

Laborator 9: SUBPROGRAME in PL/SQL (partea II - functii)

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 opțională, o parte executabilă obligatorie și o parte de tratare de excepții opțională. Exista 2 tipuri de subprograme:

1. **proceduri**;
2. **funcții** (trebuie să conțină cel puțin o comandă RETURN);

Subprogramele pot fi:

- o **locale** (în cadrul unui alt bloc PL/SQL sau subprogram)
- o **stocate** (create cu comanda CREATE) - odată create, procedurile și funcțiile sunt stocate în baza de date, motiv pentru care se numesc subprograme stocate.

Sintaxa simplificată pentru crearea unei funcții este următoarea:

```
[CREATE [OR REPLACE] ] FUNCTION nume_functie [ (lista_parametri) ]  
  RETURN tip_de_date  
  {IS | AS}  
  [declarații locale]  
BEGIN  
  partea_executabilă  
[EXCEPTION  
  partea_de_tratare_a_excepțiilor]  
END [nume_functie];
```

Lista de parametri conține specificații de parametri separate prin virgulă de forma:

nume_parametru mod_parametru tip_parametru;

O functie 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 respectiva (mutating tables)
- ✓ Poate fi folosită în lista de expresii a comenzii SELECT, în clauzele 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 FUNCTION nume_functie COMPILE;

Ștergerea unei funcții se realizează prin comenzile:

```
DROP PROCEDURE nume_proc;  
DROP FUNCTION nume_funcție;
```

Informații despre 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 ('FUNCTION');
```

Obs: STATUS – starea subprogramului (validă sau invalidă).

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.

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);**

Probleme rezolvate

a). Funcții locale

1. 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.

Soluție:

```
CREATE OR REPLACE PROCEDURE problema1 (p_dept emp.deptno%TYPE) AS  
FUNCTION nrSal (v_dept emp.deptno %TYPE)  
RETURN NUMBER IS  
    v_numar NUMBER(3);  
BEGIN  
    SELECT COUNT(*)  
    INTO v_numar  
    FROM emp  
    WHERE deptno = v_dept;  
    RETURN v_numar;  
END nrSal;
```

```
FUNCTION sumaSal(v_dept emp.deptno %TYPE)  
RETURN NUMBER IS  
    v_suma emp.sal%TYPE;  
BEGIN  
    SELECT SUM(sal)  
    INTO v_suma  
    FROM emp  
    WHERE deptno = v_dept;  
    RETURN v_suma;  
END sumaSal;
```

```
FUNCTION nrMgr(v_dept emp.deptno %TYPE)  
RETURN NUMBER IS  
    v_numar NUMBER(3);  
BEGIN  
    SELECT COUNT(DISTINCT mgr)  
    INTO v_numar  
    FROM emp  
    WHERE deptno = v_dept;  
    RETURN v_numar;  
END nrMgr;
```

```
BEGIN -- partea executabila a procedurii problema1_pnu  
    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;
```

The screenshot shows the Oracle APEX SQL Workshop interface. The 'SQL Commands' pane contains the following code:

```

27 BEGIN
28     SELECT COUNT(DISTINCT mgr)
29     INTO   v_numar
30     FROM   emp
31     WHERE  deptno = v_dept;
32     RETURN v_numar;
33 END nrMgr;
34 BEGIN -- partea executabila a procedurii problema1
35     DBMS_OUTPUT.PUT_LINE('Numarul salariatilor care lucreaza in departamentul '||p_dept|| ' este '|| nrSal(p_dept));
36     DBMS_OUTPUT.PUT_LINE('Suma salariilor angajatilor din departamentul '|| p_dept || ' este '|| sumaSal(p_dept));
37     DBMS_OUTPUT.PUT_LINE('Numarul de manageri din departamentul '|| p_dept || ' este '|| nrMgr(p_dept));
38 END;
39

```

The 'Results' pane at the bottom shows the message: "Procedure created."

The screenshot shows the Oracle APEX SQL Workshop interface with the 'Code' tab selected. The title bar indicates 'PROBLEMA1'. The code editor contains the following PL/SQL code:

```

1  create or replace PROCEDURE problema1 (p_dept emp.deptno%TYPE) AS
2  FUNCTION nrSal (v_dept emp.deptno %TYPE)
3  RETURN NUMBER IS
4  | v_numar NUMBER(3);
5  BEGIN
6  | SELECT COUNT(*)
7  | INTO   v_numar
8  | FROM   emp
9  | WHERE  deptno = v_dept;
10 | RETURN v_numar;
11 END nrSal;
12 FUNCTION sumaSal(v_dept emp.deptno %TYPE)
13 RETURN NUMBER IS
14 | v_suma emp.sal%TYPE;
15 BEGIN
16 | SELECT SUM(sal)
17 | INTO   v_suma
18 | FROM   emp
19 | WHERE  deptno = v_dept;
20 | RETURN v_suma;
21 END sumaSal;
22 FUNCTION nrMgr(v_dept emp.deptno %TYPE)
23 | RETURN NUMBER IS

```

```

22 FUNCTION nrMgr(v_dept emp.deptno %TYPE)
23 RETURN NUMBER IS
24     v_numar NUMBER(3);
25 BEGIN
26     SELECT COUNT(DISTINCT mgr)
27     INTO   v_numar
28     FROM   emp
29     WHERE  deptno = v_dept;
30     RETURN v_numar;
31 END nrMgr;
32 BEGIN -- partea executabila a procedurii problema1_pnu
33 DBMS_OUTPUT.PUT_LINE('Numarul salariatilor care lucreaza in departamentul '||p_dept|| ' este '|| nrSal(p_dept));
34 DBMS_OUTPUT.PUT_LINE('Suma salariilor angajatilor din departamentul '|| p_dept || ' este '|| sumaSal(p_dept));
35 DBMS_OUTPUT.PUT_LINE('Numarul de manageri din departamentul '|| p_dept || ' este '|| nrMgr(p_dept));
36 END;

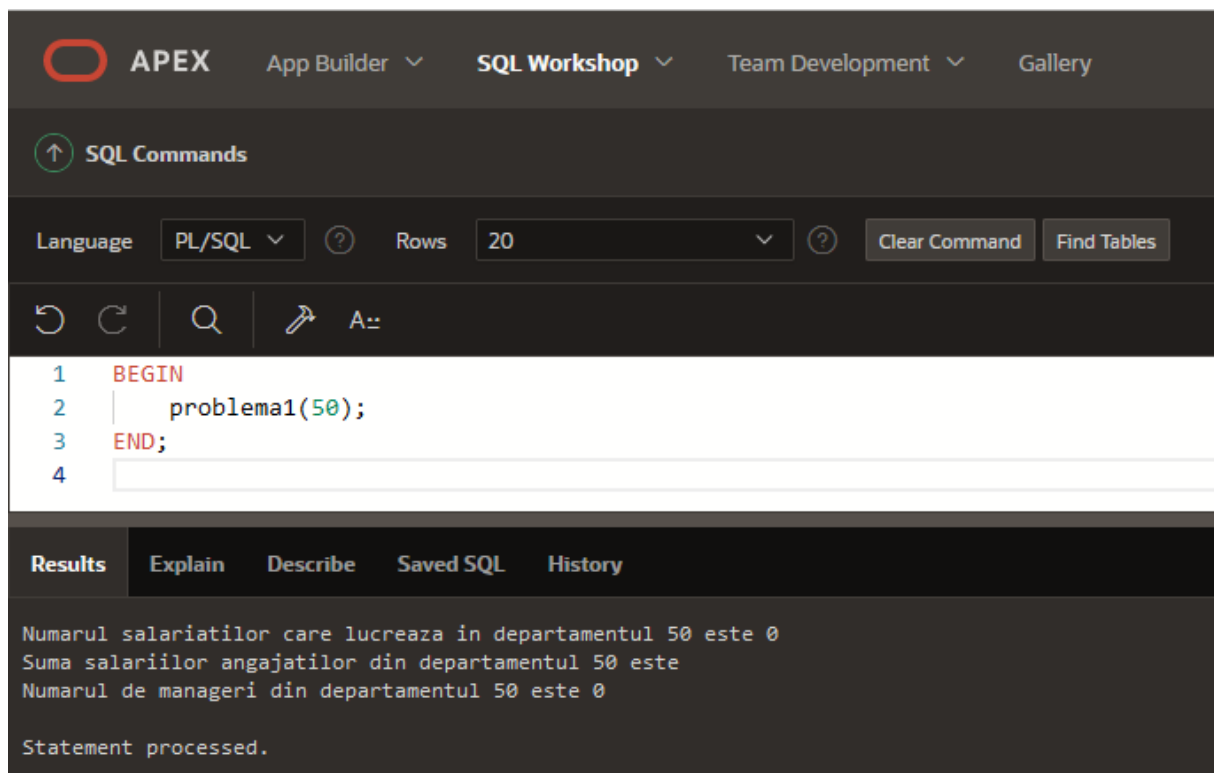
```

Pasul 2 – Executia procedurii construite prin apelare intr-un bloc anonim:

```

BEGIN
    problema1(50);
END;

```



The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs for APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below the tabs, there is a section for SQL Commands with a search icon and a list of commands. The Language is set to PL/SQL and the Rows limit is 20. There are buttons for Clear Command and Find Tables. Below the command area, there are icons for undo, redo, search, and a text input field. The main area shows the following SQL code:

```

1 BEGIN
2     problema1(50);
3 END;
4

```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the following output:

```

Numarul salariatilor care lucreaza in departamentul 50 este 0
Suma salariilor angajatilor din departamentul 50 este
Numarul de manageri din departamentul 50 este 0

Statement processed.

```

2. Să se creeze două funcții (locale) supraîncărcate (overloaded) 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

Solutie:

```
DECLARE
    medie1 NUMBER(10,2);
    medie2 NUMBER(10,2);
FUNCTION medie (v_dept emp.deptno%TYPE)
    RETURN NUMBER IS
    rezultat NUMBER(10,2);
BEGIN
    SELECT AVG(sal)
    INTO rezultat
    FROM emp
    WHERE deptno = v_dept;
    RETURN rezultat;
END;
FUNCTION medie (v_dept emp.deptno%TYPE, v_job emp.job %TYPE)
    RETURN NUMBER IS
    rezultat NUMBER(10,2);
BEGIN
    SELECT AVG(sal)
    INTO rezultat
    FROM emp
    WHERE deptno = v_dept AND job = v_job;
    RETURN rezultat;
END;
```

```
BEGIN
    medie1:=medie(30);
    DBMS_OUTPUT.PUT_LINE('Media salariilor din departamentul 30 este ' ||
medie1);
    medie2 := medie(30, 'SALESMAN');
    DBMS_OUTPUT.PUT_LINE('Media salariilor reprezentantilor de vanzari din
departamentul 30 este ' || medie2);
END;
```



↑ SQL Commands

Language

PL/SQL ▾



Rows

20 ▾



Clear Command

Find Tables



A::

```
1  DECLARE
2  medie1 NUMBER(10,2);
3  medie2 NUMBER(10,2);
4  FUNCTION medie (v_dept emp.deptno%TYPE)
5  |      RETURN NUMBER IS
6  rezultat NUMBER(10,2);
7  BEGIN
8  |      SELECT AVG(sal)
9  |      INTO rezultat
10 |      FROM emp
11 |      WHERE deptno = v_dept;
12 |      RETURN rezultat;
13 END;
```

```
14 FUNCTION medie (v_dept emp.deptno%TYPE, v_job emp.job %TYPE)
15 |      RETURN NUMBER IS
16 |      rezultat NUMBER(10,2);
17 BEGIN
18 |      SELECT AVG(sal)
19 |      INTO rezultat
20 |      FROM emp
21 |      WHERE deptno = v_dept AND job = v_job;
22 |      RETURN rezultat;
23 END;
```

```
25 BEGIN
26 medie1:=medie(30);
27 DBMS_OUTPUT.PUT_LINE('Media salariilor din departamentul 30 este ' || medie1);
28 medie2 := medie(30, 'SALESMAN');
29 DBMS_OUTPUT.PUT_LINE('Media salariilor reprezentantilor de vanzari din departamentul 30 este ' || medie2);
30 END;
```

```
31 |
```

Results

Explain

Describe

Saved SQL

History

```
Media salariilor din departamentul 30 este 2566.67
Media salariilor reprezentantilor de vanzari din departamentul 30 este 2400

Statement processed.
```

b) Funcții stocate

3. Să se creeze o funcție stocată care determină numărul de salariați din emp angajați după 1981, într-un departament dat ca parametru. Să se apeleze această funcție prin diferite modalități:

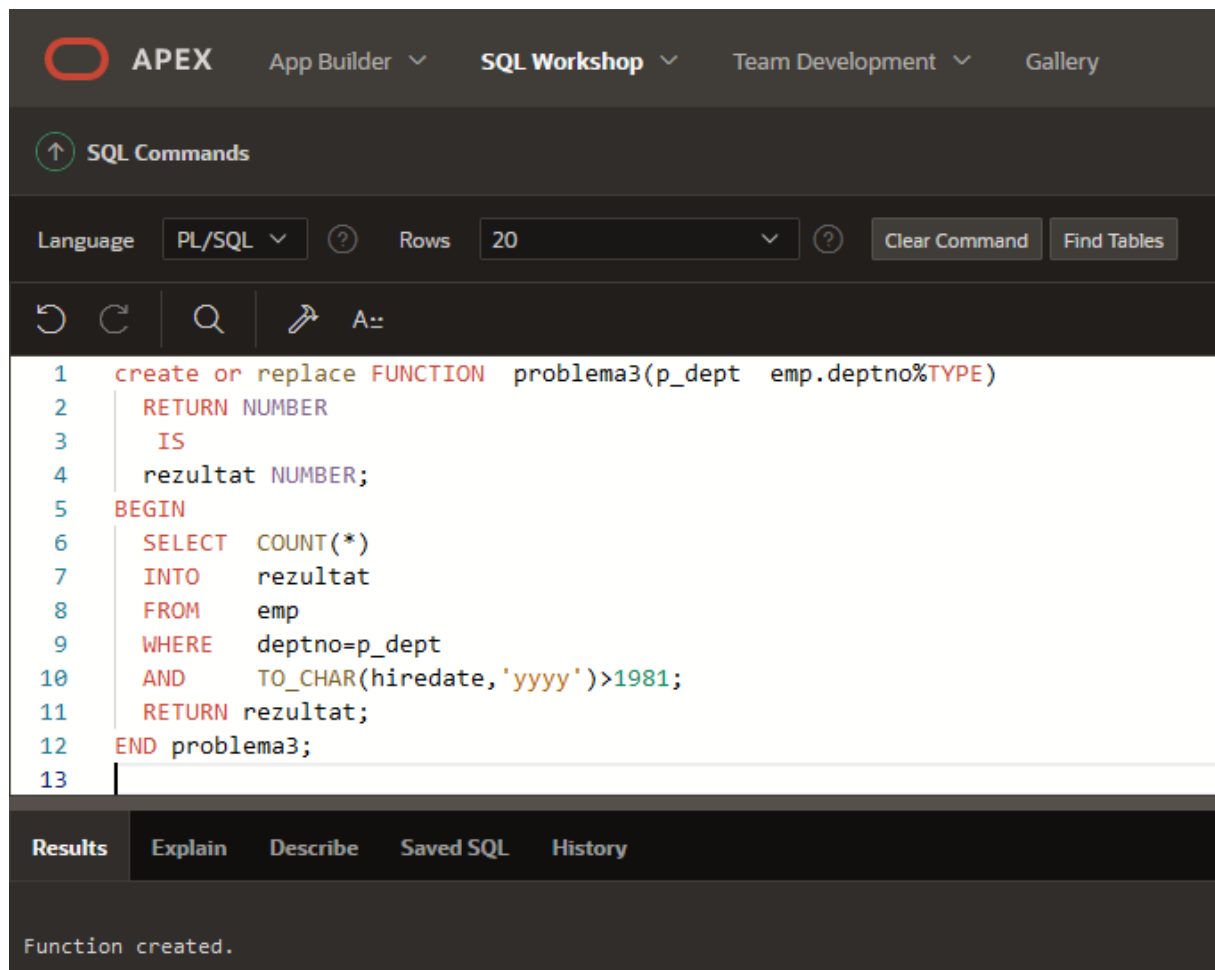
- printr-o comandă SELECT;

- într-un bloc PL/SQL.

Soluție:

Pasul 1 – Crearea funcției problema3:

```
CREATE OR REPLACE FUNCTION problema3 (p_dept emp.deptno%TYPE)
RETURN NUMBER
IS
  rezultat NUMBER;
BEGIN
  SELECT COUNT(*)
  INTO rezultat
  FROM emp
  WHERE deptno=p_dept
  AND TO_CHAR(hire_date,'yyyy')>1981;
  RETURN rezultat;
END problema3;
```

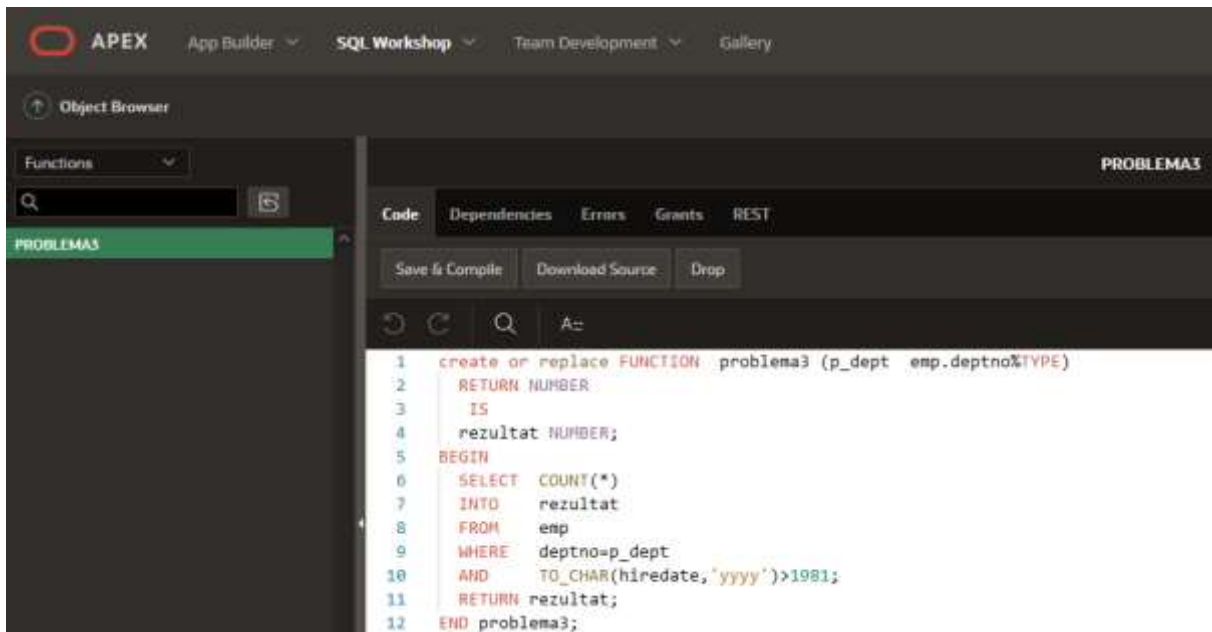


The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing a 'Language' dropdown set to 'PL/SQL' and 'Rows' set to '20'. The main editor area contains the following SQL code:

```
1 create or replace FUNCTION problema3(p_dept emp.deptno%TYPE)
2   RETURN NUMBER
3   IS
4   rezultat NUMBER;
5 BEGIN
6   SELECT COUNT(*)
7   INTO rezultat
8   FROM emp
9   WHERE deptno=p_dept
10  AND TO_CHAR(hiredate,'yyyy')>1981;
11  RETURN rezultat;
12 END problema3;
13
```

Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing the message 'Function created.'

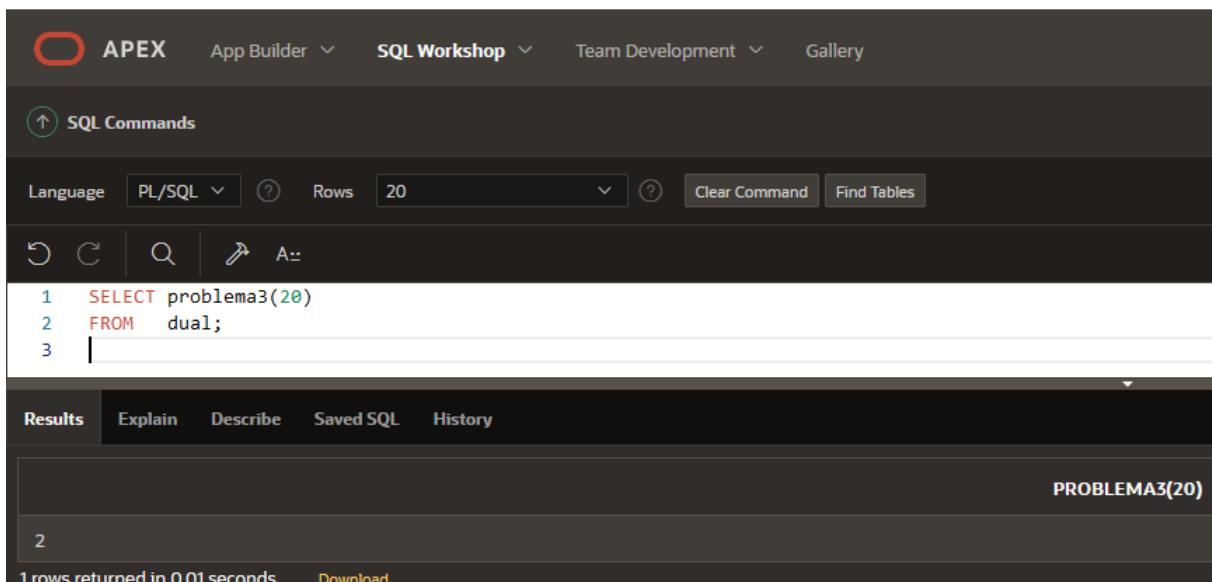
Pasul 2 – Verificarea stocării funcției. Se accesează **OBJECT BROWSER**; se selectează **FUNCTIONS**, iar apoi *se selectează numele funcției dorite din lista de funcții*. Conținutul codului funcției va apărea în zona principală a ecranului respectiv:



```
1 create or replace FUNCTION problema3 (p_dept emp.deptno%TYPE)
2 RETURN NUMBER
3 IS
4 rezultat NUMBER;
5 BEGIN
6 SELECT COUNT(*)
7 INTO rezultat
8 FROM emp
9 WHERE deptno=p_dept
10 AND TO_CHAR(hiredate,'yyyy')>1981;
11 RETURN rezultat;
12 END problema3;
```

Pasul 3 – Se execută funcția construită la pasul 1, printr-un apel în care parametrul este 20 cu semnificația că se prelucrează datele pentru departamentul cu numărul 20.

1)
SELECT problema3(20)
FROM dual;



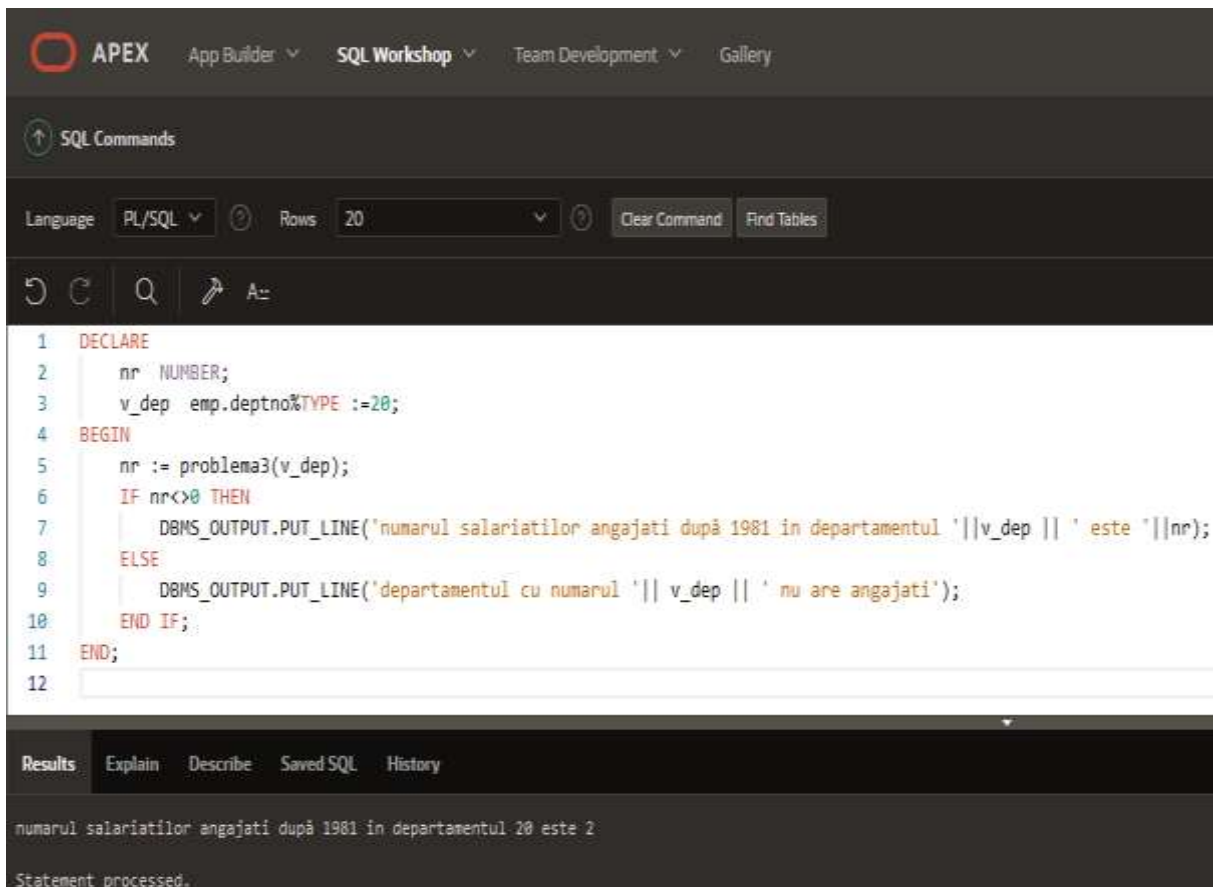
```
1 SELECT problema3(20)
2 FROM dual;
3 |
```

PROBLEMA3(20)
2

1 rows returned in 0.01 seconds [Download](#)

Pasul 4 – Se construiește un bloc anonim în care se apelează funcția construită la pasul 1 cu un parametru dat ca variabilă.

```
2)
DECLARE
  nr NUMBER;
  v_dep emp.deptno%TYPE :=20;
BEGIN
  nr := problema3(v_dep);
  IF nr<>0 THEN
    DBMS_OUTPUT.PUT_LINE('numarul salariatilor angajati după 1981 in
departamentul '||v_dep || ' este '||nr);
  ELSE
    DBMS_OUTPUT.PUT_LINE('departamentul cu numarul '|| v_dep || ' nu are
angajati');
  END IF;
END;
```



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing the language set to 'PL/SQL' and 'Rows' set to '20'. The main editor contains the following PL/SQL code:

```
1 DECLARE
2   nr NUMBER;
3   v_dep emp.deptno%TYPE :=20;
4 BEGIN
5   nr := problema3(v_dep);
6   IF nr<>0 THEN
7     DBMS_OUTPUT.PUT_LINE('numarul salariatilor angajati după 1981 in departamentul '||v_dep || ' este '||nr);
8   ELSE
9     DBMS_OUTPUT.PUT_LINE('departamentul cu numarul '|| v_dep || ' nu are angajati');
10  END IF;
11 END;
```

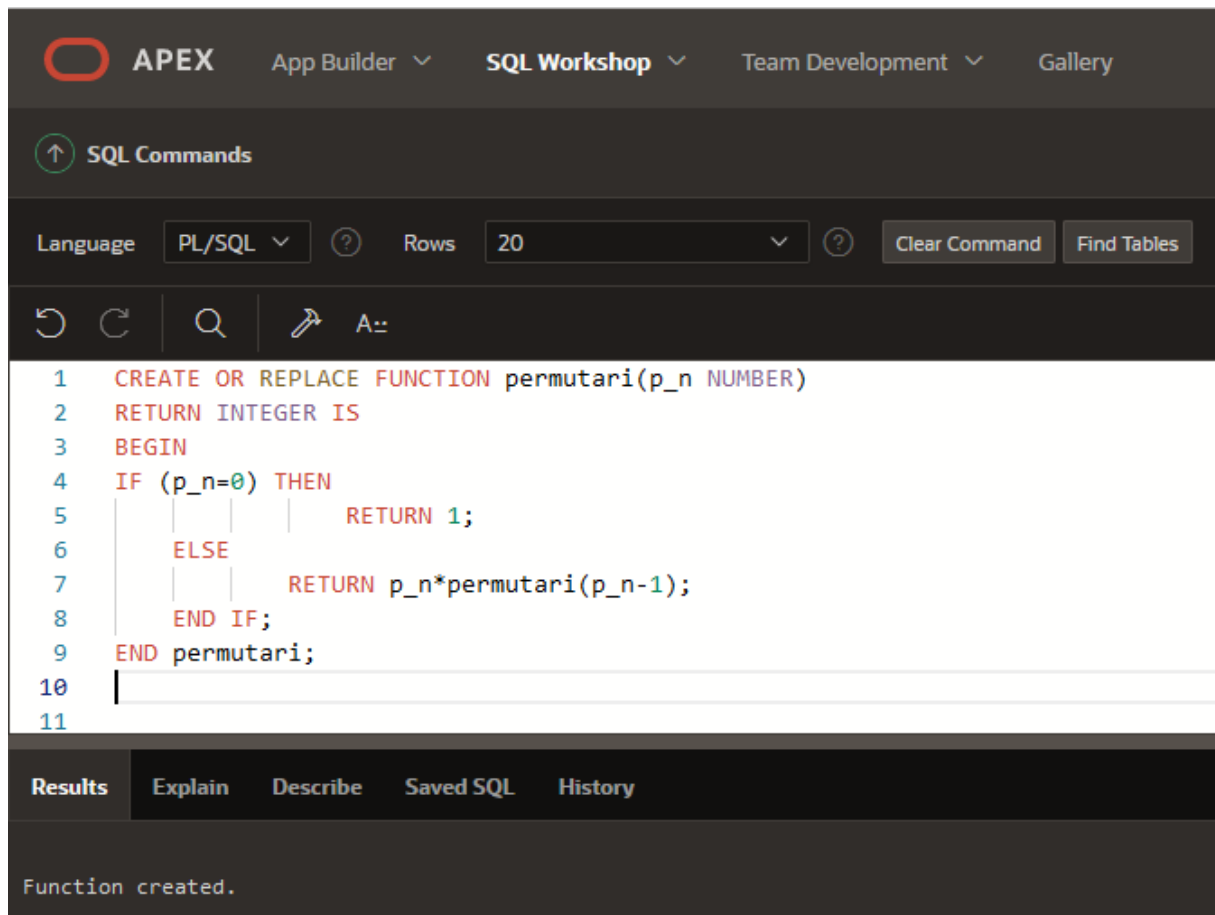
The 'Results' tab at the bottom shows the output of the execution: 'numarul salariatilor angajati după 1981 in departamentul 20 este 2'. Below the output, it states 'Statement processed.'

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

Soluție:

Pasul 1 – Se construiește funcția permutari.

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



The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs for 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is a 'SQL Commands' section with a search icon and a list of commands. The 'Language' is set to 'PL/SQL' and 'Rows' is set to '20'. There are buttons for 'Clear Command' and 'Find Tables'. The main area displays the SQL code for the 'permutari' function, with line numbers 1 through 11. The code is: 1 CREATE OR REPLACE FUNCTION permutari(p_n NUMBER), 2 RETURN INTEGER IS, 3 BEGIN, 4 IF (p_n=0) THEN, 5 | | | RETURN 1;, 6 | ELSE, 7 | | | RETURN p_n*permutari(p_n-1);, 8 | END IF;, 9 END permutari;, 10 |, 11 |. Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the message 'Function created.'

Pasul 2 – Într-un bloc anonim se apelează funcția construită la pasul 1 cu parametrul 5.

```
DECLARE valoare_permutari_n NUMBER;
BEGIN
    valoare_permutari_n := permutari(5);
    DBMS_OUTPUT.PUT_LINE('permutari de ' ||5||' = ' || valoare_permutari_n);
END;
```

The screenshot shows the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below the tabs, there's a 'SQL Commands' section with a back arrow. The main area has a 'Language' dropdown set to 'PL/SQL', a 'Rows' dropdown set to '20', and buttons for 'Clear Command' and 'Find Tables'. Below this is a toolbar with icons for undo, redo, search, and a text input field containing 'A:'. The main code editor contains the following PL/SQL script:

```
1 DECLARE valoare_permutari_n NUMBER;
2 BEGIN
3     valoare_permutari_n := permutari(5);
4     DBMS_OUTPUT.PUT_LINE('permutari de ' || 5 || ' = ' || valoare_permutari_n);
5 END;
```

Below the code editor, there's a 'Results' section with tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', 'History', and 'Bottom Splitter'. The 'Results' tab is active, showing the output of the script:

```
permutari de 5 = 120
Statement processed.
```

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

Soluție:

Pasul 1 – Se creeaza functia valoare_medie_sal pentru calculul mediei salariilor angajatilor din tabela EMP.

```
CREATE OR REPLACE FUNCTION valoare_medie_sal
RETURN NUMBER IS
medie NUMBER;
BEGIN
    SELECT AVG(sal)
INTO medie
FROM emp;
    RETURN medie;
END;
```

The screenshot shows the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below the tabs, there's a 'SQL Commands' section with a back arrow. The main area has a 'Language' dropdown set to 'PL/SQL', a 'Rows' dropdown set to '20', and buttons for 'Clear Command' and 'Find Tables'. Below this is a toolbar with icons for undo, redo, search, and a text input field containing 'A:'. The main code editor contains the following PL/SQL script:

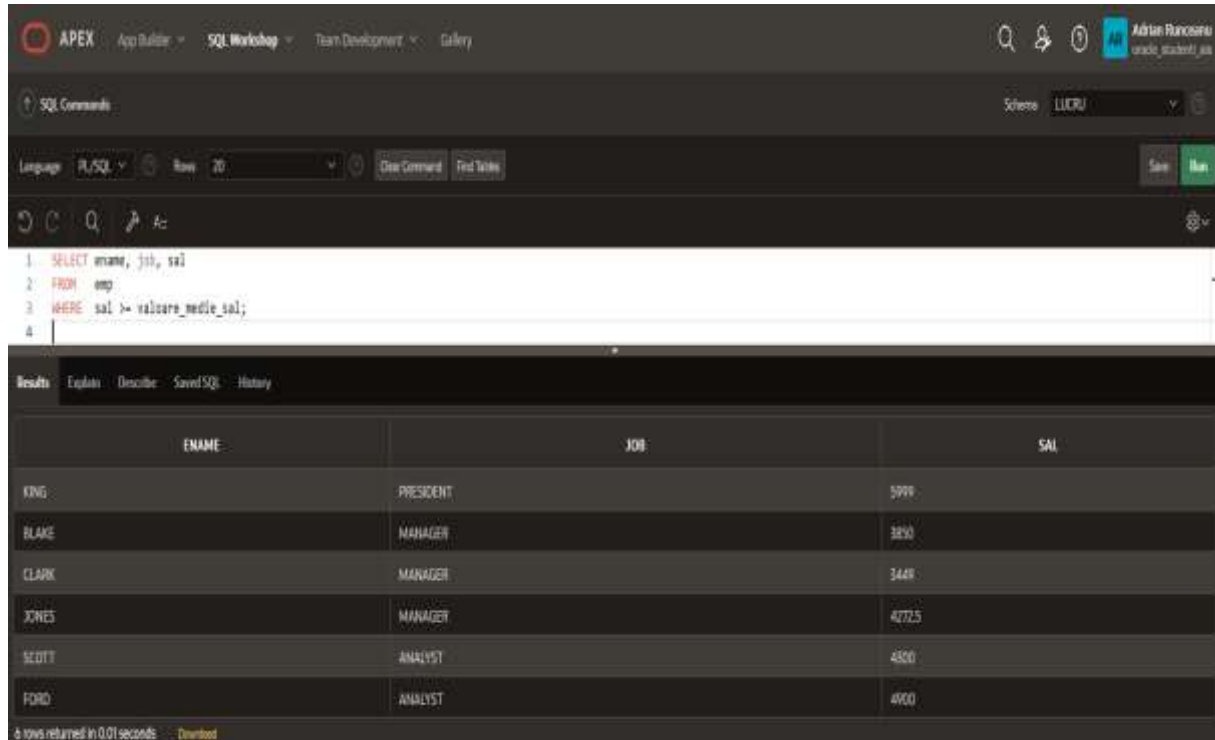
```
1 CREATE OR REPLACE FUNCTION valoare_medie_sal
2     RETURN NUMBER IS
3     medie NUMBER;
4 BEGIN
5     SELECT AVG(sal)
6     INTO medie
7     FROM emp;
8     RETURN medie;
9 END;
```

Below the code editor, there's a 'Results' section with tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the output of the script:

```
Function created.
```

Pasul 2 – Se scrie o interogare care rezolva cerinta problemei.

```
SELECT ename, job, sal
FROM emp
WHERE sal >= valoare_medie_sal;
```



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following query:

```
1. SELECT ename, job, sal
2. FROM emp
3. WHERE sal >= valoare_medie_sal;
4.
```

The Results window displays the following data:

ENAME	JOB	SAL
KING	PRESIDENT	5000
BLAKE	MANAGER	3850
CLARK	MANAGER	3440
JONES	MANAGER	4725
SCOTT	ANALYST	4300
FORD	ANALYST	4900

6 rows returned in 0.01 seconds

6. a) Analizați subprogramele create anterior în vizualizarea din dicționarul datelor USER_OBJECTS.

b) Regăsiți codul unuia dintre subprogramele create anterior în vizualizarea USER_SOURCE.

c) Aflați tipul parametrilor uneia dintre procedurile create anterior utilizând comanda DESCRIBE.

Probleme propuse spre rezolvare

1. Să se creeze o funcție stocată care determină numărul de salariați care au fost angajați ulterior tuturor subalternilor unui manager al cărui cod este dat ca parametru. Să se apeleze această funcție într-un bloc PL/SQL.
2. Să se creeze trei funcții locale cu același nume care să calculeze numărul de salariați astfel:
 - prima funcție va avea ca argument codul departamentului, adică funcția calculează numărul de salariați din departamentul specificat;
 - a doua funcție va avea două argumente, unul reprezentând codul departamentului, iar celălalt reprezentând anul angajării, adică funcția va calcula numărul de salariați din departament și care au fost angajați într-un anumit an;
 - a treia funcție va avea trei argumente, unul reprezentând codul departamentului, unul reprezentând anul angajării, iar celălalt grila de salarizare, adică funcția va calcula numărul de salariați din departament, care au fost angajați într-un anumit an și au salariul într-o anumită grilă de salarizare.
3. Să se modifice salariul unui angajat al cărui cod este introdus ca parametru astfel încât să devină media salariilor angajaților care câștigă comision dintr-un departament dat ca parametru.
4. Să se scrie o funcție care să întoarcă, pentru un angajat al cărui cod este specificat ca parametru, vechimea exprimată în luni. Să se utilizeze funcția într-o instrucțiune SELECT care să întoarcă numele angajaților, salariul și numărul de luni lucrate.

Bibliografie web:

<https://www.w3resource.com/>

<https://www.bullraider.com/database/pl-sql/pl-sql-examples>

<https://www.oracletutorial.com/plsql-tutorial/plsql-function/>

<https://docs.oracle.com/database/121/LNPLS/subprograms.htm#LNPLS00825>