Proiectarea si Interogarea

unei baze de date pt DSP

Fmi unibuc | Str. Academiei 14, Bucuresti

proiect SGBD – an 2 SEM 1

Paula Iuga

2020

CUPRINS:

[1.Baza de date și utilitatea ei 2](#_Toc60331754)

[2.Diagrama Entitate-Relatie 3](#_Toc60331755)

[3. Diagrama Conceptuală 4](#_Toc60331756)

[4. Implementarea Diagramei Conceptuale în Oracle 6](#_Toc60331757)

[5. Adăugarea informațiilor în tabelele create 14](#_Toc60331758)

[6. Un subprogram stocat care utilizează o colecție 26](#_Toc60331759)

[7. Un subprogram stocat care utilizează un cursor 32](#_Toc60331760)

[8. Un subprogram stocat de tip funcție care utilizează 3 tabele.Excepții. 35](#_Toc60331762)

[9. Un subprogram stocat de tip procedură care utilizează 5 tabele. Excepții. 43](#_Toc60331764)

[10. Un trigger de tip LMD la nivel comandă 48](#_Toc60331765)

[11. Un trigger de tip LMD la nivel linie 51](#_Toc60331766)

[12. Un trigger de tip LDD 55](#_Toc60331767)

[13. Un pachet care să conțină toate obiectele definite la cerințele anterioare 58](#_Toc60331768)

[14. Un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate 71](#_Toc60331769)

1.Baza de date și utilitatea ei

Baza de date modeleaza succint modelul unei organizații de tip DSP și este inspirata din contextul actual al pandemiei de covid 19. Este realizată în vederea unei interogari mai ușoare a pacientilor infectați, spitalizați si carantinați din cauza noului virus.

Baza de date se împarte în 3 cateorii:

* 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 teritoriali care investighează o anumită zonă( verifica 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 intr-un centru dintr-o anumită locație.

* Persoanele fizice. Acestea au un medic de familie, un pachet de servicii(ce conține o lista de servicii), se află într-o locatie 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 campul test\_covid care are tipul Bool : 0 pt pacientii negativi, 1 pt pacientii pozitivi. Data testului este trecuta in campul de tip date, data\_testare.

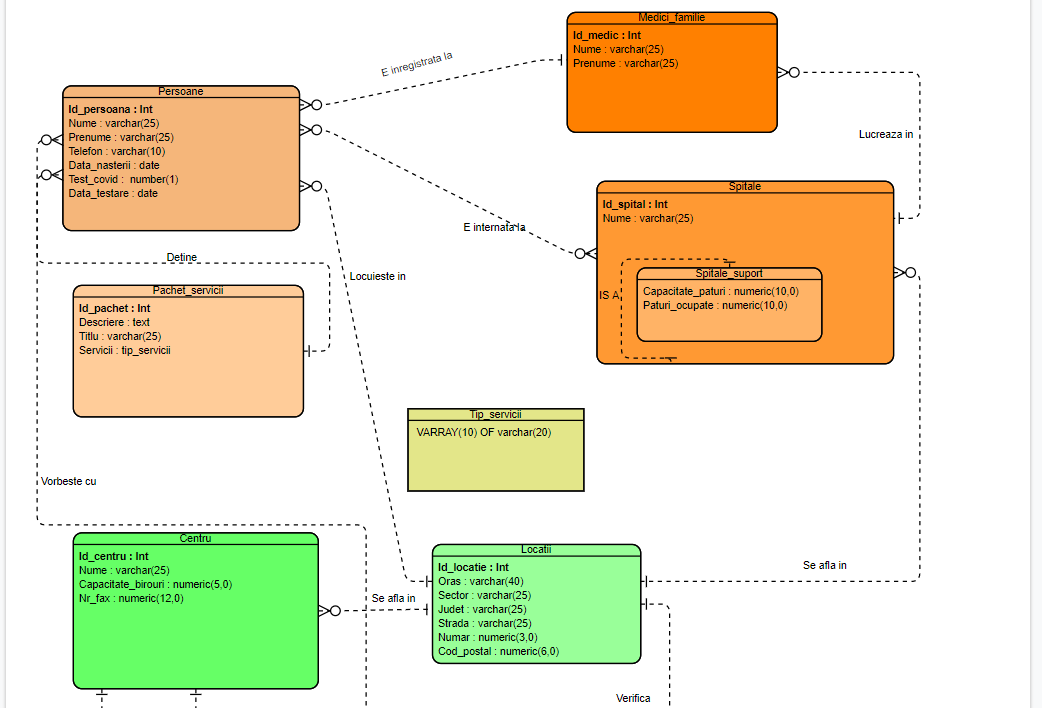
**Nota**! *Câmpurile test\_covid și data\_testare rețin datele ultimului test făcut, dacă persoana nu a mai făcut teste vor fi null.*

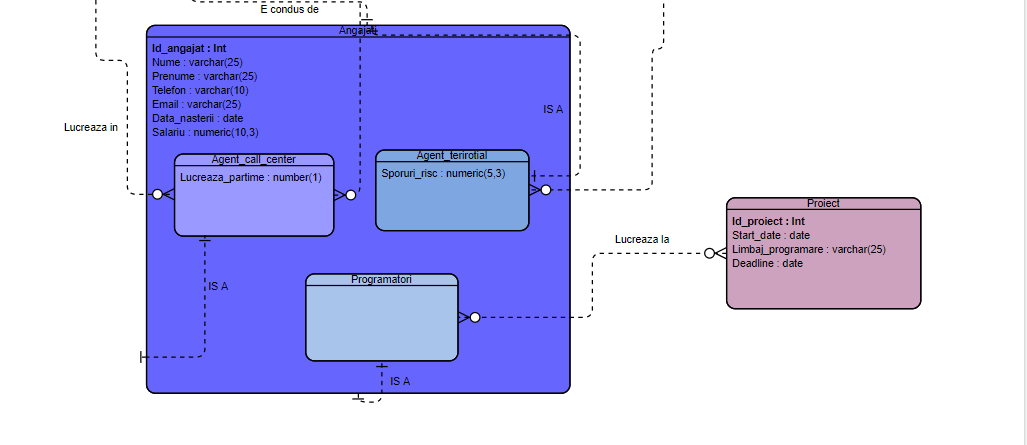
* Sistemul medical. Acesta cuprinde spitalele. In spitale lucreaza medici de familie iar unele spitale sunt suport covid adică aici se internează persoanele grav afectate de virus.

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

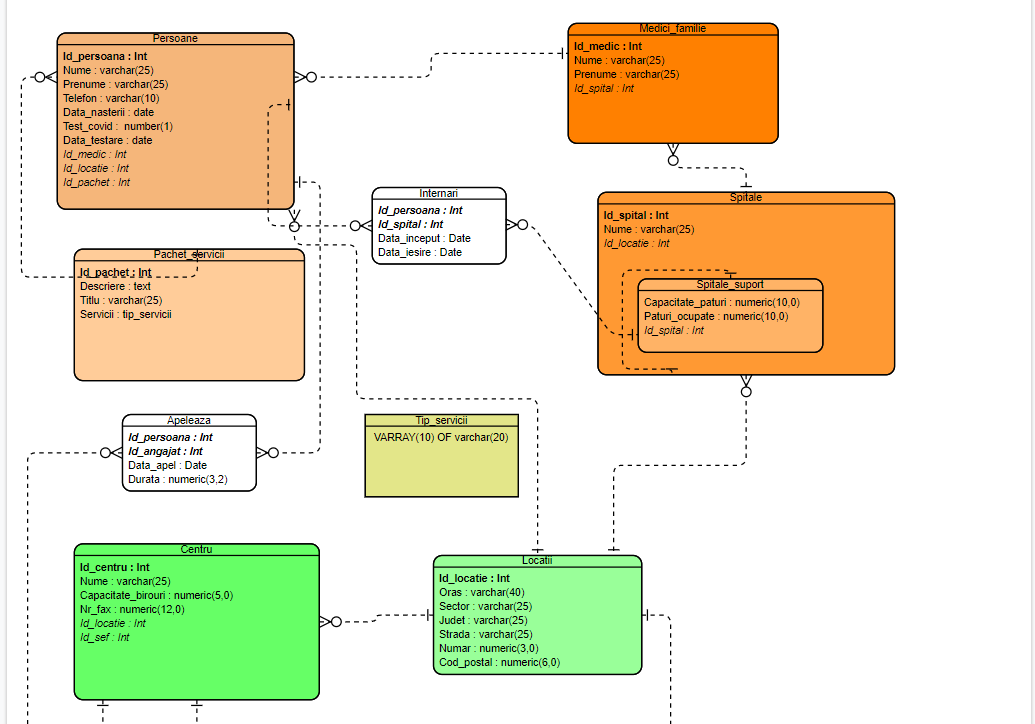
Pe baza diagramei Entitate – Relatie am construit diagrama Conceptuală transformând corespunzător relațiile și entitățile.

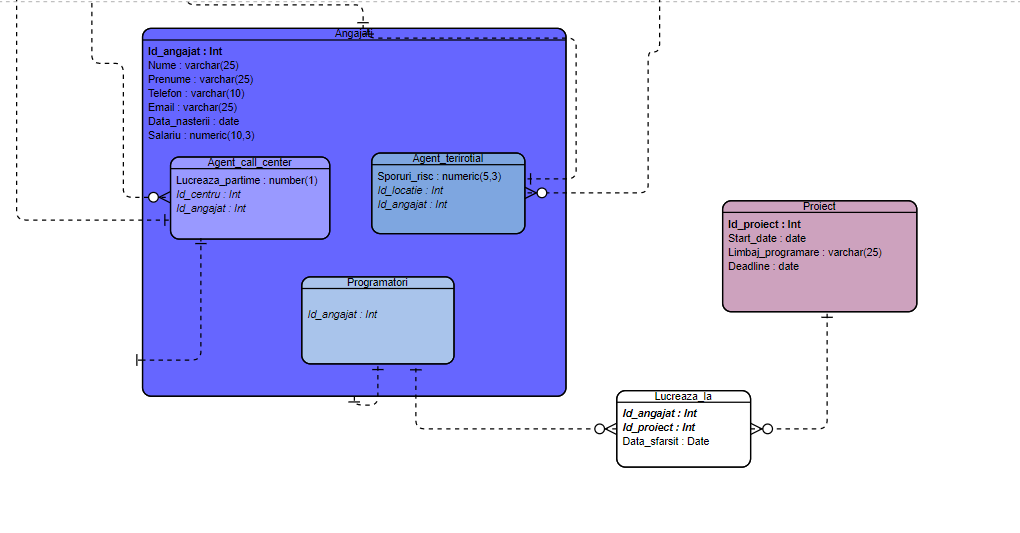
Principalele modificări au fost :

* Transformarea realțiilor 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 ceilor 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 Conceputală. 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,

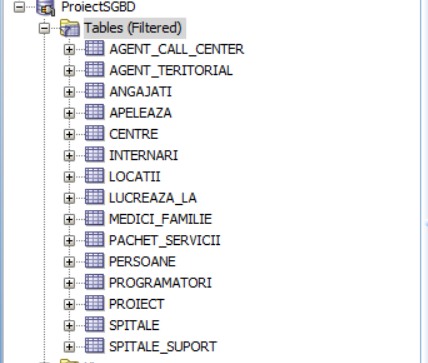
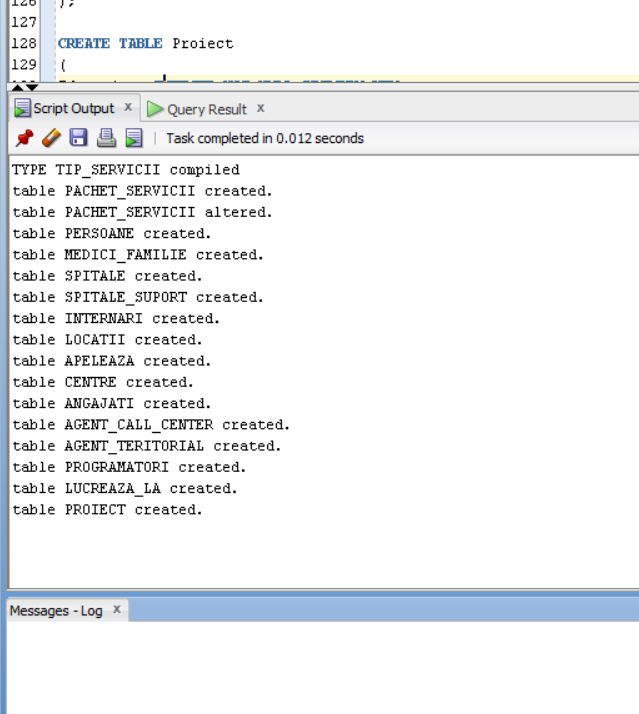
Start\_date DATE,

Limbaj\_programare VARCHAR(25),

Deadline DATE

);

Codul pt creerea tabelelor funcționează și genereaza tabelele:





Codul pt adăugarea constrângerilor de cheie externă:

ALTER TABLE Persoane

ADD CONSTRAINT P\_FK FOREIGN KEY(Id\_medic) REFERENCES Medici\_Familie(Id\_medic);

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);

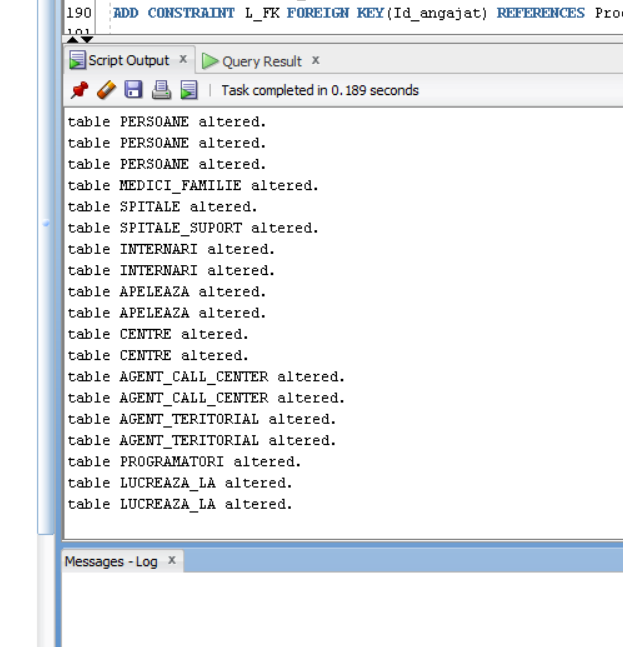
ALTER TABLE Lucreaza\_la

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 tabelelor:

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, 'Jerau', '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);

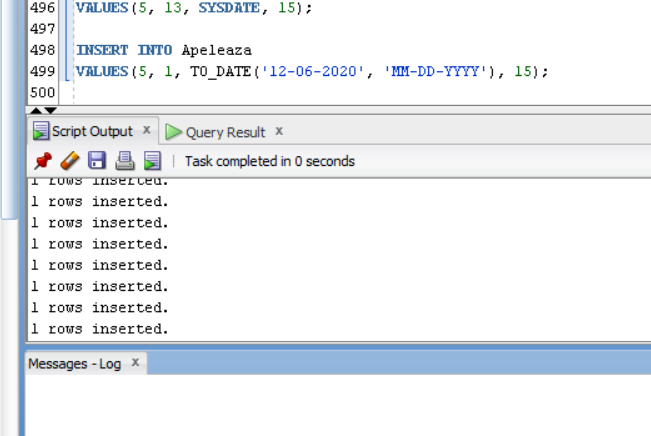
INSERT INTO Apeleaza

VALUES(5, 13, SYSDATE, 15);

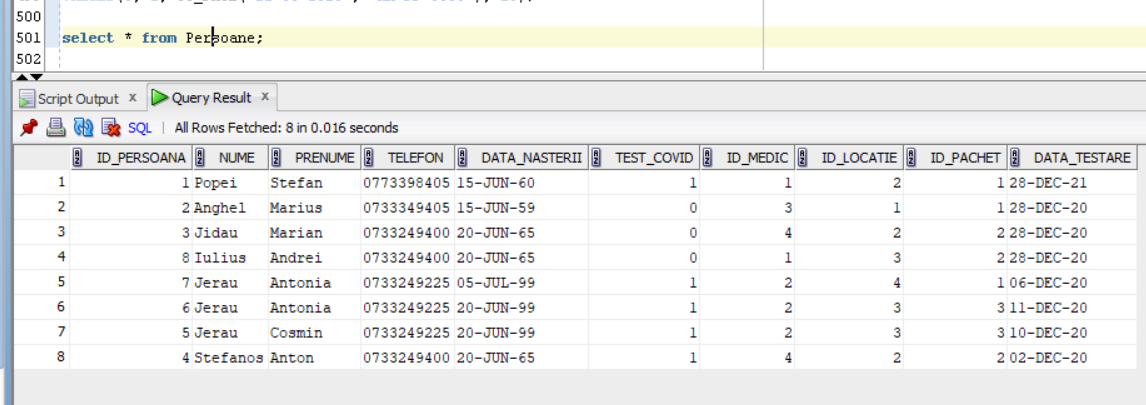
INSERT INTO Apeleaza

VALUES(5, 1, TO\_DATE('12-06-2020', 'MM-DD-YYYY'), 15);

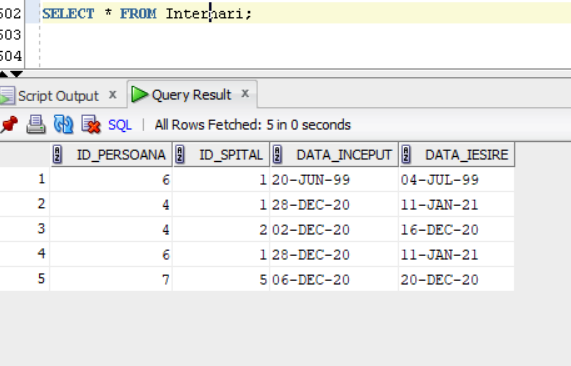
Adăugarea înregistrarilor în tabele a fost realizată cu succes, în urma acesteia am obținut o secvență:



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



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



6. Un subprogram stocat care utilizează o colecție

Voi creea 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.

1. Pentru persoanele care nu dețin pachete cu Internare voi afișa daca au de plătit internarea si unde s-au internat pana acum.
2. Pentru cele tot fară acest pachet care nu s-au internat și mai sunt încă 14 zile de la data infectarii voi afisa ca se pot interna cu conditia de plata a serviciilor
3. Pentru persoanele cu pachete care conțin și spitalizarea voi afișa fară 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 l 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.nume || '\* ');

v\_ok2 := 1;

ELSE

DBMS\_OUTPUT.PUT('\*'||l.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;

/

*Apelul procedurii*:

DECLARE

v\_nr1 NUMBER;

v\_nr2 NUMBER;

BEGIN

plata\_spitalizare(v\_nr1, v\_nr2);

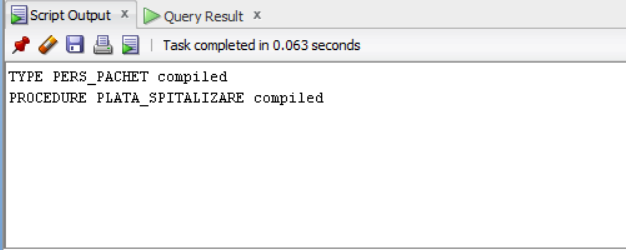
DBMS\_OUTPUT.PUT\_LINE('Numarul de persoane care au achizitionat un pachet CU internari este: ' || v\_nr1);

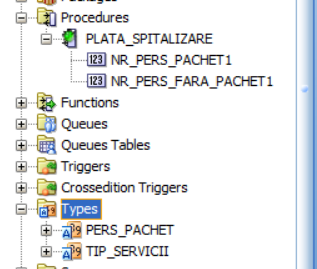
DBMS\_OUTPUT.PUT\_LINE('Numarul de Persoane care au achizitionat un pachet FARA internari este: ' || v\_nr2);

END;

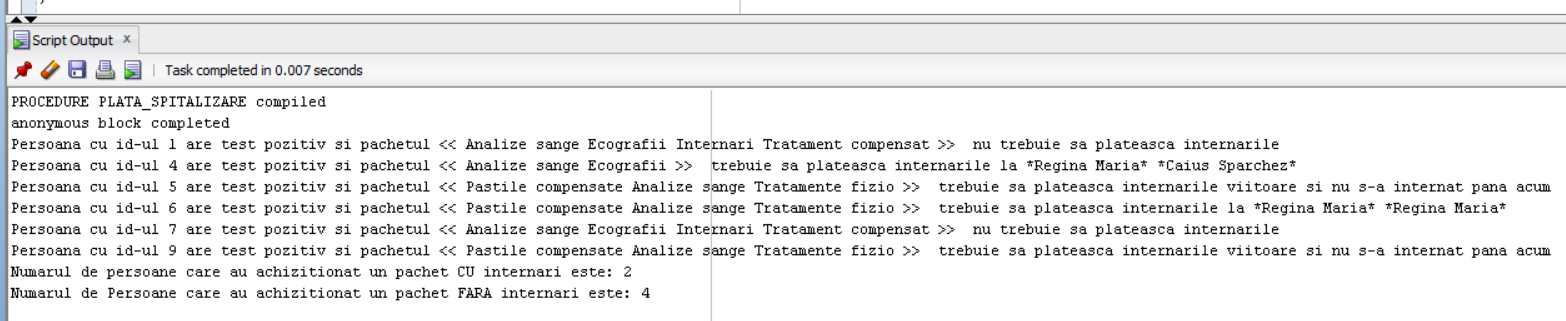
/

Obiectul și procedura compilează:





Apelul procedurii generează:



* *Explicații cod*:

Comanda SET SERVEROUTPUT ON a fost utilizată pentru a putea vedea in Script Output ce afiseaza funcțiile PUT si NEW\_LINE din pachetul DBMS\_OUTPUT.

Am creat *un obiect pers\_pachet* care retine tripuri 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(acestea sunt chiar contorul folosit pt inserarea datelor in tabel) și nr. persoanelor care au un astel 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 pozitiva si 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 pesoana *vectorul ei de servicii t(j).pachet\_servicii* si verifica 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 creea un **subprogram stocat de tip funcție**  fără parametrii care **returnează** numarul de pachete (**Number**) 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;

/

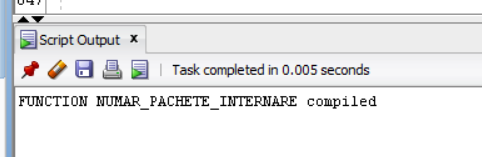
Apelul funcției in SQL\* PLUS cu variabila HOST:

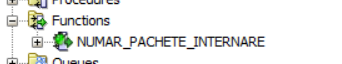
VARIABLE nr NUMBER

EXECUTE :nr := numar\_pachete\_internare;

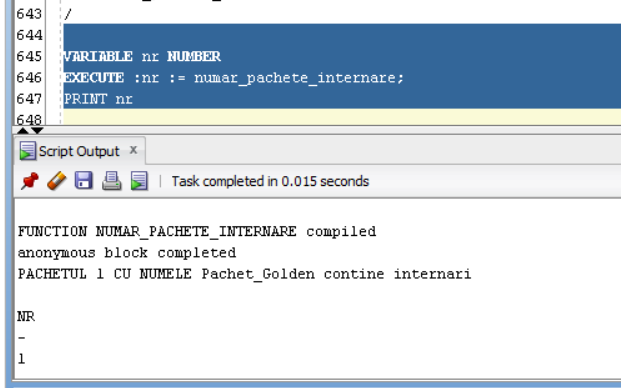
PRINT nr

Funcția compilează:





Apelul procedurii generează:



* *Explicații cod*:

Am definit un tip numit refcusor care este un **REF Cursor**, un **cursor explicit** c1 care reține pt fiecare pachet numele lui și id -ul, dar și un subcursor care renține 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 cusorul cursor\_aux. Loopul se oprește când în cursor nu mai sunt date de adăugat. Apoi parcurg datele din ref cursor adica lista cu servicii a fiecărui pachet și verific dacă unul dintre acestea este egal cu ref Internari, dacă da afișez și contorizez.

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

Excepții.

Am creeat o funcție care pentru un programator dat ca paramentru 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 funției 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 specialziare!');

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;

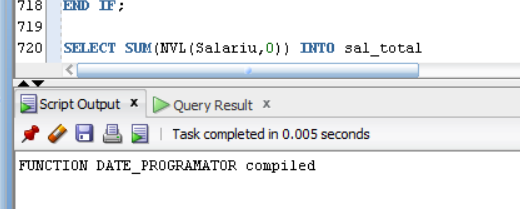
procent\_salariu := sal\*100/sal\_total;

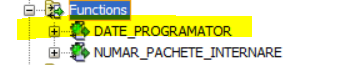
RETURN procent\_salariu;

END date\_programator;

/

Funcția compilează:



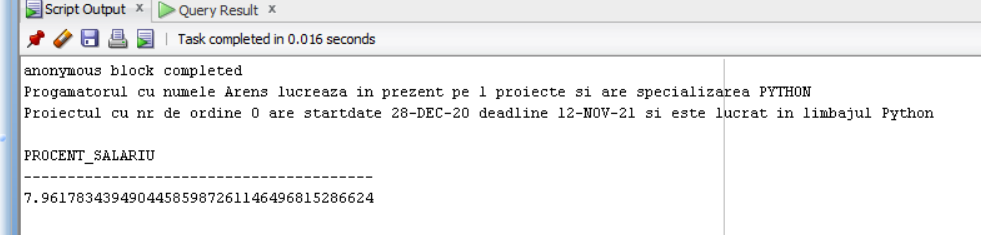


Apelul funcției:

VARIABLE procent\_salariu NUMBER

EXECUTE :procent\_salariu :=date\_programator('Arens');

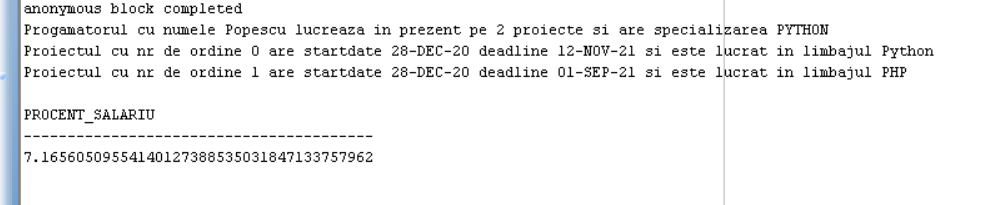
PRINT procent\_salariu



VARIABLE procent\_salariu NUMBER

EXECUTE :procent\_salariu :=date\_programator('Popescu');

PRINT procent\_salariu

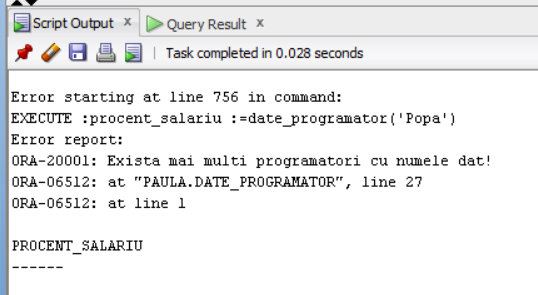


***Pt erori:***

VARIABLE procent\_salariu NUMBER

EXECUTE :procent\_salariu :=date\_programator('Popa');

PRINT 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);

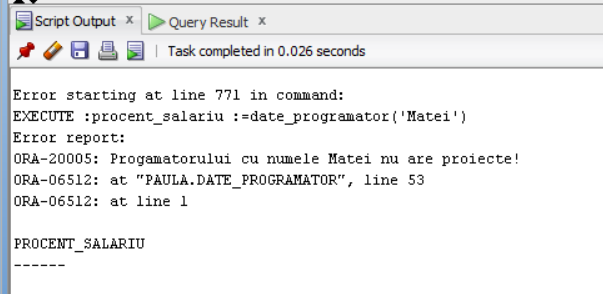
INSERT INTO PROGRAMATORI

VALUES (16,'Java');

VARIABLE procent\_salariu NUMBER

EXECUTE :procent\_salariu :=date\_programator('Matei');

PRINT procent\_salariu



--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');

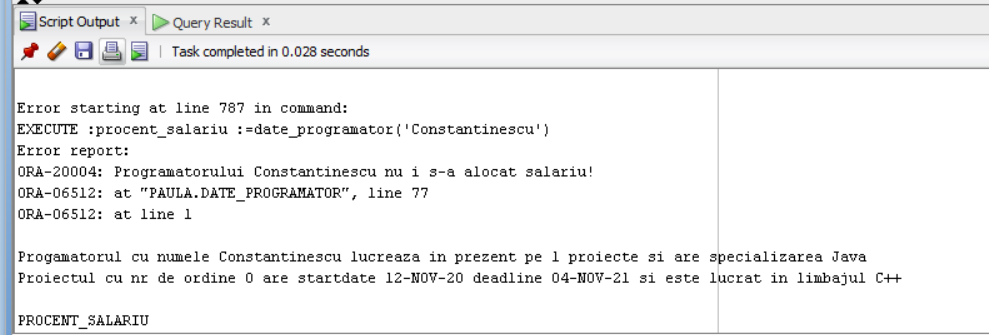
INSERT INTO Lucreaza\_la

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);

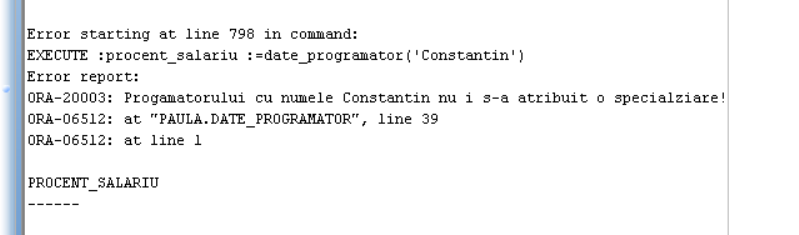
INSERT INTO PROGRAMATORI

VALUES (18,null);

VARIABLE procent\_salariu NUMBER

EXECUTE :procent\_salariu :=date\_programator('Constantin');

PRINT procent\_salariu



* *Explicații cod*:

Pt. a putea folosi **mai multe blocuri EXCEPTION** am folosit mai multe **sublocuri** 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 parametu. 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 stocata care marește cu 10 % salariul agenților call center apelați de cele mai multe ori de persoanele care au medici de familie într-un spital dat ca paramentru.

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(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

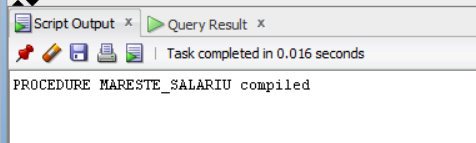
DBMS\_OUTPUT.PUT\_LINE('Salariul depaseste limita de 10000!');

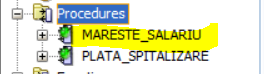
END;

END mareste\_salariu;

/

Procedura compilează:





Apelul procedurii:

DECLARE

v\_nr NUMBER;

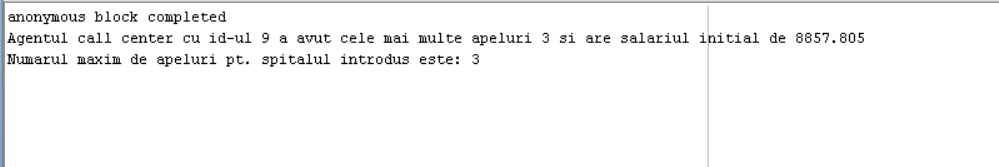
BEGIN

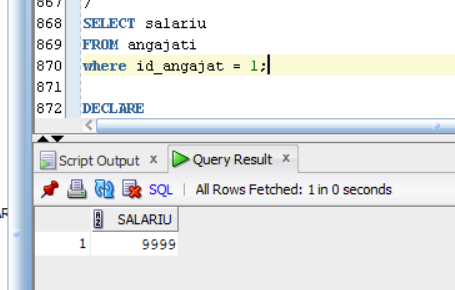
mareste\_salariu('Regina Maria',v\_nr);

DBMS\_OUTPUT.PUT\_LINE('Numarul maxim de apeluri pt. spitalul introdus este: ' || v\_nr);

END;

/





* Pt. Excepții:

DECLARE

v\_nr NUMBER;

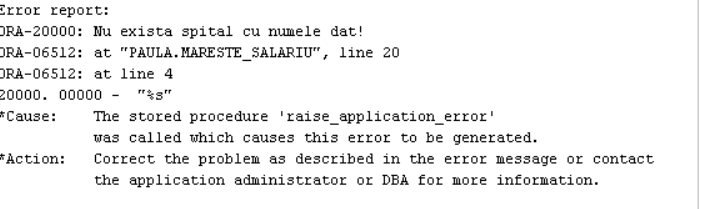
BEGIN

mareste\_salariu('Caius Sparez',v\_nr);

DBMS\_OUTPUT.PUT\_LINE('Numarul maxim de apeluri pt. spitalul introdus este: ' || v\_nr);

END;

/



DECLARE

v\_nr NUMBER;

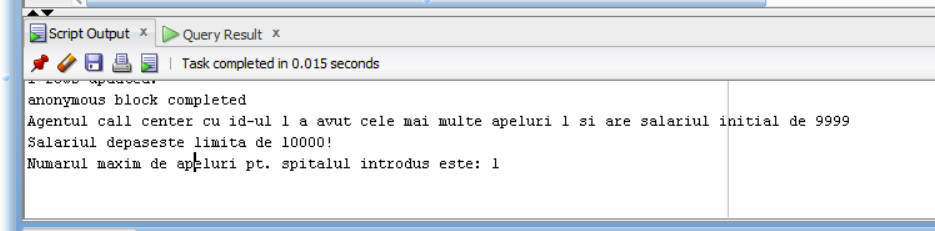
BEGIN

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 pesoanelor care au un medic de familie în spitalul dat ca parametru și am calculat maximul de apeleluri salvând si id-ul angajatului care le-a onorat în v\_id\_max.Apoi ii modific salariul daca 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 cand s-a facut modificarea, de care cine si ce se modifica.

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

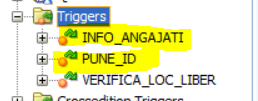
INTO :new.ID

FROM dual;

END;

/

Cei 2 triggeri și secvența compileză:

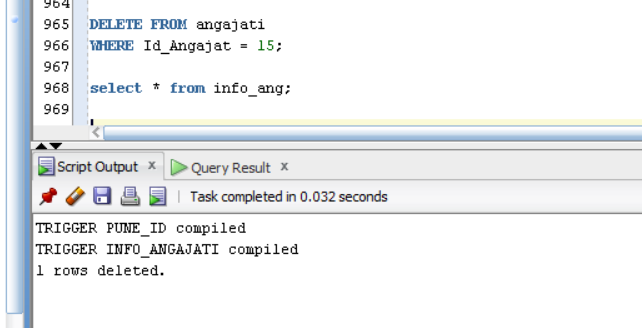
 

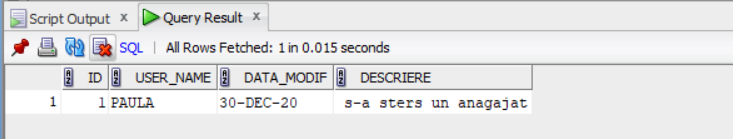
Declanșare trigger:

DELETE FROM angajati

WHERE Id\_Angajat = 15;

select \* from info\_ang;



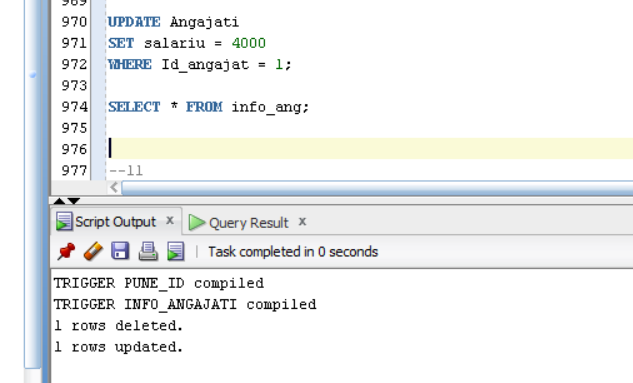


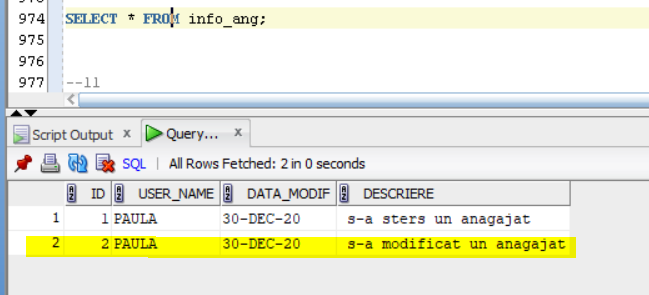
UPDATE Angajati

SET salariu = 4000

WHERE Id\_angajat = 1;

SELECT \* FROM info\_ang;





* *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 la nivel linie pentru Insert pe tabelul Spitale\_suport, care, verifică că nu pot fi mai mult de 2 spitale suport într-o locatie.

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');

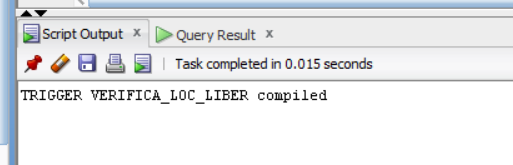
END IF;

END AFTER EACH ROW;

END verifica\_loc\_liber;

/

Trigger ul compilează:





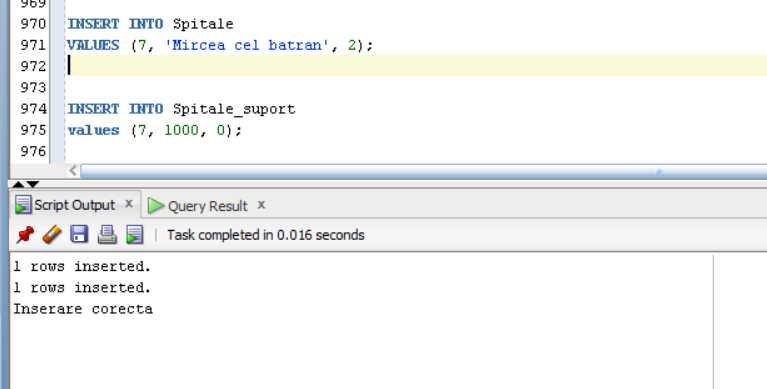
Declanșarea triggerului:

INSERT INTO Spitale

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

INSERT INTO Spitale\_suport

values (7, 1000, 0);

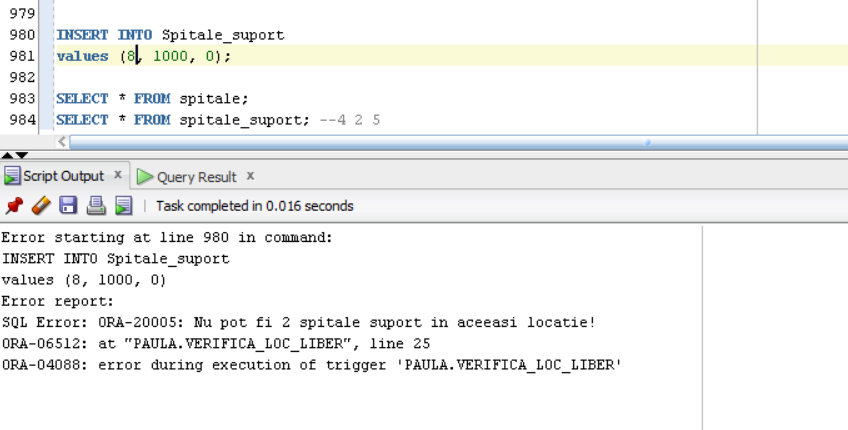


INSERT INTO Spitale

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

INSERT INTO Spitale\_suport

values (8, 1000, 0);



* *Explicații cod:*

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

12. Un trigger de tip LDD

Un trigger care apeleaza o porcedură și inserează intr-un tabel info ce user a făcut modificarea, ce tip de modificare a fost făcută, când și pe ce obiect din schema.

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.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;

/

CREATE OR REPLACE TRIGGER ldd\_trigger

BEFORE CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

insereaza(SYS.SYSEVENT,SYS.DICTIONARY\_OBJ\_NAME);

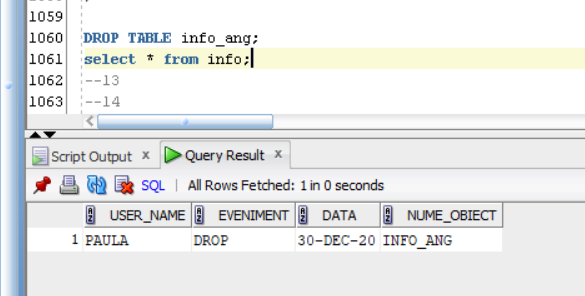
END;

/

Apelul tiggerului:

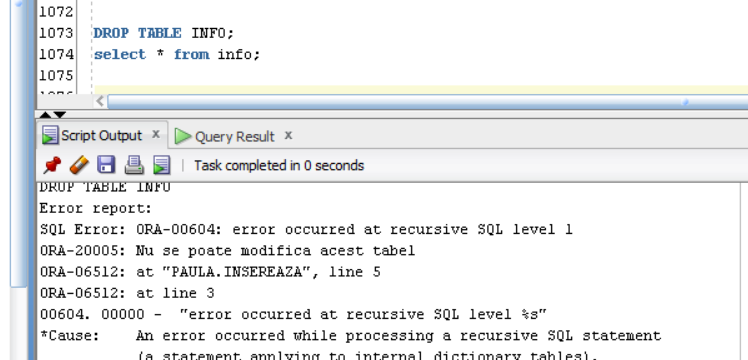
DROP TABLE info\_ang;

SELECT \* FROM info;



DROP TABLE INFO;

select \* from info;



* *Explicații cod:*

Am creat **o procedură** care primește **ca paramentrii de intrare**(tip IN) evenimentul si numele obiectului și inserează în tabelul info. Triggerul se declanșează la **orice comandă alter/create/delete** si apelează procedura. Dacă numele obiectului pe care s-a facut alter/create/delete este info, atunci se **aruncă o eroare** întrucat 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.nume\_obiect%TYPE);

PROCEDURE mareste\_salariu(nume\_spital IN Spitale.Nume%TYPE, nr\_apeluri OUT NUMBER);

FUNCTION date\_programator(param\_nume 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 specialziare!');

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;

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;

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 l 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.nume || '\* ');

v\_ok2 := 1;

ELSE

DBMS\_OUTPUT.PUT('\*'||l.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;

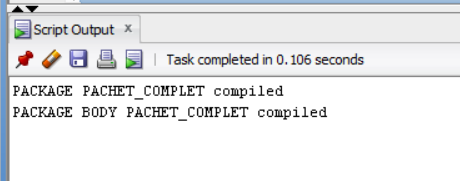
nr\_pers\_fara\_pachet := nr\_pers\_fara\_pachet - nr\_pers\_pachet;

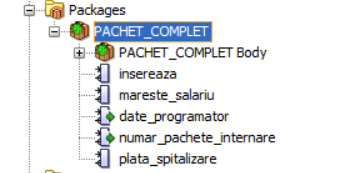
END plata\_spitalizare;

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);

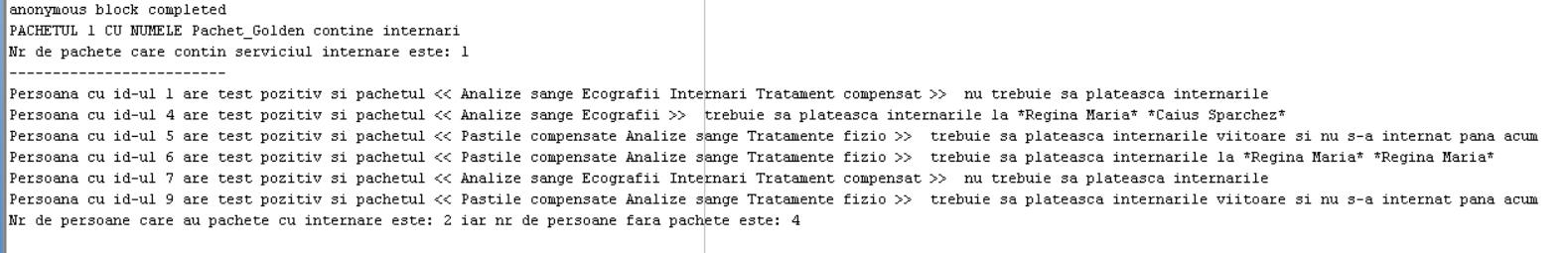
DBMS\_OUTPUT.PUT\_LINE('-------------------------');

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;

/



Triggerul ldd rescris, astfel încăt să folosească procedura insereaza din pachet:

CREATE OR REPLACE

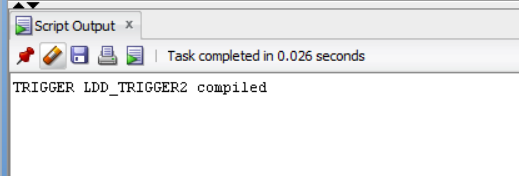
TRIGGER ldd\_trigger2

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 creea 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 li 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 creeaza\_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 ');

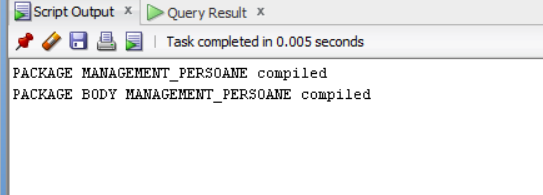
END IF;

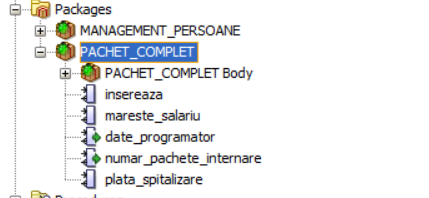
END scoate\_din\_carantina;

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 SUPORT 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!');

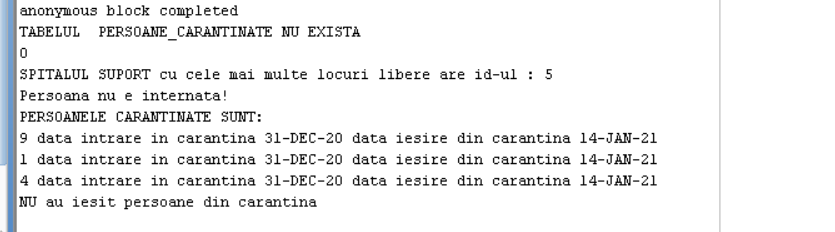
END IF;

management\_persoane.interneaza\_persoana;

management\_persoane.scoate\_din\_carantina;

END;

/



* *Explicații cod:*

Pe cod în comentarii.