

Bază de date pentru gestionarea unei afaceri cu librării

Dragoș-Constantin Țânțaru, grupa 244

January 9, 2021

1 Descrierea proiectului

Am construit o bază de date care are ca scop facilitarea conducerii unei afaceri de tip librărie. Este posibil ca afacerea să aibă mai multe librării în posesia sa. De asemenea, am considerat că un angajat poate lucra la mai mult de o singură librărie, ca exemplu ar fi un manager ce se ocupă de mai multe librării sau un IT-ist care administrează sistemele informatice de la diferite librării.

Am implementat și funcționalitatea bazei de date de a admite „membrii” ai afacerii. Un membru este un client ce a achiziționat un tip de abonament. Abonamentele reprezintă o plată lunară făcută de client, în schimbul căreia are acces la o mulțime de cărți în format digital.

Baza de date asigură toate aceste utilități și multe altele. Fără o bază de date, ar fi dificil pentru o afacere să poată menține o evidență corectă, nemaivorbind de diversele aplicații care se pot implementa facil cu o bază de date, dar care ar fi greu de realizat altfel.

2 Diagrama Entitate-Relație

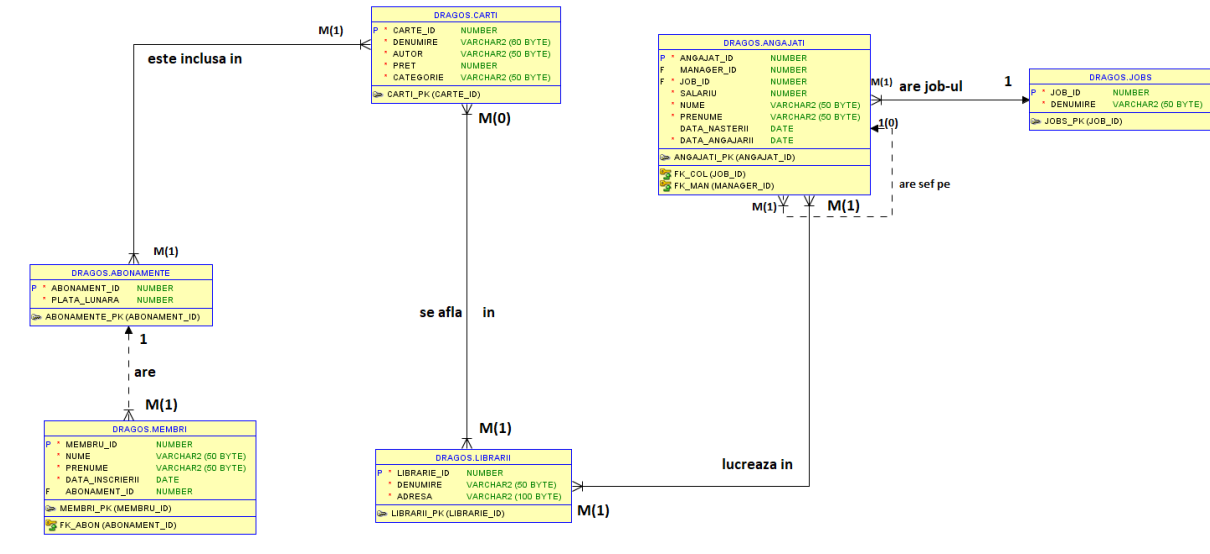


Figure 1: diagrama ER

3 Diagrama Conceptuală

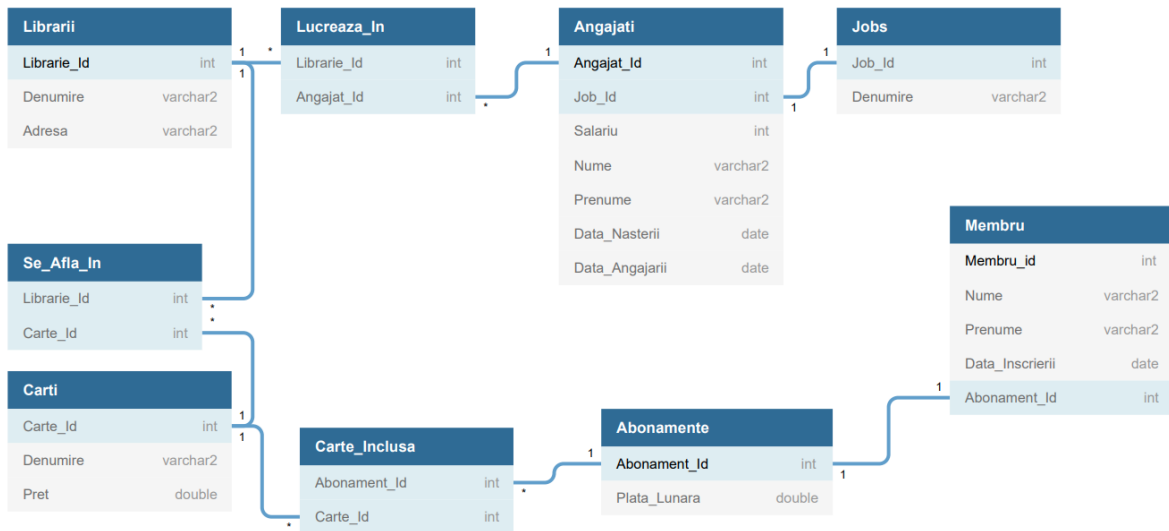


Figure 2: diagrama Conceptuala

4 Definirea tabelor

```

1 CREATE TABLE Librarii (
2 Librarie_Id NUMBER NOT NULL,

```

```
3 Denumire VARCHAR2(50) NOT NULL ,
4 Adresa VARCHAR2(100) NOT NULL ,
5 PRIMARY KEY (Librarie_Id)
6 );
7
8 CREATE TABLE Carti (
9 Carte_Id NUMBER NOT NULL ,
10 Denumire VARCHAR2(60) NOT NULL ,
11 Autor VARCHAR(50) NOT NULL ,
12 Pret NUMBER NOT NULL ,
13 Categorie VARCHAR2(50) NOT NULL ,
14 PRIMARY KEY (Carte_Id)
15 );
16
17 CREATE TABLE Abonamente(
18 Abonament_Id NUMBER NOT NULL ,
19 Plata_Lunara NUMBER NOT NULL ,
20 PRIMARY KEY (Abonament_Id)
21 );
22
23 CREATE TABLE Membri (
24 Membru_Id NUMBER NOT NULL ,
25 Nume VARCHAR2(50) NOT NULL ,
26 Prenume VARCHAR2(50) NOT NULL ,
27 Data_Inscrierii DATE NOT NULL ,
28 Abonament_Id NUMBER ,
29 PRIMARY KEY (Membru_Id),
30 CONSTRAINT fk_abon
31 FOREIGN KEY (Abonament_Id)
32 REFERENCES Abonamente(Abonament_Id)
33 );
34
35 CREATE TABLE Carte_Inclusa(
36 Abonament_Id NUMBER NOT NULL ,
37 Carte_Id NUMBER NOT NULL ,
38 PRIMARY KEY (Abonament_Id, Carte_Id),
39 CONSTRAINT fk_col_ab
40 FOREIGN KEY (Abonament_Id)
41 REFERENCES Abonamente(Abonament_Id),
42 CONSTRAINT fk_col_car
43 FOREIGN KEY (Carte_Id)
44 REFERENCES Carti(Carte_Id)
45 );
46
47 CREATE TABLE Se_Afla_In (
```

```
48 Librarie_Id NUMBER NOT NULL ,
49 Carte_Id NUMBER NOT NULL ,
50 PRIMARY KEY (Librarie_Id, Carte_Id),
51 CONSTRAINT fk_asoc_sb
52 FOREIGN KEY (Librarie_Id)
53 REFERENCES Librarii(Librarie_Id),
54 CONSTRAINT fk_asoc_sc
55 FOREIGN KEY (Carte_Id)
56 REFERENCES Carti(Carte_Id)
57 );
58
59
60 CREATE TABLE Jobs (
61 Job_Id NUMBER NOT NULL ,
62 Denumire VARCHAR2(50) NOT NULL ,
63 PRIMARY KEY (Job_Id)
64 );
65
66 CREATE TABLE Angajati (
67 Angajat_Id NUMBER NOT NULL ,
68 Manager_Id NUMBER DEFAULT NULL ,
69 Job_Id NUMBER NOT NULL ,
70 Salariu NUMBER NOT NULL ,
71 Nume VARCHAR2(50) NOT NULL ,
72 Prenume VARCHAR2(50) NOT NULL ,
73 Data_Nasterii DATE DEFAULT NULL ,
74 Data_Angajarii DATE NOT NULL ,
75 PRIMARY KEY (Angajat_Id),
76 CONSTRAINT fk_col
77 FOREIGN KEY (Job_Id)
78 REFERENCES Jobs(Job_Id),
79 CONSTRAINT fk_man
80 FOREIGN KEY (Manager_Id)
81 REFERENCES Angajati(Angajat_Id)
82 );
83
84 CREATE TABLE Lucreaza_In (
85 Librarie_Id NUMBER NOT NULL ,
86 Angajat_Id NUMBER NOT NULL ,
87 PRIMARY KEY(Librarie_Id, Angajat_Id),
88 CONSTRAINT fk_asoc_b
89 FOREIGN KEY (Librarie_Id)
90 REFERENCES Librarii(Librarie_Id),
91 CONSTRAINT fk_asoc_a
92 FOREIGN KEY (Angajat_Id)
```

```

93 REFERENCES Angajati(Angajat_Id)
94 );

```

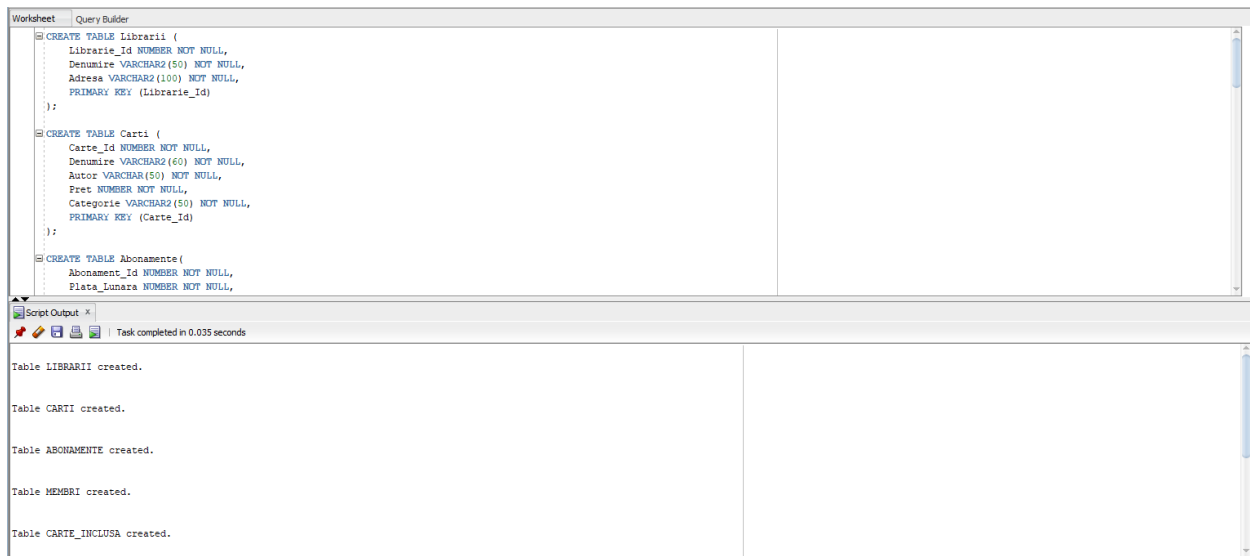


Figure 3: creates 1

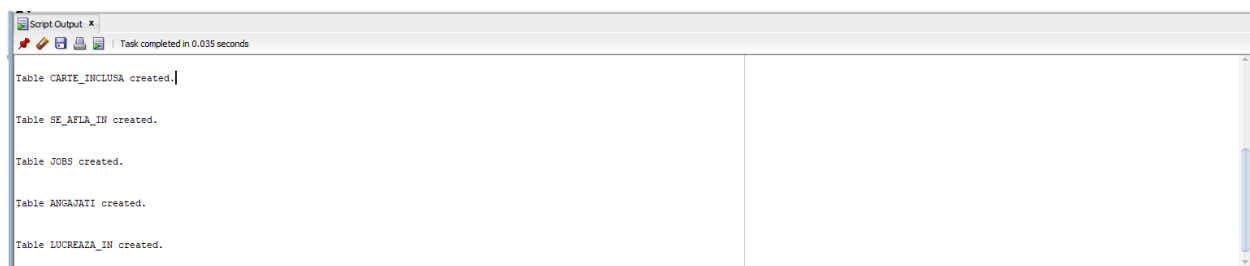


Figure 4: creates 2

5 Inserări în tabel

```

1 INSERT INTO Librarii
2 VALUES (1, 'Libraria_abc', 'Strada_Unirii_nr._23');
3 INSERT INTO Librarii
4 VALUES (2, 'Mihai_Eminescu', 'Strada_Octavian_Goga_nr._26');
5 INSERT INTO Librarii
6 VALUES (3, 'Libraria_veche', 'Bulevardul_Mare_nr._1');
7
8 INSERT INTO Jobs
9 VALUES(1, 'Manager');
10 INSERT INTO Jobs

```

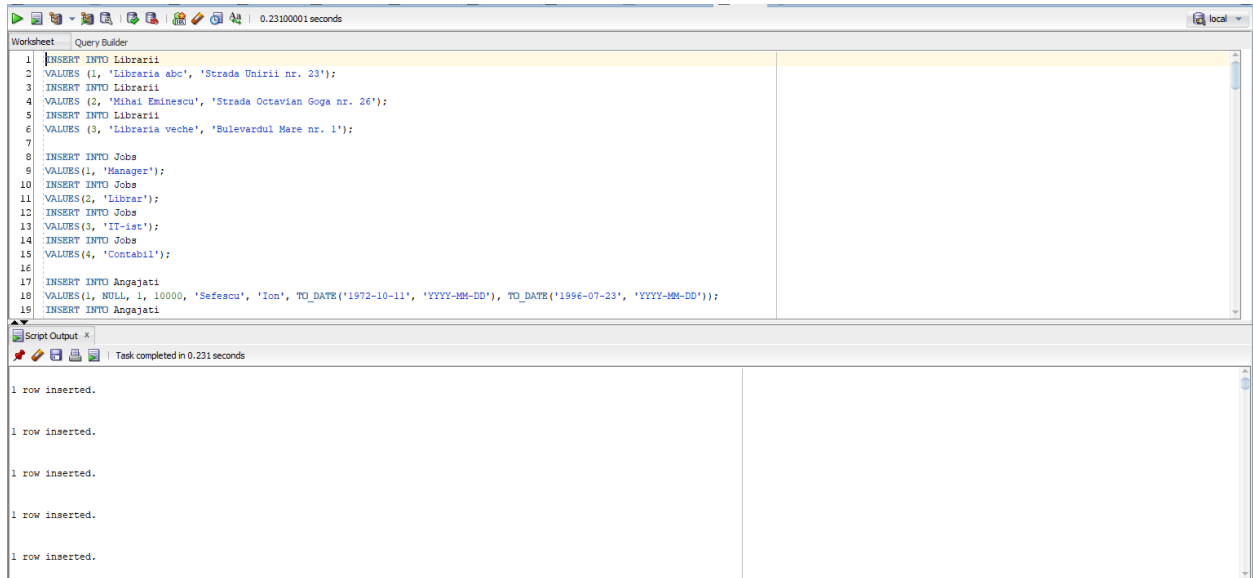
```
11 VALUES(2, 'Librar');
12 INSERT INTO Jobs
13 VALUES(3, 'IT-ist');
14 INSERT INTO Jobs
15 VALUES(4, 'Contabil');
16
17 INSERT INTO Angajati
18 VALUES(1, NULL, 1, 10000, 'Sefescu', 'Ion', TO_DATE('1972-10-11', 'YYYY-MM-DD'), TO_DATE('1996-07-23', 'YYYY-MM-DD'));
19 INSERT INTO Angajati
20 VALUES(2, 1, 2, 3000, 'Pop', 'Andrei', TO_DATE('1975-07-02', 'YYYY-MM-DD'), TO_DATE('1998-02-03', 'YYYY-MM-DD'));
21 INSERT INTO Angajati
22 VALUES(3, 1, 2, 3000, 'Bob', 'Alex', TO_DATE('1981-04-23', 'YYYY-MM-DD'), TO_DATE('1999-06-10', 'YYYY-MM-DD'));
23 INSERT INTO Angajati
24 VALUES(4, 1, 4, 2500, 'Ionescu', 'Radu', TO_DATE('1983-02-07', 'YYYY-MM-DD'), TO_DATE('2001-10-30', 'YYYY-MM-DD'));
25 INSERT INTO Angajati
26 VALUES(5, 1, 3, 5000, 'Petrescu', 'Alex', TO_DATE('1987-05-11', 'YYYY-MM-DD'), TO_DATE('2007-01-17', 'YYYY-MM-DD'));
27 INSERT INTO Angajati
28 VALUES(6, 1, 2, 4000, 'Marcu', 'Carl', TO_DATE('1979-06-13', 'YYYY-MM-DD'), TO_DATE('1998-03-01', 'YYYY-MM-DD'));
29 INSERT INTO Angajati
30 VALUES(7, 1, 2, 4100, 'Andreescu', 'Andrei', TO_DATE('1969-10-10', 'YYYY-MM-DD'), TO_DATE('1998-02-07', 'YYYY-MM-DD'));
31 INSERT INTO Angajati
32 VALUES(8, 1, 2, 4000, 'Adriana', 'Maria', TO_DATE('1989-06-13', 'YYYY-MM-DD'), TO_DATE('2008-04-19', 'YYYY-MM-DD'));
33
34 INSERT INTO Lucreaza_In
35 VALUES (1, 1);
36 INSERT INTO Lucreaza_In
37 VALUES (2, 1);
38 INSERT INTO Lucreaza_In
39 VALUES (3, 1);
40 INSERT INTO Lucreaza_In
41 VALUES (1, 2);
42 INSERT INTO Lucreaza_In
43 VALUES (1, 3);
44 INSERT INTO Lucreaza_In
45 VALUES (1, 4);
46 INSERT INTO Lucreaza_In
```

```
47 VALUES (2, 4);
48 INSERT INTO Lucreaza_In
49 VALUES (3, 4);
50 INSERT INTO Lucreaza_In
51 VALUES (1, 5);
52 INSERT INTO Lucreaza_In
53 VALUES (2, 5);
54 INSERT INTO Lucreaza_In
55 VALUES (3, 5);
56 INSERT INTO Lucreaza_In
57 VALUES (2, 6);
58 INSERT INTO Lucreaza_In
59 VALUES (3, 7);
60 INSERT INTO Lucreaza_In
61 VALUES (3, 8);
62
63 INSERT INTO Carti
64 VALUES (1, 'Stapanul_Inelelor_1', 'J.R.R._Tolkien', 60, '
    Fantezie');
65 INSERT INTO Carti
66 VALUES (2, 'Stapanul_Inelelor_2', 'J.R.R._Tolkien', 60, '
    Fantezie');
67 INSERT INTO Carti
68 VALUES (3, 'Stapanul_Inelelor_3', 'J.R.R._Tolkien', 60, '
    Fantezie');
69 INSERT INTO Carti
70 VALUES (4, '2001:_0_odisee_spatiala', 'Arthur_C._Clarke', 83, '
    SF');
71 INSERT INTO Carti
72 VALUES (5, 'O_mie_noua_sute_optzeci_si_patru', 'George_Orwell',
    72, 'Politic');
73 INSERT INTO Carti
74 VALUES (6, 'Ion', 'Liviu_Rebreanu', 25, 'Roman_social');
75 INSERT INTO Carti
76 VALUES (7, 'Fundatia', 'Isaac_Asimov', 65, 'SF');
77
78 INSERT INTO Se_Afla_In
79 VALUES(1,1);
80 INSERT INTO Se_Afla_In
81 VALUES(1,2);
82 INSERT INTO Se_Afla_In
83 VALUES(1,3);
84 INSERT INTO Se_Afla_In
85 VALUES(1,4);
86 INSERT INTO Se_Afla_In
```

```
87 VALUES (1,6);
88 INSERT INTO Se_Afla_In
89 VALUES (1,7);
90 INSERT INTO Se_Afla_In
91 VALUES (2,1);
92 INSERT INTO Se_Afla_In
93 VALUES (2,2);
94 INSERT INTO Se_Afla_In
95 VALUES (2,3);
96 INSERT INTO Se_Afla_In
97 VALUES (2,4);
98 INSERT INTO Se_Afla_In
99 VALUES (3,1);
100 INSERT INTO Se_Afla_In
101 VALUES (3,2);
102 INSERT INTO Se_Afla_In
103 VALUES (3,3);
104 INSERT INTO Se_Afla_In
105 VALUES (3,5);
106 INSERT INTO Se_Afla_In
107 VALUES (3,6);
108
109 INSERT INTO Abonamente
110 VALUES (1, 10);
111 INSERT INTO Abonamente
112 VALUES (2, 15);
113 INSERT INTO Abonamente
114 VALUES (3, 25);
115 INSERT INTO Abonamente
116 VALUES (4, 10);
117
118 INSERT INTO Membri
119 VALUES (1, 'Andrei', 'Andrei', TO_DATE('1999-04-04', 'YYYY-MM-DD'), 1);
120 INSERT INTO Membri
121 VALUES (2, 'Popescu', 'Marina', TO_DATE('2001-03-22', 'YYYY-MM-DD'), 1);
122 INSERT INTO Membri
123 VALUES (3, 'Ionila', 'Ioana', TO_DATE('2002-10-10', 'YYYY-MM-DD'), 4);
124 INSERT INTO Membri
125 VALUES (4, 'Paun', 'Silviu', TO_DATE('2000-10-01', 'YYYY-MM-DD'), 2);
126 INSERT INTO Membri
127 VALUES (5, 'Stefanescu', 'Stefan', TO_DATE('2003-05-10', 'YYYY-
```



```
        MM-DD'), 3);
128 INSERT INTO Membri
129 VALUES (6, 'Stancu', 'Loredana', TO_DATE('2001-11-02', 'YYYY-MM
        -DD'), 2);
130
131 INSERT INTO Carte_Inclusa
132 VALUES (1,1);
133 INSERT INTO Carte_Inclusa
134 VALUES (1,2);
135 INSERT INTO Carte_Inclusa
136 VALUES (1,3);
137 INSERT INTO Carte_Inclusa
138 VALUES (2,1);
139 INSERT INTO Carte_Inclusa
140 VALUES (2,2);
141 INSERT INTO Carte_Inclusa
142 VALUES (2,3);
143 INSERT INTO Carte_Inclusa
144 VALUES (2,4);
145 INSERT INTO Carte_Inclusa
146 VALUES (2,7);
147 INSERT INTO Carte_Inclusa
148 VALUES (3,1);
149 INSERT INTO Carte_Inclusa
150 VALUES (3,2);
151 INSERT INTO Carte_Inclusa
152 VALUES (3,3);
153 INSERT INTO Carte_Inclusa
154 VALUES (3,4);
155 INSERT INTO Carte_Inclusa
156 VALUES (3,5);
157 INSERT INTO Carte_Inclusa
158 VALUES (3,7);
159 INSERT INTO Carte_Inclusa
160 VALUES (4, 5);
161 INSERT INTO Carte_Inclusa
162 VALUES (4, 7);
```



The screenshot shows the SQL Developer interface with a query script in the 'Query Builder' tab. The script contains 19 lines of SQL code for inserting data into three tables: Librarii, Jobs, and Angajati. The 'Script Output' tab below shows the execution results, indicating that 1 row was inserted for each of the five insert statements.

```
1 INSERT INTO Librarii
2 VALUES (1, 'Libraria abc', 'Strada Unirii nr. 23');
3 INSERT INTO Librarii
4 VALUES (2, 'Mihai Eminescu', 'Strada Octavian Goga nr. 26');
5 INSERT INTO Librarii
6 VALUES (3, 'Libraria veche', 'Bulevardul Mare nr. 1');
7
8 INSERT INTO Jobs
9 VALUES (1, 'Manager');
10 INSERT INTO Jobs
11 VALUES (2, 'Librar');
12 INSERT INTO Jobs
13 VALUES (3, 'IT-ist');
14 INSERT INTO Jobs
15 VALUES (4, 'Contabil');
16
17 INSERT INTO Angajati
18 VALUES (1, NULL, 1, 10000, 'Sefescu', 'Ion', TO_DATE('1972-10-11', 'YYYY-MM-DD'), TO_DATE('1996-07-23', 'YYYY-MM-DD'));
19 INSERT INTO Angajati
```

Script Output: Task completed in 0.231 seconds

1 row inserted.


1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Figure 5: inserts 1



The screenshot shows the SQL Developer interface with a query script in the 'Query Builder' tab. The script contains 47 lines of SQL code for inserting data into three tables: Angajati, Lucreaza_In, and Lucreaza_In. The 'Script Output' tab below shows the execution results, indicating that 1 row was inserted for each of the five insert statements.

```
29 INSERT INTO Angajati
30 VALUES (7, 1, 2, 4100, 'Andrescu', 'Andrei', TO_DATE('1969-10-10', 'YYYY-MM-DD'), TO_DATE('1998-02-07', 'YYYY-MM-DD'));
31 INSERT INTO Angajati
32 VALUES (8, 1, 2, 4000, 'Adriana', 'Maria', TO_DATE('1969-06-13', 'YYYY-MM-DD'), TO_DATE('2008-04-19', 'YYYY-MM-DD'));
33
34 INSERT INTO Lucreaza_In
35 VALUES (1, 1);
36 INSERT INTO Lucreaza_In
37 VALUES (2, 1);
38 INSERT INTO Lucreaza_In
39 VALUES (3, 1);
40 INSERT INTO Lucreaza_In
41 VALUES (1, 2);
42 INSERT INTO Lucreaza_In
43 VALUES (1, 3);
44 INSERT INTO Lucreaza_In
45 VALUES (1, 4);
46 INSERT INTO Lucreaza_In
47 VALUES (2, 4);
```

Script Output: Task completed in 0.231 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Figure 6: inserts 2

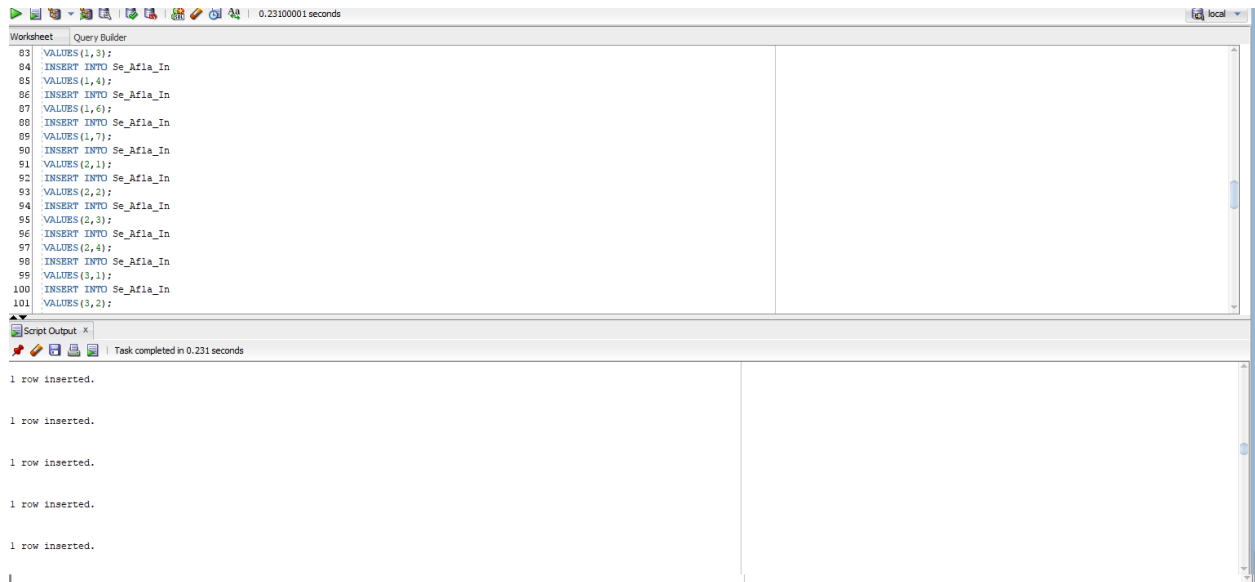


Figure 7: inserts 3

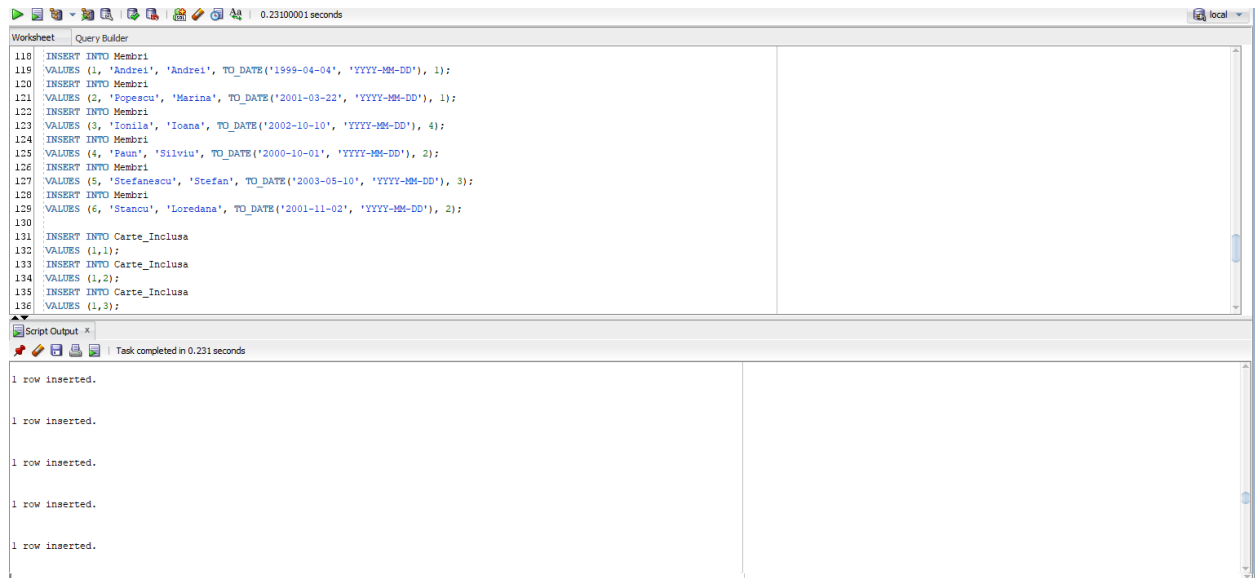


Figure 8: inserts 4

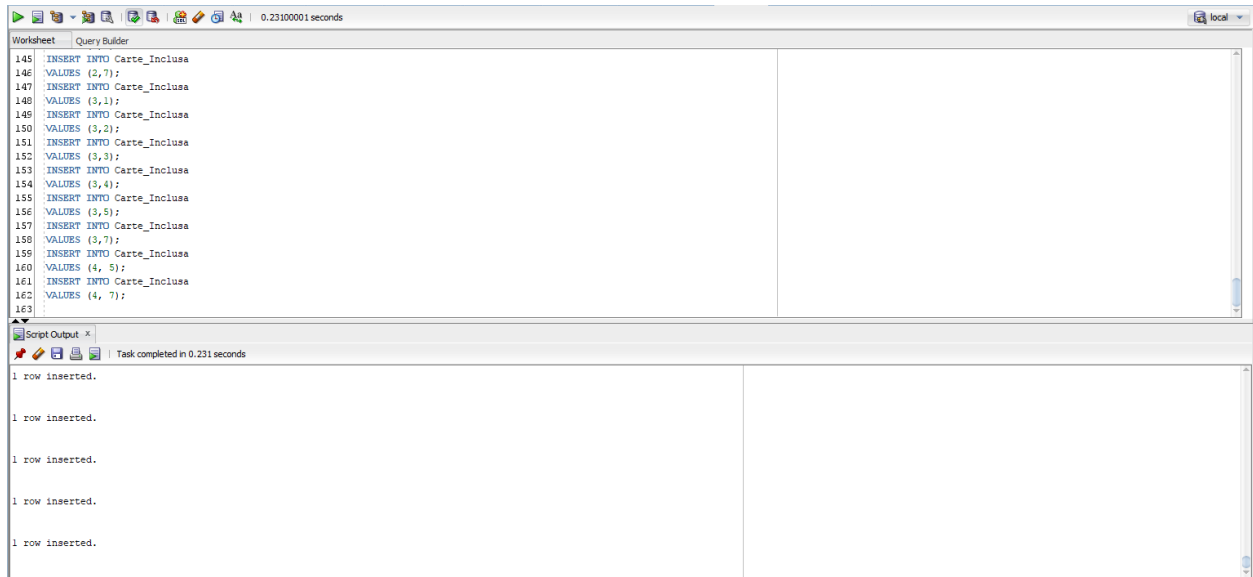


Figure 9: inserts 5

6 Subprogram cu un tip de colecție studiat

Am scris un subprogram care determină în care librării se găsește cartea trimisă ca parametru și ce abonamente o includ.

```
1  -- Un subprogram care determina in care librarii se gaseste
    cartea introdusa ca parametru si ce abonamente o includ.
2  CREATE OR REPLACE PROCEDURE ex6 (v_nume_carte carti.denumire%
    TYPE )IS
3  TYPE string_tabel IS TABLE OF VARCHAR2(60) INDEX BY PLS_INTEGER
    ;
4  TYPE number_tabel IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
5  v_id_carte carti.carte_id%TYPE := NULL;
6  v_librarii string_tabel;
7  v_abonamente string_tabel;
8  v_pret abonamente.plata_lunara%TYPE;
9  BEGIN
10 SELECT carte_id INTO v_id_carte
11 FROM carti
12 WHERE v_nume_carte = carti.denumire;
13
14 SELECT l.denumire BULK COLLECT INTO v_librarii
15 FROM librarii l, se_afla_in sai
16 WHERE sai.carte_id = v_id_carte AND sai.librarie_id = l.
    librarie_id;
17
18 DBMS_OUTPUT.put(v_nume_carte || ' se gaseste in librariile: ');
19 FOR i IN 1..v_librarii.last LOOP
20 DBMS_OUTPUT.put(v_librarii(i) || ' ');
21 END LOOP;
22 DBMS_OUTPUT.NEW_LINE();
23
24 SELECT a.abonament_id BULK COLLECT INTO v_abonamente
25 FROM abonamente a, carte_inclusa ci
26 WHERE ci.carte_id = v_id_carte AND ci.abonament_id = a.
    abonament_id;
27
28 IF v_abonamente.count = 0 THEN
29 DBMS_OUTPUT.PUT_LINE('Cartea nu este inclusa in niciun
    abonament. ');
30 ELSE
31 DBMS_OUTPUT.PUT(v_nume_carte || ' este inclusa in format
    digital in abonamentele: ');
32 FOR i in 1..v_abonamente.last LOOP
```

```
33 SELECT a.plata_lunara INTO v_pret
34 FROM abonamente a
35 WHERE a.abonament_id = v_abonamente(i);
36 DBMS_OUTPUT.PUT('abonamentul_' || v_abonamente(i) || '_cu_'
    pretul_' || v_pret || ',');
37 END LOOP;
38 DBMS_OUTPUT.NEW_LINE();
39 END IF;
40
41 EXCEPTION
42 WHEN NO_DATA_FOUND THEN
43 DBMS_OUTPUT.PUT_LINE('Cartea_nu_exista. ');
44 END;
45 /
```

The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL procedure named 'EX6'. The procedure uses a loop to iterate through a range of values, querying the 'abonamente' table for the monthly fee ('plata_lunara') of each subscription. It then outputs the subscription ID and the fee. An exception handler is included to catch the 'NO_DATA_FOUND' error, which occurs because the specified subscription ID does not exist in the table. The bottom pane, titled 'Script Output', shows the successful compilation of the procedure. The 'Dbms Output' pane at the bottom shows the execution results, indicating that the book 'Ion' is not included in any subscription.

```
32  FOR i in 1..v_abonamente.last LOOP
33      SELECT a.plata_lunara INTO v_pret
34      FROM abonamente a
35      WHERE a.abonament_id = v_abonamente(i);
36      DBMS_OUTPUT.PUT('abonamentul ' || v_abonamente(i) || ' cu pretul ' || v_pret || ', ');
37  END LOOP;
38  DBMS_OUTPUT.NEW_LINE();
39  END IF;
40
41  EXCEPTION
42      WHEN NO_DATA_FOUND THEN
43          DBMS_OUTPUT.PUT_LINE('Cartea nu exista.');
```

44 END;

45 /

46

47 BEGIN

48 ex6('Ion');

49 END;

50 /

Script Output x

Task completed in 0.038 seconds

Procedure EX6 compiled

PL/SQL procedure successfully completed.

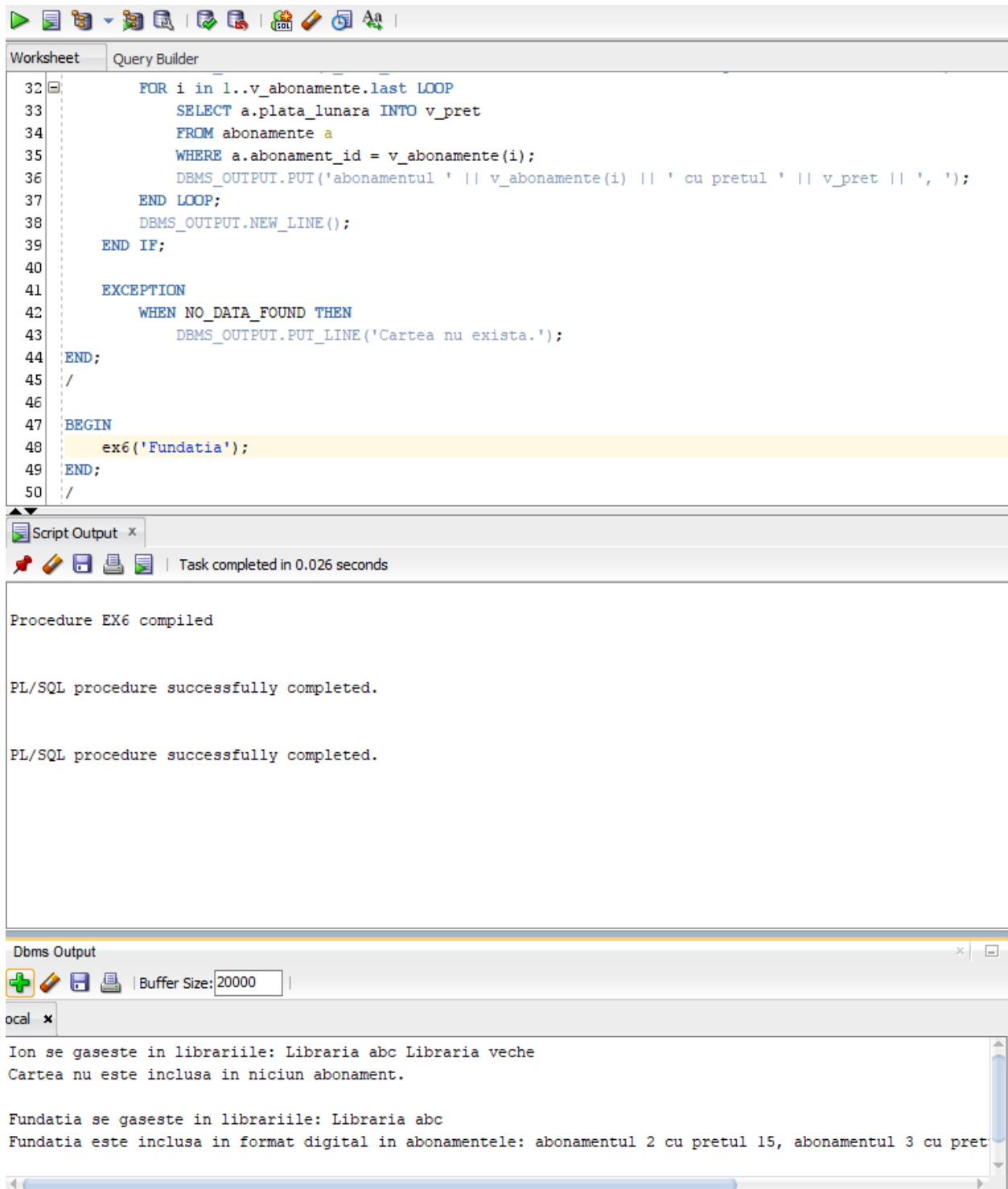
Dbms Output

Buffer Size: 20000

local x

Ion se gaseste in librariile: Libraria abc Libraria veche
Cartea nu este inclusa in niciun abonament.

Figure 10: Carte care nu există în vreun abonament



The screenshot displays the Oracle SQL Developer environment. The top pane shows a PL/SQL procedure named EX6. The procedure iterates through a list of subscriptions (v_abonamente) and for each, it queries the monthly fee (plata_lunara) from the abonamente table. It then outputs the subscription name and its fee. The procedure also includes an exception handler for the case where no data is found, outputting a message. The bottom pane shows the script output, indicating that the procedure was compiled and executed successfully. The bottom-most pane shows the DBMS output, which displays the results of the procedure's execution.

```
32 FOR i in 1..v_abonamente.last LOOP
33     SELECT a.plata_lunara INTO v_pret
34     FROM abonamente a
35     WHERE a.abonament_id = v_abonamente(i);
36     DBMS_OUTPUT.PUT('abonamentul ' || v_abonamente(i) || ' cu pretul ' || v_pret || ', ');
37 END LOOP;
38 DBMS_OUTPUT.NEW_LINE();
39 END IF;
40
41 EXCEPTION
42 WHEN NO_DATA_FOUND THEN
43     DBMS_OUTPUT.PUT_LINE('Cartea nu exista.');
```

END;

/

BEGIN

ex6('Fundatia');

END;

/

Script Output x

Task completed in 0.026 seconds

Procedure EX6 compiled

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Dbms Output

Buffer Size: 20000

ocal x

Ion se gaseste in librariile: Libraria abc Libraria veche

Cartea nu este inclusa in niciun abonament.

Fundatia se gaseste in librariile: Libraria abc

Fundatia este inclusa in format digital in abonamentele: abonamentul 2 cu pretul 15, abonamentul 3 cu pret

Figure 11: Carte inclusă în abonamente

The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL procedure named 'EX6'. The procedure iterates through a range of values (1 to v_abonamente.last) and for each value, it selects the monthly payment (plata_lunara) from the 'abonamente' table. It then outputs the subscription ID and the monthly payment. An exception handler is included to catch 'NO_DATA_FOUND' and output a message 'Cartea nu exista.'.

```
32 FOR i in 1..v_abonamente.last LOOP
33     SELECT a.plata_lunara INTO v_pret
34     FROM abonamente a
35     WHERE a.abonament_id = v_abonamente(i);
36     DBMS_OUTPUT.PUT('abonamentul ' || v_abonamente(i) || ' cu pretul ' || v_pret || ', ');
37 END LOOP;
38 DBMS_OUTPUT.NEW_LINE();
39 END IF;
40
41 EXCEPTION
42 WHEN NO_DATA_FOUND THEN
43     DBMS_OUTPUT.PUT_LINE('Cartea nu exista.');
```

The bottom pane, titled 'Script Output', shows the execution results. It indicates that the procedure 'EX6' was compiled successfully and executed without errors. The output is as follows:

```
Procedure EX6 compiled
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
```

The 'Dbms Output' pane at the bottom shows the actual output of the procedure:

```
Fundatia se gaseste in librariile: Libraria abc
Fundatia este inclusa in format digital in abonamentele: abonamentul 2 cu pretul 15, abonamentul 3 cu pret:
Cartea nu exista.
```

Figure 12: Carte inexistentă

7 Subprogram cu un tip de cursor studiat

Am scris o procedură care caută angajații cu o vechime dată ca parametru cursorului dintr-o librărie dată, de asemenea, ca parametru, și le mărește salariul cu 15

```
1  --0 procedura care cauta angajatii cu vechime data ca parametru
    dintr-o librărie data de asemenea ca parametru si le
        mareste
2  --salariul cu 15%.
3  CREATE OR REPLACE PROCEDURE ex7 (vechime NUMBER,
4  cod_librarie librării.librarie_id%TYPE)
5  IS
6  CURSOR c (pvec NUMBER, pcod librării.librarie_id%TYPE) IS
7  SELECT a.angajat_id, a.nume, a.prenume, a.manager_id
8  FROM angajati a, lucreaza_in li
9  WHERE TRUNC ((SYSDATE) - a.data_angajarii) / 365.5 >= pvec AND
        li.angajat_id = a.angajat_id AND li.librarie_id = pcod;
10 BEGIN
11 FOR i in c(vechime, cod_librarie) LOOP
12 IF i.manager_id IS NOT NULL THEN
13 UPDATE angajati
14 SET salariu = salariu + salariu * 15 / 100
15 WHERE angajat_id = i.angajat_id;
16 DBMS_OUTPUT.PUT_LINE('A_fost_marit_salariul_anagajtului\ei_' ||
        i.nume || '_' || i.prenume || '_');
17 END IF;
18 END LOOP;
19 END;
20 /
```

The screenshot displays a SQL development environment with three main panes:

- Query Builder:** Contains a PL/SQL script. The script starts with a WHERE clause filtering by date and employee ID. It then enters a loop over a collection 'c' containing library codes. For each library, it checks if an employee has a manager. If so, it updates the employee's salary by 15% and outputs a message. The script ends with a SELECT statement to view the 'angajati' table.
- Query Result:** Shows the result of the SELECT statement. It is a table with 8 rows and 8 columns: ANGAJAT_ID, MANAGER_ID, JOB_ID, SALARIU, NUME, PRENUME, DATA_NASTERII, and DATA_ANGAJARII.
- Dbms Output:** Shows the output of the DBMS_OUTPUT.PUT_LINE statements. It displays four messages, one for each employee whose salary was updated: Pop Andrei, Bob Alex, Ionescu Radu, and Petrescu Alex.

	ANGAJAT_ID	MANAGER_ID	JOB_ID	SALARIU	NUME	PRENUME	DATA_NASTERII	DATA_ANGAJARII
1	1	(null)	1	10000	Sefescu	Ion	11-OCT-72	23-JUL-96
2	2	1	2	3450	Pop	Andrei	02-JUL-75	03-FEB-98
3	3	1	2	3450	Bob	Alex	23-APR-81	10-JUN-99
4	4	1	4	2875	Ionescu	Radu	07-FEB-83	30-OCT-01
5	5	1	3	5750	Petrescu	Alex	11-MAY-87	17-JAN-07
6	6	1	2	4000	Marcu	Carl	13-JUN-79	01-MAR-98
7	7	1	2	4100	Andreescu	Andrei	10-OCT-69	07-FEB-98
8	8	1	2	4000	Adriana	Maria	13-JUN-89	19-APR-08

Dbms Output:

```

A fost marit salariul anagajtului\ei Pop Andrei .
A fost marit salariul anagajtului\ei Bob Alex .
A fost marit salariul anagajtului\ei Ionescu Radu .
A fost marit salariul anagajtului\ei Petrescu Alex .

```

Figure 13: Exemplu de execuție + tabel schimbat

8 Subprogram de tip funcție care lucrează cu minim 3 tabele

Am scris o funcție care, dându-se denumirea unui job, returnează un șir de numere ce reprezintă câți angajați au job-ul respectiv în fiecare librărie.

```
1 CREATE OR REPLACE TYPE int_table IS VARRAY(3) OF NUMBER;
2 /
3 --Dandu-se denumirea unui job, vrem un sir de numere ce
   reprezinta
4 --cati angajati au job-ul respectiv in fiecare librerie.
5 CREATE OR REPLACE FUNCTION f8 (job_name jobs.denumire%TYPE)
6 RETURN int_table
7 IS
8 v_job_id angajati.job_id%TYPE;
9 v_check NUMBER;
10 v_old_lib_val NUMBER := -1;
11 v_counter NUMBER := 1;
12 v_found_job NUMBER := 0;
13 retval int_table := int_table(0,0,0);
14 BEGIN
15 SELECT job_id INTO v_job_id
16 FROM jobs
17 WHERE denumire = job_name;
18
19 SELECT COUNT(*) INTO v_check -- verific daca exista angajati cu
   job-ul dat
20 FROM angajati a
21 WHERE a.job_id = v_job_id;
22 IF v_check = 0 THEN
23 RAISE_APPLICATION_ERROR(-20000, 'Nu exista angajati cu job-ul
   dat. ');
24 END IF;
25
26 FOR i IN (SELECT l.librarie_id, a.job_id, COUNT(*) c
27 FROM angajati a, lucreaza_in li, librarii l
28 WHERE a.angajat_id = li.angajat_id AND li.librarie_id = l.
   librarie_id
29 GROUP BY l.librarie_id, a.job_id
30 ORDER BY l.librarie_id) LOOP
31 IF v_old_lib_val = -1 THEN -- prima intrare in for
32 v_old_lib_val := i.librarie_id;
33 END IF;
34 IF i.librarie_id != v_old_lib_val THEN -- am ajuns in gruparea
   pentru urmatoarea librerie
35 v_counter := v_counter + 1;
36 v_old_lib_val := i.librarie_id;
37 IF v_found_job = 0 THEN
38 retval(v_counter - 1) := 0;
```

```
39  END IF;
40  v_found_job := 0;
41  END IF;
42
43  IF i.job_id = v_job_id THEN
44  v_found_job := 1;
45  retval(v_counter) := i.c;
46  END IF;
47  END LOOP;
48  RETURN retval;
49
50  EXCEPTION
51  WHEN NO_DATA_FOUND THEN
52  RAISE_APPLICATION_ERROR(-20000, 'Nu există jobul dat');
53  END;
54  /
```

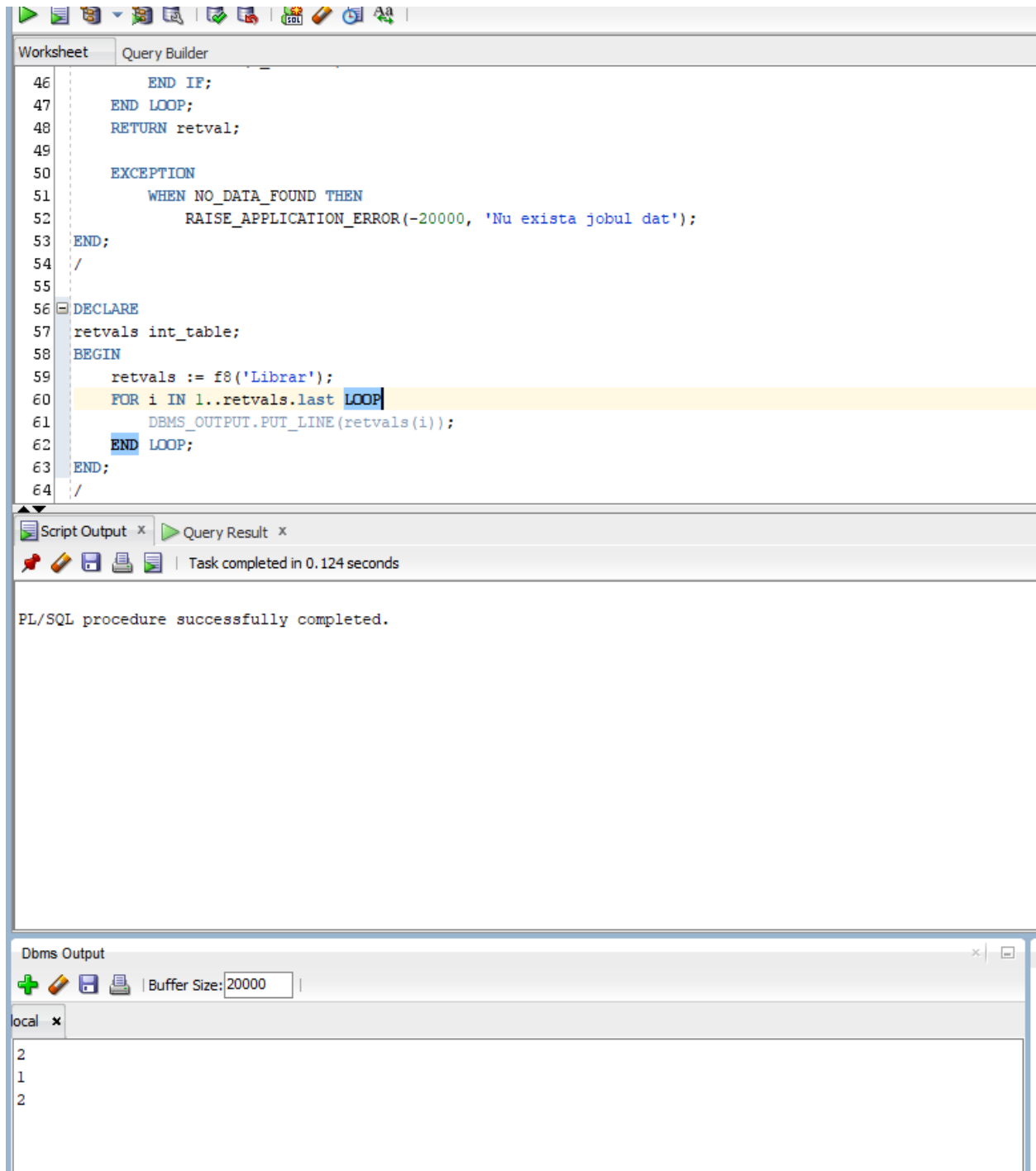


Figure 14: Fără excepții

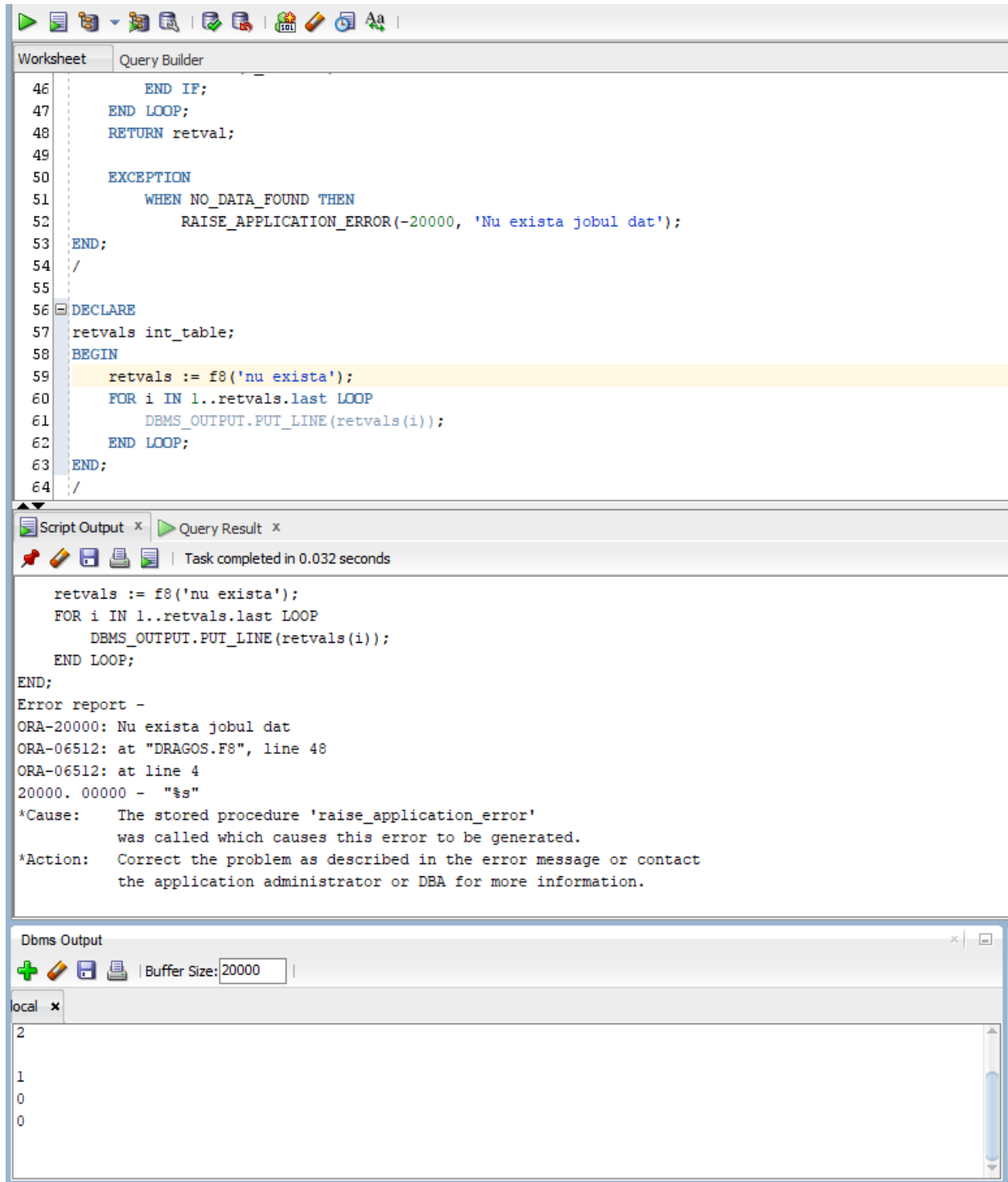
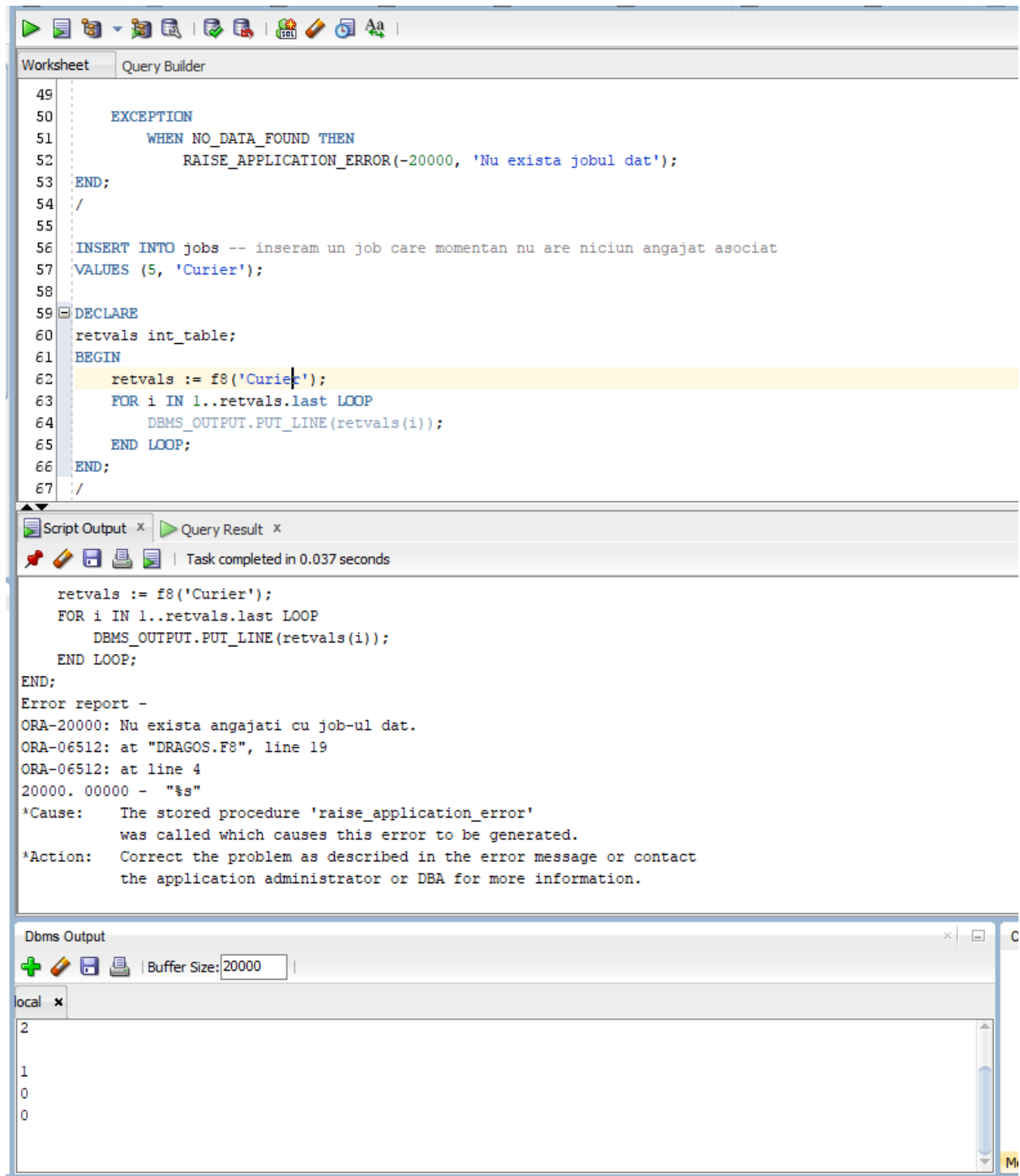


Figure 15: Job care nu există



```
49
50     EXCEPTION
51         WHEN NO_DATA_FOUND THEN
52             RAISE_APPLICATION_ERROR(-20000, 'Nu exista jobul dat');
53     END;
54 /
55
56 INSERT INTO jobs -- inseram un job care momentan nu are niciun angajat asociat
57 VALUES (5, 'Curier');
58
59 DECLARE
60     retvals int_table;
61 BEGIN
62     retvals := f8('Curier');
63     FOR i IN 1..retvals.last LOOP
64         DBMS_OUTPUT.PUT_LINE(retvals(i));
65     END LOOP;
66 END;
67 /
```

Script Output x Query Result x

Task completed in 0.037 seconds

```
retvals := f8('Curier');
FOR i IN 1..retvals.last LOOP
    DBMS_OUTPUT.PUT_LINE(retvals(i));
END LOOP;
END;
Error report -
ORA-20000: Nu exista angajati cu job-ul dat.
ORA-06512: at "DRAGOS.F8", line 19
ORA-06512: at line 4
20000. 00000 - "%s"
*Cause:      The stored procedure 'raise_application_error'
              was called which causes this error to be generated.
*Action:     Correct the problem as described in the error message or contact
              the application administrator or DBA for more information.
```

Dbms Output

Buffer Size: 20000

local x

```
2
1
0
0
```

Figure 16: Nu avem angajați cu job-ul dat

9 Subprogram de tip procedură care lucrează cu 5 tabele

Dându-se denumirea unei librării ca parametru, vreau să obțin toți membrii care au abonamente cu cărți din librărie.

```
1  --Vreau sa obtin o lista cu toti membrii care au abonamente cu
    carti din libraria data in procedura.
2  --Lista contine si numele abonamentului despre care este vorba.
3  CREATE OR REPLACE PROCEDURE p9 (lib_name librarii.denumire%TYPE
    )
4  IS
5  TYPE nume IS RECORD (abonament_id abonamente.abonament_id%TYPE,
6  nume membri.nume%TYPE,
7  prenume membri.prenume%TYPE);
8  TYPE name_table IS TABLE OF nume INDEX BY PLS_INTEGER;
9  v_lib_id librarii.librarie_id%TYPE;
10 flag NUMBER := 0;
11 v_data name_table;
12 BEGIN
13 SELECT librarie_id INTO v_lib_id
14 FROM librarii
15 WHERE denumire = lib_name;
16
17 SELECT a.abonament_id, m.nume, m.prenume BULK COLLECT INTO
    v_data
18 FROM membri m, abonamente a, carte_inclusa ci, carti c,
    se_afla_in sai
19 WHERE sai.librarie_id = v_lib_id AND sai.carte_id = c.carte_id
    AND c.carte_id = ci.carte_id AND ci.abonament_id = a.
    abonament_id AND m.abonament_id = a.abonament_id
20 GROUP BY a.abonament_id, m.nume, m.prenume;
21
22 IF v_data.count = 0 THEN
23 RAISE_APPLICATION_ERROR(-20000, 'Nu exista membrii cu
    abonamente care includ carti in libraria ceruta.');
```

```
24 END IF;
25
26 FOR i in 1..v_data.last LOOP
27 DBMS_OUTPUT.PUT_LINE('Abonamentul ' || v_data(i).abonament_id
    || ', detinut de membrul ' || v_data(i).nume || ' ' ||
    v_data(i).prenume || '.');
```

```
28 END LOOP;
29
30 EXCEPTION
```

```
31 WHEN NO_DATA_FOUND THEN
32 RAISE_APPLICATION_ERROR(-20000, 'Nu există librăria cerută. ');
33 END;
34 /
```

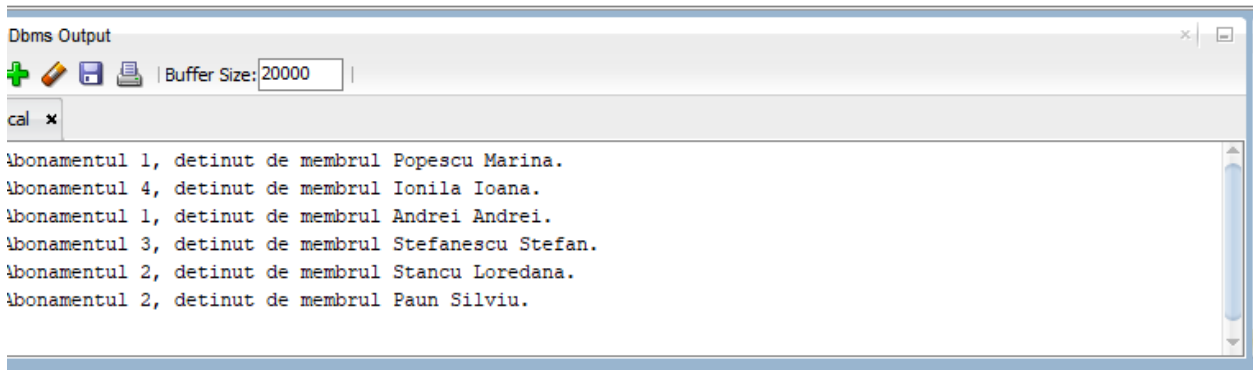
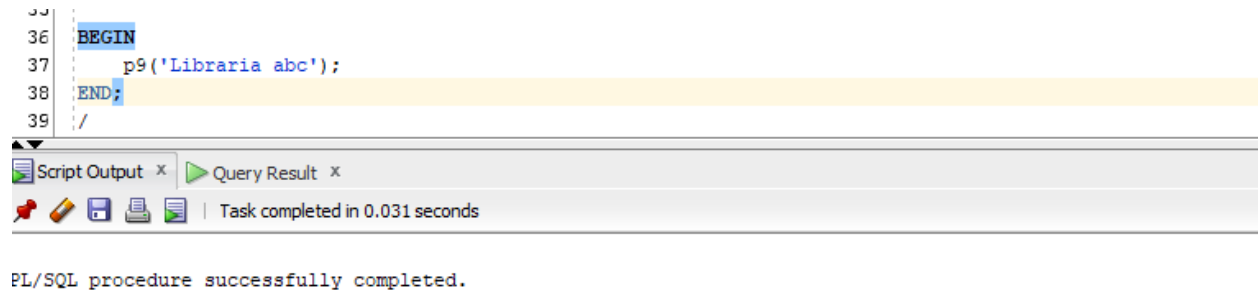


Figure 17: Fără excepții

```
34 /
35
36 BEGIN
37     p9('nu exista');
38 END;
39 /
```

Script Output x Query Result x

Task completed in 0.036 seconds

Error starting at line : 36 in command -
BEGIN
 p9('nu exista');
END;
Error report -
ORA-20000: Nu exista libraria ceruta.
ORA-06512: at "DRAGOS.P9", line 30
ORA-06512: at line 2
20000. 00000 - "%s"
*Cause: The stored procedure 'raise_application_error'
 was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
 the application administrator or DBA for more information.

Dbms Output

Buffer Size: 20000

ocal x

Abonamentul 1, detinut de membrul Popescu Marina.
Abonamentul 4, detinut de membrul Ionila Ioana.
Abonamentul 1, detinut de membrul Andrei Andrei.
Abonamentul 3, detinut de membrul Stefanescu Stefan.
Abonamentul 2, detinut de membrul Stancu Loredana.
Abonamentul 2, detinut de membrul Paun Silviu.

Figure 18: Librărie care nu există

```
34 /
35
36 INSERT INTO Librarii -- niste dummy data pt demonstrarea exceptiei cand libraria nu contine carti din vreun abonament
37 VALUES (4, 'Dummy', 'none');
38 BEGIN
39     p9('Dummy');
40 END;
```

Script Output x Query Result x

Task completed in 0.029 seconds

Error starting at line : 38 in command -

```
BEGIN
    p9('Dummy');
END;
```

Error report -

```
ORA-20000: Nu exista membrii cu abonamente care includ carti in libraria ceruta.
ORA-06512: at "DRAGOS.P9", line 21
ORA-06512: at line 2
20000. 00000 - "%s"
*Cause:      The stored procedure 'raise_application_error'
              was called which causes this error to be generated.
*Action:     Correct the problem as described in the error message or contact
              the application administrator or DBA for more information.
```

Figure 19: Librărie care nu conține cărți din vreun abonament

10 Trigger LMD la nivel de comandă

Am scris un trigger care interzice schimbarea tabelului Membrii în weekend sau în afara orelor de program ale IT-istului.

```
1 --Nu e permis sa fie schimbati membrii in timpul weekend-ului
   sau in afara orelor de program.
2 CREATE OR REPLACE TRIGGER T10
3 BEFORE INSERT OR DELETE OR UPDATE ON Membri
4 BEGIN
5 IF (TO_CHAR(SYSDATE, 'D') = 1 OR TO_CHAR(SYSDATE, 'D') = 7)
6 OR TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 8 AND 21 THEN
7 RAISE_APPLICATION_ERROR(-20001, 'Nu se pot face modificari la
   tabelul membrilor in acest interval orar');
8 END IF;
9 END;
10 /
```

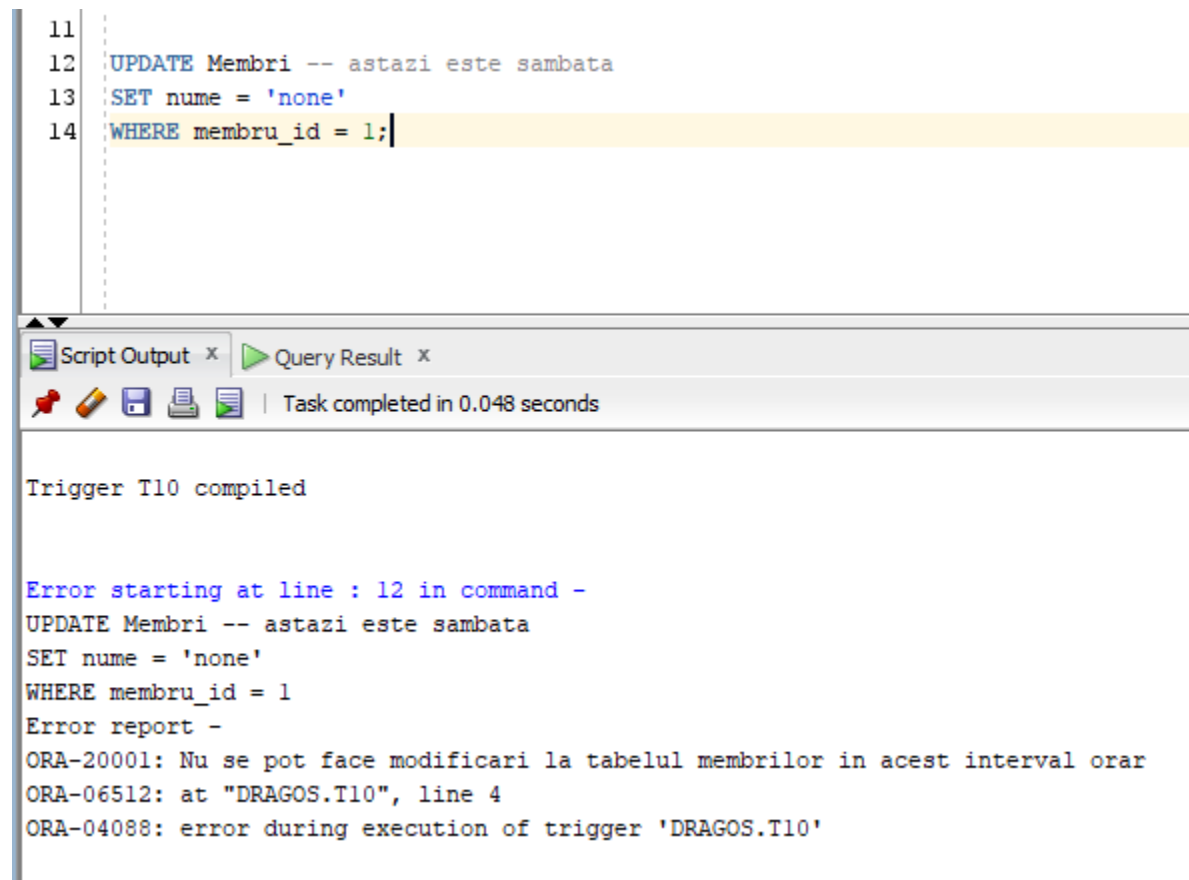


Figure 20: Încercare de update sâmbăta

11 Trigger LMD la nivel de linie

Am scris un trigger care, atunci când prețul unei cărți este schimbat, va reflecta schimbarea în prețul abonamentelor ce includ cartea. Deoarece abonamentele sunt deja destul de ieftine, schimbarea va fi efectuată doar când prețul unei cărți este mărit sau când, din păcate, este scoasă din stocul afacerii.

Deoarece un astfel de trigger ar solicita un select pe tabelul Cărți, pentru a calcula cât reprezintă ca procentaj din valoarea unui abonament o anumită carte, am avea de a face cu un trigger pe tabele mutating. Pentru a evita eroarea, am folosit un package în care, folosind un trigger LMD la nivel de comandă, voi stoca tabelul Cărți într-o colecție. Deși nu este necesar, am stocat și tabelul Abonamente, pentru a mai simplifica din codul trigger-ului la nivel de linie.

```
1  --Cand pretul unei carti este schimbat, vrem ca schimbarea sa
   fie reflectata si in abonament, in cazul in care scumpim
   cartea
2  --Daca scoatem o carte din stoc, vom ieftini abonamentul cu
   procentajul pe care pretul cartii il reprezenta din intregul
   pachet al abonamentului.
3  --Pt a evita trigger mutating, vom folosi o colectie intr-un
   pachet, care este initializata cu un trigger la nivel de
   comanda.
4  CREATE OR REPLACE PACKAGE trig_help
5  AS
6  TYPE abo IS TABLE OF abonamente%ROWTYPE INDEX BY PLS_INTEGER;
7  TYPE car IS TABLE OF carti%ROWTYPE INDEX BY PLS_INTEGER;
8  TYPE ci  IS TABLE OF carte_inclusa%ROWTYPE INDEX BY PLS_INTEGER
9  ;
10 abonamente_c abo;
11 carte_c car;
12 carte_inclusa_c ci;
13 end;
14 /
15 CREATE OR REPLACE TRIGGER T11_HELP
16 BEFORE UPDATE OR DELETE ON carti
17 BEGIN
18 SELECT * BULK COLLECT INTO trig_help.abonamente_c
19 FROM abonamente;
20 SELECT * BULK COLLECT INTO trig_help.carte_c
21 FROM carti;
22 SELECT * BULK COLLECT INTO trig_help.carte_inclusa_c
23 FROM carte_inclusa;
24 END;
25 /
```

```
25
26 CREATE OR REPLACE TRIGGER T11
27 BEFORE UPDATE OR DELETE ON carti
28 FOR EACH ROW
29 DECLARE
30 buffr number;
31 v_pret_total NUMBER(4);
32 v_pret_carte NUMBER(4);
33 v_crestere_prop NUMBER(4);
34 BEGIN
35 IF UPDATING THEN
36 IF :NEW.carte_id != :OLD.carte_id THEN
37 RAISE_APPLICATION_ERROR(-20001, 'Nu este permisa schimbarea
    cheiilor primare din tabelul Carte');
38 END IF;
39
40 FOR i in 1..trig_help.abonamente_c.last LOOP
41 v_pret_total := 0;
42 IF :NEW.pret > :OLD.pret THEN -- ne intereseaza sa modificam
    doar daca pretul cartii e crescut
43 FOR j in 1..trig_help.carte_c.last LOOP
44 buffr := 0;
45 SELECT COUNT(*) INTO buffr --nu o sa avem mutating, pt ca id-ul
    cartii nu se schimba
46 FROM carte_inclusa ci
47 WHERE ci.abonament_id = trig_help.abonamente_c(i).abonament_id
    AND ci.carte_id = trig_help.carte_c(j).carte_id;
48
49 IF buffr > 0 THEN
50 v_pret_total := v_pret_total + trig_help.carte_c(j).pret; --
    deoarece valorile sunt luate cu un trigger before, e ca si
    cum am apela :OLD pt valoarea schimbata
51 END IF;
52 END LOOP;
53
54 v_crestere_prop := :NEW.pret / :OLD.pret;
55
56 UPDATE abonamente -- crestem proportia
57 SET plata_lunara = plata_lunara + (:NEW.pret / v_pret_total) *
    plata_lunara
58 WHERE abonament_id = trig_help.abonamente_c(i).abonament_id;
59 END IF;
60 END LOOP;
61 ELSE --DELETING
62 FOR i in 1..trig_help.abonamente_c.last LOOP
```

```
63 v_pret_total := 0;
64 FOR j in 1..trig_help.carte_c.last LOOP
65   buffr := 0;
66   -- trebuie sa folosim colectia din pachet, deoarece id-ul
        cartii este scos si din carte_inclusa
67   FOR k in 1..trig_help.carte_inclusa_c.last LOOP
68     IF trig_help.carte_inclusa_c(k).abonament_id = trig_help.
        abonamente_c(i).abonament_id
69     AND trig_help.carte_inclusa_c(k).carte_id = trig_help.carte_c(j
        ).carte_id THEN
70       buffr := 1;
71     END IF;
72   END LOOP;
73
74   IF buffr > 0 THEN
75     v_pret_total := v_pret_total + trig_help.carte_c(j).pret; --
        deoarece valorile sunt luate cu un trigger before, e ca si
        cum am apela :OLD pt valoarea schimbata
76   END IF;
77   END LOOP;
78
79   UPDATE abonamente -- crestem proportia
80   SET plata_lunara = plata_lunara - (:OLD.pret / v_pret_total) *
        plata_lunara
81   WHERE abonament_id = trig_help.abonamente_c(i).abonament_id;
82   END LOOP;
83   END IF;
84   END;
85   /
```



```

6  nume_obiect VARCHAR2(50),
7  data DATE);
8
9  CREATE OR REPLACE TRIGGER T12
10 AFTER CREATE OR DROP OR ALTER ON SCHEMA
11 BEGIN
12 INSERT INTO audit_lib
13 VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
14 SYS.DICTIONARY_OBJ_NAME, SYSDATE);
15 END;
16 /

```

9
10 `select * from audit_lib;`

Script Output x Query Result x

SQL | Fetched 50 rows in 0.003 seconds

	UTILIZATOR	NUME_BD	EVENIMENT	NUME_OBIECT	DATA
1	DRAGOS	XE	CREATE	PROIECT	08-JAN-21
2	DRAGOS	XE	CREATE	PROIECT	08-JAN-21
3	DRAGOS	XE	ALTER	LIBRARIE	09-JAN-21
4	DRAGOS	XE	ALTER	CARTE	09-JAN-21
5	DRAGOS	XE	ALTER	ANGAJAT	09-JAN-21
6	DRAGOS	XE	ALTER	MEMBRU	09-JAN-21
7	DRAGOS	XE	ALTER	ABONAMENT	09-JAN-21
8	DRAGOS	XE	DROP	LUCREAZA_IN	09-JAN-21
9	DRAGOS	XE	DROP	SE_AFLA_IN	09-JAN-21
10	DRAGOS	XE	DROP	CARTE_INCLUSA	09-JAN-21
11	DRAGOS	XE	DROP	LIBRARII	09-JAN-21

Figure 23: Tabelul audit a fost creat înaintea restului de tabele

13 Pachet ce conține toate obiectele definite în cadrul proiectului

```
1 CREATE OR REPLACE PACKAGE proiect AS
2 TYPE int_table IS VARRAY(3) OF NUMBER;
3
4 PROCEDURE ex6 (v_num_carte carti.denumire%TYPE);
5 PROCEDURE ex7 (vechime NUMBER,
6 cod_librarie librarii.librarie_id%TYPE);
7 FUNCTION f8 (job_name jobs.denumire%TYPE)
8 RETURN int_table;
9 PROCEDURE p9 (lib_name librarii.denumire%TYPE);
10 END proiect;
11 /
12
13 CREATE OR REPLACE PACKAGE BODY proiect AS
14 PROCEDURE ex6 (v_num_carte carti.denumire%TYPE )IS
15 TYPE string_tabel IS TABLE OF VARCHAR2(60) INDEX BY PLS_INTEGER
16 ;
17 TYPE number_tabel IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
18 v_id_carte carti.carte_id%TYPE := NULL;
19 v_librarii string_tabel;
20 v_abonamente string_tabel;
21 v_pret abonamente.plata_lunara%TYPE;
22 BEGIN
23 SELECT carte_id INTO v_id_carte
24 FROM carti
25 WHERE v_num_carte = carti.denumire;
26
27 SELECT l.denumire BULK COLLECT INTO v_librarii
28 FROM librarii l, se_afla_in sai
29 WHERE sai.carte_id = v_id_carte AND sai.librarie_id = l.
30 librerie_id;
31
32 DBMS_OUTPUT.put(v_num_carte || ' se gaseste in librariile: ');
33 FOR i IN 1..v_librarii.last LOOP
34 DBMS_OUTPUT.put(v_librarii(i) || ' ');
35 END LOOP;
36 DBMS_OUTPUT.NEW_LINE();
37
38 SELECT a.abonament_id BULK COLLECT INTO v_abonamente
39 FROM abonamente a, carte_inclusa ci
40 WHERE ci.carte_id = v_id_carte AND ci.abonament_id = a.
41 abonament_id;
42
43 IF v_abonamente.count = 0 THEN
44 DBMS_OUTPUT.PUT_LINE('Cartea nu este inclusa in niciun
45 abonament.');
```

```
42 ELSE
43 DBMS_OUTPUT.PUT(v_nume_carte || ' este inclusă în format
    digital în abonamentele: ');
44 FOR i in 1..v_abonamente.last LOOP
45 SELECT a.plata_lunara INTO v_pret
46 FROM abonamente a
47 WHERE a.abonament_id = v_abonamente(i);
48 DBMS_OUTPUT.PUT('abonamentul ' || v_abonamente(i) || ' cu
    pretul ' || v_pret || ', ');
49 END LOOP;
50 DBMS_OUTPUT.NEW_LINE();
51 END IF;
52
53 EXCEPTION
54 WHEN NO_DATA_FOUND THEN
55 DBMS_OUTPUT.PUT_LINE('Cartea nu există. ');
56 END ex6;
57
58 PROCEDURE ex7 (vechime NUMBER,
59 cod_librarie librarii.librarie_id%TYPE)
60 IS
61 CURSOR c (pvec NUMBER, pcod librarii.librarie_id%TYPE) IS
62 SELECT a.angajat_id, a.nume, a.prenume, a.manager_id
63 FROM angajati a, lucreaza_in li
64 WHERE TRUNC ((SYSDATE) - a.data_angajarii) / 365.5 >= pvec AND
    li.angajat_id = a.angajat_id AND li.librarie_id = pcod;
65 BEGIN
66 FOR i in c(vechime, cod_librarie) LOOP
67 IF i.manager_id IS NOT NULL THEN
68 UPDATE angajati
69 SET salariu = salariu + salariu * 15 / 100
70 WHERE angajat_id = i.angajat_id;
71 DBMS_OUTPUT.PUT_LINE('A fost marit salariul angajatului\ei ' ||
    i.nume || ' ' || i.prenume || '. ');
72 END IF;
73 END LOOP;
74 END ex7;
75
76 FUNCTION f8 (job_name jobs.denumire%TYPE)
77 RETURN int_table
78 IS
79 v_job_id angajati.job_id%TYPE;
80 v_check NUMBER;
81 v_old_lib_val NUMBER := -1;
82 v_counter NUMBER := 1;
```

```
83 v_found_job NUMBER := 0;
84 retval int_table := int_table(0,0,0);
85 BEGIN
86 SELECT job_id INTO v_job_id
87 FROM jobs
88 WHERE denumire = job_name;
89
90 SELECT COUNT(*) INTO v_check -- verific daca exista angajati cu
    job-ul dat
91 FROM angajati a
92 WHERE a.job_id = v_job_id;
93 IF v_check = 0 THEN
94 RAISE_APPLICATION_ERROR(-20000, 'Nu exista angajati cu job-ul
    dat. ');
95 END IF;
96
97 FOR i IN (SELECT l.librarie_id, a.job_id, COUNT(*) c
98 FROM angajati a, lucreaza_in li, librarii l
99 WHERE a.angajat_id = li.angajat_id AND li.librarie_id = l.
    librarie_id
100 GROUP BY l.librarie_id, a.job_id
101 ORDER BY l.librarie_id) LOOP
102 IF v_old_lib_val = -1 THEN -- prima intrare in for
103 v_old_lib_val := i.librarie_id;
104 END IF;
105 IF i.librarie_id != v_old_lib_val THEN -- am ajuns in gruparea
    pentru urmatoarea librarie
106 v_counter := v_counter + 1;
107 v_old_lib_val := i.librarie_id;
108 IF v_found_job = 0 THEN
109 retval(v_counter - 1) := 0;
110 END IF;
111 v_found_job := 0;
112 END IF;
113
114 IF i.job_id = v_job_id THEN
115 v_found_job := 1;
116 retval(v_counter) := i.c;
117 END IF;
118 END LOOP;
119 RETURN retval;
120
121 EXCEPTION
122 WHEN NO_DATA_FOUND THEN
123 RAISE_APPLICATION_ERROR(-20000, 'Nu exista jobul dat');
```

```
124 END f8;
125
126 PROCEDURE p9 (lib_name librarii.denumire%TYPE)
127 IS
128 TYPE nume IS RECORD (abonament_id abonamente.abonament_id%TYPE,
129 nume membri.nume%TYPE,
130 prenume membri.prenume%TYPE);
131 TYPE name_table IS TABLE OF nume INDEX BY PLS_INTEGER;
132 v_lib_id librarii.librarie_id%TYPE;
133 flag NUMBER := 0;
134 v_data name_table;
135 BEGIN
136 SELECT librarie_id INTO v_lib_id
137 FROM librarii
138 WHERE denumire = lib_name;
139
140 SELECT a.abonament_id, m.nume, m.prenume BULK COLLECT INTO
141     v_data
142 FROM membri m, abonamente a, carte_inclusa ci, carti c,
143     se_afla_in sai
144 WHERE sai.librarie_id = v_lib_id AND sai.carte_id = c.carte_id
145     AND c.carte_id = ci.carte_id AND ci.abonament_id = a.
146     abonament_id AND m.abonament_id = a.abonament_id
147 GROUP BY a.abonament_id, m.nume, m.prenume;
148
149 IF v_data.count = 0 THEN
150 RAISE_APPLICATION_ERROR(-20000, 'Nu există membrii cu
151     abonamente care includ carti in biblioteca ceruta. ');
152 END IF;
153
154 FOR i in 1..v_data.last LOOP
155 DBMS_OUTPUT.PUT_LINE('Abonamentul ' || v_data(i).abonament_id
156     || ', detinut de membrul ' || v_data(i).nume || ' ' ||
157     v_data(i).prenume || '. ');
158 END LOOP;
159
160 EXCEPTION
161 WHEN NO_DATA_FOUND THEN
162 RAISE_APPLICATION_ERROR(-20000, 'Nu există biblioteca ceruta. ');
163 END p9;
164 END proiect;
165 /
```

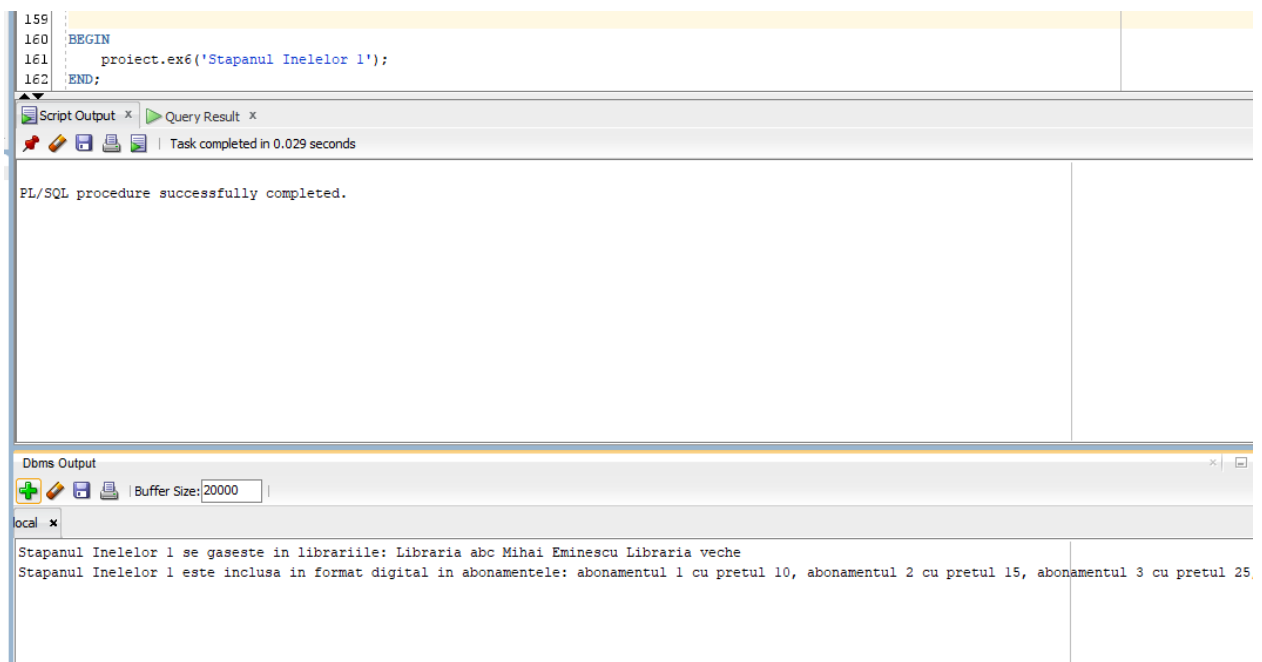


Figure 24: Ex 6 din proiect

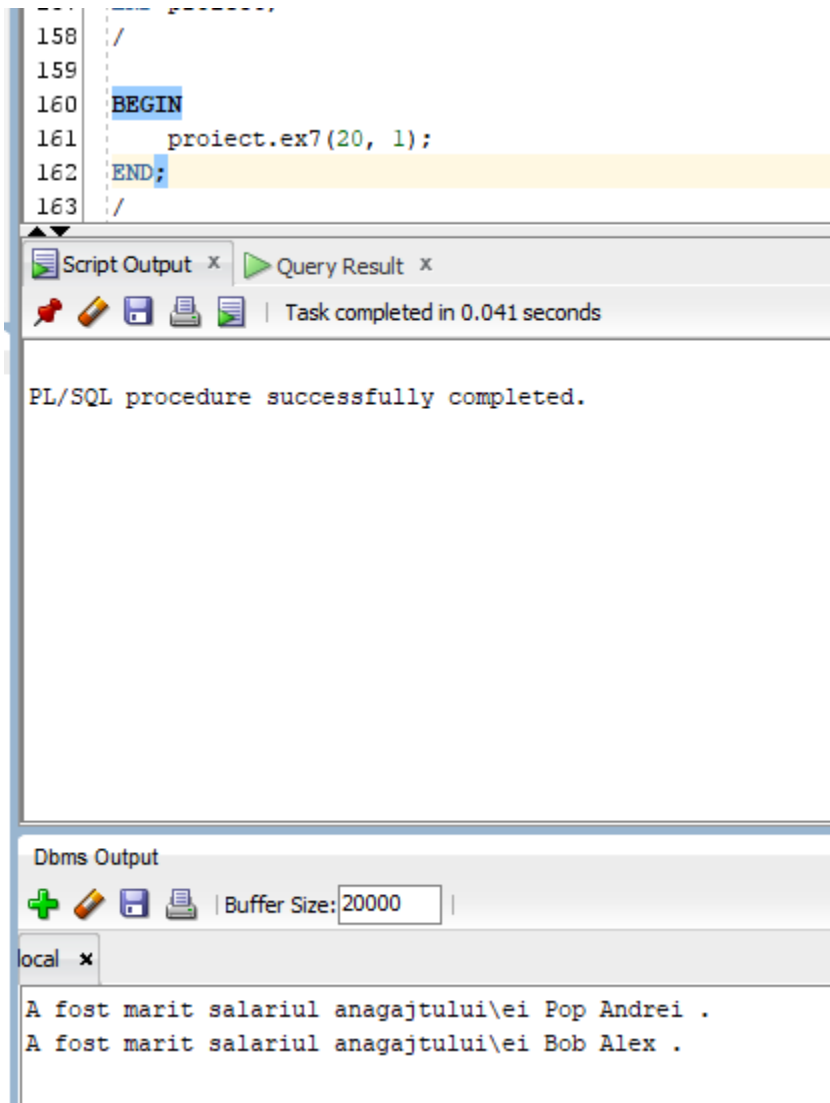


Figure 25: Ex 7 din proiect


```
159
160 DECLARE
161     angs proiect.int_table;
162 BEGIN
163     angs := proiect.f8('Librar');
164     FOR i IN 1..angs.last LOOP
165         dbms_output.put_line(angs(i));
166     END LOOP;
167 END;
168 /
```

Script Output x Query Result x

Task completed in 0.036 seconds

PL/SQL procedure successfully completed.

Dbms Output

Buffer Size: 20000

local x

```
2
1
2
```

Figure 26: Ex 8 din proiect

```
159  
160 BEGIN  
161     proiect.p9('Mihai Eminescu');  
162 END;  
163 /
```

Script Output x Query Result x

Task completed in 0.024 seconds

PL/SQL procedure successfully completed.

Dbms Output

Buffer Size: 20000

local x

```
Abonamentul 1, detinut de membrul Popescu Marina.  
Abonamentul 1, detinut de membrul Andrei Andrei.  
Abonamentul 3, detinut de membrul Stefanescu Stefan.  
Abonamentul 2, detinut de membrul Stancu Loredana.  
Abonamentul 2, detinut de membrul Paun Silviu.
```

Figure 27: Ex 9 din proiect