

Proiectarea și Interogarea unei baze de date pt DSP

PROIECT SGBD – AN 2 SEM 1
PAULA IUGA

CUPRINS:

1.Baza de date și utilitatea ei.....	2
2.Diagrama Entitate-Relatie.....	3
3. Diagrama Conceptuală	4
4. Implementarea Diagramei Conceptuale în Oracle	6
5. Adăugarea informațiilor în tabelele create.....	14
6. Un subprogram stocat care utilizează o colecție.....	26
7. Un subprogram stocat care utilizează un cursor	32
8. Un subprogram stocat de tip funcție care utilizează 3 tabele.Excepții.	35
9. Un subprogram stocat de tip procedură care utilizează 5 tabele. Excepții.....	43
10. Un trigger de tip LMD la nivel comandă.....	48
11. Un trigger de tip LMD la nivel linie	51
12. Un trigger de tip LDD	55
13. Un pachet care să conțină toate obiectele definite la cerințele anterioare	58
14. Un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate	71

1. Baza de date și utilitatea ei

Baza de date modelează succint modelul unei organizații de tip DSP și este inspirată din contextul actual al pandemiei de covid 19. Este realizată în vederea unei interogări mai ușoare a pacienților infectați, spitalizați și carantinați din cauza noului virus.

Baza de date se împarte în 3 categorii:

- Sistemul central de organizare. Acesta cuprinde angajații care pot fi de 3 feluri:
 1. programatorii care lucrează pe proiecte (de exemplu site-uri, aplicații web, robot telefonic) cu un anumit limbaj de programare.
 2. agenții teritoriale care investighează o anumită zonă (verifică dacă restaurantele, hotelurile și magazinele respectă normele de distanțare socială)
 3. agenții sau operatorii call-center care răspund persoanelor la apeluri telefonice. Aceștia se află într-un centru dintr-o anumită locație.
- Persoanele fizice. Acestea au un medic de familie, un pachet de servicii (ce conține o listă de servicii), se află într-o locație și anunță că au simptome sau au intrat în contact cu o persoană infectată. Li se poate lua un test covid, rezultatul acestuia este trecut în tabela lor în câmpul `test_covid` care are tipul Bool, adică în Oracle, Number(1) : 0 pt pacienții negativi, 1 pt pacienții pozitivi. Data testului este trecută în câmpul de tip date, `data_testare`.

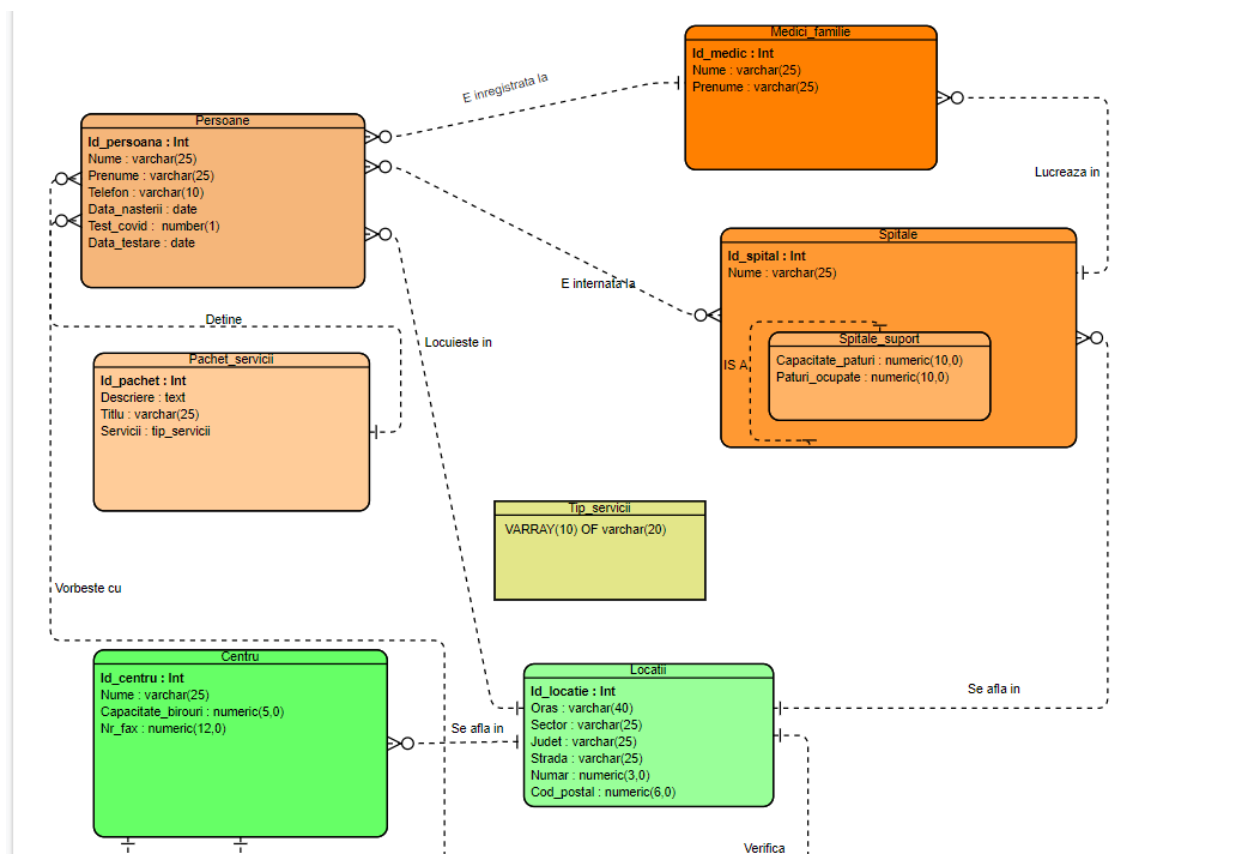
Nota! Câmpurile `test_covid` și `data_testare` rețin datele ultimului test făcut, dacă persoana nu a mai făcut teste vor fi null.

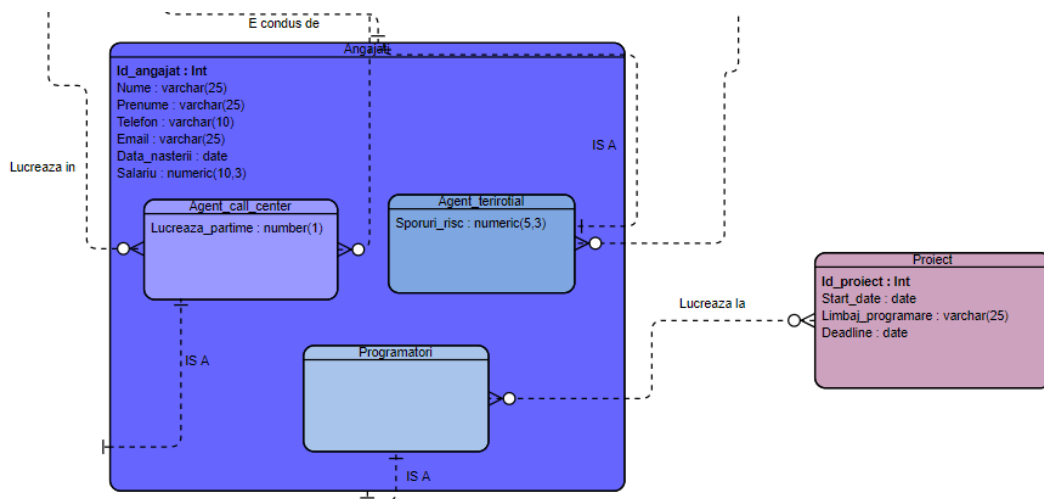
- Sistemul medical. Acesta cuprinde spitalele. In spitale lucreaza medici de familie iar unele spitale sunt suport covid adică aici se internează persoanele grav afectate de virus.

2.Diagrama Entitate-Relatie

Mai jos am inserat capturi de ecran cu diagrama Entitate-Relatie a modelului descris la punctul anterior. Aceasta a fost realizată în VP Online.

Nota! In folderul cu proiectul am atașat un pdf cu aceasta.





3. Diagrama Conceptuală

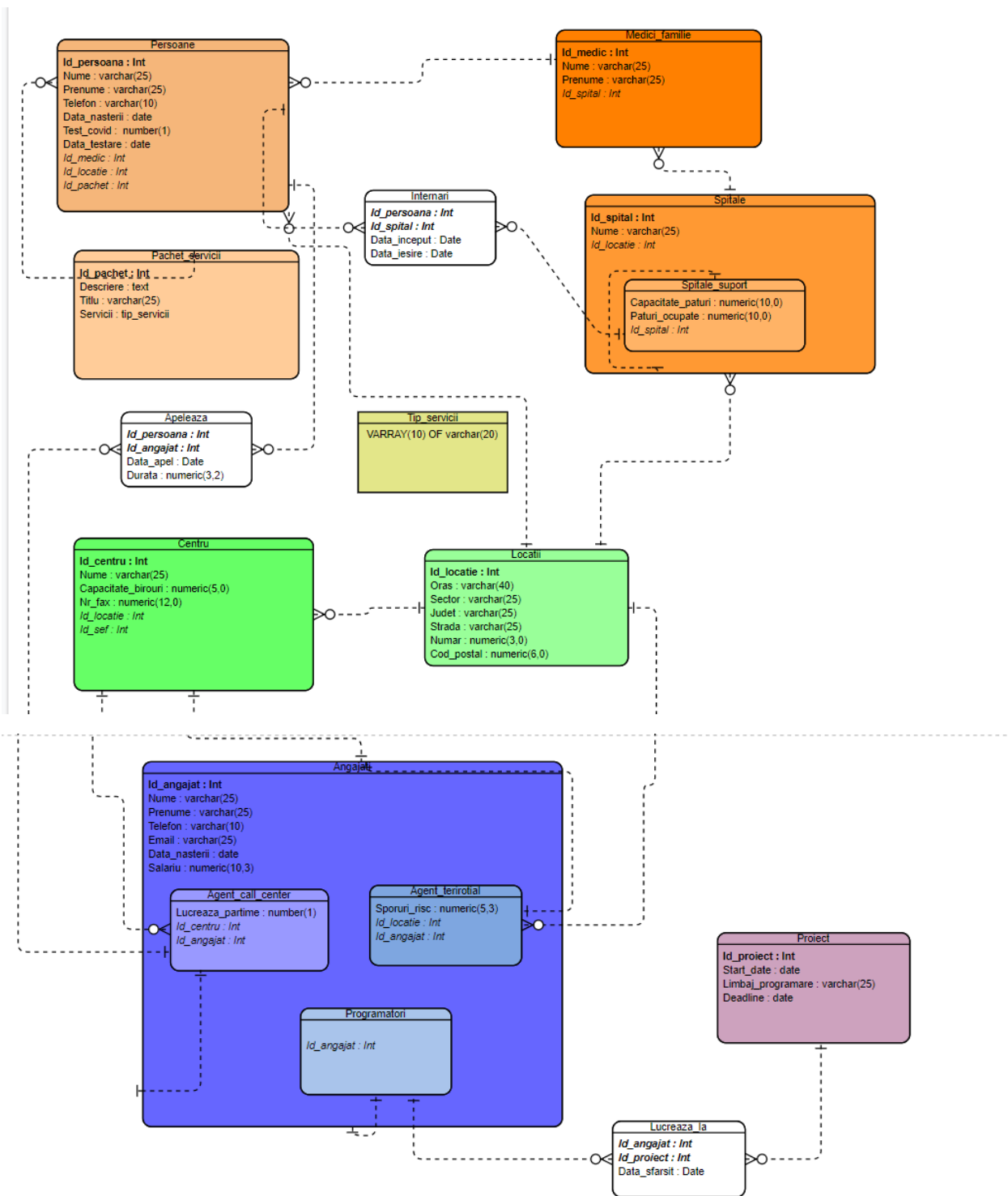
Pe baza diagramei Entitate – Relatie am construit diagrama Conceptuală transformând corespunzător relațiile și entitățile.

Principalele modificări au fost :

- Transformarea relațiilor de tip Many-to-Many în tabele asociative. Acestea au 2 chei externe pentru cele 2 tabele asociate iar cheia lor primară este compusă din combinarea celor 2 chei externe.
- Plasarea cheilor externe. In relatia Many-to-One cheia externă se plasează în tabelul care a determinat Many, iar în relația One-to-One cheia externă se plasează în tabelul care conține mai multe linii.

Mai jos am inserat capturi de ecran cu diagrama Conceptuală. Aceasta a fost realizată în VP Online.

Nota! In folderul cu proiectul am atașat un pdf cu aceasta.



4. Implementarea Diagramei Conceptuale în Oracle

Codul pt. creerea tabelelor și adăugarea constrângerilor în SQL Oracle:

```
CREATE OR REPLACE
```

```
TYPE TIP_SERVICII IS VARRAY(10) OF VARCHAR(20);
```

```
/
```

```
CREATE TABLE Pachet_servicii
```

```
(
```

```
Id_pachet INTEGER NOT NULL PRIMARY KEY,
```

```
Descriere VARCHAR2(200),
```

```
Titlu VARCHAR(25)
```

```
);
```

```
ALTER TABLE Pachet_servicii ADD Servicii TIP_SERVICII;
```

```
CREATE TABLE PERSOANE
```

```
(
```

```
Id_persoana INTEGER NOT NULL PRIMARY KEY,
```

```
Nume VARCHAR(25) NOT NULL,
```

```
Prenume VARCHAR(25) NOT NULL,
```

```
Telefon VARCHAR(10),
```

```
DATA_NASTERII DATE,
```

```
Test_covid Number(1),---Boolean, se pare ca sql nu are bool asa ca voi folosi number(1)
```

```
Id_medic INTEGER NOT NULL,  
Id_locatie INTEGER NOT NULL,  
Id_pachet INTEGER  
);
```

```
CREATE TABLE medici_familie  
(  
id_medic INTEGER NOT NULL PRIMARY KEY,  
nume VARCHAR(25),  
prenume VARCHAR(25),  
Id_spital INTEGER  
);
```

```
CREATE TABLE Spitale  
(  
Id_spital INTEGER NOT NULL PRIMARY KEY,  
Nume VARCHAR(25),  
Id_locatie INTEGER  
);
```

```
CREATE TABLE Spitale_suport  
(  
Id_spital INTEGER NOT NULL PRIMARY KEY,  
Capacitate_paturi NUMERIC(10,0),  
Paturi_ocupate NUMERIC(10,0)  
);
```

```
CREATE TABLE Internari  
(
```



```
Id_persoana INTEGER NOT NULL,  
Id_spital INTEGER NOT NULL,  
Data_inceput DATE not null,  
Data_iesire DATE,  
PRIMARY KEY (Id_persoana, Id_spital, Data_inceput)  
);
```

```
CREATE TABLE Locatii  
(  
Id_locatie INTEGER NOT NULL PRIMARY KEY,  
Oras VARCHAR(40),  
Sector VARCHAR(25),  
Judet VARCHAR(25),  
Strada VARCHAR(25),  
Numar NUMERIC(3,0),  
Cod_postal NUMERIC(6,0)  
);
```

```
CREATE TABLE Apeleaza  
(  
Id_persoana INTEGER NOT NULL,  
Id_angajat INTEGER NOT NULL,  
Data_apel DATE NOT NULL,  
DURATA NUMERIC(3,2),  
PRIMARY KEY (Id_persoana, Id_angajat),  
CONSTRAINT A_chk_durata CHECK (DURATA <120)  
);
```

```
CREATE TABLE Centre
```

```
(  
Id_centru INTEGER NOT NULL PRIMARY KEY,  
Nume VARCHAR(25),  
Capacitate_birouri NUMERIC(5,0),  
Nr_fax NUMERIC(12,0),  
Id_locatie INTEGER,  
Id_sef INTEGER  
);
```

CREATE TABLE ANGAJATI

```
(  
Id_angajat INTEGER NOT NULL PRIMARY KEY,  
Nume VARCHAR(25) NOT NULL,  
Prenume VARCHAR(25) NOT NULL,  
Telefon VARCHAR(10),  
Email VARCHAR(25),  
Data_nasterii DATE,  
Salariu NUMERIC(10,3)  
);
```

CREATE TABLE Agent_call_center

```
(  
Id_angajat INTEGER NOT NULL PRIMARY KEY,  
Lucreaza_partime NUMBER(1),  
Id_centru INTEGER  
);
```

CREATE TABLE Agent_teritorial

```
(
```

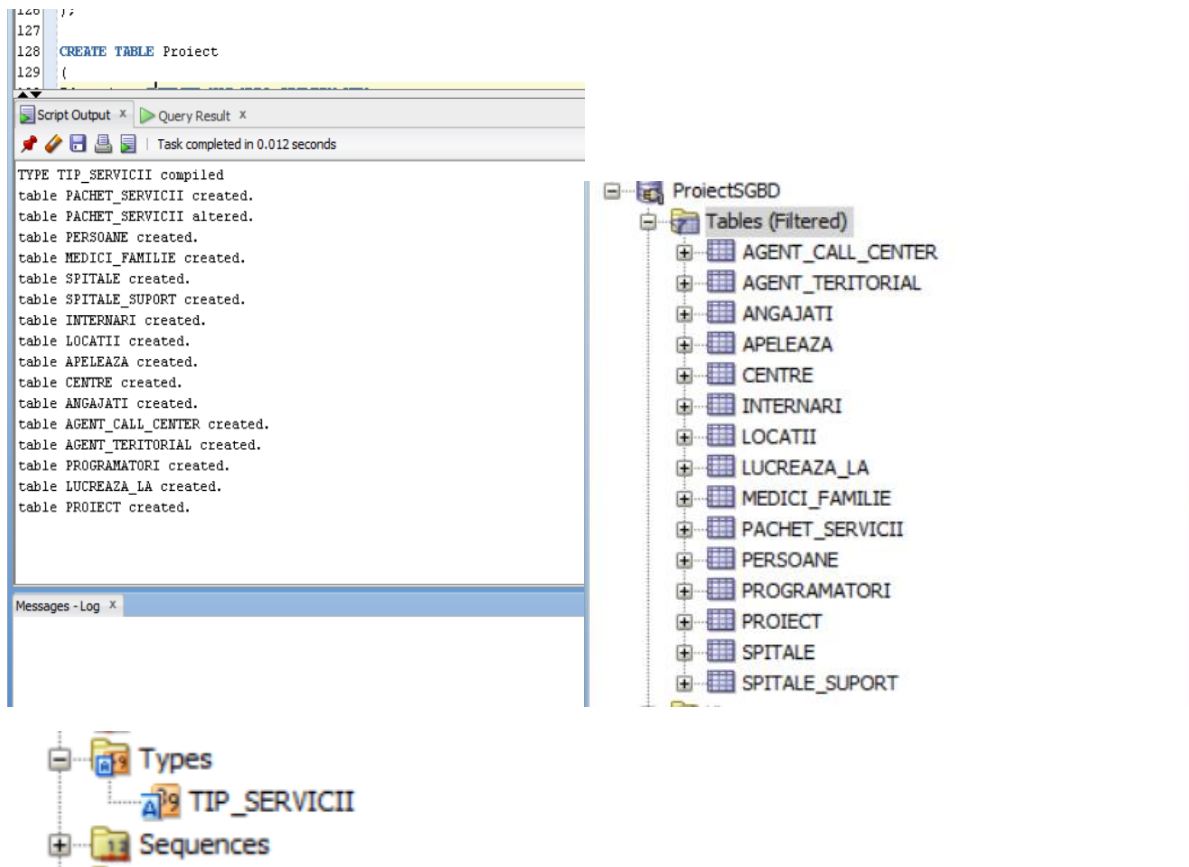
```
Id_angajat INTEGER NOT NULL PRIMARY KEY,  
Sporuri_risc NUMERIC(5,3),  
Id_locatie INTEGER  
);
```

```
CREATE TABLE Programatori  
(  
Id_angajat INTEGER NOT NULL PRIMARY KEY,  
Specializare_limbaj VARCHAR(45)  
);
```

```
CREATE TABLE Lucreaza_la  
(  
Id_angajat INTEGER NOT NULL,  
Id_proiect INTEGER NOT NULL,  
Data_sfarsit DATE,  
PRIMARY KEY (Id_proiect, Id_angajat)  
);
```

```
CREATE TABLE Proiect  
(  
Id_proiect INTEGER NOT NULL PRIMARY KEY,  
Start_date DATE,  
Limbaj_programare VARCHAR(25),  
Deadline DATE  
);
```

Codul pt creerea tabelelor funcționează și generează tabelele:



Codul pt adăugarea constrângerilor de cheie externă:

ALTER TABLE Persoane

```
ADD CONSTRAINT P_FK FOREIGN KEY(Id_medec) REFERENCES Medici_Familie(Id_medec);
```

ALTER TABLE Persoane

```
ADD CONSTRAINT P_FK2 FOREIGN KEY(Id_pachet) REFERENCES Pachet_Servicii(Id_pachet);
```

ALTER TABLE Persoane

```
ADD CONSTRAINT p_FK3 FOREIGN KEY(Id_locatie) REFERENCES Locatii(Id_locatie);
```

```
ALTER TABLE Medici_Familie
```

```
ADD CONSTRAINT MF_FK FOREIGN KEY(Id_spital) REFERENCES Spitale(Id_spital);
```

```
ALTER TABLE Spitale
```

```
ADD CONSTRAINT S_FK FOREIGN KEY(Id_locatie) REFERENCES Locatii(Id_locatie);
```

```
ALTER TABLE Spitale_Suport
```

```
ADD CONSTRAINT SS_FK FOREIGN KEY(Id_spital) REFERENCES Spitale(Id_spital);
```

```
ALTER TABLE Internari
```

```
ADD CONSTRAINT I_FK FOREIGN KEY(Id_persoana) REFERENCES Persoane(Id_persoana);
```

```
ALTER TABLE Internari
```

```
ADD CONSTRAINT I_FK2 FOREIGN KEY(Id_spital) REFERENCES Spitale(Id_spital);
```

```
ALTER TABLE Apeleaza
```

```
ADD CONSTRAINT A_FK FOREIGN KEY(Id_persoana) REFERENCES Persoane(Id_persoana);
```

```
ALTER TABLE Apeleaza
```

```
ADD CONSTRAINT A_FK2 FOREIGN KEY(Id_angajat) REFERENCES Agent_call_center(Id_angajat);
```

```
ALTER TABLE Centre
```

```
ADD CONSTRAINT C_FK FOREIGN KEY(Id_locatie) REFERENCES Locatii(Id_locatie);
```

```
ALTER TABLE Centre
```

```
ADD CONSTRAINT C_FK2 FOREIGN KEY(Id_sef) REFERENCES Angajati(Id_angajat);
```

```
ALTER TABLE Agent_call_center
```

```
ADD CONSTRAINT Ag_FK FOREIGN KEY(Id_angajat) REFERENCES Angajati(Id_angajat);
```

```
ALTER TABLE Agent_call_center
```

```
ADD CONSTRAINT Ag_FK2 FOREIGN KEY(Id_centru) REFERENCES Centre(Id_centru);
```

```
ALTER TABLE Agent_teritorial
```

```
ADD CONSTRAINT Agt_FK FOREIGN KEY(Id_angajat) REFERENCES Angajati(Id_angajat);
```

```
ALTER TABLE Agent_teritorial
```

```
ADD CONSTRAINT Agt_FK2 FOREIGN KEY(Id_locatie) REFERENCES Locatii(Id_locatie);
```

```
ALTER TABLE Programatori
```

```
ADD CONSTRAINT PR_FK FOREIGN KEY(Id_angajat) REFERENCES Angajati(Id_angajat);
```

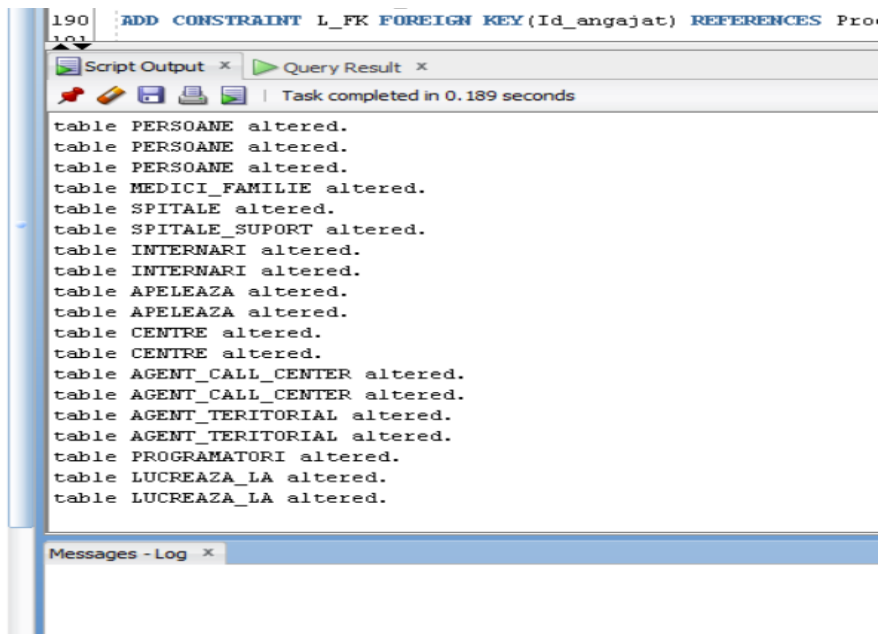
```
ALTER TABLE Lucreaza_la
```

```
ADD CONSTRAINT L_FK FOREIGN KEY(Id_angajat) REFERENCES Programatori(Id_angajat);
```

```
ALTER TABLE Lucreaza_la
```

```
ADD CONSTRAINT L_FK2 FOREIGN KEY(Id_proiect) REFERENCES Proiect(Id_proiect);
```

Codul pt adăugarea constrângerilor funcționează:



5. ADĂUGAREA INFORMAȚIILOR ÎN TABELELE CREATE

Codul pt popularea cu date a tabelor:

```
INSERT INTO LOCATII
```

```
VALUES(1,'Brasov', NULL, 'Brasov', 'Muresenilor', 24, 505600);
```

```
INSERT INTO LOCATII
```

```
VALUES(2,'Zarnesti', NULL, 'Brasov', 'Zorilor', null, 505800);
```

INSERT INTO LOCATII

VALUES(3, 'Codlea', NULL, 'Brasov', 'Apolodor', 10, 505900);

INSERT INTO LOCATII

VALUES(4, 'Harman', NULL, 'Brasov', 'Hiercher', 8, 505700);

INSERT INTO LOCATII

VALUES(5, 'Brasov', NULL, 'Brasov', 'Florilor', 13, 505600);

INSERT INTO LOCATII

VALUES(6, 'Bucuresti', '3', 'Bucuresti', 'Gura Calitei', 24 ,100590);

INSERT INTO LOCATII

VALUES(7, 'Bucuresti', '4', 'Bucuresti', 'Marioara', 10 ,101590);

INSERT INTO LOCATII

VALUES(8, 'Bucuresti', '1', 'Bucuresti', 'Academiei', null ,100690);

INSERT INTO LOCATII

VALUES(9, 'Bucuresti', '1', 'Bucuresti', 'Aerodromului', 9 ,100690);

INSERT INTO PROIECT

VALUES(1, SYSDATE, 'Python', TO_DATE('11-12-2021', 'MM-DD-YYYY'));

INSERT INTO PROIECT

VALUES(2, TO_DATE('11-12-2020', 'MM-DD-YYYY'), 'C++', TO_DATE('11-04-2021', 'MM-DD-YYYY'));

INSERT INTO PROIECT

VALUES(3, TO_DATE('08-12-2020', 'MM-DD-YYYY'), 'C++', TO_DATE('01-01-2021', 'MM-DD-YYYY'));

INSERT INTO PROIECT

VALUES(4, sysdate, 'PHP', TO_DATE('09-01-2021', 'MM-DD-YYYY'));

INSERT INTO PACHET_SERVICII

VALUES(1, 'Pachet complet analize si spitalizare', 'Pachet_Golden', tip_servicii('Analize sange', 'Ecografii', 'Internari', 'Tratament compensat'));

INSERT INTO PACHET_SERVICII

VALUES(2, 'Pachet doar pt analize medicale', 'Pachet_Silver', tip_servicii('Analize sange', 'Ecografii'));

INSERT INTO PACHET_SERVICII

VALUES(3, 'Pachet pt medicamente si tratamente', 'Pachet_White', tip_servicii('Pastile compensate', 'Analize sange', 'Tratamente fizio'));

INSERT INTO ANGAJATI

VALUES (1, 'Popa', 'Andrei', '0772298777', 'andreip@mail.com', TO_DATE('09-01-1970', 'MM-DD-YYYY'), 3500);

INSERT INTO ANGAJATI

VALUES (2, 'Popa', 'Diana Sandra', '0770178777', 'popasandrad@mail.com', TO_DATE('08-14-1975', 'MM-DD-YYYY'), 3500);

INSERT INTO ANGAJATI

VALUES (3, 'Anton', 'Ileana', '0770178112', 'antonile@mail.com', TO_DATE('10-30-1975', 'MM-DD-YYYY'), 4500);

INSERT INTO ANGAJATI

VALUES (4, 'Antonescu', 'Marcel', '0720938112', 'antonescu.marcel@mail.com', TO_DATE('10-31-1976', 'MM-DD-YYYY'), 4500);

INSERT INTO ANGAJATI

VALUES (5, 'Petru', 'Rares', '0720938132', 'petcu_raresica@mail.com', TO_DATE('04-11-1988', 'MM-DD-YYYY'), 4500);

INSERT INTO ANGAJATI

VALUES (6, 'Mircea', 'David', '0720008132', 'mircea_dav@mail.com', TO_DATE('04-11-1991', 'MM-DD-YYYY'), 4500);

INSERT INTO ANGAJATI

VALUES (7, 'Popescu', 'Denisa Antonia', '0751208132', 'popescu_antoniaD@mail.com', TO_DATE('06-12-1991', 'MM-DD-YYYY'), 4500);

INSERT INTO ANGAJATI

VALUES (8, 'Arens', 'Patricia', '0721248149', 'arensPatty@mail.com', TO_DATE('02-16-1993', 'MM-DD-YYYY'), 5000);

INSERT INTO ANGAJATI

VALUES (9, 'Wentzel', 'Sandra', '0720009992', 'wentzelSandra@mail.com', TO_DATE('04-17-1991', 'MM-DD-YYYY'), 5000);

INSERT INTO ANGAJATI

VALUES (10, 'Marginean', 'Laurentiu', '0721234132', 'margineanL@mail.com', TO_DATE('05-25-1990', 'MM-DD-YYYY'), 5000);

```
ALTER TABLE Agent_teritorial  
MODIFY (SPORURI_RISC NUMERIC(7,3) DEFAULT 1000);
```

```
INSERT INTO Agent_teritorial  
values (5, null, 1);
```

```
INSERT INTO Agent_teritorial (Id_angajat, Id_locatie)  
values(6, 4);
```

```
INSERT INTO Agent_teritorial (Id_angajat, Id_locatie)  
values(4, 6);
```

```
INSERT INTO Programatori  
VALUES (1, 'C++');
```

```
INSERT INTO Programatori  
VALUES (2, 'C++');
```

```
INSERT INTO Programatori  
VALUES (3, 'PHP');
```

```
INSERT INTO Programatori  
VALUES (7, 'PYTHON');
```

```
INSERT INTO Programatori  
VALUES (8, 'PYTHON');
```

```
SELECT * FROM Proiect;
```

```
INSERT INTO Lucreaza_la  
VALUES(1, 2, null);
```

```
INSERT INTO Lucreaza_la  
VALUES(2, 2, null);
```

```
INSERT INTO Lucreaza_la  
VALUES(7, 1, null);
```

```
INSERT INTO Lucreaza_la  
VALUES(8, 1, null);
```

```
INSERT INTO Lucreaza_la  
VALUES(3, 4, null);
```

```
INSERT INTO Lucreaza_la  
Values(7, 4, null);
```

```
INSERT INTO Centre  
VALUES (1, 'Centru DSP Bv1', 500, 123456789000, 1, 11);
```

```
INSERT INTO Centre  
VALUES (2, 'Centru DSP Buc1', 1000, 123456232000, 8, 10);
```

```
INSERT INTO Centre  
VALUES (3, 'Centru DSP Buc2', 2000, 122956232000, 7, 9);
```

```
INSERT INTO Agent_call_center  
VALUES(9, 0, 3);
```

```
INSERT INTO Agent_call_center  
VALUES(11, 0, 1);
```

```
INSERT INTO Agent_call_center  
VALUES(12, 1, 1);
```

```
INSERT INTO Agent_call_center  
VALUES(13, 1, 1);
```

```
INSERT INTO Agent_call_center  
VALUES(8, 0, 2);
```

```
INSERT INTO Agent_call_center  
VALUES(1, 0, 2);
```

```
INSERT INTO Agent_call_center  
VALUES(3, 0, 2);
```

```
INSERT INTO Spitale  
VALUES (1, 'Regina Maria', 1);
```

```
INSERT INTO Spitale  
VALUES (2, 'Caius Sparchez', 2);
```

```
INSERT INTO Spitale
```

```
VALUES (3, 'Sf Anton', 3);
```

```
INSERT INTO Spitale
```

```
VALUES (4, 'Infectioase', 5);
```

```
INSERT INTO Spitale
```

```
VALUES (5, 'Matei Bals', 8);
```

```
INSERT INTO Spitale
```

```
VALUES (6, 'Matei Basarab', 9);
```

```
INSERT INTO Spitale_Suport
```

```
VALUES (4, 2000, 100);
```

```
INSERT INTO Spitale_Suport
```

```
VALUES (2, 40, 10);
```

```
INSERT INTO Spitale_Suport
```

```
VALUES (5, 4000, 500);
```

```
INSERT INTO Medici_familie
```

```
VALUES(1, 'Pleasa', 'Mirela', 1);
```

```
INSERT INTO Medici_familie
```

```
VALUES(3, 'Matasa', 'Ciprian', 1);
```

```
INSERT INTO Medici_familie
```

```
VALUES(4, 'Iustin', 'Alin', 1);
```

```
INSERT INTO Medici_familie  
VALUES(2, 'Petrica', 'Radu', 2);
```

```
INSERT INTO Medici_familie  
VALUES(5, 'Amariei', 'Natalia', 2);
```

```
INSERT INTO Persoane  
VALUES (1, 'Popei', 'Stefan', '0773398405', TO_DATE('06-15-1960', 'MM-DD-YYYY'), 1, 1, 2, 1,  
TO_DATE('12-28-2021', 'MM-DD-YYYY'));
```

```
INSERT INTO Persoane  
VALUES (2, 'Anghel', 'Marius', '0733349405', TO_DATE('06-15-1959', 'MM-DD-YYYY'), 0, 3, 1, 1, sysdate);
```

```
INSERT INTO Persoane  
VALUES (3, 'Jidau', 'Marian', '0733249400', TO_DATE('06-20-1965', 'MM-DD-YYYY'), 0, 4, 2, 2, sysdate);
```

```
INSERT INTO Persoane  
VALUES (8, 'Iulius', 'Andrei', '0733249400', TO_DATE('06-20-1965', 'MM-DD-YYYY'), 0, 1, 3, 2, SYSDATE);
```

```
INSERT INTO Persoane  
VALUES (4, 'Stefanos', 'Anton', '0733249400', TO_DATE('06-20-1965', 'MM-DD-YYYY'), 1, 4, 2, 2,  
TO_DATE('12-02-2020', 'MM-DD-YYYY'));
```

```
INSERT INTO Persoane  
VALUES (5, 'Jeranu', 'Cosmin', '0733249225', TO_DATE('06-20-1999', 'MM-DD-YYYY'), 1, 2, 3, 3,  
TO_DATE('12-10-2020', 'MM-DD-YYYY'));
```

```
INSERT INTO Persoane
```

```
VALUES (6, 'Jerau', 'Antonia', '0733249225', TO_DATE('06-20-1999', 'MM-DD-YYYY'), 1, 2, 3, 3,  
TO_DATE('12-11-2020', 'MM-DD-YYYY'));
```

```
INSERT INTO Persoane
```

```
VALUES (7, 'Jerau', 'Antonia', '0733249225', TO_DATE('07-05-1999', 'MM-DD-YYYY'), 1, 2, 4, 1,  
TO_DATE('12-06-2020', 'MM-DD-YYYY'));
```

```
INSERT INTO Internari
```

```
VALUES(4,1,sysdate, sysdate + 14);
```

```
INSERT INTO Internari
```

```
VALUES(4,2,TO_DATE('12-02-2020', 'MM-DD-YYYY'), TO_DATE('12-02-2020', 'MM-DD-YYYY') + 14);
```

```
INSERT INTO Internari
```

```
VALUES(6,1,sysdate, sysdate + 14);
```

```
INSERT INTO Internari
```

```
VALUES(6,1,TO_DATE('06-20-1999', 'MM-DD-YYYY'), TO_DATE('06-20-1999', 'MM-DD-YYYY') + 14);
```

```
INSERT INTO Internari
```

```
VALUES(7,5, TO_DATE('12-06-2020', 'MM-DD-YYYY'), TO_DATE('12-06-2020', 'MM-DD-YYYY') + 14);
```

```
ALTER TABLE APELEAZA
```

```
MODIFY (Durata NUMERIC(4,0));
```

```
INSERT INTO Apeleaza
```

```
values(2, 9, sysdate, 5);
```



```
INSERT INTO Apeleaza
```

```
values(4, 11, sysdate, 10);
```

```
INSERT INTO Apeleaza
```

```
VALUES(4, 8, SYSDATE, 100);
```

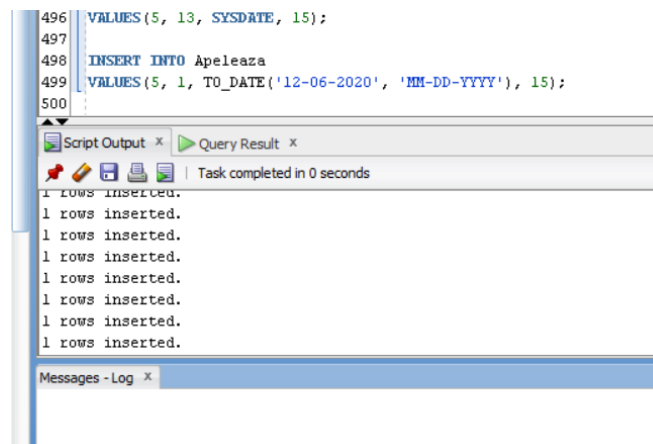
```
INSERT INTO Apeleaza
```

```
VALUES(5, 13, SYSDATE, 15);
```

```
INSERT INTO Apeleaza
```

```
VALUES(5, 1, TO_DATE('12-06-2020', 'MM-DD-YYYY'), 15);
```

Adăugarea înregistrărilor în tabele a fost realizată cu succes, în urma acestora am obținut o secvență:



Un exemplu de tabel populat independent ar fi tabela “Persoane”:

The screenshot shows a SQL query window with the following query:

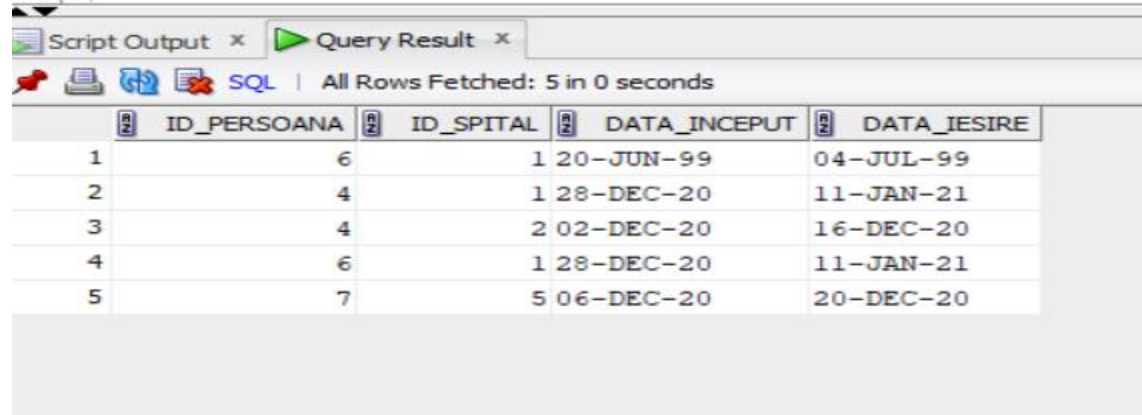
```
500
501 select * from Persoane;
502
```

Below the query, the 'Query Result' window displays the following data:

ID_PERSONA	NUME	PRENUME	TELEFON	DATA_NASTERII	TEST_COVID	ID_MEDIC	ID_LOCATIE	ID_PACHET	DATA_TESTARE
1	Popel	Stefan	0773398405	15-JUN-60	1	1	2		128-DEC-21
2	Anghel	Marius	0733349405	15-JUN-59	0	3	1		128-DEC-20
3	Jidau	Marian	0733249400	20-JUN-65	0	4	2		228-DEC-20
4	Iulius	Andrei	0733249400	20-JUN-65	0	1	3		228-DEC-20
5	Jerau	Antonia	0733249225	05-JUL-99	1	2	4		106-DEC-20
6	Jerau	Antonia	0733249225	20-JUN-99	1	2	3		311-DEC-20
7	Jerau	Cosmin	0733249225	20-JUN-99	1	2	3		310-DEC-20
8	Stefanos	Anton	0733249400	20-JUN-65	1	4	2		202-DEC-20

Iar un exemplu de tabel asociativ populat ar fi tabela "Internari":

```
502 SELECT * FROM Internari;  
503  
504
```



The screenshot shows a SQL query result window with the following data:

	ID_PERSOANA	ID_SPITAL	DATA_INCEPUT	DATA_IESIRE
1	6	1	20-JUN-99	04-JUL-99
2	4	1	28-DEC-20	11-JAN-21
3	4	2	02-DEC-20	16-DEC-20
4	6	1	28-DEC-20	11-JAN-21
5	7	5	06-DEC-20	20-DEC-20

6. Un subprogram stocat care utilizează o colecție

Voi crea un **subprogram stocat de tip procedură** care are **2 paramertii de ieșire** (primul care va întoarce câte persoane pozitive dețin pachete cu internare, iar al doilea numărul de persoane care nu dețin astfel de pachete, deci nu se pot interna la un spital de stat pe gratis).

Procedura va rezolva următoarea cerință:

Pentru toate persoanele testate pozitiv voi salva într-o **colecție de tip tablou imbricat obiecte** ce conțin id-ul lor și lista serviciilor din pachetul achiziționat.

- I. Pentru persoanele care nu dețin pachete cu Internare voi afișa daca au de plătit internarea si unde s-au internat până acum.
- II. Pentru cele tot fără acest pachet care nu s-au internat și mai sunt încă 14 zile de la data infectării voi afișa că se pot interna dacă plătesc serviciile.
- III. Pentru persoanele cu pachete care conțin și spitalizarea voi afișa fără plată.

Codul pentru sarcina descrisă mai sus:

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE TYPE pers_pachet IS OBJECT (cod_pesoana INTEGER, data_test DATE, pachet_servicii  
tip_servicii);
```

```
/
```

```
CREATE OR REPLACE PROCEDURE plata_spitalizare(nr_pers_pachet OUT NUMBER, nr_pers_fara_pachet OUT  
NUMBER)
```

```
IS
```

```
TYPE tablou_imbricat IS TABLE OF pers_pachet;
```

```
t tablou_imbricat := tablou_imbricat();
```

```
v_id_pers persoane.Id_persoana%TYPE;
```

```
v_id_pachet persoane.Id_pachet%TYPE;
```

```
v_data persoane.Data_testare%TYPE;
```

```

v_pachet tip_servicii;

contor NUMBER;

contor2 NUMBER;

v_ok NUMBER;

v_ok2 NUMBER;

v_nr NUMBER;


BEGIN


nr_pers_pachet := 0;

nr_pers_fara_pachet := 0;

contor := 1;


FOR i in (SELECT Id_persoana, Id_pachet, Data_testare
          FROM Persoane
          WHERE test_covid = 1
          ORDER BY Id_persoana)

LOOP

SELECT servicii INTO v_pachet
FROM pachet_servicii
WHERE Id_pachet = i.Id_pachet;

t.EXTEND;

t(contor) := pers_pachet(i.Id_persoana, i.Data_testare, v_pachet);

contor := contor + 1;

END LOOP;


FOR j IN t.FIRST..t.LAST

LOOP

```

```
DBMS_OUTPUT.PUT('Persoana cu id-ul ' || t(j).cod_pesoana || ' are test pozitiv si pachetul << ');
```

```
v_ok := 0;
```

```
v_nr := 0;
```

```
FOR k IN t(j).pachet_servicii.FIRST..t(j).pachet_servicii.LAST
```

```
LOOP
```

```
DBMS_OUTPUT.PUT(t(j).pachet_servicii(k) || ' ');
```

```
IF 'Internari' = t(j).pachet_servicii(k) AND v_ok = 0 THEN
```

```
v_ok := 1;
```

```
nr_pers_pachet := nr_pers_pachet + 1;
```

```
END IF;
```

```
END LOOP;
```

```
DBMS_OUTPUT.PUT('>> ');
```

```
IF v_ok = 0 THEN --nu are pachet care sa contina internari deci trebuie sa le plateasca
```

```
DBMS_OUTPUT.PUT(' trebuie sa plateasca internarile ');
```

```
v_ok2 := 0;
```

```
SELECT count(*) INTO v_nr
```

```
FROM spitale S JOIN internari i ON (S.id_spital = i.id_spital)
```

```
WHERE i.id_persoana = t(j).cod_pesoana;
```

```
IF v_nr = 0 THEN
```

```
DBMS_OUTPUT.PUT(' viitoare si nu s-a internat pana acum');
```

```
END IF;
```

```
FOR I IN (SELECT nume
```

```
FROM spitale S JOIN internari i ON (S.id_spital = i.id_spital)
```

```
WHERE i.id_persoana = t(j).cod_pesoana)
```

```
LOOP
```

```
IF v_ok2 = 0 THEN
```

```

        DBMS_OUTPUT.PUT('la ');

        DBMS_OUTPUT.PUT('*' || l.ume || '* ');

        v_ok2 := 1;
    ELSE
        DBMS_OUTPUT.PUT('*' || l.ume || '* ');
    END IF;
END LOOP;

ELSE
    DBMS_OUTPUT.PUT(' nu trebuie sa plateasca internarile');
END IF;

DBMS_OUTPUT.NEW_LINE();
END LOOP;

SELECT count(*) INTO nr_pers_fara_pachet
FROM Persoane
where Test_covid = 1;

nr_pers_fara_pachet := nr_pers_fara_pachet - nr_pers_pachet;

END plata_spitalizare;

/

```

Apelul procedurii:

```

DECLARE
v_nr1 NUMBER;
v_nr2 NUMBER;
BEGIN

```

```
plata_spitalizare(v_nr1, v_nr2);
```

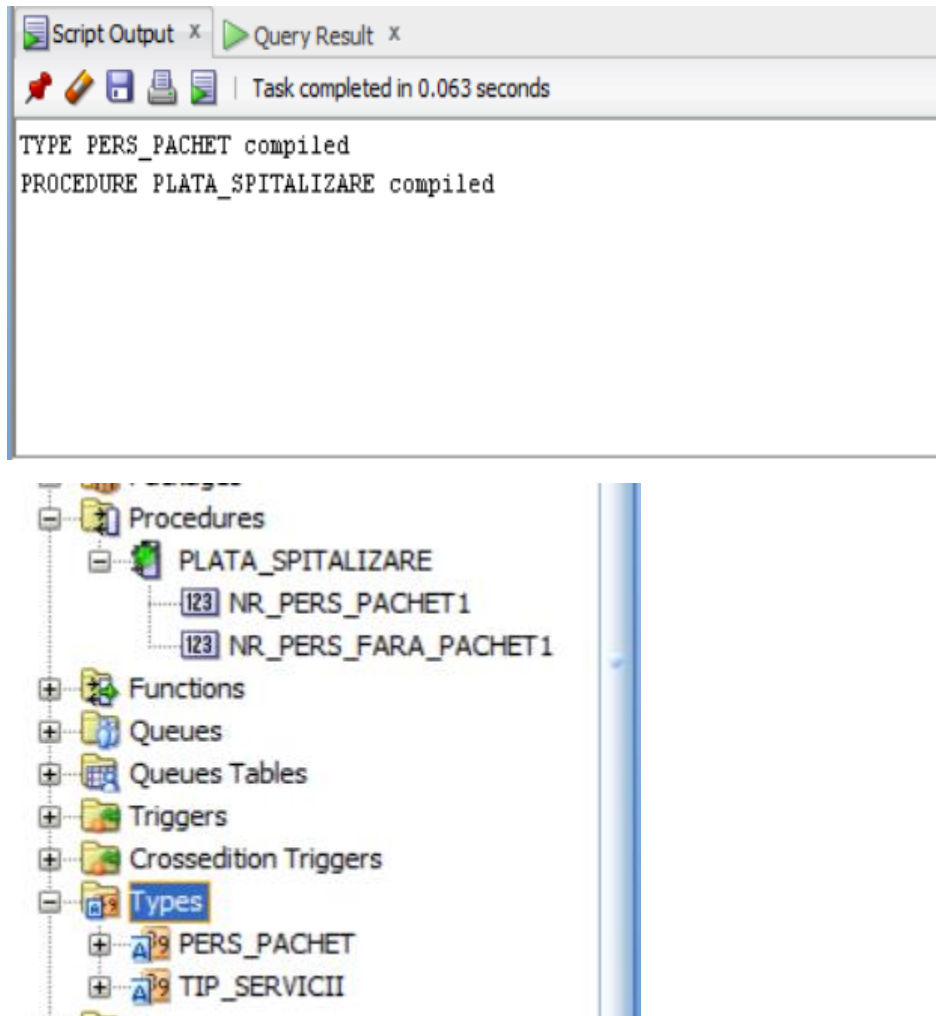
```
DBMS_OUTPUT.PUT_LINE('Numarul de persoane care au achizitionat un pachet CU internari este: ' || v_nr1);
```

```
DBMS_OUTPUT.PUT_LINE('Numarul de Persoane care au achizitionat un pachet FARA internari este: ' || v_nr2);
```

```
END;
```

```
/
```

Obiectul și procedura compilează:



Apelul procedurii generează:

```

Script Output x
Task completed in 0.007 seconds

PROCEDURE PLATA_SPITALIZARE compiled
anonymous block completed
Persoana cu id-ul 1 are test pozitiv si pachetul << Analize sange Ecografii Internari Tratament compensat >> nu trebuie sa plateasca internarile
Persoana cu id-ul 4 are test pozitiv si pachetul << Analize sange Ecografii >> trebuie sa plateasca internarile la "Regina Maria" "Caius Sparchez"
Persoana cu id-ul 5 are test pozitiv si pachetul << Pastile compensate Analize sange Tratamente fizio >> trebuie sa plateasca internarile viitoare si nu s-a internat pana acum
Persoana cu id-ul 6 are test pozitiv si pachetul << Pastile compensate Analize sange Tratamente fizio >> trebuie sa plateasca internarile la "Regina Maria" "Regina Maria"
Persoana cu id-ul 7 are test pozitiv si pachetul << Analize sange Ecografii Internari Tratament compensat >> nu trebuie sa plateasca internarile
Persoana cu id-ul 9 are test pozitiv si pachetul << Pastile compensate Analize sange Tratamente fizio >> trebuie sa plateasca internarile viitoare si nu s-a internat pana acum
Numarul de persoane care au achizitionat un pachet CU internari este: 2
Numarul de Persoane care au achizitionat un pachet FARA internari este: 4

```

- *Explicații cod:*

Comanda SET SERVEROUTPUT ON a fost utilizată pentru a putea vedea în Script Output ce afiseaza funcțiile PUT si NEW_LINE din pachetul DBMS_OUTPUT.

Am creat *un obiect pers_pachet* care retine tipuri de tip (int, data, vector de varchar uri).

Am folosit *o procedura stocata* (cu ajutorul comenzii *create*) cu *doi paramentrii de iesire (OUT)* pt. nr. pesoanelor infectate care au achiziționat un pachet ce conține spitalizare (acest număr este egal la final cu variabila de tip contor folosită pt. inserarea datelor in tabel) și nr. persoanelor care nu au un astfel de pachet (nr. total de persoane infectate minus primul paramentru de tip out deja determinat).

În interiorul procedurii am definit *un tablou imbricat vid de tip obiect pers_pachet* (fiecare linie inserată va fi de acest tip).

Cu un *ciclu-cursor cu subcereri* parcurg fiecare persoana pozitivă și îi salvez in *v_pachet* pachetul de servicii achizitionat, apoi inserez in tabelul *t*.

Al doilea *for loop* parcurge tabelul obținut și afișează informații despre persoanele pozitive. Al treilea *for loop* parcurge pt fiecare persoană *vectorul ei de servicii t(j).pachet_servicii* și verifică dacă unul din aceastea este egal cu Internari, apoi parcurg internările persoanei utilizând și *tabela asociativă internări*.

7. Un subprogram stocat care utilizează un cursor

Voi crea un **subprogram stocat de tip funcție** fără parametrii care **returnează** numărul de pachete (**Number**) care au serviciul Internare inclus folosind un **refcursor**.

Codul pentru sarcina descrisă mai sus:

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE FUNCTION numar_pachete_internare RETURN NUMBER
```

```
IS
```

```
TYPE refcursor IS REF CURSOR;
```

```
CURSOR c1 IS
```

```
    SELECT Id_pachet, Titlu,
```

```
           CURSOR(SELECT S.*
```

```
                FROM pachet_servicii P2, TABLE (P2.servicii) S
```

```
                WHERE P2.Id_Pachet = P.Id_Pachet)
```

```
    FROM Pachet_servicii P;
```

```
cursor_aux refcursor;
```

```
v_id Pachet_servicii.Id_pachet%TYPE;
```

```
v_titlu Pachet_servicii.Titlu%TYPE;
```

```
v_nr NUMBER;
```

```
serviciu varchar(20);
```

```
BEGIN
```

```
    v_nr := 0;
```

```
    OPEN c1;
```

LOOP

FETCH c1 INTO v_id, v_titlu, cursor_aux;

EXIT WHEN c1%NOTFOUND;

LOOP

FETCH cursor_aux INTO serviciu;

EXIT WHEN cursor_aux%NOTFOUND;

IF 'Internari' = Initcap(serviciu) THEN

v_nr := v_nr + 1;

DBMS_OUTPUT.PUT_LINE('PACHETUL ' || v_id || ' CU NUMELE ' || v_titlu || ' contine internari');

END IF;

END LOOP;

END LOOP;

CLOSE c1;

RETURN v_nr;

IF v_nr = 0 THEN

RAISE_APPLICATION_ERROR(-20001,'Nu exista pachet cu internare!');

RETURN -1;

END IF;

END numar_pachete_internare;

/

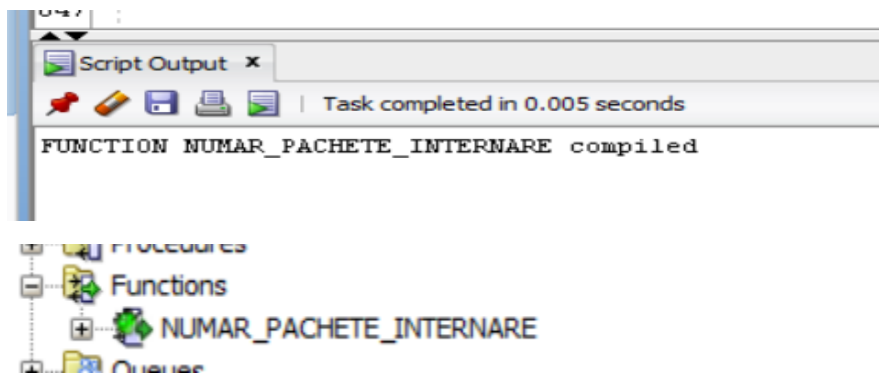
Apelul funcției în SQL* PLUS cu variabila HOST:

VARIABLE nr NUMBER

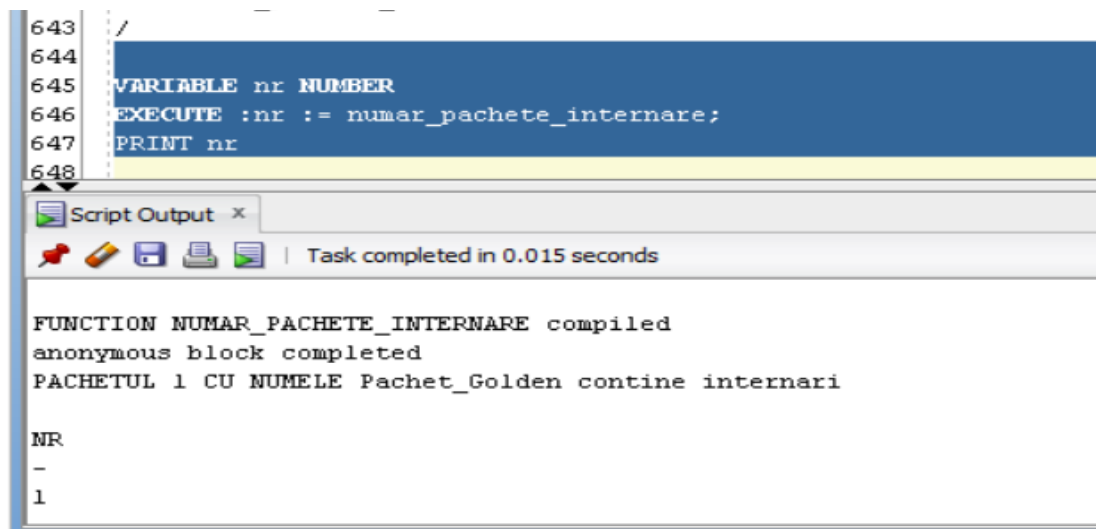
EXECUTE :nr := numar_pachete_internare;

PRINT nr

Funcția compilează:



Apelul funcției generează:



- *Explicații cod:*

Am definit un tip numit refcursor care este un **REF Cursor**, un **cursor explicit** c1 care reține pt fiecare pachet numele lui și id -ul, dar și un subcursor care reține pt acel pachet componentele pachetului de servicii (pt. aceasta am folosit **operatorul TABLE**, deoarece în tabelul Pachet_servicii, Serviciile unui pachet se află listate sub forma de **vector de varchar** și am vrut să le obțin pe fiecare separat).

Deschid cursorul c1, apoi într-un loop salvez fiecare informație în variabilele v_id, v_titlu și in Ref cursorul cursor_aux. Loopul se oprește când în cursor nu mai sunt date de adăugat. Apoi parcurg datele din ref cursor adică lista cu

servicii a fiecărui pachet și verific dacă unul dintre acestea este egal cu „Internari”, dacă da afișez și contorizez.

8. Un subprogram stocat de tip funcție care utilizează 3 tabele. Excepții.

Am creat o funcție care pentru un programator dat ca parametru cu numele de familie afișează domeniul său de specialitate, pe câte proiecte lucrează în prezent, deadline-ul lor și limbajul de programare. Funcția întoarce cât la sută reprezintă salariul lui din toate salariile alocate DSP-ului.

EXCEPȚII pentru:

- Nu există programator cu numele dat. Există mai mulți programatori cu numele dat.
- Programatorul nu are atribuită o specialitate
- Programatorul nu lucrează pe niciun proiect
- Programatorul nu are atribuit un salariu

Codul funcției descrise mai sus:

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE FUNCTION date_programator(param_nume Angajati.Nume%TYPE) RETURN FLOAT  
IS  
v_id Angajati.Id_angajat%TYPE;  
date_proiect Proiect%ROWTYPE;
```

```

procent_salariu FLOAT := 0.00;

specialitate Programatori.Specializare_limbaj%TYPE;

nr_proiecte NUMBER;

contor NUMBER := 0;

sal_total Angajati.Salariu%TYPE;

sal Angajati.Salariu%TYPE;


CURSOR c(param Angajati.Id_angajat%TYPE) IS

    SELECT Id_Proiect

    FROM Lucreaza_la

    where Id_angajat = param;


BEGIN

BEGIN

SELECT Id_angajat INTO v_id

FROM Programatori JOIN Angajati USING (Id_Angajat)

WHERE Nume = param_nume;

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        RAISE_APPLICATION_ERROR(-20000, 'Nu exista programator cu numele dat!');

        RETURN -1.00;

    WHEN TOO_MANY_ROWS THEN

        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi programatori cu numele dat!');

        RETURN -2.00;

    WHEN OTHERS THEN

        RAISE_APPLICATION_ERROR(-20002, 'S-a generat alta eroare!');

        RETURN -3.00;

END;


BEGIN

    SELECT Specializare_limbaj INTO specialitate

```

```

FROM Programatori

WHERE Id_Angajat = v_id;

IF specialitate is null THEN

    RAISE_APPLICATION_ERROR(-20003, 'Progamatorului cu numele ' || param_nume || ' nu i s-a atribuit o
    specializare!');

    RETURN -4.00;

END IF;

END;

BEGIN

SELECT COUNT(*) INTO nr_proiecte

FROM Lucreaza_la

WHERE Id_angajat = v_id

GROUP BY Id_angajat;

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        RAISE_APPLICATION_ERROR(-20005, 'Progamatorului cu numele ' || param_nume || ' nu are proiecte!');

        RETURN -5.00;

END;

DBMS_OUTPUT.PUT_LINE('Progamatorul cu numele ' || param_nume || ' lucreaza in prezent pe ' ||
nr_proiecte || ' proiecte si are specializarea ' || specialitate);

IF nr_proiecte <> 0 THEN --daca lcureaza pe mai mult de 0 proiecte vedem care sunt

FOR i IN c(v_id) LOOP

    SELECT * INTO date_proiect

    FROM PROIECT

    WHERE Id_proiect = i.Id_proiect;

```

```

        DBMS_OUTPUT.PUT_LINE('Proiectul cu nr de ordine ' || contor || ' are startdate ' ||
date_proiect.Start_date || ' deadline ' || date_proiect.Deadline || ' si este lucrat in limbajul ' ||
date_proiect.Limbaj_programare);

        contor := contor + 1;

    END LOOP;

END IF;


SELECT SUM(NVL(Salariu,0)) INTO sal_total
FROM ANGAJATI;


BEGIN

SELECT Salariu INTO sal
FROM ANGAJATI
WHERE Id_angajat = v_id;

IF sal is null THEN

    RAISE_APPLICATION_ERROR(-20004, 'Programatorului ' || param_nume || ' nu i s-a alocat salariu!');

    RETURN -4.00;

END IF;


END;


procent_salariu := sal*100/sal_total;

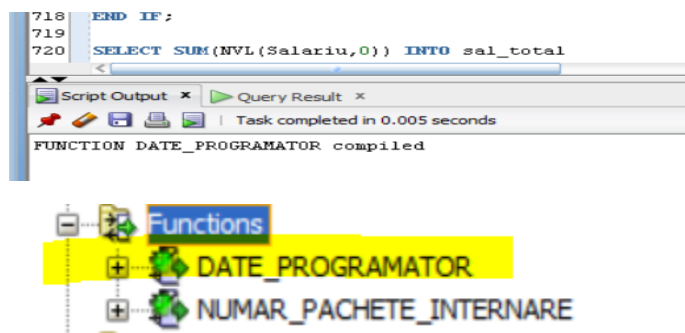
RETURN procent_salariu;


END date_programator;

/

```

Funcția compilează:

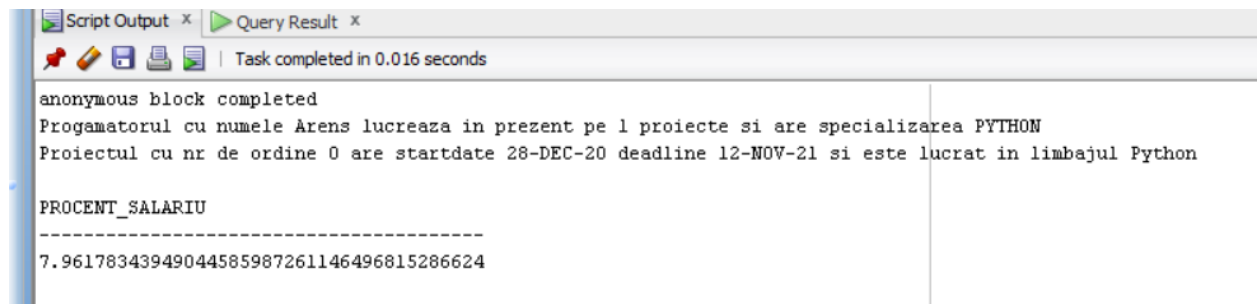


Apelul funcției:

```
VARIABLE procent_salariu NUMBER
```

```
EXECUTE :procent_salariu :=date_programator('Arens');
```

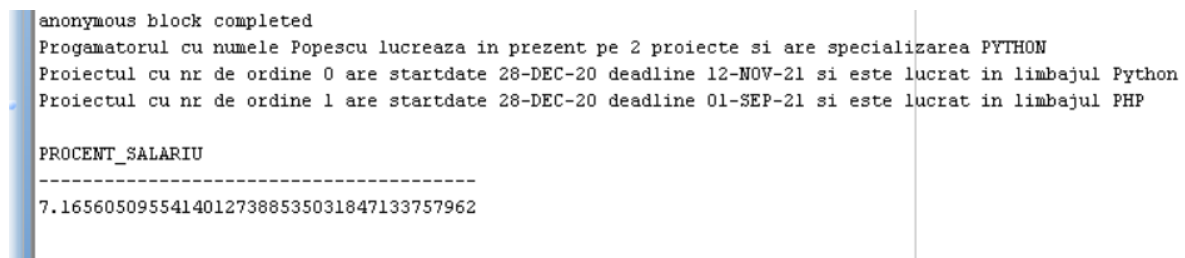
```
PRINT procent_salariu
```



```
VARIABLE procent_salariu NUMBER
```

```
EXECUTE :procent_salariu :=date_programator('Popescu');
```

```
PRINT procent_salariu
```



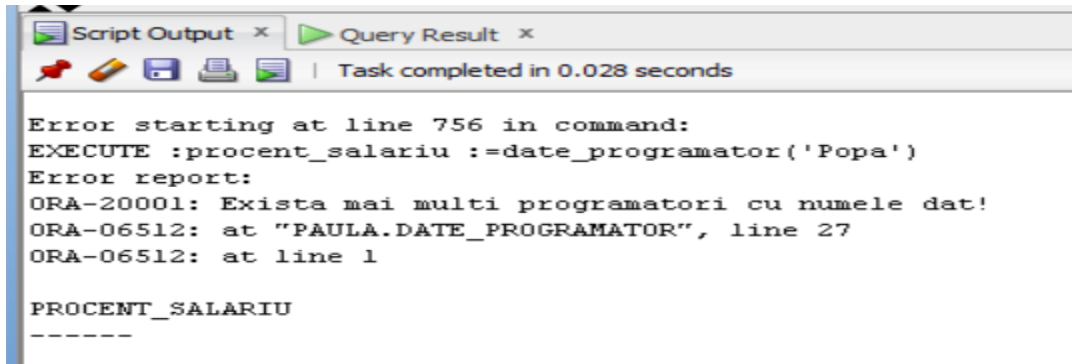
Pt excepții:

```
VARIABLE procent_salariu NUMBER
```



```
EXECUTE :procent_salariu :=date_programator('Popa');
```

```
PRINT procent_salariu
```



The screenshot shows a 'Script Output' window in SQL Developer. The title bar includes 'Script Output x' and 'Query Result x'. Below the title bar, there are icons for redo, undo, save, and print, followed by the text 'Task completed in 0.028 seconds'. The main text area displays the following error message:

```
Error starting at line 756 in command:
EXECUTE :procent_salariu :=date_programator('Popa')
Error report:
ORA-20001: Exista mai multi programatori cu numele dat!
ORA-06512: at "PAULA.DATE_PROGRAMATOR", line 27
ORA-06512: at line 1

PROCENT_SALARIU
-----
```

--inseram un programator fara proiecte se observa ca exceptia de proiecte era prinsa inaintea celei de salariu null

```
INSERT INTO ANGAJATI
```

```
VALUES (16, 'Matei', 'Aelxandru', '0720938672', 'mateiAlex@mail.com', TO_DATE('04-11-1988', 'MM-DD-YYYY'), NULL);
```

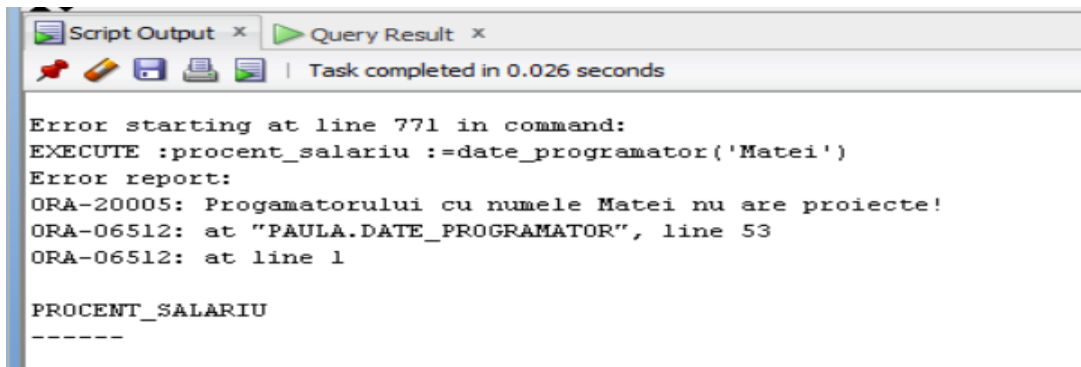
```
INSERT INTO PROGRAMATORI
```

```
VALUES (16,'Java');
```

```
VARIABLE procent_salariu NUMBER
```

```
EXECUTE :procent_salariu :=date_programator('Matei');
```

```
PRINT procent_salariu
```



The screenshot shows a 'Script Output' window in SQL Developer. The title bar includes 'Script Output x' and 'Query Result x'. Below the title bar, there are icons for redo, undo, save, and print, followed by the text 'Task completed in 0.026 seconds'. The main text area displays the following error message:

```
Error starting at line 771 in command:
EXECUTE :procent_salariu :=date_programator('Matei')
Error report:
ORA-20005: Programatorului cu numele Matei nu are proiecte!
ORA-06512: at "PAULA.DATE_PROGRAMATOR", line 53
ORA-06512: at line 1

PROCENT_SALARIU
-----
```

--inseram un programator cu salariul null dar cu proiecte

```
INSERT INTO ANGAJATI
```

```
VALUES (17, 'Constantinescu', 'David', '0720932372', 'ConstDavid@mail.com', TO_DATE('04-11-1988', 'MM-DD-YYYY'), NULL);
```

```
INSERT INTO PROGRAMATORI
```

```
VALUES (17, 'Java');
```

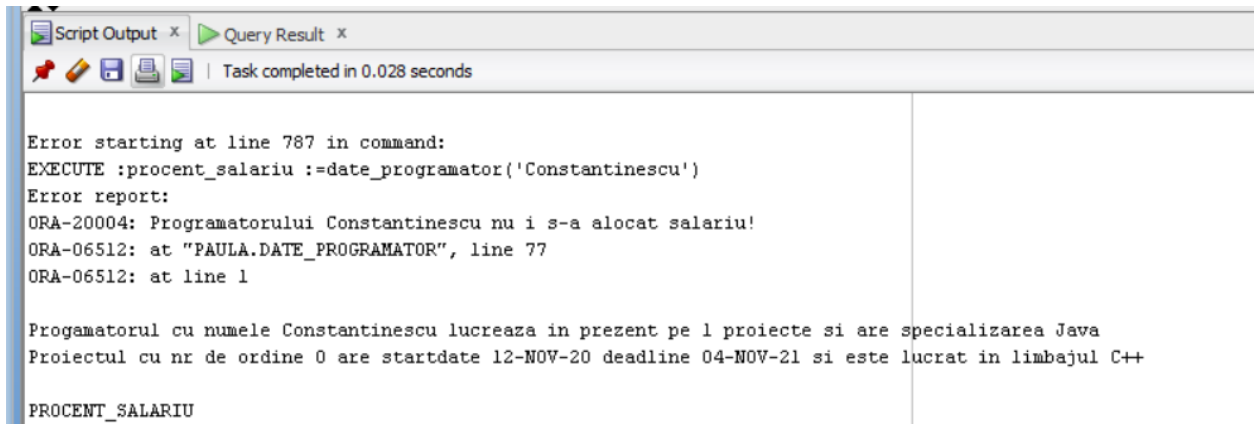
```
INSERT INTO Lucreaza_la
```

```
VALUES (17, 2, null);
```

```
VARIABLE procent_salariu NUMBER
```

```
EXECUTE :procent_salariu := date_programator('Constantinescu');
```

```
PRINT procent_salariu
```



```
--inseram un programator cu fara specialitate
```

```
INSERT INTO ANGAJATI
```

```
VALUES (18, 'Constantin', 'Ilie', '0720932372', 'IlieC@mail.com', TO_DATE('04-11-1988', 'MM-DD-YYYY'), 300);
```

```
INSERT INTO PROGRAMATORI
```

```
VALUES (18, null);
```

VARIABLE procent_salariu NUMBER

EXECUTE :procent_salariu :=date_programator('Constantin');

PRINT procent_salariu

```
Error starting at line 798 in command:
EXECUTE :procent_salariu :=date_programator('Constantin')
Error report:
ORA-20003: Programatorului cu numele Constantin nu i s-a atribuit o specializare!
ORA-06512: at "PAULA.DATE_PROGRAMATOR", line 39
ORA-06512: at line 1

PROCENT_SALARIU
-----
```

- *Explicații cod:*

Pt. a putea folosi **mai multe blocuri EXCEPTION** am folosit mai multe **subblocuri** în corpul funcției. Erorile le-am abordat pe rând așa cum sunt descrise mai sus.

Cursorul c este **parametrizat** și reține toate id -urile proiectelor la care lucrează programatorul dat ca parametru. Pt fiecare Id_proiect salvat în cursor selectez într-un **record** (de tip **rowtype**) informațiile despre acel proiect apoi afișez și calculez procentul salariului.

9. Un subprogram stocat de tip procedură care utilizează 5 tabele. Excepții.

Am creat o procedură stocată care mărește cu 10 % salariul agenților call center apelați de cele mai multe ori de persoanele care au medici de familie într-un spital dat ca parametru.

EXCEPȚII pentru:

- Nu există spitalul dat ca parametru.
- Salariul nou depășește constrângerea de salariu maxim.

Codul pt procedura de mai sus:

```
ALTER TABLE Angajati
```

```
ADD CONSTRAINT Ang_sal_max
```

```
CHECK (salariu < 10000);
```

```
CREATE OR REPLACE PROCEDURE mareste_salariu(ume_spital IN Spitale.Nume%TYPE, nr_apeluri  
OUT NUMBER)
```

```
IS
```

```
id_sp Spitale.Id_spital%TYPE;
```

```
max_apeluri NUMBER := 0;
```

```
v_nr NUMBER := 0;
```

```
v_id Angajati.Id_angajat%TYPE;
```

```
v_id_max Angajati.Id_angajat%TYPE;
```

```
v_sal Angajati.Salariu%TYPE;
```

```

CHECK_CONSTRAINT_VIOLATED EXCEPTION;

PRAGMA EXCEPTION_INIT(CHECK_CONSTRAINT_VIOLATED, -2290);


BEGIN

    BEGIN

        SELECT Id_Spital INTO id_sp

        FROM Spitale

        WHERE Nume = nume_spital;

    EXCEPTION

        WHEN NO_DATA_FOUND THEN

            RAISE_APPLICATION_ERROR(-20000, 'Nu exista spital cu numele dat!');

    END;


BEGIN

    FOR i IN (SELECT COUNT(*) nr , Id_angajat id

        FROM Apeleaza

        WHERE Id_persoana in (SELECT Id_persoana

            FROM Persoane P JOIN Medici_familie M ON (P.Id_medic = M.Id_medic)

            JOIN Spitale S ON (M.Id_spital = S.Id_spital)

            WHERE S.Id_spital = id_sp)

        GROUP BY Id_angajat)

    LOOP

        IF (i.nr > max_apeluri) THEN

            max_apeluri := i.nr;

            v_id_max := i.id;

        END IF;

    END LOOP;

```

```

nr_apeluri := max_apeluri;

SELECT salariu INTO v_sal
FROM angajati
WHERE id_angajat = v_id_max;

DBMS_OUTPUT.PUT_LINE('Agentul call center cu id-ul ' || v_id_max || ' a avut cele mai multe
apeluri ' || nr_apeluri || ' si are salariul initial de ' || v_sal);

UPDATE Angajati
SET salariu = salariu * 10/100 + salariu
WHERE id_angajat = v_id_max;

EXCEPTION
WHEN CHECK_CONSTRAINT_VIOLATED THEN
DBMS_OUTPUT.PUT_LINE('Salariul depaseste limita de 10000!');

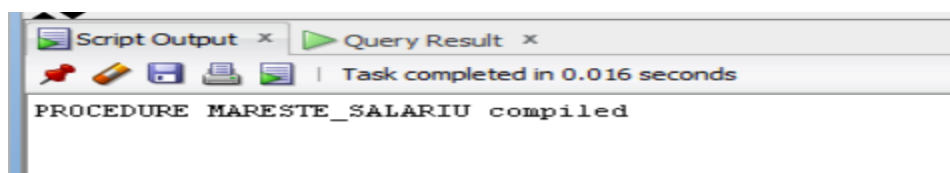
END;

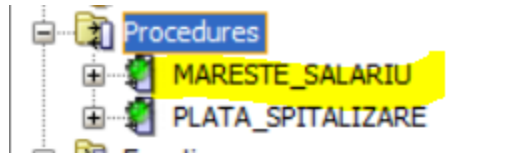
END maresteste_salariu;

/

```

Procedura compilează:

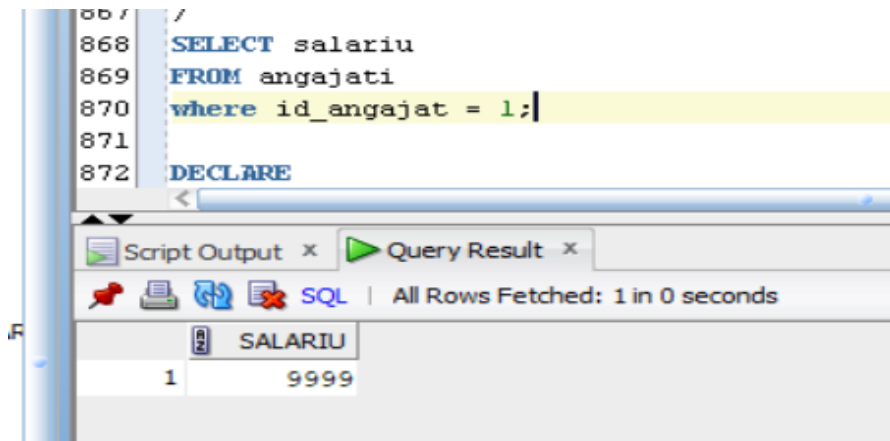




Apelul procedurii:

```
DECLARE
v_nr NUMBER;
BEGIN
mareste_salariu('Regina Maria',v_nr);
DBMS_OUTPUT.PUT_LINE('Numarul maxim de apeluri pt. spitalul introdus este: ' || v_nr);
END;
/
```

```
anonymous block completed
Agentul call center cu id-ul 9 a avut cele mai multe apeluri 3 si are salariul initial de 8857.805
Numarul maxim de apeluri pt. spitalul introdus este: 3
```



- Pt. excepții:

```
DECLARE
v_nr NUMBER;
BEGIN
mareste_salariu('Caius Sparez',v_nr);
DBMS_OUTPUT.PUT_LINE('Numarul maxim de apeluri pt. spitalul introdus este: ' || v_nr);
END;
/
```

```

Error report:
ORA-20000: Nu exista spital cu numele dat!
ORA-06512: at "PAULA.MARESTE_SALARIU", line 20
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.

```

DECLARE

v_nr NUMBER;

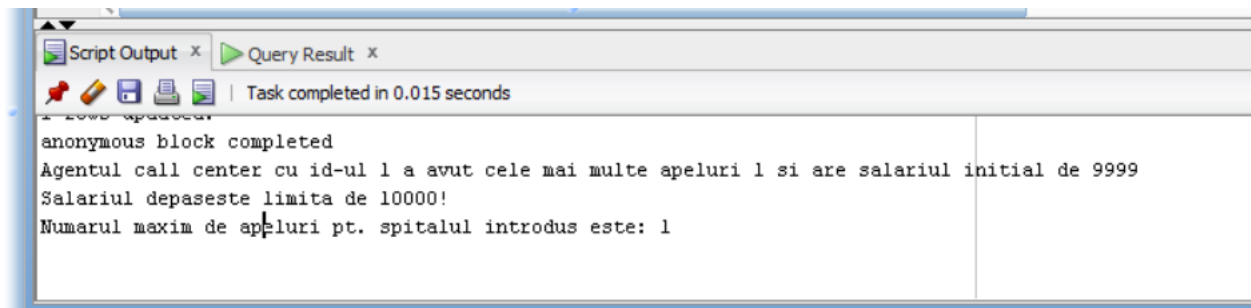
BEGIN

mareste_salariu('Caius Sparchez',v_nr);

DBMS_OUTPUT.PUT_LINE('Numarul maxim de apeluri pt. spitalul introdus este: ' || v_nr);

END;

/



- *Explicații cod:*

Am definit o excepție proprie cu numele CHECK_CONSTRAINT_VIOLATED care capează excepția de violare a constrangerii de salariu mai mic decât 10000 și anume captează eroarea cu numarul -2290. Cu **EXCEPTION_INIT** am asociat numele excepției cu un număr de eroare Oracle. **Pragma** semnifică faptul ca asocierea nume-număr excepție este o directivă pt. compilator.

Cu un ciclu cursor cu subcereri am parcurs fiecare agent call center și numărul de apeluri răspunse persoanelor care au un medic de familie în spitalul dat ca parametru și am calculat maximum de apeleluri salvând în

v_id_max id-ul angajatului care le-a onorat. Apoi îi modific salariul dacă se poate.

10. Un trigger de tip LMD la nivel comandă

Voi defini un declanșator la nivel instrucțiune care pentru insert/delete/update pe tabelul Angajati inserează într-un tabel nou info_angajati data la care s-a făcut modificarea, de către cine și ce se modifică.

Codul pt. cerința de mai sus:

```
CREATE SEQUENCE id_info START WITH 1;
```

```
CREATE TABLE INFO_ANG
```

```
( ID INTEGER NOT NULL PRIMARY KEY,
```

```
  user_name VARCHAR2(50),
```

```
  data_modif date,
```

```
  DESCRIERE VARCHAR2(100)
```

```
);
```

```
CREATE OR REPLACE TRIGGER info_angajati
```

```
AFTER INSERT OR UPDATE OR DELETE ON angajati
```

```
BEGIN
```

```
IF DELETING THEN
```

```
  INSERT INTO info_ang (user_name, data_modif, descriere)
```

```
  VALUES(SYS.LOGIN_USER, SYSDATE, ' s-a sters un anagajat');
```

```
ELSIF UPDATING THEN
```

```
    INSERT INTO info_ang (user_name, data_modif, descriere)
```

```
    VALUES(SYS.LOGIN_USER, SYSDATE, ' s-a modificat un anagajat');
```

```
ELSE
```

```
    INSERT INTO info_ang (user_name, data_modif, descriere)
```

```
    VALUES(SYS.LOGIN_USER, SYSDATE, ' s-a inserat un anagajat');
```

```
END IF;
```

```
END info_angajati;
```

```
/
```

```
CREATE OR REPLACE TRIGGER pune_id
```

```
BEFORE INSERT ON info_ang
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    SELECT id_info.NEXTVAL
```

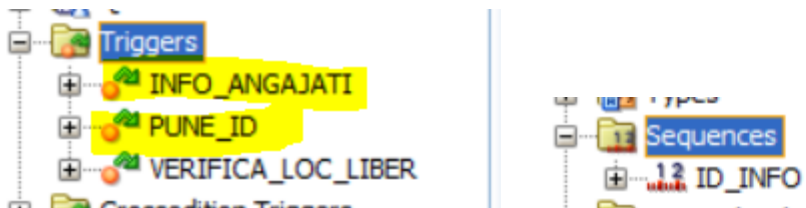
```
    INTO :new.ID
```

```
    FROM dual;
```

```
END;
```

```
/
```

Cei 2 triggeri și secvența compilează:

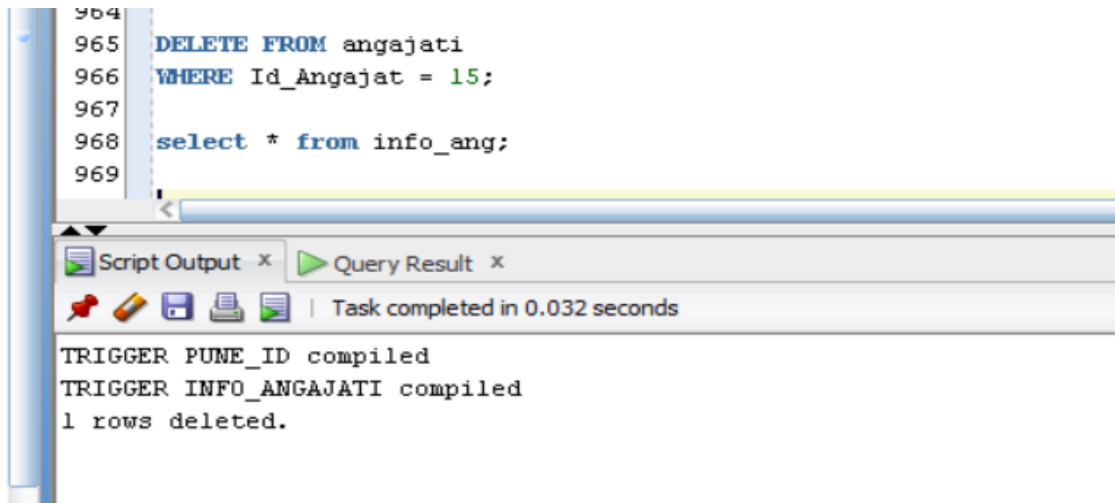


Declanșare trigger:

DELETE FROM angajati

WHERE Id_Angajat = 15;

select * from info_ang;



Script Output x Query Result x
SQL | All Rows Fetched: 1 in 0.015 seconds

ID	USER_NAME	DATA_MODIF	DESCRIERE
1	PAULA	30-DEC-20	s-a sters un anagajat

UPDATE Angajati

SET salariu = 4000

WHERE Id_angajat = 1;

SELECT * FROM info_ang;

```

969
970 UPDATE Angajati
971 SET salariu = 4000
972 WHERE Id_angajat = 1;
973
974 SELECT * FROM info_ang;
975
976
977 --11

```


Script Output x Query Result x

Task completed in 0 seconds

TRIGGER PUNE_ID compiled
 TRIGGER INFO_ANGAJATI compiled
 1 rows deleted.
 1 rows updated.

```
974 SELECT * FROM info_ang;
975
976
977 --11
```

Script Output x Query... x

 All Rows Fetched: 2 in 0 seconds

	ID	USER_NAME	DATA_MODIF	DESCRIERE
1	1 PAULA	30-DEC-20	s-a sters un anagajat	
2	2 PAULA	30-DEC-20	s-a modificat un anagajat	

- *Explicații cod:*

Am creat tabelul info_ang în care **triggerul** inserează informații atunci când se modifică tabelul Angajati. Pt. a incrementa automat id-ul informațiilor inserate am folosit o **secvența id_info** care începe de la 1 și un **trigger la nivel de linie** care updateaza id-ul pt. noua înregistrare inserată.

11. Un trigger de tip LMD la nivel linie

Voi defini un declanșator LMD la nivel linie pentru Insert pe tabelul Spitale_suport, care, verifică că nu pot fi mai mult de 2 spitale suport într-o locație.

Codul pt cerința de mai sus:

```
CREATE OR REPLACE TRIGGER verifica_loc_liber
FOR INSERT ON spitale_suport
compound TRIGGER

TYPE tablou_indexat IS TABLE OF Spitale_suport.Capacitate_paturi%TYPE
INDEX BY BINARY_INTEGER;

t tablou_indexat;
v_ok NUMBER;

BEFORE statement IS
BEGIN
SELECT count(*)
BULK COLLECT INTO t
FROM Spitale_Suport SS JOIN Spitale S ON(SS.Id_spital = S.Id_spital)
GROUP BY S.Id_locatie;

END BEFORE statement;

AFTER EACH ROW IS
BEGIN
v_ok := 1;
FOR i IN 1..t.LAST
LOOP
```

```

IF t(i) + 1 > 2 THEN

    v_ok := 0;

    RAISE_APPLICATION_ERROR(-20005, 'Nu pot fi 2 spitale suport in aceeasi locatie!');

end if;

END LOOP;


IF v_ok <> 0 THEN

DBMS_OUTPUT.PUT_LINE('Inserare corecta');

END IF;

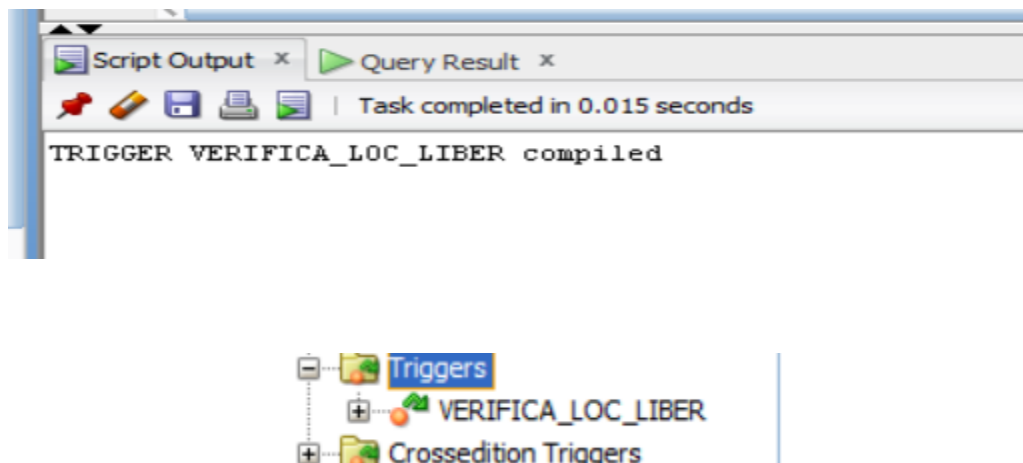
END AFTER EACH ROW;


END verifica_loc_liber;

/

```

Trigger ul compilează:



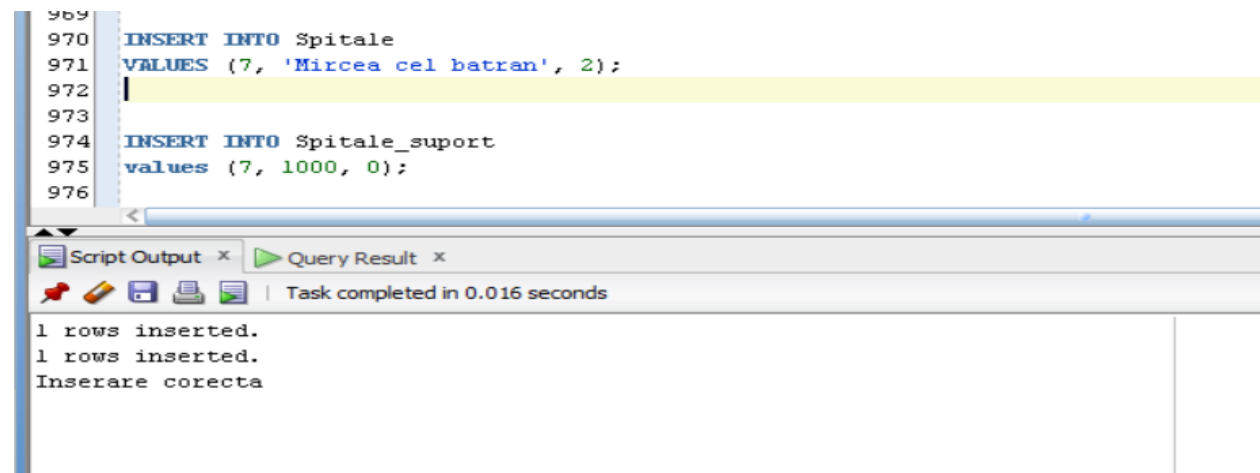
Declanșarea triggerului:

```
INSERT INTO Spitale
```

```
VALUES (7, 'Mircea cel batran', 2);
```

```
INSERT INTO Spitale_suport
```

```
values (7, 1000, 0);
```



The screenshot shows a SQL script editor with the following code:

```
969  
970 INSERT INTO Spitale  
971 VALUES (7, 'Mircea cel batran', 2);  
972  
973  
974 INSERT INTO Spitale_suport  
975 values (7, 1000, 0);  
976
```

Below the script editor is a window titled "Script Output" and "Query Result". It displays the following output:

```
1 rows inserted.  
1 rows inserted.  
Inserare corecta
```

The window also shows a status bar indicating "Task completed in 0.016 seconds".

```
INSERT INTO Spitale
```

```
VALUES (8, 'Sf. Stefan ', 2);
```

```
INSERT INTO Spitale_suport
```

```
values (8, 1000, 0);
```

```
979
980 INSERT INTO Spitale_suport
981 values (8, 1000, 0);
982
983 SELECT * FROM spitale;
984 SELECT * FROM spitale_suport; --4 2 5
```

Script Output x Query Result x

Task completed in 0.016 seconds

Error starting at line 980 in command:
INSERT INTO Spitale_suport
values (8, 1000, 0)
Error report:
SQL Error: ORA-20005: Nu pot fi 2 spitale suport in aceeaasi locatie!
ORA-06512: at "PAULA.VERIFICA_LOC_LIBER", line 25
ORA-04088: error during execution of trigger 'PAULA.VERIFICA_LOC_LIBER'

- *Explicații cod:*

Pt. a nu primi eroarea *Mutating table*, atunci cand interoghez tabelul care se actualizează am folosit un **compound trigger** în loc de 2 trigeri obișnuiți.

12. Un trigger de tip LDD

Un trigger LDD care apelează o porcedură și inserează într-un tabel info ce user a făcut modificarea, ce tip de modificare a fost făcută, când și pe ce obiect din schemă.

Codul pt. sarcina descrisă mai sus:

```
CREATE TABLE info
(
  user_name VARCHAR2(30),
  eveniment VARCHAR2(20),
  DATA DATE,
  nume_obiect VARCHAR2(30)
```



```
);
```

```
CREATE OR REPLACE PROCEDURE insereaza(ev IN info.eveniment%TYPE, ob IN  
info.num_e_object%TYPE)
```

```
IS
```

```
BEGIN
```

```
IF ob = 'INFO' THEN
```

```
    RAISE_APPLICATION_ERROR(-20005, 'Nu se poate modifica acest tabel');
```

```
END IF;
```

```
INSERT INTO info
```

```
VALUES(SYS.LOGIN_USER, ev, SYSDATE, ob);
```

```
END insereaza;
```

```
/
```

```
CREATE OR REPLACE TRIGGER ldd_trigger
```

```
BEFORE CREATE OR DROP OR ALTER ON SCHEMA
```

```
BEGIN
```

```
    insereaza(SYS.SYSEVENT,SYS.DICTIONARY_OBJ_NAME);
```

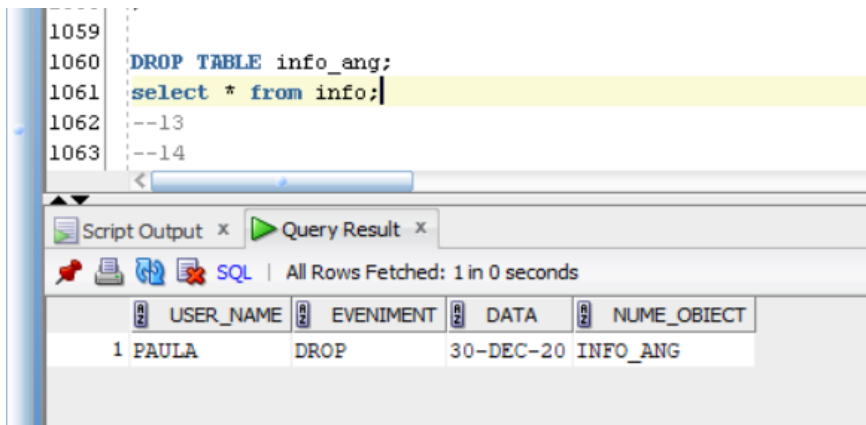
```
END;
```

```
/
```

Apelul triggerului:

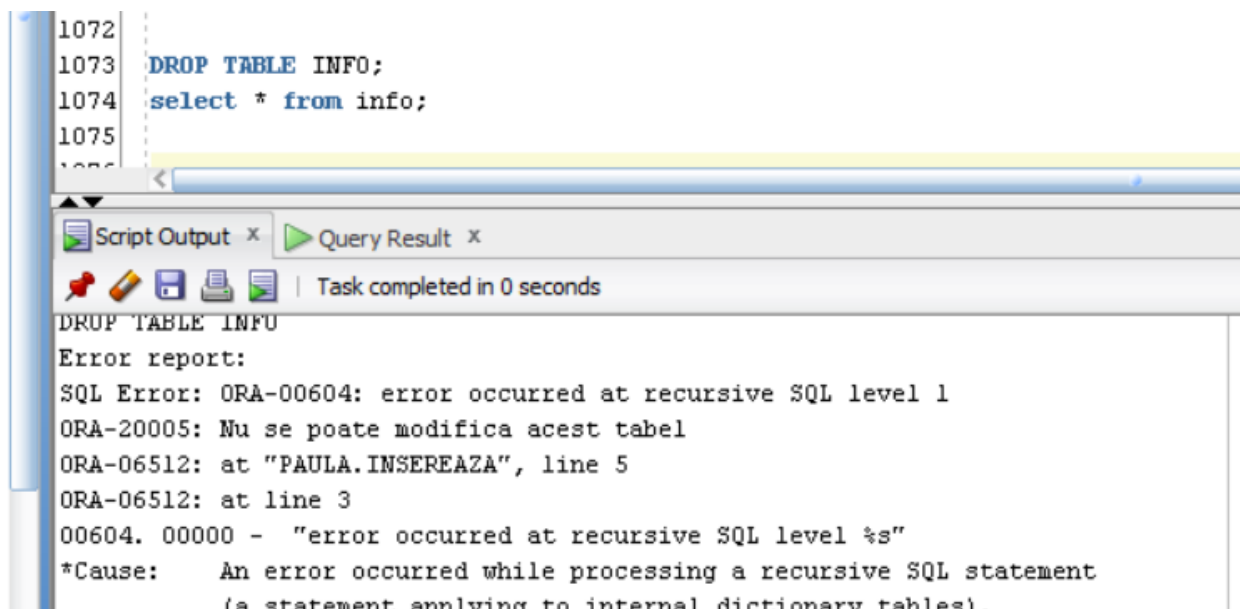
```
DROP TABLE info_ang;
```

```
SELECT * FROM info;
```



DROP TABLE INFO;

select * from info;



- *Explicații cod:*

Am creat o **procedură** care primește ca **parametrii de intrare** (de tip IN) evenimentul și numele obiectului și inserează în tabelul info. Triggerul se declanșează la **orice comandă alter/create/delete** și apelează procedura. Dacă

numele obiectului pe care s-a facut alter/create/delete este info, atunci se **aruncă o eroare** întrucât acesta nu poate fi modificat.

13. Un pachet care să conțină toate obiectele definite la cerințele anterioare

Am creat un pachet care conține obiectele, procedurile și funcțiile definite la subpunctele anterioare.

Triggerii nu se pot include într-un pachet, în schimb i-am rescris astfel încât să folosească funcții din pachet.

Codul pachetului este:

```
CREATE OR REPLACE PACKAGE pachet_complet AS
```

```
PROCEDURE insereaza(ev IN info.eveniment%TYPE, ob IN info.ume_obiect%TYPE);
```

```
PROCEDURE mareste_salariu(ume_spital IN Spitale.Nume%TYPE, nr_apeluri OUT NUMBER);
```

```
FUNCTION date_programator(param_ume Angajati.Nume%TYPE) RETURN FLOAT;
```

```
FUNCTION numar_pachete_internare RETURN NUMBER;
```

```
PROCEDURE plata_spitalizare(nr_pers_pachet OUT NUMBER, nr_pers_fara_pachet OUT NUMBER);
```

```
END pachet_complet;
```

```
/
```

```
CREATE OR REPLACE PACKAGE BODY pachet_complet AS
```

```
PROCEDURE insereaza(ev IN info.eveniment%TYPE, ob IN info.nume_obiect%TYPE)
```

```
IS
```

```
BEGIN
```

```
IF ob = 'INFO' THEN
```

```
    RAISE_APPLICATION_ERROR(-20005, 'Nu se poate modifica acest tabel');
```

```
END IF;
```

```
INSERT INTO info
```

```
VALUES(SYS.LOGIN_USER, ev, SYSDATE, ob);
```

```
END insereaza;
```

```
PROCEDURE mareste_salariu(nume_spital IN Spitale.Nume%TYPE, nr_apeluri OUT NUMBER)
```

```
IS
```

```
id_sp Spitale.Id_spital%TYPE;
```

```
max_apeluri NUMBER := 0;
```

```
v_nr NUMBER := 0;
```

```
v_id Angajati.Id_angajat%TYPE;
```

```
v_id_max Angajati.Id_angajat%TYPE;
```

```
v_sal Angajati.Salariu%TYPE;
```

```
CHECK_CONSTRAINT_VIOLATED EXCEPTION;
```

```
PRAGMA EXCEPTION_INIT(CHECK_CONSTRAINT_VIOLATED, -2290);
```

```
BEGIN
```

```
    BEGIN
```

```
        SELECT Id_Spital INTO id_sp
```

```
        FROM Spitale
```

```
        WHERE Nume = nume_spital;
```

```
    EXCEPTION
```

```
        WHEN NO_DATA_FOUND THEN
```

```
            RAISE_APPLICATION_ERROR(-20000, 'Nu exista spital cu numele dat!');
```

```
    END;
```

```
BEGIN
```

```
FOR i IN (SELECT COUNT(*) nr , Id_angajat id
```

```
    FROM Apeleaza
```

```
    WHERE Id_persoana in (SELECT Id_persoana
```

```
        FROM Persoane P JOIN Medici_familie M ON (P.Id_medic = M.Id_medic)
```

```
        JOIN Spitale S ON (M.Id_spital = S.Id_spital)
```

```
        WHERE S.Id_spital = id_sp)
```

```
    GROUP BY Id_angajat)
```

```
LOOP
```

```
    IF (i.nr > max_apeluri) THEN
```

```
        max_apeluri := i.nr;
```

```
        v_id_max := i.id;
```

```
    END IF;
```

```
END LOOP;
```

```
nr_apeluri := max_apeluri;
```

```
SELECT salariu INTO v_sal
```

```
FROM angajati
```

```
WHERE id_angajat = v_id_max;
```

```
DBMS_OUTPUT.PUT_LINE('Agentul call center cu id-ul ' || v_id_max || ' a avut cele mai multe apeluri ' || nr_apeluri || ' si are salariul initial de ' || v_sal);
```

```
UPDATE Angajati
```

```
SET salariu = salariu * 10/100 + salariu
```

```
WHERE id_angajat = v_id_max;
```

```
EXCEPTION
```

```
WHEN CHECK_CONSTRAINT_VIOLATED THEN
```

```
DBMS_OUTPUT.PUT_LINE('Salariul depaseste limita de 10000!');
```

```
END;
```

```
END mareste_salariu;
```

```
FUNCTION date_programator(param_nume Angajati.Nume%TYPE) RETURN FLOAT
```

```
IS
```

```
v_id Angajati.Id_angajat%TYPE;
```

```
date_proiect Proiect%ROWTYPE;
```

```
procent_salariu FLOAT := 0.00;
```

```
specialitate Programatori.Specializare_limbaj%TYPE;
```

```

nr_proiecte NUMBER;

contor NUMBER := 0;

sal_total Angajati.Salariu%TYPE;

sal Angajati.Salariu%TYPE;


CURSOR c(param Angajati.Id_angajat%TYPE) IS

    SELECT Id_Proiect
    FROM Lucreaza_la
    where Id_angajat = param;


BEGIN

BEGIN

SELECT Id_angajat INTO v_id
FROM Programatori JOIN Angajati USING (Id_Angajat)
WHERE Nume = param_nume;

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        RAISE_APPLICATION_ERROR(-20000, 'Nu exista programator cu numele dat!');

        RETURN -1.00;

    WHEN TOO_MANY_ROWS THEN

        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi programatori cu numele dat!');

        RETURN -2.00;

    WHEN OTHERS THEN

        RAISE_APPLICATION_ERROR(-20002, 'S-a generat alta eroare!');

        RETURN -3.00;

END;


BEGIN

    SELECT Specializare_limbaj INTO specialitate

```

```

FROM Programatori
WHERE Id_Angajat = v_id;

IF specialitate is null THEN

    RAISE_APPLICATION_ERROR(-20003, 'Progamatorului cu numele ' || param_nume || ' nu i s-a
atribuit o specializare!');

    RETURN -4.00;

END IF;

```

```

END;

```

```

BEGIN

SELECT COUNT(*) INTO nr_proiecte
FROM Lucreaza_la
WHERE Id_angajat = v_id
GROUP BY Id_angajat;

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        RAISE_APPLICATION_ERROR(-20005, 'Progamatorului cu numele ' || param_nume || ' nu are
proiecte!');

        RETURN -5.00;

END;

```

```

DBMS_OUTPUT.PUT_LINE('Progamatorul cu numele ' || param_nume || ' lucreaza in prezent pe ' ||
nr_proiecte || ' proiecte si are specializarea ' || specialitate);

```

```

IF nr_proiecte <> 0 THEN --daca lcucreaza pe mai mult de 0 proiecte vedem care sunt
FOR i IN c(v_id) LOOP

    SELECT * INTO date_proiect

    FROM PROIECT

```



```

WHERE Id_proiect = i.Id_proiect;

DBMS_OUTPUT.PUT_LINE('Proiectul cu nr de ordine ' || contor || ' are startdate ' ||
date_proiect.Start_date || ' deadline ' || date_proiect.Deadline || ' si este lucrat in limbajul ' ||
date_proiect.Limbaj_programare);

    contor := contor + 1;

END LOOP;

END IF;


SELECT SUM(NVL(Salariu,0)) INTO sal_total
FROM ANGAJATI;


BEGIN

SELECT Salariu INTO sal
FROM ANGAJATI
WHERE Id_angajat = v_id;

IF sal is null THEN

    RAISE_APPLICATION_ERROR(-20004, 'Programatorului ' || param_nume || ' nu i s-a alocat salariu!');

    RETURN -4.00;

END IF;


END;


procent_salariu := sal*100/sal_total;

RETURN procent_salariu;


END date_programator;


FUNCTION numar_pachete_internare RETURN NUMBER

```

```

IS
TYPE refcursor IS REF CURSOR;

CURSOR c1 IS
    SELECT Id_pachet, Titlu,
           CURSOR(SELECT S.*
                   FROM pachet_servicii P2, TABLE (P2.servicii) S
                   WHERE P2.Id_Pachet = P.Id_Pachet)
    FROM Pachet_servicii P;

cursor_aux refcursor;
v_id Pachet_servicii.Id_pachet%TYPE;
v_titlu Pachet_servicii.Titlu%TYPE;
v_nr NUMBER;
serviciu varchar(20);

BEGIN
    v_nr := 0;
    OPEN c1;
    LOOP
        FETCH c1 INTO v_id, v_titlu, cursor_aux;
        EXIT WHEN c1%NOTFOUND;
        LOOP
            FETCH cursor_aux INTO serviciu;
            EXIT WHEN cursor_aux%NOTFOUND;
            IF 'Internari' = Initcap(serviciu) THEN
                v_nr := v_nr + 1;
                DBMS_OUTPUT.PUT_LINE('PACHETUL ' || v_id || ' CU NUMELE ' || v_titlu || ' contine internari');
            END IF;
        END LOOP;
    END LOOP;

```

```

END LOOP;

CLOSE c1;


RETURN v_nr;

IF v_nr = 0 THEN

RAISE_APPLICATION_ERROR(-20001,'Nu exista pachet cu internare!');

RETURN -1;

END IF;

END numar_pachete_internare;


PROCEDURE plata_spitalizare(nr_pers_pachet OUT NUMBER, nr_pers_fara_pachet OUT NUMBER)
IS
TYPE tablou_imbricat IS TABLE OF pers_pachet;
t tablou_imbricat := tablou_imbricat();
v_id_pers persoane.Id_persoana%TYPE;
v_id_pachet persoane.Id_pachet%TYPE;
v_data persoane.Data_testare%TYPE;
v_pachet tip_servicii;
contor NUMBER;
contor2 NUMBER;
v_ok NUMBER;
v_ok2 NUMBER;
v_nr NUMBER;

BEGIN

nr_pers_pachet := 0;

nr_pers_fara_pachet := 0;

contor := 1;

```

```

FOR i in (SELECT Id_persoana, Id_pachet, Data_testare
          FROM Persoane
          WHERE test_covid = 1
          ORDER BY Id_persoana)

LOOP

SELECT servicii INTO v_pachet
FROM pachet_servicii
WHERE Id_pachet = i.Id_pachet;

t.EXTEND;
t(contor) := pers_pachet(i.Id_persoana, i.Data_testare, v_pachet);
contor := contor + 1;
END LOOP;

FOR j IN t.FIRST..t.LAST
LOOP
DBMS_OUTPUT.PUT('Persoana cu id-ul ' || t(j).cod_persoana || ' are test pozitiv si pachetul << ');

v_ok := 0;
v_nr := 0;
FOR k IN t(j).pachet_servicii.FIRST..t(j).pachet_servicii.LAST
LOOP
DBMS_OUTPUT.PUT(t(j).pachet_servicii(k) || ' ');
IF 'Internari' = t(j).pachet_servicii(k) AND v_ok = 0 THEN

v_ok := 1;

```

```

nr_pers_pachet := nr_pers_pachet + 1;

END IF;

END LOOP;

DBMS_OUTPUT.PUT('>> ');

IF v_ok = 0 THEN --nu are pachet care sa contina internari deci trebuie sa le plateasca

    DBMS_OUTPUT.PUT(' trebuie sa plateasca internarile ');

    v_ok2 := 0;

    SELECT count(*) INTO v_nr
    FROM spitale S JOIN internari i ON (S.id_spital = i.id_spital)
        WHERE i.id_persoana = t(j).cod_pesoana;

    IF v_nr = 0 THEN

        DBMS_OUTPUT.PUT(' viitoare si nu s-a internat pana acum');

    END IF;

    FOR I IN (SELECT nume
        FROM spitale S JOIN internari i ON (S.id_spital = i.id_spital)
        WHERE i.id_persoana = t(j).cod_pesoana)
    LOOP
        IF v_ok2 = 0 THEN

            DBMS_OUTPUT.PUT('la ');

            DBMS_OUTPUT.PUT('*' || I.nume || '* ');

            v_ok2 := 1;

        ELSE

            DBMS_OUTPUT.PUT('*' || I.nume || '* ');

        END IF;

    END LOOP;

    ELSE

```

```
DBMS_OUTPUT.PUT(' nu trebuie sa plateasca internarile');  
END IF;  
DBMS_OUTPUT.NEW_LINE();  
END LOOP;
```

```
SELECT count(*) INTO nr_pers_fara_pachet  
FROM Persoane  
where Test_covid = 1;
```

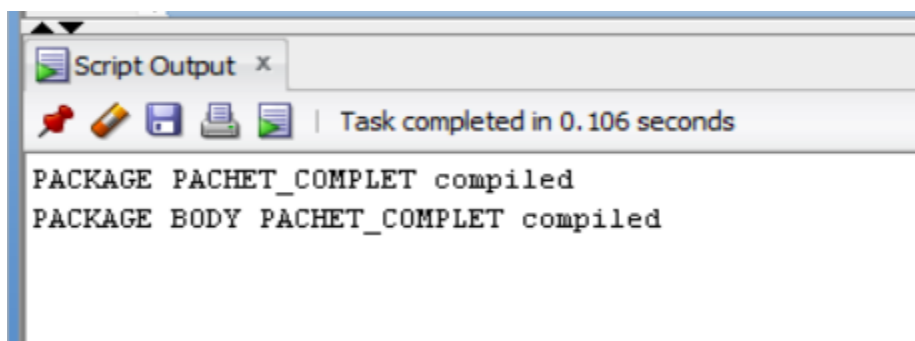
```
nr_pers_fara_pachet := nr_pers_fara_pachet - nr_pers_pachet;
```

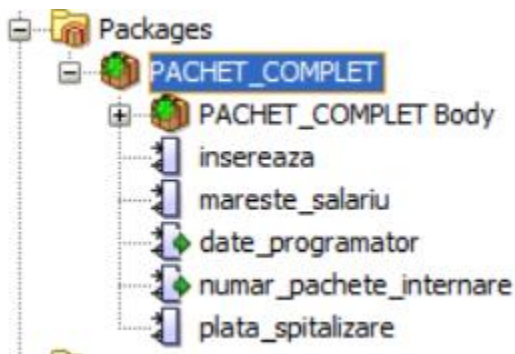
```
END plata_spitalizare;
```

```
END pachet_complet;
```

```
/
```

Pachetul compilează:





Apelarea unei funcții și a unei proceduri din pachet:

DECLARE

nr_pachete NUMBER;

nr_pers_pachet NUMBER;

nr_pers_fara_pachet NUMBER;

BEGIN

nr_pachete := pachet_complet.numar_pachete_internare;

DBMS_OUTPUT.PUT_LINE('Nr de pachete care contin serviciul internare este: ' || nr_pachete);

DBMS_OUTPUT.PUT_LINE('-----');

plata_spitalizare(nr_pers_pachet, nr_pers_fara_pachet);

DBMS_OUTPUT.PUT_LINE('Nr de persoane care au pachete cu internare este: ' || nr_pers_pachet ||
' iar nr de persoane fara pachete este: ' || nr_pers_fara_pachet);

END;

/

```

anonymous block completed
PACHETUL 1 CU NUMELE Pachet_Golden contine internari
Nr de pachete care contin serviciul internare este: 1
-----
Persoana cu id-ul 1 are test pozitiv si pachetul << Analize sange Ecografii Internari Tratament compensat >> nu trebuie sa plateasca internarile
Persoana cu id-ul 4 are test pozitiv si pachetul << Analize sange Ecografii >> trebuie sa plateasca internarile la *Regina Maria* *Caius Sparchez*
Persoana cu id-ul 5 are test pozitiv si pachetul << Pastile compensate Analize sange Tratamente fizio >> trebuie sa plateasca internarile viitoare si nu s-a internat pana acum
Persoana cu id-ul 6 are test pozitiv si pachetul << Pastile compensate Analize sange Tratamente fizio >> trebuie sa plateasca internarile la *Regina Maria* *Regina Maria*
Persoana cu id-ul 7 are test pozitiv si pachetul << Analize sange Ecografii Internari Tratament compensat >> nu trebuie sa plateasca internarile
Persoana cu id-ul 9 are test pozitiv si pachetul << Pastile compensate Analize sange Tratamente fizio >> trebuie sa plateasca internarile viitoare si nu s-a internat pana acum
Nr de persoane care au pachete cu internare este: 2 iar nr de persoane fara pachete este: 4
  
```

Triggerul LDD rescris, astfel încât să folosească procedura insereaza din pachet:

```
CREATE OR REPLACE
```

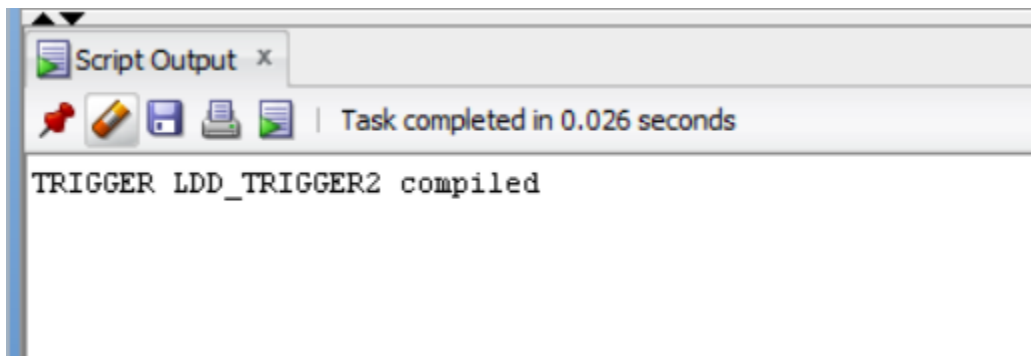
```
TRIGGER ldd_trigger2
```

```
BEFORE CREATE OR DROP OR ALTER ON SCHEMA
```

```
BEGIN
```

```
    pachet_complet.insereaza(SYS.SYSEVENT,SYS.DICTIONARY_OBJ_NAME);
```

```
END;
```



14. Un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate

Voi crea un pachet numit management_persoane care se ocupă de gestionarea persoanelor:

1. Persoanele pozitive pot alege daca se intrenează sau nu.
2. Internarea se face prin interserarea unei linii în tabelul Internari, Id_spitalului trb sa fie un Id al unui spital suport.
3. Se va alege spitalul suport cu cele mai multe locuri libere.
4. Persoanele care refuză internarea vor fi inserate într-un colecție tablou indexat de tip record (nume_persoana, data_test, data_iesire_din_internare).
5. Persoanelor la care au trecut 14 zile de la carantinare se scot din tabel.

Codul pt pachetul descris mai sus:

```
CREATE OR REPLACE PACKAGE management_persoane AS
  CURSOR c1 RETURN Persoane%ROWTYPE; -- pt persoanele pozitive
  TYPE sp_tip IS REF CURSOR RETURN Spitale_Suport%ROWTYPE; -- tip pt cusor dinamic pt spitalele suport care mai au locuri libere
  TYPE pers_carantinate IS RECORD (cod_pesoana INTEGER, data_test DATE, iesire_izolare DATE);
  TYPE tabel_carantinate IS TABLE OF pers_carantinate
    INDEX BY BINARY_INTEGER;
  FUNCTION creeaza_tabel RETURN NUMBER; -- RETURNEAZA 1 DACA EXISTA tabel CU
  PERSOANELE CARANTINATE SI 0 IN CAZ CONTRAR
  FUNCTION alege_spital RETURN INTEGER;
  FUNCTION verifica_daca_e_internata(param PERSOANE.Id_persoana%TYPE) RETURN NUMBER;
  --daca persoana cu id ul dat e internata returneaza 1 altfel 0
  PROCEDURE interneaza_persoana; -- daca se citeste de la tastatura pt fiecare persoana din
  cursorul c1 se interneaza, daca nu se insereaza in t1
  PROCEDURE scoate_din_carantina;
  t1 tabel_carantinate; --colectie tip tablou indexat pt. persoanele carantinate
END management_persoane;
/
```

```
CREATE OR REPLACE PACKAGE BODY management_persoane AS
```

```
  CURSOR c1 RETURN Persoane%ROWTYPE IS
  SELECT * FROM PERSOANE
  WHERE Test_covid = 1;

  FUNCTION creeaza_tabel RETURN NUMBER IS
  v_nr NUMBER;
  BEGIN
  SELECT COUNT(*) into v_nr
```

```

FROM USER_TABLES
WHERE TABLE_NAME = 'PERSOANE_CARANTINATE';
IF v_nr <> 0 THEN
DBMS_OUTPUT.PUT_LINE('TABELUL PERSOANE_CARANTINATE EXISTA DEJA');
RETURN 1;
ELSE
DBMS_OUTPUT.PUT_LINE('TABELUL PERSOANE_CARANTINATE NU EXISTA');
RETURN 0;
END IF;
END creeza_tabel;

```

```

FUNCTION alege_spital RETURN INTEGER IS
c2 sp_tip;
v_id INTEGER;
v_nr_max NUMBER;
v_spital Spitale_suport%ROWTYPE;
BEGIN
OPEN c2 FOR
    SELECT * FROM spitale_suport
    WHERE capacitate_paturi - paturi_ocupate >= 1;

v_nr_max := 0;
LOOP
    FETCH c2 INTO v_spital;
    EXIT WHEN c2%NOTFOUND;
    IF (v_spital.capacitate_paturi - v_spital.paturi_ocupate) > v_nr_max THEN
        v_nr_max := v_spital.capacitate_paturi - v_spital.paturi_ocupate;
        v_id := v_spital.Id_spital;
    END IF;
END LOOP;

RETURN v_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NU MAI SUNT SPITALE SUPORT CU LOCURI LIBERE!');

END alege_spital;

```

```

FUNCTION verifica_daca_e_internata(param PERSOANE.Id_persoana%TYPE) RETURN NUMBER
IS
v_nr NUMBER;

```

```

BEGIN
SELECT COUNT(*) INTO v_nr
FROM Internari
WHERE id_persoana = param;

IF v_nr <> 0 THEN --e internata
    RETURN 1;
ELSE
    RETURN 0;
END IF;
END verifica_daca_e_internata;

PROCEDURE interneaza_persoana IS
v_id_spital INTEGER;
v_pers Persoane%ROWTYPE;
raspuns VARCHAR(20);
ran FLOAT;
p pers_carantinate;
contor NUMBER;
verifica NUMBER;

BEGIN
v_id_spital := alege_spital;
contor := 1;

OPEN c1;
LOOP
    FETCH c1 INTO v_pers;
    EXIT WHEN c1%NOTFOUND;

    SELECT dbms_random.VALUE(1,20) INTO ran FROM dual;

    verifica := management_persoane.verifica_daca_e_internata(v_pers.Id_persoana);

    IF ran <= 10 AND verifica = 0 THEN
        raspuns := 'DA';
    ELSE
        raspuns := 'NU';
    END IF;

    IF upper(raspuns) = 'DA' THEN
        INSERT INTO INTERNARI
        VALUES(v_pers.Id_persoana, v_id_spital,SYSDATE,null);
        UPDATE Spitale_suport

```

```

        SET paturi_ocupate = paturi_ocupate + 1
        where Id_spital = v_id_spital;
        v_id_spital := alege_spital;

    ELSE
        p.cod_pesoana := v_pers.Id_persoana;
        p.data_test := SYSDATE;
        p.iesire_izolare := SYSDATE+14;
        t1(contor) := p;
        contor := contor + 1;
        --t1(t1.LAST+1) := p;

    END IF;

END LOOP;

CLOSE c1;

DBMS_OUTPUT.PUT_LINE('PERSOANELE CARANTINATE SUNT:');
FOR i IN 1..t1.LAST
LOOP
    DBMS_OUTPUT.PUT_LINE(t1(i).cod_pesoana || ' data intrare in carantina ' || t1(i).data_test ||
    ' data iesire din carantina ' || t1(i).iesire_izolare);
END LOOP;
END interneaza_persoana;

PROCEDURE scoate_din_carantina IS
TYPE vector IS VARRAY(20) OF NUMBER;
t vector:= vector();
contor NUMBER;
BEGIN
    contor := 1;
    FOR i IN 1..t1.LAST
    LOOP
        IF t1(i).iesire_izolare >=SYSDATE THEN
            t.EXTEND;
            t(contor) := i;
            --t1.delete(i);
        END IF;
    END LOOP;

```

```

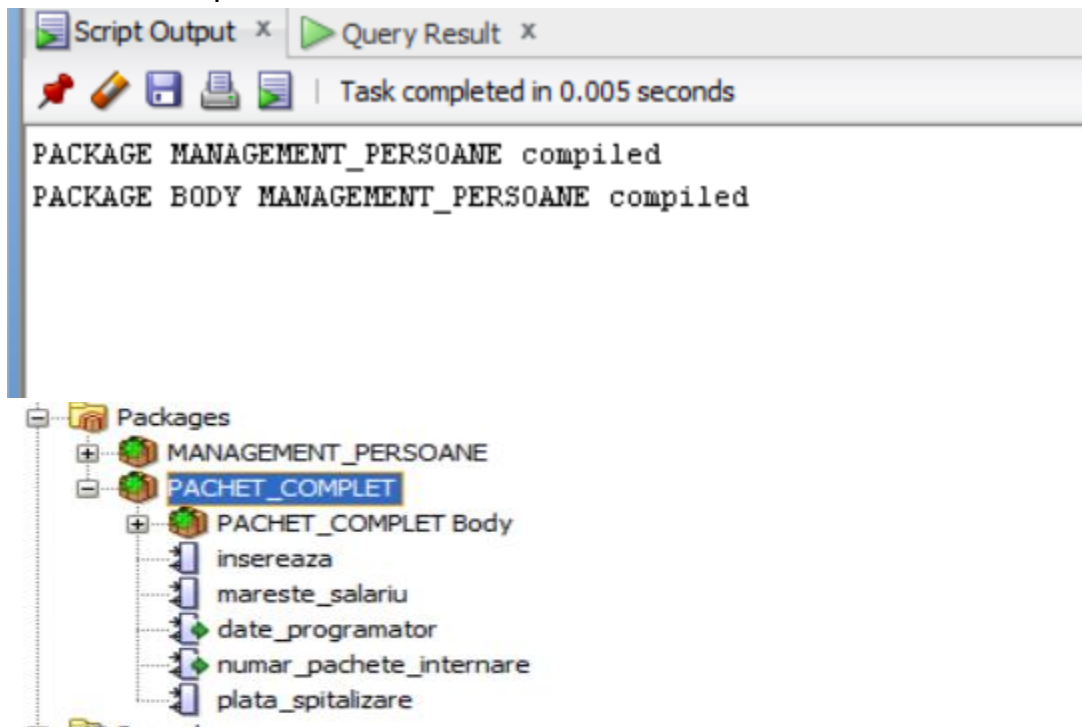
FOR i IN 1..t.LAST
LOOP
t1.DELETE(t(i));
END LOOP;

IF contor -1 <> 0 THEN
DBMS_OUTPUT.PUT_LINE('Au iesit ' || contor -1 || ' persoane din carantina ');
ELSE
DBMS_OUTPUT.PUT_LINE('NU au iesit persoane din carantina ');
END IF;
END scoate_din_carantina;

END management_persoane;
/

```

Pachetul compilează:



Apelul funcțiilor/procedurilor din pachet:

```

DECLARE
exista_tabel NUMBER;
id_spital INTEGER;
verifica_inter INTEGER;

```

```

BEGIN

exista_tabel := management_persoane.creeaza_tabel;
DBMS_OUTPUT.PUT_LINE(exista_tabel);
id_spital := management_persoane.alege_spital;
DBMS_OUTPUT.PUT_LINE('SPITALUL SUPT cu cele mai multe locuri libere are id-ul : ' ||
id_spital);
verifica_inter := management_persoane.verifica_daca_e_internata(10);
IF verifica_inter = 0 THEN
DBMS_OUTPUT.PUT_LINE('Persoana nu e internata!');
ELSE
DBMS_OUTPUT.PUT_LINE('Persoana e internata!');
END IF;

management_persoane.interneaza_persoana;
management_persoane.scoate_din_carantina;
END;
/

```

```

anonymous block completed
TABELUL PERSOANE_CARANTINATE NU EXISTA
0
SPITALUL SUPT cu cele mai multe locuri libere are id-ul : 5
Persoana nu e internata!
PERSOANELE CARANTINATE SUNT:
9 data intrare in carantina 31-DEC-20 data iesire din carantina 14-JAN-21
1 data intrare in carantina 31-DEC-20 data iesire din carantina 14-JAN-21
4 data intrare in carantina 31-DEC-20 data iesire din carantina 14-JAN-21
NU au iesit persoane din carantina

```

- *Explicații cod:*

Pe cod în comentarii.