**GESTIUNEA UNUI SPITAL**

**-Proiect SISTEME DE GESTIUNE A BAZELOR DE DATE-**

**Ion Apelia-Cosmina**

**Grupa 231**

**Universitatea din Bucuresti**

**Facultatea de Matematica si Informatica**

Contents

[--1 Descrierea modelului 2](#_Toc92484396)

[--1.2 Restrictii de functionare 2](#_Toc92484397)

[--2 Diagrama E/R 3](#_Toc92484398)

[--3 Diagrama conceptuala 4](#_Toc92484399)

[--4&5 Implementare diagramă conceptuală și inserare informații în tabele 5](#_Toc92484400)

[--4&5.1Print-Screen-uri ce demonstrează rularea codului 14](#_Toc92484401)

[--6 Problema rezolvata folosind un subprogram care utilizează 2 tipuri de colecții studiate 15](#_Toc92484402)

[--6.1 Print-screen ce dovedeste rularea codului in Oracle 16](#_Toc92484403)

[--7 Problema rezolvata folosind un subprogram care utilizează un tip de cursor 17](#_Toc92484404)

[--7.1 Print-screen ce dovedeste rularea codului in Oracle 18](#_Toc92484405)

[--8 Functie ce utilizează într-o singură comanda SQL 3 dintre tabelele din baza de date 18](#_Toc92484406)

[--8.1 Print-screen ce dovedeste rularea codului in Oracle 20](#_Toc92484407)

[--9 Procedura ce utilizează într-o singură comanda SQL 5 dintre tabelele din baza de date 21](#_Toc92484408)

[--9.1 Print-screen ce dovedeste rularea codului in Oracle 23](#_Toc92484409)

[--10 Trigger LMD la nivel de comandă 23](#_Toc92484410)

[--10.1 Print-screen ce dovedeste rularea codului in Oracle 24](#_Toc92484411)

[--11 Trigger LMD la nivel de linie 25](#_Toc92484412)

[--11.1 Print-screen ce dovedeste rularea codului in Oracle 26](#_Toc92484413)

[--12 Trigger de tip LDD 26](#_Toc92484414)

[--12.1 Print-screen ce dovedeste rularea codului in Oracle 27](#_Toc92484415)

[--13 Pachet care contine obiectele definite 27](#_Toc92484416)

[--13.1 10.1 Print-screen ce dovedeste rularea codului in Oracle 31](#_Toc92484417)

# --1 Descrierea modelului

Modelul de date ce urmeaza a fi descris va gestiona informatii legate de organizarea si functionarea unui spital, mai exact gestiunea lucratorilor si a pacientilor internati in cadrul acestuia.

Spitalul are mai multe **cladiri**, fiecare avand un numar variabil de **etaje**. In cadrul fiecarui etaj se incadreaza **sali** de diferite tipuri. Un etaj poate gazdui un singur **departament** (ATI, Medicina interna, Boli infectioase, etc). Departamentul reprezinta principala unitate organizatorica a spitalului, ce imparte angajatii in functie de specializarea lor, pentru a putea oferi pacientilor cel mai potrivit tratament, in functiile de nevoile acestora.

**Angajatii** pot avea mai multe **job**-uri, pornind de la cele de maxima importanta pentru spital, medici, asistente, pana la agenti de securitate, ingrijjitori etc. Medicii si asistentele pot avea in grija pacienti si le pot pune un **diagnostic**. Diagnosticul determina aplicarea unui **tratament**.

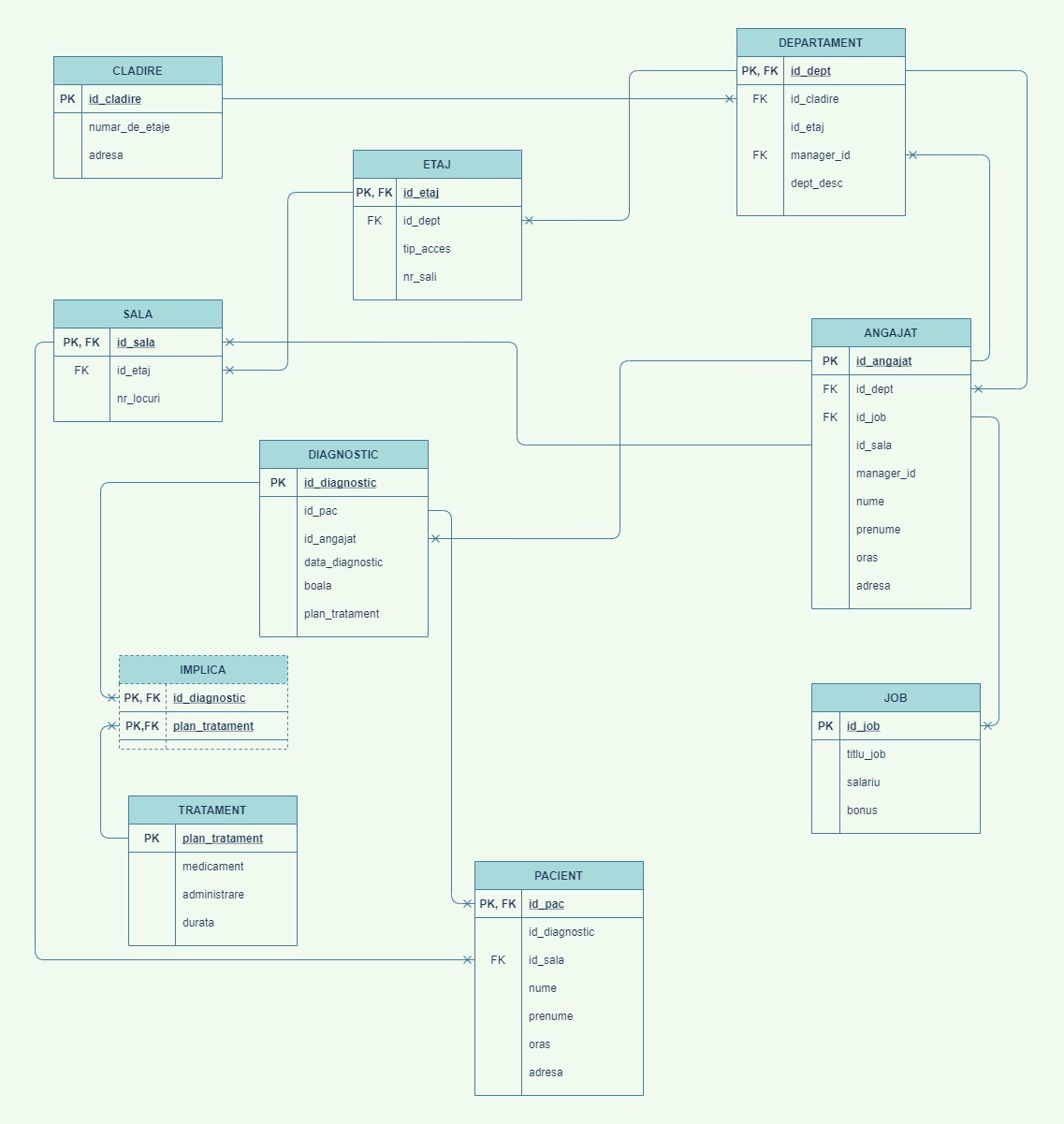
**Pacientii** sunt internati in sali.

## --1.2 Restrictii de functionare

* Intr-o cladire pot exista mai multe departamente.
* Un departament poate ocupa un singur etaj dintr-o cladire.
* Un angajat poate avea un singur job in cadrul spitalului, deci lucreaza intr-un singur departament.
* Un departament poate fi condus de un singur angajat (mannager).
* Un pacient poate avea un singur diagnostic in cadrul internarii curente.

# --2 Diagrama E/RC:\Users\ionap\Desktop\PROIECT BD DIAGRAMA E_R (2).jpg

# --3 Diagrama conceptuala



# --4&5 Implementare diagramă conceptuală și inserare informații în tabele

--Cladire----------------------------------------------------------------------

CREATE TABLE CLADIRE (

id\_cladire VARCHAR(5) PRIMARY KEY,

nr\_de\_etaje NUMBER(2),

adresa VARCHAR(25)

);

INSERT INTO CLADIRE (ID\_CLADIRE, NR\_DE\_ETAJE, ADRESA)

VALUES ('A1', 3, ' Floreasca, 5');

INSERT INTO CLADIRE (ID\_CLADIRE, NR\_DE\_ETAJE, ADRESA)

VALUES ('A2', 2, 'Floreasca, 7');

INSERT INTO CLADIRE (ID\_CLADIRE, NR\_DE\_ETAJE, ADRESA)

VALUES ('A3', 1, 'Floreasca, 11');

INSERT INTO CLADIRE (ID\_CLADIRE, NR\_DE\_ETAJE, ADRESA)

VALUES ('B1', 3, 'Floreasca, 9');

INSERT INTO CLADIRE (ID\_CLADIRE, NR\_DE\_ETAJE, ADRESA)

VALUES ('C1', 3, 'Floreasca, 12');

--Departament---------------------------------------------------------------------

CREATE TABLE DEPARTAMENT (

id\_dept VARCHAR(5) PRIMARY KEY,

id\_cladire VARCHAR(5),

id\_etaj varchar(5),

id\_angajat number(5),

manager\_id number(5),

dept\_descr VARCHAR(20)

);

INSERT INTO DEPARTAMENT (ID\_DEPT, ID\_CLADIRE, ID\_ETAJ, MANAGER\_ID, DEPT\_DESCR)

VALUES ('U', 'A1', 'A1\_1', 1, 'Urgente');

INSERT INTO DEPARTAMENT (ID\_DEPT, ID\_CLADIRE, ID\_ETAJ, MANAGER\_ID, DEPT\_DESCR)

VALUES ('ATI', 'A1', 'A1\_2', 12, 'Terapie intensiva');

INSERT INTO DEPARTAMENT (ID\_DEPT, ID\_CLADIRE, ID\_ETAJ, MANAGER\_ID, DEPT\_DESCR)

VALUES ('MI', 'A1', 'A1\_3', 5, 'Medicina interna');

INSERT INTO DEPARTAMENT (ID\_DEPT, ID\_CLADIRE, ID\_ETAJ, MANAGER\_ID, DEPT\_DESCR)

VALUES ('OFT', 'A2', 'A2\_1', 6, 'Oftalmologie');

INSERT INTO DEPARTAMENT (ID\_DEPT, ID\_CLADIRE, ID\_ETAJ, MANAGER\_ID, DEPT\_DESCR)

VALUES ('DERM', 'A2', 'A2\_2', 10, 'Dermatologie');

INSERT INTO DEPARTAMENT (ID\_DEPT, ID\_CLADIRE, ID\_ETAJ, MANAGER\_ID, DEPT\_DESCR)

VALUES ('NEF', 'B1', 'B3\_1', 8, 'Nefrologie');

--Etaj----------------------------------------------------------------------------

CREATE TABLE ETAJ (

id\_etaj varchar(5) PRIMARY KEY,

id\_dept VARCHAR(5),

tip\_acces VARCHAR(10),

nr\_sali NUMBER (2)

);

INSERT INTO ETAJ (ID\_ETAJ, ID\_DEPT, TIP\_ACCES, NR\_SALI)

VALUES ('A1\_1', 'U', 'general', 5);

INSERT INTO ETAJ (ID\_ETAJ, ID\_DEPT, TIP\_ACCES, NR\_SALI)

VALUES ('A1\_2', 'ATI', 'angajati', 7);

INSERT INTO ETAJ (ID\_ETAJ, ID\_DEPT, TIP\_ACCES, NR\_SALI)

VALUES ('A1\_3', 'MI', 'angajati', 4);

INSERT INTO ETAJ (ID\_ETAJ, ID\_DEPT, TIP\_ACCES, NR\_SALI)

VALUES ('A2\_1', 'OFT', 'general', 2);

INSERT INTO ETAJ (ID\_ETAJ, ID\_DEPT, TIP\_ACCES, NR\_SALI)

VALUES ('A2\_2', 'DERM', 'general', 2);

INSERT INTO ETAJ (ID\_ETAJ, ID\_DEPT, TIP\_ACCES, NR\_SALI)

VALUES ('B1\_1', 'NEF', 'general', 6);

--SALA-----------------------------------------------------------------------------

CREATE TABLE SALA(

id\_sala VARCHAR(5) PRIMARY KEY,

id\_etaj varchar(5),

nr\_locuri NUMBER(3)

);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('U1', 'A1\_1', 3);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('Ati1', 'A1\_2', 15);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('Ati2', 'A1\_2', 20);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('Oft1', 'A2\_2', 4);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('Nef1', 'B1\_1', 2);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('MI1', 'A1\_3', 10);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('MI2', 'A1\_3', 7);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('Mi3', 'A1\_3', 1);

INSERT INTO SALA (ID\_SALA, ID\_ETAJ, NR\_LOCURI)

VALUES ('U2', 'A1\_1', 2);

--LEGATURI\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE DEPARTAMENT

ADD CONSTRAINT leg\_cladire\_dept

FOREIGN KEY (id\_cladire) REFERENCES CLADIRE (id\_cladire) ON DELETE SET NULL;

ALTER TABLE ETAJ

ADD CONSTRAINT leg\_dept\_etaj

FOREIGN KEY (id\_dept) REFERENCES DEPARTAMENT (id\_dept) ON DELETE SET NULL;

ALTER TABLE SALA

ADD CONSTRAINT leg\_etaj\_sala

FOREIGN KEY (id\_etaj) REFERENCES ETAJ (id\_etaj) ON DELETE SET NULL;

--Angajat---------------------------------------------------------------------------------

CREATE TABLE ANGAJAT (

id\_angajat NUMBER(5) PRIMARY KEY,

id\_dept VARCHAR(5),

id\_job VARCHAR(5),

id\_sala VARCHAR(5),

manager\_id NUMBER(5),

nume VARCHAR(10),

prenume VARCHAR(20),

oras VARCHAR(10),

adresa VARCHAR(15)

);

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (1, 'U', 'Med', 'U1', NULL, 'Nelu', 'Constantin', 'Bucuresti', 'str. sarii');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (2, 'U', 'Asist', NULL, 1, 'Ion', 'Agripina', 'Cluj', 'str. Siret');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (3, 'MI', 'Med', NULL, 3, 'Maria', 'Andreea', 'Giurgiu', 'Str. Campaniei');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (4, 'MI', 'An', NULL, 3, 'Georgescu', 'Adrian', 'Bucuresti', 'Str. Proiect');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (5, 'MI', 'Ch', 'Mi3', NULL, 'Neculce', 'Marian', 'Bucuresti', 'Str Scanteii');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (6, 'OFT', 'Med', NULL, NULL, 'Stan', 'Maria', 'Bucuresti', 'str. Teilor');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (7, 'OFT', 'Asist', NULL, 6, 'Dinu', 'Alexandra', 'Giurgiu', 'str. Pietrelor');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (8, 'NEF', 'Med', 'Nef1', NULL, 'Stanescu', 'Paul', 'Pitesti', 'str. Valurilor');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (9, 'NEF', 'Asist', NULL, 8, 'Velicu', 'Cristina', 'Bucuresti', 'str. Sadoveanu');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (10, 'DERM', 'Med', NULL, NULL, 'Oprea', 'Luminita', 'Ploiesti', 'str. Unirii');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (11, 'DERM', 'Asist', NULL, 10, 'Oancea', 'Georgiana', 'Galati', 'str. Plopilor');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (12, 'ATI', 'Med', NULL, NULL, 'Udrea', 'Ileana', 'Iasi', 'str. Depozit');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (13, 'ATI', 'Med', NULL, 12, 'Mihail ', 'Constantin', 'Bucuresti', 'str. Florilor');

INSERT INTO ANGAJAT (ID\_ANGAJAT, ID\_DEPT, ID\_JOB, ID\_SALA, MANAGER\_ID, NUME, PRENUME, ORAS, ADRESA)

VALUES (14, 'ATI', 'Asist', NULL, 12, 'Manea', 'Thomas', 'Bucuresti', 'bd. Timisoara');

--LEGATURI\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE ANGAJAT

ADD CONSTRAINT leg\_dept\_angajat

FOREIGN KEY (id\_dept) REFERENCES DEPARTAMENT (id\_dept) ON DELETE SET NULL;

ALTER TABLE DEPARTAMENT

ADD CONSTRAINT leg\_angajat\_dept

FOREIGN KEY (manager\_id) REFERENCES angajat(id\_angajat) ON DELETE SET NULL;

--JOB------------------------------------------------------------------------------

CREATE TABLE JOB (

id\_job VARCHAR(5) PRIMARY KEY,

titlu\_job VARCHAR(15),

salariu NUMBER(6),

bonus NUMBER(5)

);

INSERT INTO JOB (ID\_JOB, TITLU\_JOB, SALARIU, BONUS)

VALUES ('Med', 'Medic Rezident', 5000, 1000);

INSERT INTO JOB (ID\_JOB, TITLU\_JOB, SALARIU, BONUS)

VALUES ('Asist', 'Asistent', 3000, 500);

INSERT INTO JOB (ID\_JOB, TITLU\_JOB, SALARIU, BONUS)

VALUES ('Sec', 'Agent sec.', 1500, 1500);

INSERT INTO JOB (ID\_JOB, TITLU\_JOB, SALARIU, BONUS)

VALUES ('An', 'Anestezist', 7000, 4500);

INSERT INTO JOB (ID\_JOB, TITLU\_JOB, SALARIU, BONUS)

VALUES ('Ch', 'Chirurg', 10000, 6000);

--LEG\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE Angajat

ADD CONSTRAINT leg\_angajat\_job

FOREIGN KEY (id\_job) REFERENCES Job (id\_job) ON DELETE SET NULL;

--Diagnostic------------------------------------------------------------------------

CREATE TABLE DIAGNOSTIC (

id\_diagnostic VARCHAR(10) PRIMARY KEY,

id\_pac VARCHAR(10),

id\_angajat NUMBER(5),

data\_diagnostic date default (sysdate),

boala VARCHAR(20)

);

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('frac', '1', 1, to\_date('10-06-2020', 'DD-MM-YYYY'), 'Fractura');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('ir', '2', 8, to\_date('11-08-2020', 'DD-MM-YYYY'), 'Insuficienta Renala');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('mio', '3', 6, to\_date('14-03-21', 'DD-MM-YYYY'), 'miopie');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('ast', '4', 6, to\_date('19-11-19', 'DD-MM-YYYY'), 'astigmatism');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('derm', '5', 10, NULL, 'dermatita');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('traum', '6', 1, NULL, 'traumatim');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('apend', '7', 3, NULL, 'apendicita acuta');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('inf', '8', 1, to\_date('03-03-2021', 'DD-MM-YYYY'), 'infectie');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('sept', '9', 1, NULL, 'septicemie');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('lit', '10', 8, to\_date('01/09/2021', 'DD-MM-YYYY'), 'litiaza renala');

INSERT INTO DIAGNOSTIC (ID\_DIAGNOSTIC, ID\_PAC, ID\_ANGAJAT, DATA\_DIAGNOSTIC, BOALA)

VALUES ('infa', '11', 13, to\_date('05-07-2021', 'DD-MM-YYYY'), 'infarct');

--LEG\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE Diagnostic

ADD CONSTRAINT leg\_angajat\_diagnostic

FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT (id\_angajat) ON DELETE SET NULL;

--PACIENT-----------------------------------------------------------------

CREATE TABLE PACIENT (

id\_pac number(10) PRIMARY KEY,

id\_diagnostic VARCHAR(10),

--id\_angajat NUMBER(5),

id\_sala VARCHAR(5),

nume VARCHAR(10),

prenume VARCHAR(20),

oras VARCHAR(10),

adresa VARCHAR(15)

);

--SECVENTA~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

CREATE SEQUENCE secventa

start with 1

increment by 1

minvalue 1

maxvalue 100000

nocycle;

--LEG\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE Pacient

ADD CONSTRAINT leg\_diagnostic\_pacient

FOREIGN KEY (id\_diagnostic) REFERENCES Diagnostic (id\_diagnostic) ON DELETE SET NULL;

ALTER TABLE PACIENT

ADD CONSTRAINT leg\_sala\_pacient

FOREIGN KEY (id\_sala) REFERENCES SALA (id\_sala) ON DELETE SET NULL;

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'frac', 'U1', 'Ilioiu', 'Ana', 'Bucuresti', 'Str. Astra');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'traum', 'U1', 'Sambure', 'Costin', 'Cluj', 'Str. Jiului');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'mio', 'Oft1', 'Capsuna', 'Marian', 'Bucuresti', 'Str. Macesului');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'ir', 'Nef1', 'Marin', 'Andreea', 'Giurgiu', 'Str. Turcilor');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'ast', 'Oft1', 'Hodosan', 'Toma', 'Bucuresti', 'Str. Pierzaniei');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'ir', 'Nef1', 'Neculce', 'Elena', 'Bucuresti', 'Str. Crinului');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'traum', 'U1', 'Tomsa', 'Popa', 'Bucuresti', 'Str. Ploii');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'frac', 'U1', 'Nedelciu', 'Adriana', 'Iasi', 'Str. Castanelor');

INSERT INTO PACIENT (ID\_PAC, ID\_DIAGNOSTIC, ID\_SALA, NUME, PRENUME, ORAS, ADRESA)

VALUES (secventa.nextval, 'frac', 'U2', 'Dinu', 'Cosmin', 'Bucuresti', 'Str. Iasomiei');

--TRATAMENT-------------------------------------------------------------------

CREATE TABLE TRATAMENT (

plan\_tratament NUMBER(4) PRIMARY KEY,

medicament VARCHAR(20),

administrare VARCHAR(20),

durata VARCHAR(20)

);

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (1, 'Ghips', 'imediata', '4 saptamani');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (2, 'dializa', NULL, 'nedeterminata');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (3, 'prescriptie ochelari', NULL, 'nedeterminata');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (4, 'prescriptie ochelari', NULL, 'nedeterminata');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (5, 'antibiotic', 'zilnica', '2 saptamani');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (6, 'Analgezic', 'zilnica', '1 saptamana');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (7,'operatie','imediata','2 zile');

INSERT INtO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (8,'antibiotice','zilnica','3 saptamani');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (9,'antibiotice','zilnica','3 saptamani');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (10,'operatie, tratament','imediata','1 saptamana');

INSERT INTO TRATAMENT (PLAN\_TRATAMENT, MEDICAMENT, ADMINISTRARE, DURATA)

VALUES (11,'anticoagulante','imediata','5 zile');

--IMPLICA---------------------------------------------------------------------

CREATE TABLE IMPLICA (

id\_diagnostic VARCHAR(10),

plan\_tratament NUMBER(4),

CONSTRAINT PK\_implica PRIMARY KEY (id\_diagnostic, plan\_tratament)

);

--LEG\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE IMPLICA

ADD CONSTRAINT leg\_diagnostic\_implica

FOREIGN KEY (id\_diagnostic) REFERENCES Diagnostic (id\_diagnostic) ON DELETE SET NULL;

--LEG\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ALTER TABLE IMPLICA

ADD CONSTRAINT leg\_tratament\_implica

FOREIGN KEY (plan\_tratament) REFERENCES TRATAMENT (plan\_tratament) ON DELETE SET NULL;

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('frac', 1);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('ir', 2);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('mio', 3);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('ast', 4);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('derm', 5);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('traum', 6);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('apend', 7);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('inf', 8);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('sept', 9);

INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

VALUES ('lit', 10);

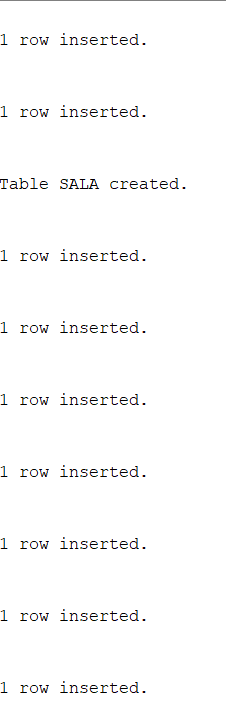
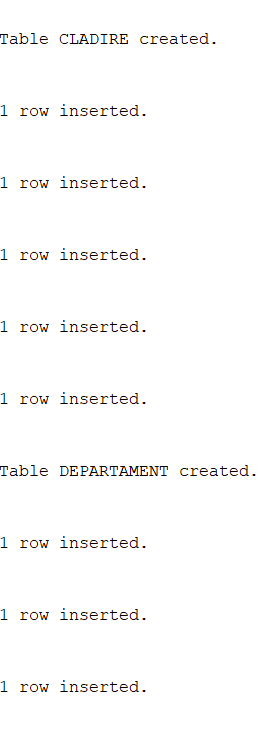
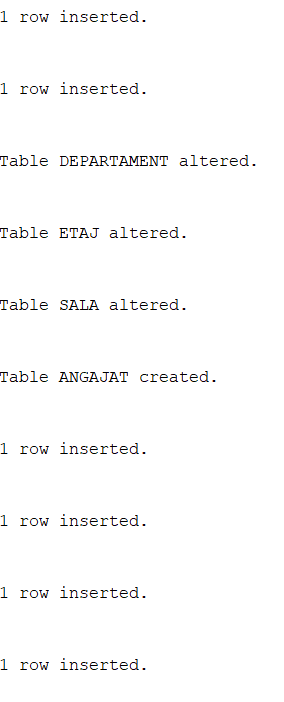
INSERT INTO IMPLICA (ID\_DIAGNOSTIC, PLAN\_TRATAMENT)

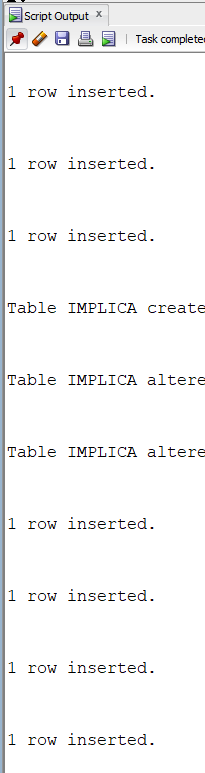
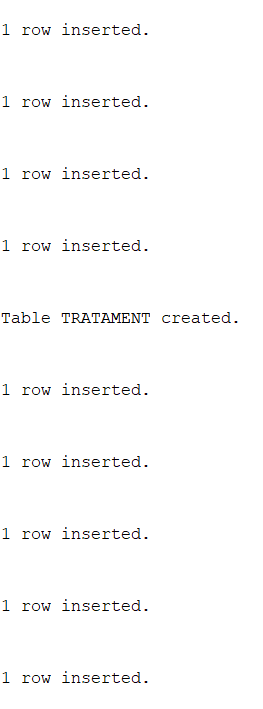
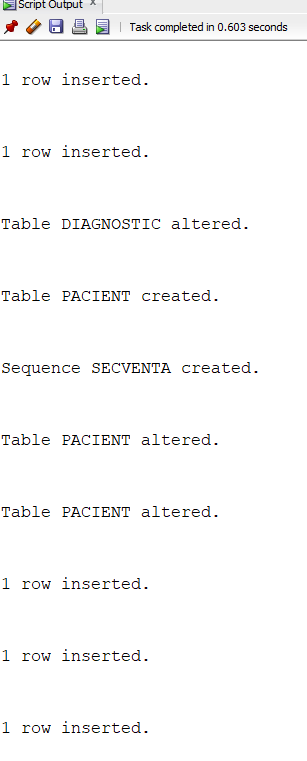
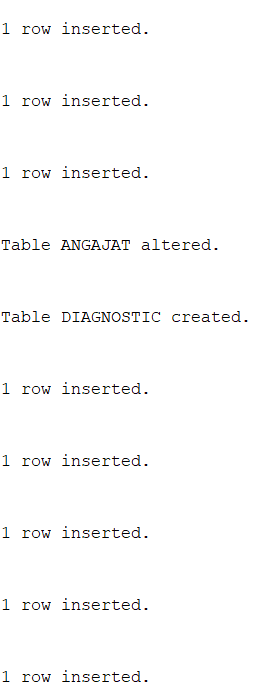
VALUES ('inf', 11);

ALTER TABLE DEPARTAMENT

DROP COLUMN id\_angajat;

## --4&5.1Print-Screen-uri ce demonstrează rularea codului





# --6 Problema rezolvata folosind un subprogram care utilizează 2 tipuri de colecții studiate

Cerinta:

Rezolvați urmatoarea problemă folosind un subprogram stocat:

Rețineți intr-o colecție (de tip vector) id-urile pacientilor ce urmeaza sa fie externati si intr-o alta

colectie (de tip tabel imbricat), numele pacienților.

Stergeti din tabelul pacient pacientii ce au codurile stocate in colectiea de tip vector iar apoi

afisati numele tuturor pacientilor folosind colectia corespunzatoare.

CREATE OR REPLACE PROCEDURE c6 IS

FARA\_PACIENTI EXCEPTION;

TYPE tip\_cod IS VARRAY(10) OF pacient.id\_pac%type;

coduri\_pacienti tip\_cod := tip\_cod(9,10);

TYPE tip\_nume IS TABLE OF pacient.nume%type;

nume\_pacienti tip\_nume := tip\_nume();

BEGIN

FORALL i IN coduri\_pacienti.FIRST..coduri\_pacienti.LAST

DELETE FROM pacient

WHERE id\_pac = coduri\_pacienti(i);

Select nume bulk collect into nume\_pacienti

from pacient;

IF nume\_pacienti.count>0 THEN

FOR i IN 1..nume\_pacienti.count LOOP

DBMS\_OUTPUT.PUT\_LINE (nume\_pacienti(i));

END LOOP;

ELSE

RAISE FARA\_PACIENTI;

END IF;

EXCEPTION

WHEN FARA\_PACIENTI THEN

DBMS\_OUTPUT.PUT\_LINE ('Nu exita pacienti in baza de date!');

END c6;

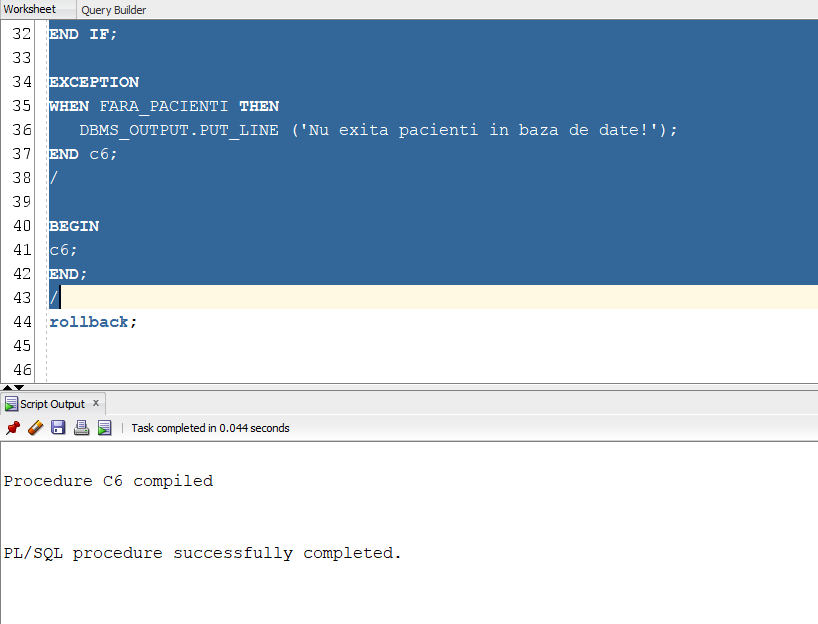
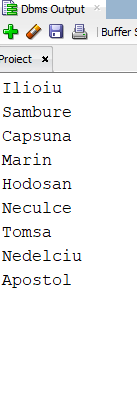
/

BEGIN

c6;

END;

## --6.1 Print-screen ce dovedeste rularea codului in Oracle



# --7 Problema rezolvata folosind un subprogram care utilizează un tip de cursor

Cerinta:

Rezolvati urmatoarea problema folosind un subprogram stocat:

Pentru fiecare job in parte, obtineti lista angajatilor (nume, prenume) care lucreaza pe jobul respectiv, sau mesajul nu exita angajati, daca nu lucreaza nimeni pe acel job.

Folostiti un ciclu cursor!

CREATE OR REPLACE PROCEDURE c7 IS

CURSOR j IS SELECT id\_job, titlu\_job FROM job;

CURSOR e(job\_curent VARCHAR2) IS SELECT nume, prenume FROM angajat WHERE id\_job = job\_curent;

v\_nr NUMBER(2);

BEGIN

FOR i in j LOOP

EXIT WHEN j%notfound;

dbms\_output.new\_line;

DBMS\_OUTPUT.PUT\_LINE(i.titlu\_job);

v\_nr:=0;

for k in e(i.id\_job) LOOP

DBMS\_OUTPUT.PUT\_LINE('Ang ' || k.nume || ' ' || k.prenume);

v\_nr:=v\_nr+1;

END LOOP;

IF v\_nr = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Fara ang');

END IF;

END LOOP;

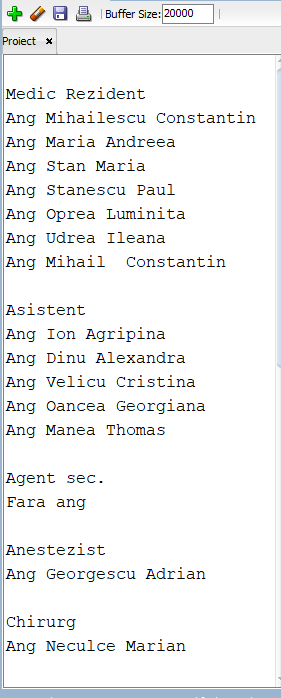
END c7;

/

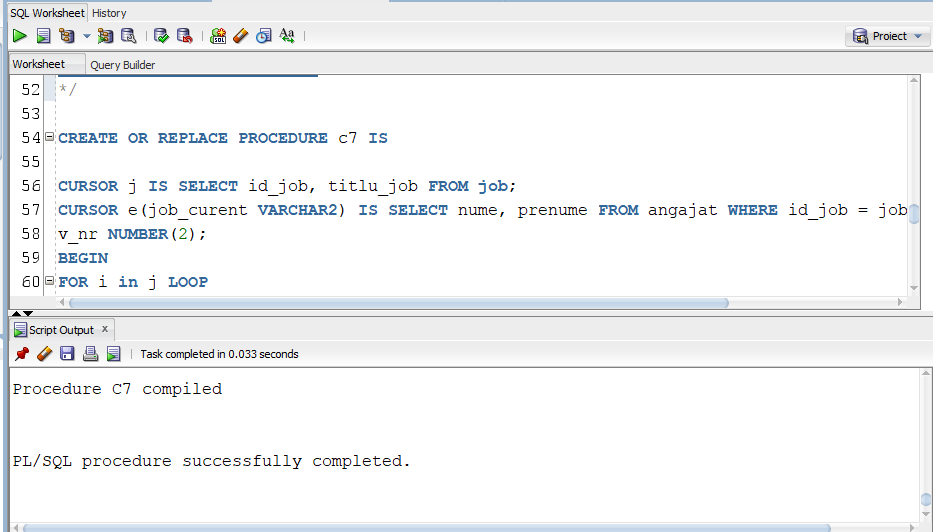
BEGIN

c7;

END;



## --7.1 Print-screen ce dovedeste rularea codului in Oracle



# --8 Functie ce utilizează într-o singură comanda SQL 3 dintre tabelele din baza de date

Cerința:

Scrieti o functie care sa returneze id-ul pacientului care a fost tratat de un angajat al carui id este dat;

Tratati toate exceptiile care pot aparea!

CREATE OR REPLACE FUNCTION c8 (v\_id\_ang angajat.id\_angajat%type DEFAULT 13)

RETURN pacient.id\_pac%type

IS

id pacient.id\_pac%type;

ANG\_INEXISTENT EXCEPTION;

TYPE tip\_id IS TABLE OF angajat.id\_angajat%type;

lista\_id\_ang tip\_id := tip\_id();

BEGIN

select id\_angajat bulk collect into lista\_id\_ang from angajat;

IF lista\_id\_ang.EXISTS(v\_id\_ang) THEN

SELECT p.id\_pac into id

FROM pacient p

JOIN diagnostic d ON d.id\_pac=p.id\_pac

JOIN angajat a ON a.id\_angajat=d.id\_angajat

WHERE a.id\_angajat=v\_id\_ang;

RETURN id;

ELSE

RAISE ANG\_INEXISTENT;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista pacient tratat de doctorul dat.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Doctorul dat trateaza mai multi pacienti!');

WHEN ANG\_INEXISTENT THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul cu codul dat nu exista!!');

END c8;

/

DECLARE

v\_rezultat pacient.id\_pac%type;

BEGIN

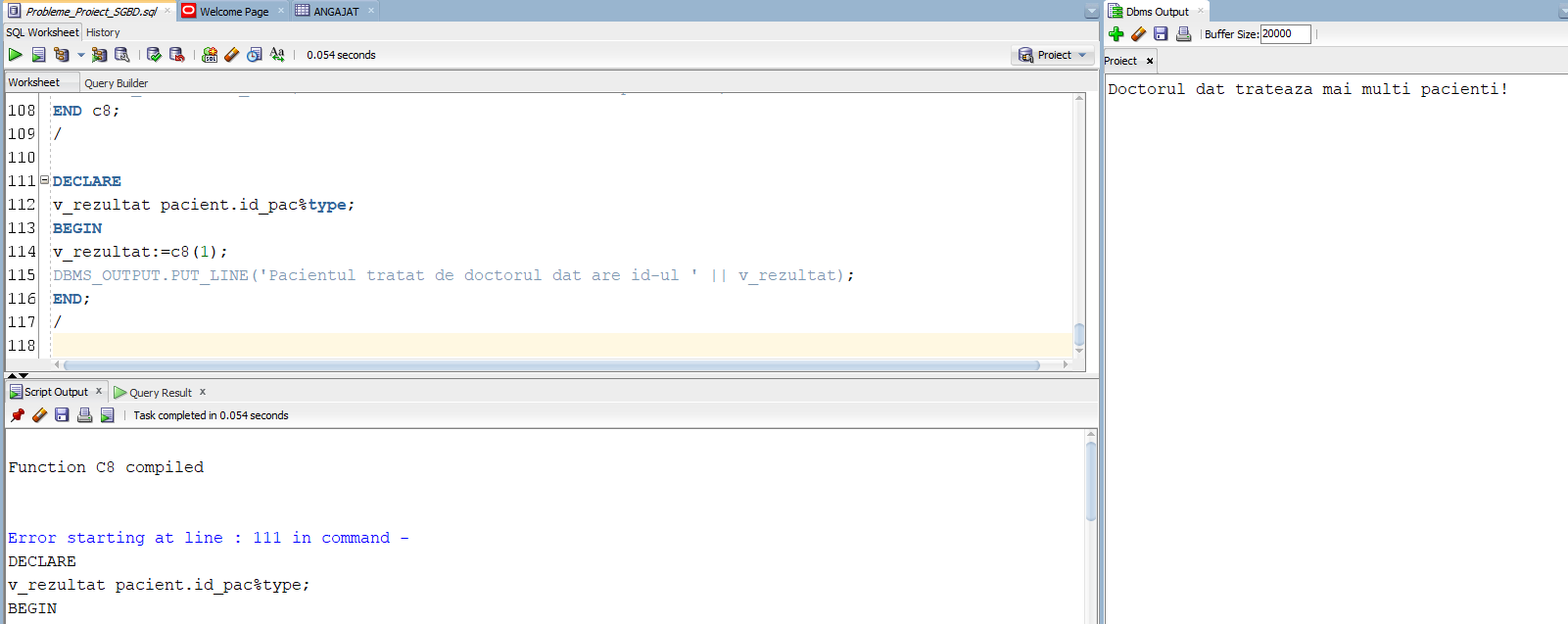
v\_rezultat:=c8(24);

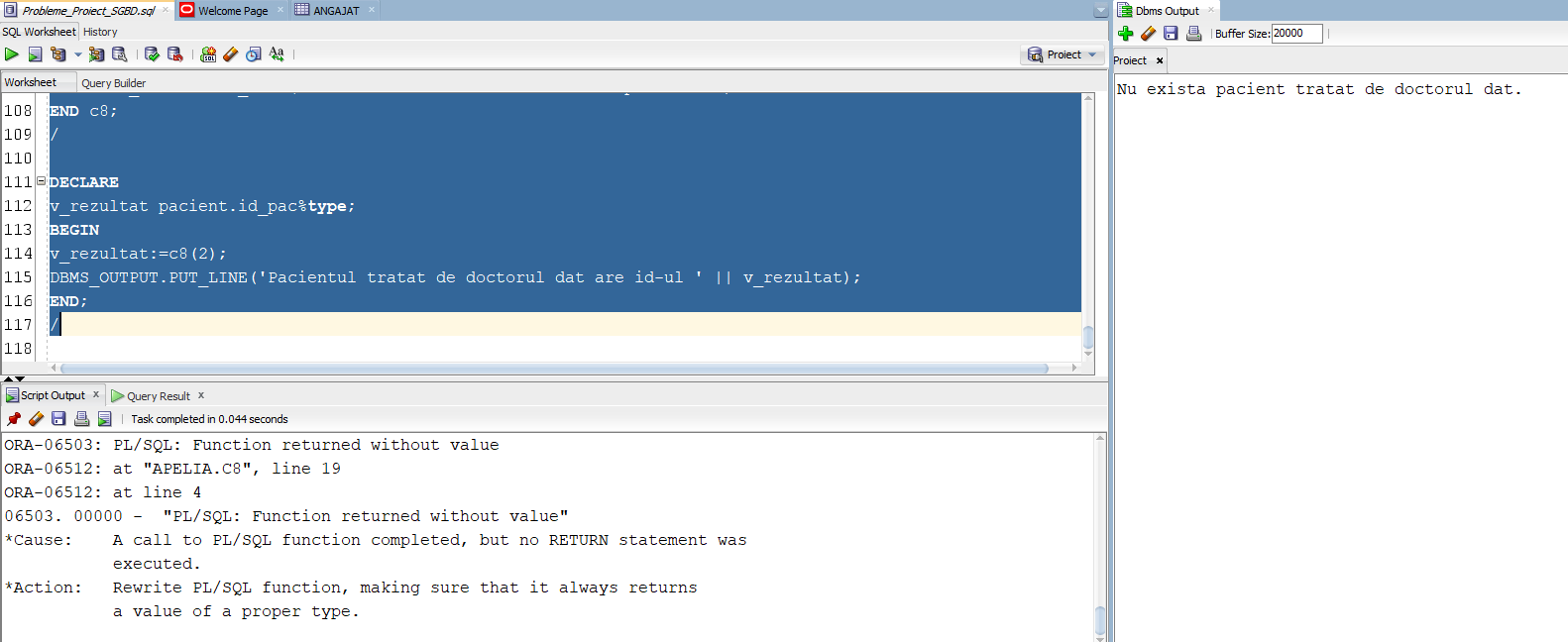
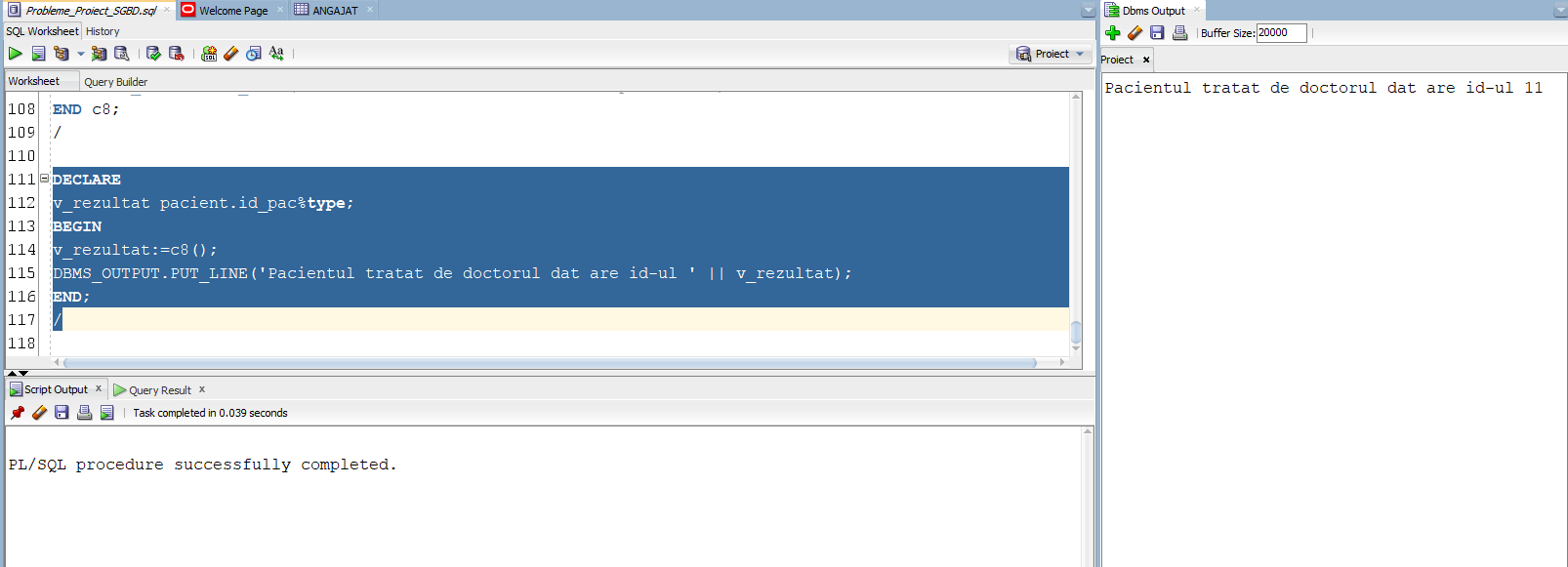
DBMS\_OUTPUT.PUT\_LINE('Pacientul tratat de doctorul dat are id-ul ' || v\_rezultat);

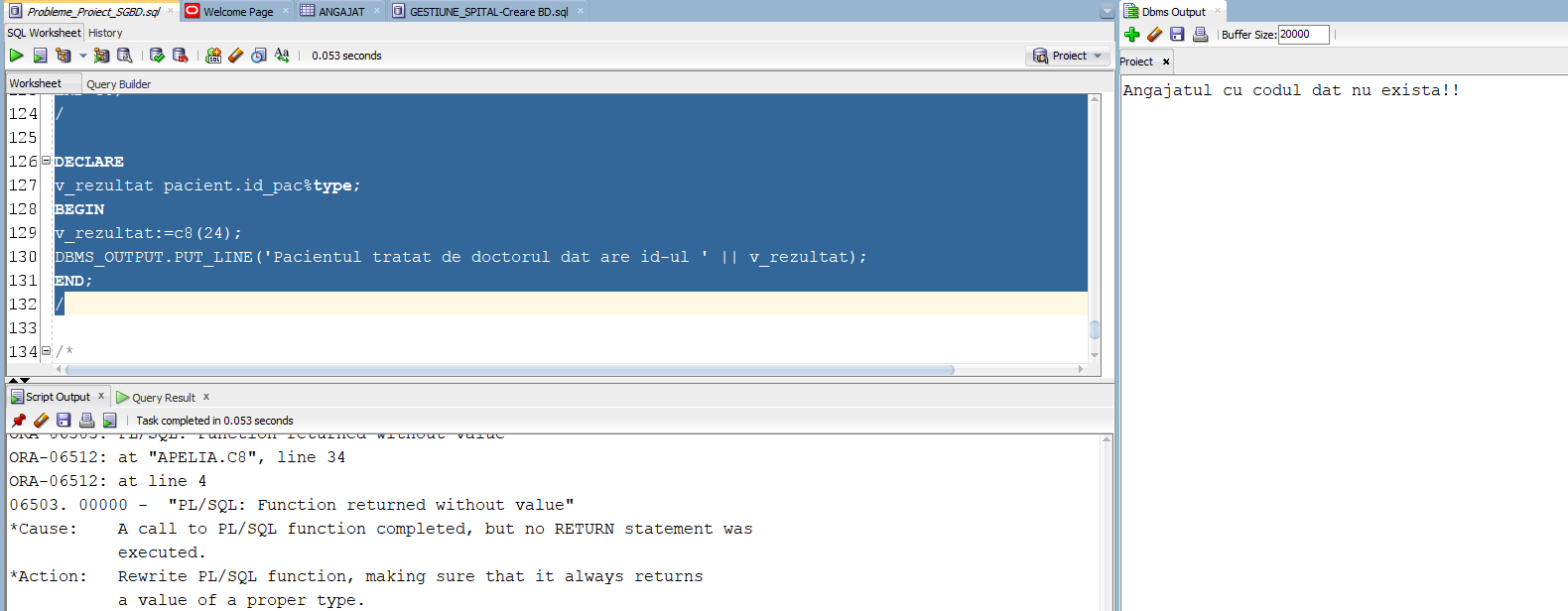
END;

/

## --8.1 Print-screen ce dovedeste rularea codului in Oracle







# --9 Procedura ce utilizează într-o singură comanda SQL 5 dintre tabelele din baza de date

Cerinta:

Scrieti o procedura care sa afiseze cu ajutorul pachetului DBMS Output id-ul etajului la nivelul caruia

are loc un anumit tratament (plan de tratament).

Tratati exceptiile ce pot aparea!

CREATE OR REPLACE PROCEDURE c9 (v\_plan\_trat tratament.plan\_tratament%TYPE) IS

v\_id\_etaj etaj.id\_etaj%type;

TYPE tip\_tratament IS TABLE OF tratament.plan\_tratament%type;

lista\_planuri\_trat tip\_tratament := tip\_tratament();

Plan\_inexistent EXCEPTION;

BEGIN

SELECT plan\_tratament bulk collect INTO lista\_planuri\_trat from implica;

IF lista\_planuri\_trat.EXISTS(v\_plan\_trat) THEN

SELECT e.id\_etaj into v\_id\_etaj

FROM etaj e

JOIN sala s ON s.id\_etaj=e.id\_etaj

JOIN pacient p on s.id\_sala=p.id\_sala

JOIN diagnostic d ON d.id\_pac=p.id\_pac

JOIN implica i ON i.id\_diagnostic=d.id\_diagnostic

WHERE i.plan\_tratament=v\_plan\_trat;

DBMS\_OUTPUT.PUT\_LINE(v\_id\_etaj);

ELSE

RAISE Plan\_inexistent;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista un etaj pentru tratamentul dat;');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Tratamentul se poate realiza in cadrul a mai multe etaje');

WHEN Plan\_inexistent THEN

DBMS\_OUTPUT.PUT\_LINE('Planul de tratament dat nu exista!!');

END c9;

/

--OBSERVATIE : Procedura nu va intra pe exceptia too many rows sau no data found!

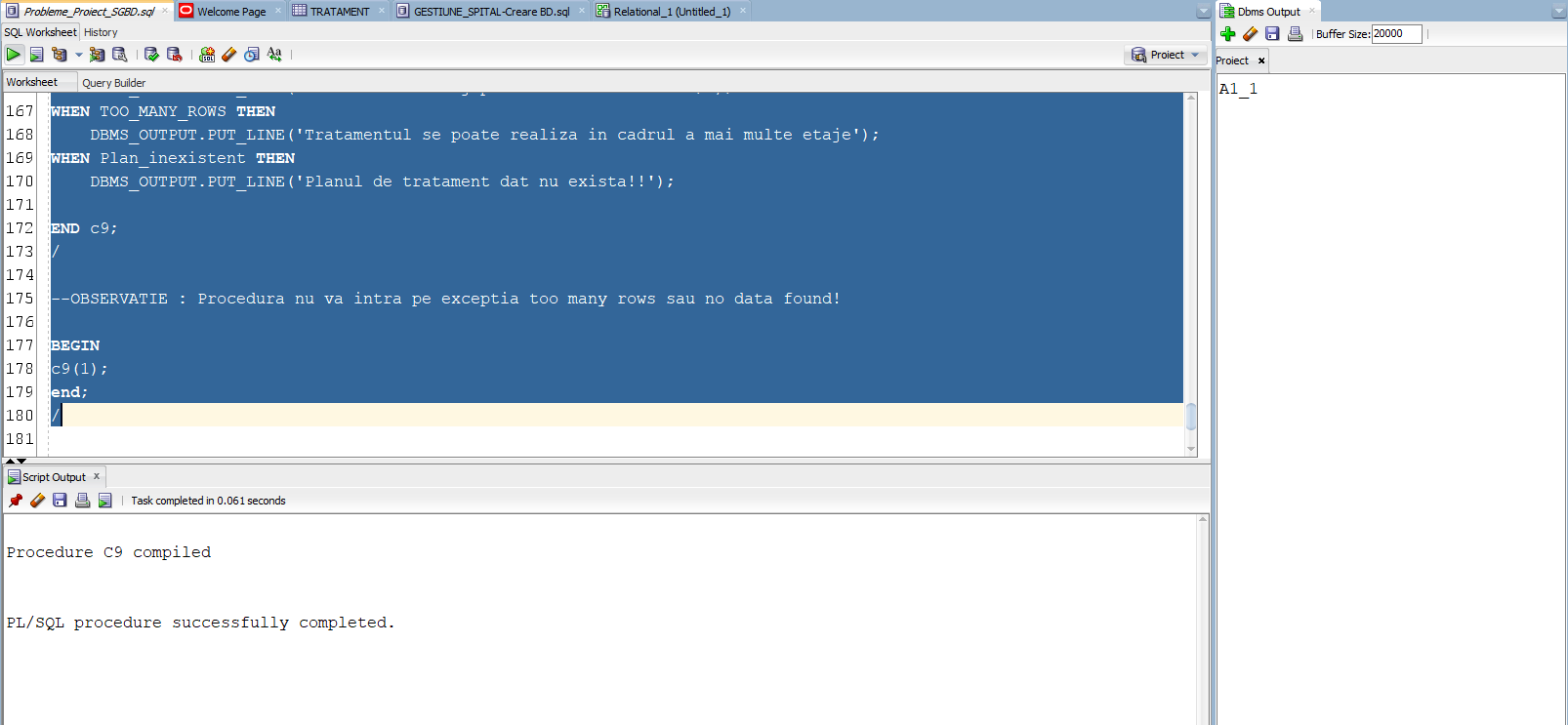
BEGIN

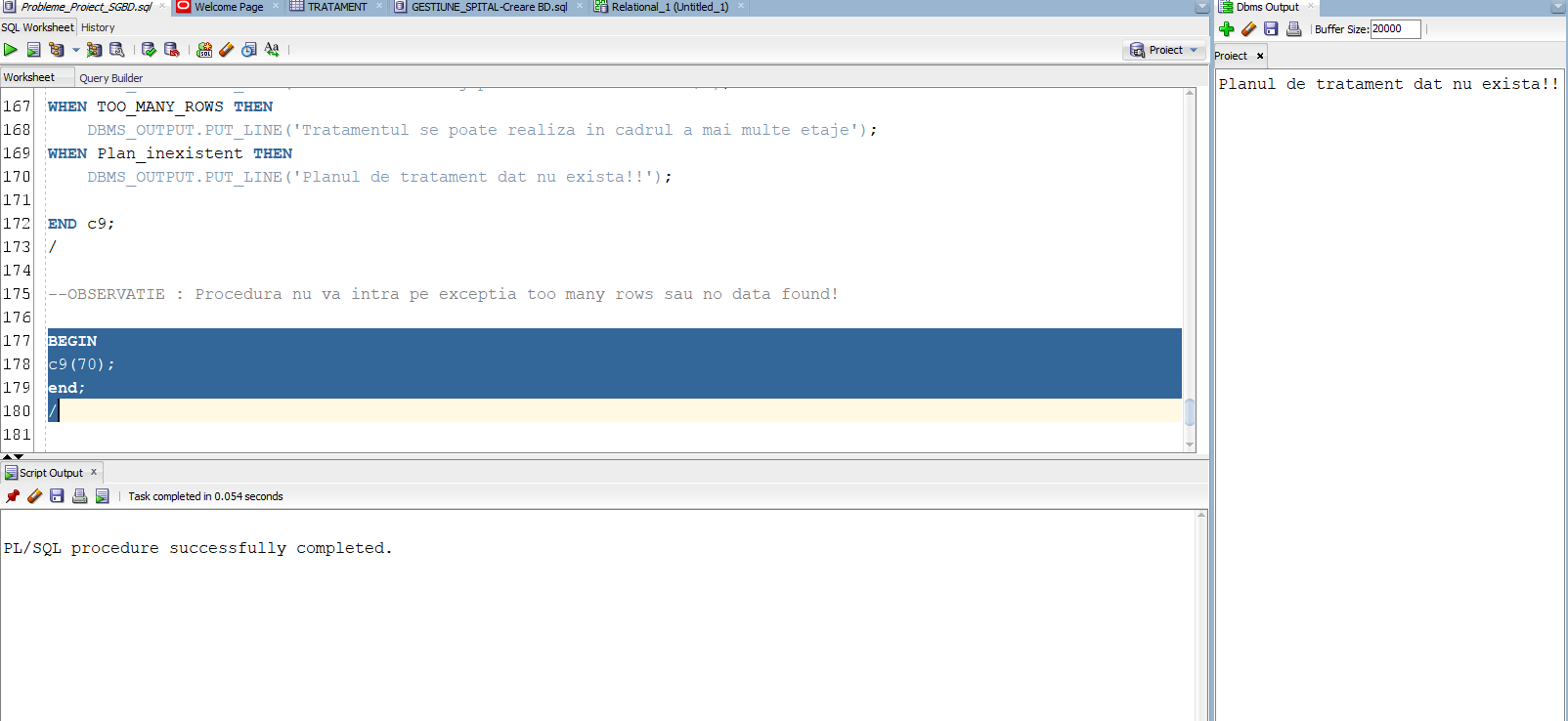
c9(1);

end;

/

## --9.1 Print-screen ce dovedeste rularea codului in Oracle





# --10 Trigger LMD la nivel de comandă

Definiti un trigger care sa permita externarea pacientilor (Delete in cadrul tabelului pacient)

doar in intervalul orar 8:00-21:00, de luni pana vineri.

CREATE OR REPLACE TRIGGER c10

BEFORE DELETE ON pacient

BEGIN IF (TO\_CHAR(SYSDATE,'D') = 1) OR (TO\_CHAR(SYSDATE,'D') = 7) OR (TO\_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 21)

THEN RAISE\_APPLICATION\_ERROR(-20001,'Pacientul nu poate fi externat inca!');

END IF;

END;

/

--Apelare declansator

BEGIN

DELETE FROM pacient

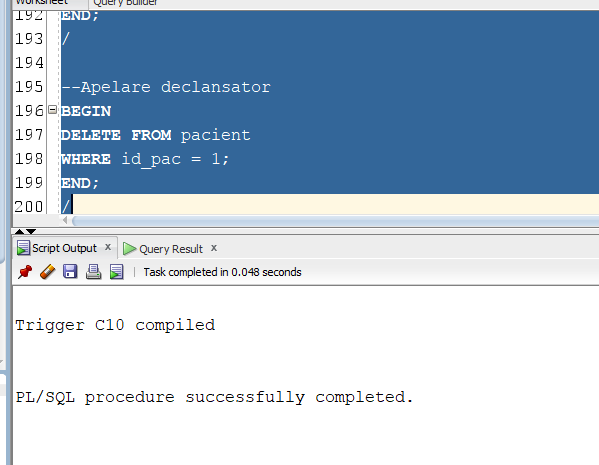
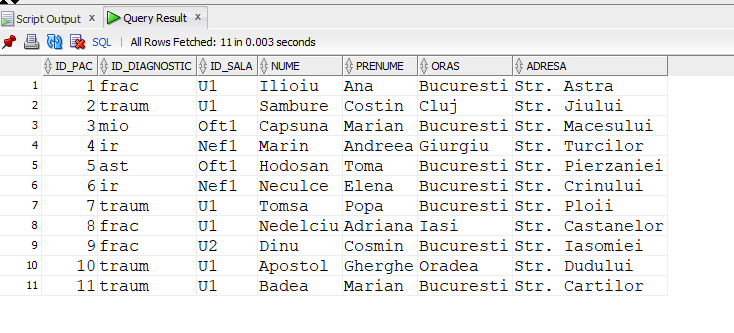
WHERE id\_pac = 1;

END;

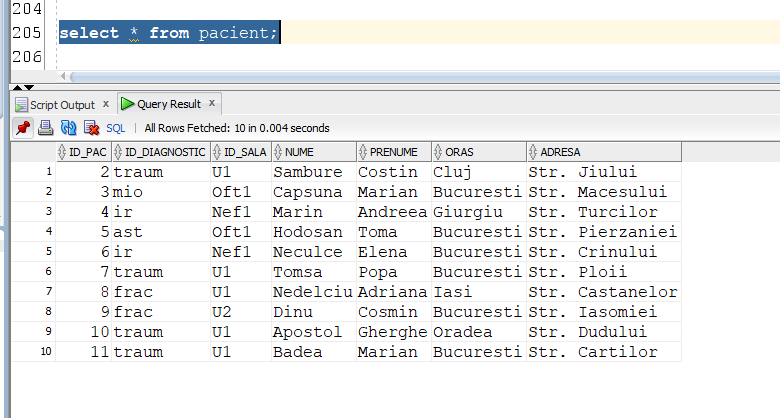
/

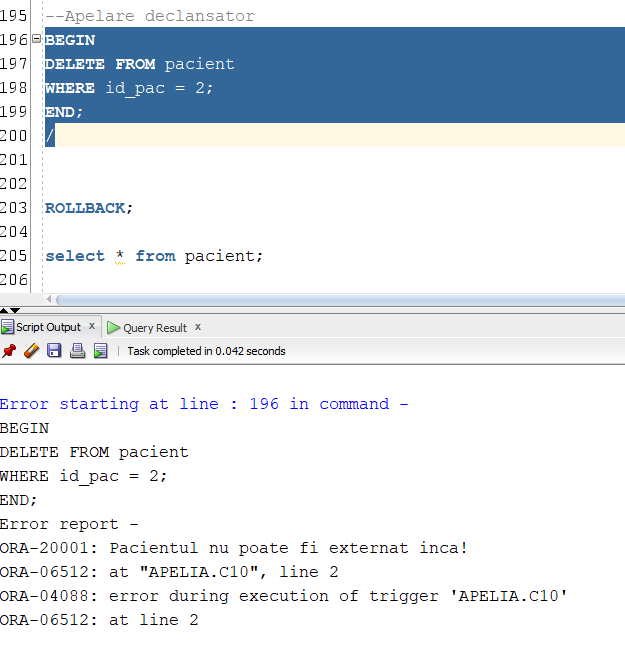
## --10.1 Print-screen ce dovedeste rularea codului in Oracle

1) Stergere reusita:

Datele din tabelul pacient inainte de stergere:

Datele din tabelul pacient dupa stergere:



In afara intervalului specificat:

# --11 Trigger LMD la nivel de linie

Definiti un trigger pentru tabelul ETAJ care sa nu permita modificarea numarului de sali

din cadrul acelui etaj.

CREATE OR REPLACE TRIGGER c11

BEFORE UPDATE OF nr\_sali ON etaj

FOR EACH ROW

BEGIN

IF (:NEW.nr\_sali < :OLD.nr\_sali)

THEN RAISE\_APPLICATION\_ERROR(-20002,'Numarul de sali nu poate fi modificat!!');

END IF;

END;

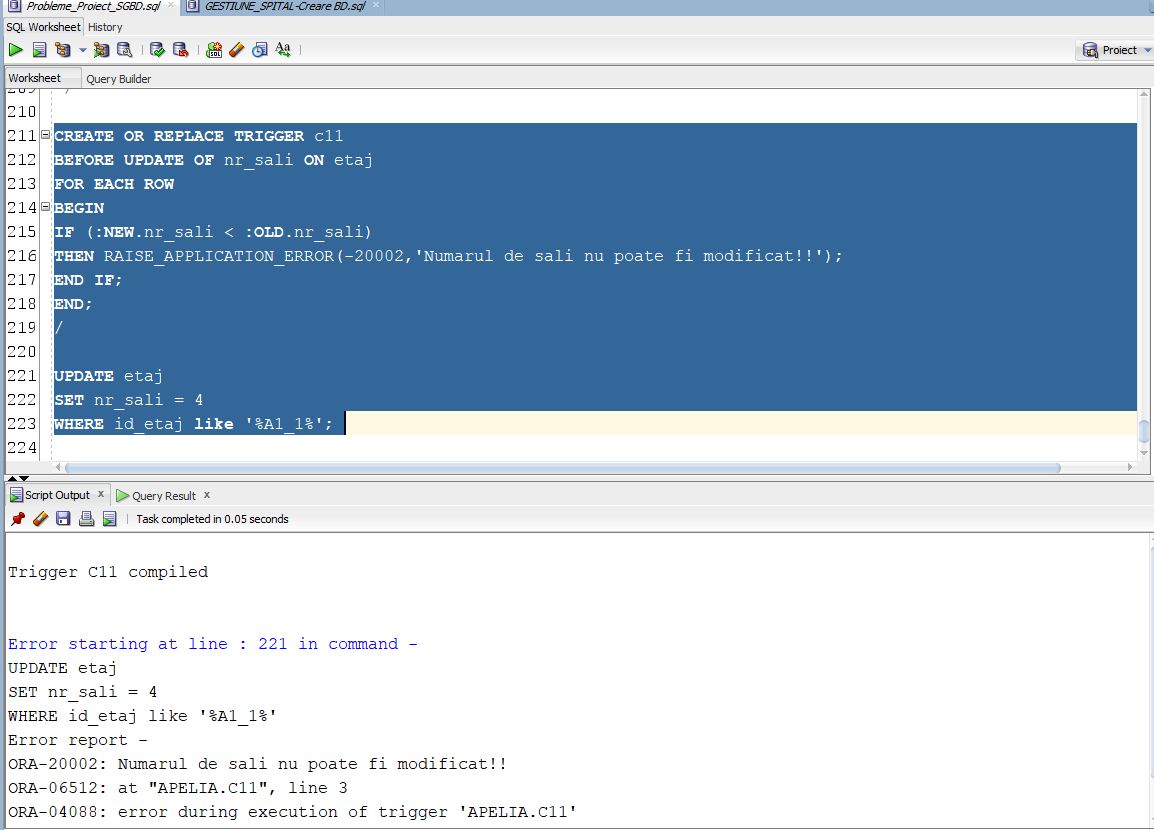
/

UPDATE etaj

SET nr\_sali = 4

WHERE id\_etaj like '%A1\_1%';

## --11.1 Print-screen ce dovedeste rularea codului in Oracle



# --12 Trigger de tip LDD

Definiti un trigger de tip DDL care sa nu permita stergerea tabelelor din baza de date.

Acest trigger va preveni pierderea accidentala a datelor.

CREATE OR REPLACE TRIGGER c12

BEFORE DROP ON DATABASE

BEGIN

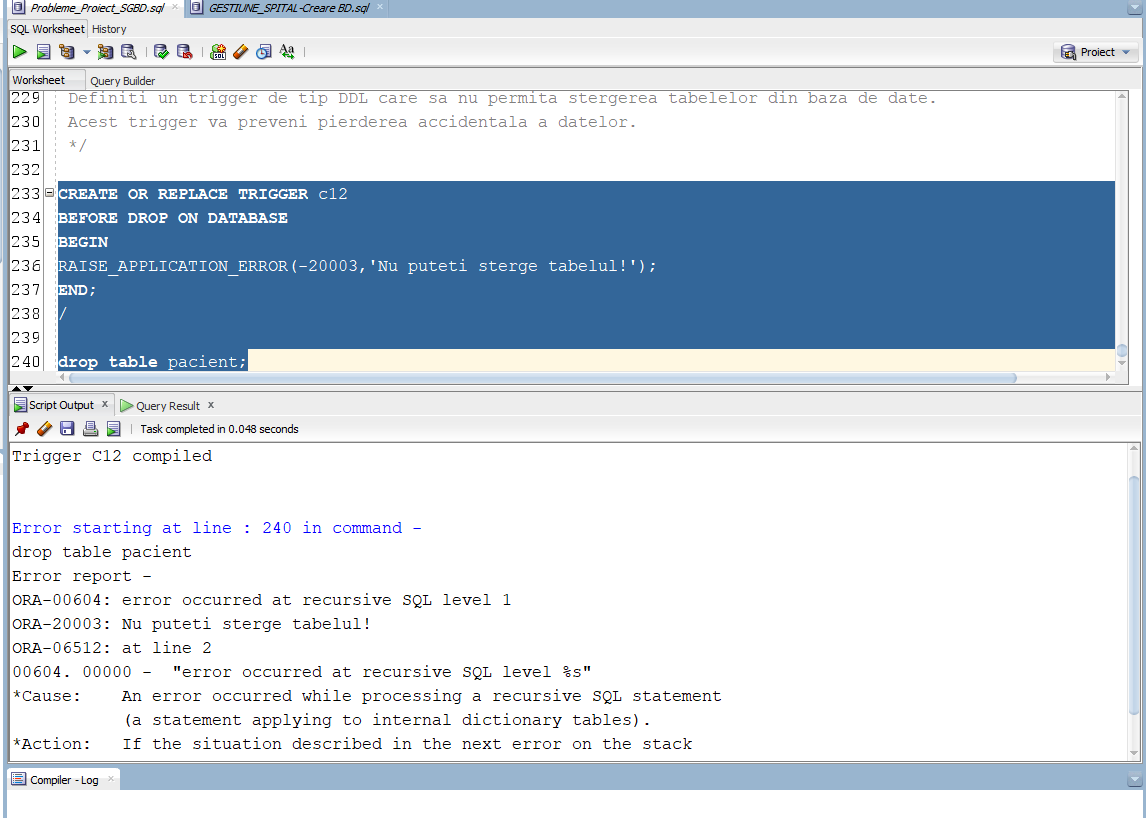
RAISE\_APPLICATION\_ERROR(-20003, 'Nu puteti sterge tabelul!');

END;

/

drop table pacient;

## --12.1 Print-screen ce dovedeste rularea codului in Oracle



# --13 Pachet care contine obiectele definite

CREATE OR REPLACE PACKAGE c13 IS

PROCEDURE c6;

PROCEDURE c7;

FUNCTION c8 (v\_id\_ang angajat.id\_angajat%type DEFAULT 13) RETURN pacient.id\_pac%type;

PROCEDURE c9 (v\_plan\_trat tratament.plan\_tratament%TYPE);

END c13;

/

CREATE OR REPLACE PACKAGE BODY c13 IS

--------------------------------------------

PROCEDURE c6 IS

FARA\_PACIENTI EXCEPTION;

TYPE tip\_cod IS VARRAY(10) OF pacient.id\_pac%type;

coduri\_pacienti tip\_cod := tip\_cod(9,10);

TYPE tip\_nume IS TABLE OF pacient.nume%type;

nume\_pacienti tip\_nume := tip\_nume();

BEGIN

FORALL i IN coduri\_pacienti.FIRST..coduri\_pacienti.LAST

DELETE FROM pacient

WHERE id\_pac = coduri\_pacienti(i);

Select nume bulk collect into nume\_pacienti

from pacient;

IF nume\_pacienti.count>0 THEN

FOR i IN 1..nume\_pacienti.count LOOP

DBMS\_OUTPUT.PUT\_LINE (nume\_pacienti(i));

END LOOP;

ELSE

RAISE FARA\_PACIENTI;

END IF;

EXCEPTION

WHEN FARA\_PACIENTI THEN

DBMS\_OUTPUT.PUT\_LINE ('Nu exita pacienti in baza de date!');

END c6;

-------------------------------------------------------------------

PROCEDURE c7 IS

CURSOR j IS SELECT id\_job, titlu\_job FROM job;

CURSOR e(job\_curent VARCHAR2) IS SELECT nume, prenume FROM angajat WHERE id\_job = job\_curent;

v\_nr NUMBER(2);

BEGIN

FOR i in j LOOP

EXIT WHEN j%notfound;

dbms\_output.new\_line;

DBMS\_OUTPUT.PUT\_LINE(i.titlu\_job);

v\_nr:=0;

for k in e(i.id\_job) LOOP

DBMS\_OUTPUT.PUT\_LINE('Ang ' || k.nume || ' ' || k.prenume);

v\_nr:=v\_nr+1;

END LOOP;

IF v\_nr = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Fara ang');

END IF;

END LOOP;

END c7;

---------------------------------------------------------------------

FUNCTION c8 (v\_id\_ang angajat.id\_angajat%type DEFAULT 13)

RETURN pacient.id\_pac%type

IS

id pacient.id\_pac%type;

ANG\_INEXISTENT EXCEPTION;

TYPE tip\_id IS TABLE OF angajat.id\_angajat%type;

lista\_id\_ang tip\_id := tip\_id();

BEGIN

select id\_angajat bulk collect into lista\_id\_ang from angajat;

IF lista\_id\_ang.EXISTS(v\_id\_ang) THEN

SELECT p.id\_pac into id

FROM pacient p

JOIN diagnostic d ON d.id\_pac=p.id\_pac

JOIN angajat a ON a.id\_angajat=d.id\_angajat

WHERE a.id\_angajat=v\_id\_ang;

RETURN id;

ELSE

RAISE ANG\_INEXISTENT;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista pacient tratat de doctorul dat.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Doctorul dat trateaza mai multi pacienti!');

WHEN ANG\_INEXISTENT THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul cu codul dat nu exista!!');

END c8;

-----------------------------------------------------------------------------------

PROCEDURE c9 (v\_plan\_trat tratament.plan\_tratament%TYPE) IS

v\_id\_etaj etaj.id\_etaj%type;

TYPE tip\_tratament IS TABLE OF tratament.plan\_tratament%type;

lista\_planuri\_trat tip\_tratament := tip\_tratament();

Plan\_inexistent EXCEPTION;

BEGIN

SELECT plan\_tratament BULK COLLECT INTO lista\_planuri\_trat from implica;

IF lista\_planuri\_trat.EXISTS(v\_plan\_trat) THEN

SELECT e.id\_etaj into v\_id\_etaj

FROM etaj e

JOIN sala s ON s.id\_etaj=e.id\_etaj

JOIN pacient p on s.id\_sala=p.id\_sala

JOIN diagnostic d ON d.id\_pac=p.id\_pac

JOIN implica i ON i.id\_diagnostic=d.id\_diagnostic

WHERE i.plan\_tratament=v\_plan\_trat;

DBMS\_OUTPUT.PUT\_LINE(v\_id\_etaj);

ELSE

RAISE Plan\_inexistent;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista un etaj pentru tratamentul dat;');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Tratamentul se poate realiza in cadrul a mai multe etaje');

WHEN Plan\_inexistent THEN

DBMS\_OUTPUT.PUT\_LINE('Planul de tratament dat nu exista!!');

END c9;

------------------------------------------------------------------------

END c13;

/

## --13.1 10.1 Print-screen ce dovedeste rularea codului in Oracle

