astfel: dacă salariul este mai mic decât 3000, valoarea lui crește cu 20%, dacă este cuprins între 3000 și 7000 valoarea lui crește cu 15%, dacă este mai mare decât 7000 va fi mărit cu 10%, iar dacă este null va lua valoarea 1000.

b) Să se declare o variabilă gazdă g_sal (VARIABLE). Să se scrie un bloc anonim PL/SQL prin care se va atribui variabilei g_sal valoarea unei variabile de substituție citite de la tastatură (ACCEPT). Să se apeleze procedura pentru această valoare și să se afișeze valoarea returnată.

```
CREATE OR REPLACE PROCEDURE p9l4_pnu (p_sal IN OUT NUMBER)
IS
BEGIN
  CASE
    WHEN p_sal < 3000 THEN
      p_sal := p_sal * 1.2;
    WHEN p_sal BETWEEN 3000 AND 7000 THEN
      p_sal := p_sal * 1.15;
    WHEN p_sal > 7000 THEN
      p_sal := p_sal * 1.1;
    ELSE p_sal := 1000;
  END CASE;
END;
/

VARIABLE g_sal NUMBER
SET VERIFY OFF
ACCEPT p_sal PROMPT 'Introduceti salariul '
BEGIN
  :g_sal:=&p_sal;
END;
/
PRINT g_sal
EXECUTE p9l4_pnu (:g_sal)
PRINT g_sal
```

III. [Funcții locale]

10. Să se creeze o procedură stocată care pentru un anumit cod de departament (dat ca parametru) calculează prin intermediul unor funcții locale numărul de salariați care lucrează în el, suma salariilor și numărul managerilor salariaților care lucrează în departamentul respectiv.

```
CREATE OR REPLACE PROCEDURE p11l4_pnu
(p_dept employees.department_id%TYPE)
AS
    FUNCTION nrSal (v_dept employees.department_id %TYPE)
        RETURN NUMBER IS
            v_numar NUMBER(3);
        BEGIN
            SELECT COUNT(*)
            INTO v_numar
            FROM employees
            WHERE department_id = v_dept;
            RETURN v_numar;
        END nrSal;
    FUNCTION sumaSal(v_dept employees.department_id %TYPE)
        RETURN NUMBER IS
            v_suma employees.salary%TYPE;
        BEGIN
            SELECT SUM(salary)
            INTO v_suma
            FROM employees
            WHERE department_id = v_dept;
            RETURN v_suma;
        END sumaSal;
    FUNCTION nrMgr(v_dept employees.department_id %TYPE)
        RETURN NUMBER IS
            v_numar NUMBER(3);
        BEGIN
            SELECT COUNT(DISTINCT manager_id)
            INTO v_numar
            FROM employees
            WHERE department_id = v_dept;
            RETURN v_numar;
        END nrMgr;
BEGIN -- partea executabila a procedurii p11l4
    DBMS_OUTPUT.PUT_LINE('Numarul salariatilor care lucreaza in departamentul
'||p_dept|| ' este
'|| nrSal(p_dept));
    DBMS_OUTPUT.PUT_LINE('Suma salariilor angajatilor din departamentul '||
p_dept || ' este '||
    sumaSal(p_dept));
    DBMS_OUTPUT.PUT_LINE('Numarul de manageri din departamentul '|| p_dept ||
' este '||
    nrMgr(p_dept));
END;
/
```


Laborator 4 PL/SQL

```
EXECUTE p1114_pnu(50);
```

11. Să se creeze două funcții (locale) supraîncărcate (overload) care să calculeze media salariilor astfel:

- prima funcție va avea ca argument codul departamentului, adică funcția calculează media salariilor din departamentul specificat;
- a doua funcție va avea două argumente, unul reprezentând codul departamentului, iar celălalt reprezentând job-ul, adică funcția va calcula media salariilor dintr-un anumit departament și care aparțin unui job specificat.

```
DECLARE
    medie1 NUMBER(10,2);
    medie2 NUMBER(10,2);
    FUNCTION medie (v_dept employees.department_id%TYPE)
        RETURN NUMBER IS
        rezultat NUMBER(10,2);
    BEGIN
        SELECT AVG(salary)
        INTO rezultat
        FROM employees
        WHERE department_id = v_dept;
        RETURN rezultat;
    END;
    FUNCTION medie (v_dept employees.department_id%TYPE,
                    v_job employees.job_id %TYPE)
        RETURN NUMBER IS
        rezultat NUMBER(10,2);
    BEGIN
        SELECT AVG(salary)
        INTO rezultat
        FROM employees
        WHERE department_id = v_dept AND job_id = v_job;
        RETURN rezultat;
    END;
BEGIN
    medie1:=medie(80);
    DBMS_OUTPUT.PUT_LINE('Media salariilor din departamentul 80 este ' || medie1);
    medie2 := medie(80, 'SA_REP');
    DBMS_OUTPUT.PUT_LINE('Media salariilor reprezentantilor de vanzari din
    departamentul 80 este ' || medie2);
END;
/
```

IV. [Funcții stocate]

12. Să se creeze o funcție stocată care determină numărul de salariați din employees angajați după 1995, într-un departament dat ca parametru. Să se apeleze această funcție prin diferite modalități:

- a) printr-o variabilă de legătură;

Laborator 4 PL/SQL

- b) folosind comanda CALL;
- c) printr-o comandă SELECT;
- d) într-un bloc PL/SQL.

```
CREATE OR REPLACE FUNCTION p14l4_pnu (p_dept employees.department_id%TYPE)
RETURN NUMBER IS
    rezultat NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO rezultat
    FROM employees
    WHERE department_id = p_dept
    AND TO_CHAR(hire_date,'yyyy') > 1995;
    RETURN rezultat;
END p14l4_pnu;
/
--a)
VARIABLE nr NUMBER
EXECUTE :nr := p14l4_pnu (80);
PRINT nr
--b)
VARIABLE nr NUMBER
CALL p14l4_pnu (50) INTO :nr;
PRINT nr
--c)
SELECT p14l4_pnu (80)
FROM dual;
--d)
UNDEFINE p_dep
DEFINE p_dep = 50
-- sau ACCEPT p_dep PROMPT 'Introduceti codul departamentului '
DECLARE
    nr NUMBER;
    v_dep employees.department_id%TYPE := &p_dep;
BEGIN
    nr := p14l4_pnu (v_dep);
    IF nr <> 0 THEN
        DBMS_OUTPUT.PUT_LINE('numarul salariatilor angajati dupa 1995 in
departamentul '||v_dep||' este '||nr);
    ELSE
        DBMS_OUTPUT.PUT_LINE('departamentul cu numarul '|| v_dep || ' nu are
angajati');
    END IF;
END;
/
```

13. a) Creați o funcție numită VALID_DEPTID_pnu pentru validarea unui cod de departament specificat ca parametru. Funcția va întoarce o valoare booleană (TRUE dacă departamentul există).

Laborator 4 PL/SQL

b) Creați o procedură numită ADD_EMP_pnu care adaugă un angajat în tabelul EMP_pnu. Linia respectivă va fi adăugată în tabel doar dacă departamentul specificat este valid, altfel utilizatorul va primi un mesaj adecvat.

Procedura va avea următorii parametri, cu valorile DEFAULT specificate între paranteze: first_name, last_name, email, job_id (SA_REP), manager_id (145), salary (1000), commission_pct (0), department_id (30).

Pentru codul angajatului se va utiliza o secvență EMP_SEQ_pnu, iar data angajării se consideră a fi TRUNC(SYSDATE).

c) Testați procedura, adăugând un angajat pentru care se specifică numele, prenumele, codul departamentului = 15, iar restul parametrilor se lasă DEFAULT.

Adăugați un angajat pentru care se specifică numele, prenumele, codul departamentului = 80, iar restul parametrilor rămân la valorile DEFAULT.

Adăugați un angajat precizând valori pentru toți parametrii procedurii.

```
CREATE SEQUENCE emp_seq_pnu
START WITH 207
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE OR REPLACE FUNCTION valid_deptid_pnu
(p_deptid IN departments.department_id%TYPE)
RETURN boolean IS
v_aux VARCHAR2(1);
BEGIN
SELECT 'x'
INTO v_aux
FROM departments
WHERE department_id = p_deptid;
RETURN (TRUE);
EXCEPTION
WHEN NO_DATA_FOUND
THEN RETURN(FALSE);
END valid_deptid_pnu;
/

CREATE OR REPLACE PROCEDURE add_emp_pnu
(p_lname employees.last_name%TYPE,
p_fname employees.first_name%TYPE,
p_email employees.email%TYPE,
p_job employees.job_id%TYPE DEFAULT 'SA_REP',
p_mgr employees.manager_id%TYPE DEFAULT 145,
p_sal employees.salary%TYPE DEFAULT 1000,
p_comm employees.commission_pct%TYPE DEFAULT 0,
p_deptid employees.department_id%TYPE DEFAULT 30)
```

Laborator 4 PL/SQL

```
IS
BEGIN
    IF valid_deptid_pnu(p_deptid) THEN
        INSERT INTO emp_pnu
            VALUES (emp_seq_pnu.NEXTVAL, p_fname, p_lname, p_email, NULL,
                TRUNC(SYSDATE), p_job, p_sal, p_comm, p_mgr, p_deptid);
    ELSE
        RAISE_APPLICATION_ERROR(-20204, 'Cod invalid de departament.');
```

END IF;

END add_emp_pnu;

/

EXECUTE add_emp_pnu(p_lname=>'Harris', p_fname=>'Jane', p_email=>'JHarris',
p_deptid=>15);

EXECUTE add_emp_pnu(p_lname=>'Harris', p_fname=>'Joe', p_email=>'JoHarris',
p_deptid=>80);

SELECT *

FROM emp_pnu

WHERE department_id = 80;

14. Să se calculeze recursiv numărul de permutări ale unei mulțimi cu n elemente, unde n va fi transmis ca parametru.

```
CREATE OR REPLACE FUNCTION nr_permutari_pnu(n NUMBER)
    RETURN INTEGER IS
BEGIN
    IF (n = 0) THEN
        RETURN 1;
    ELSE
        RETURN n * nr_permutari_pnu (n-1);
    END IF;
END nr_permutari_pnu;
/

VARIABLE g_n NUMBER
EXECUTE :g_n := nr_permutari_pnu (5);
PRINT g_n

SELECT nr_permutari_pnu(5)
FROM dual;
```

15. Să se afișeze numele, job-ul și salariul angajaților al căror salariu este mai mare decât media salariilor din tabelul employees.

```
CREATE OR REPLACE FUNCTION medie_pnu
    RETURN NUMBER IS
    medie NUMBER;
BEGIN
    SELECT AVG(salary)
```

Laborator 4 PL/SQL

```
INTO medie  
FROM employees;  
RETURN medie;  
END;  
/  
  
SELECT last_name, job_id, salary  
FROM employees  
WHERE salary >= medie_pnu;
```