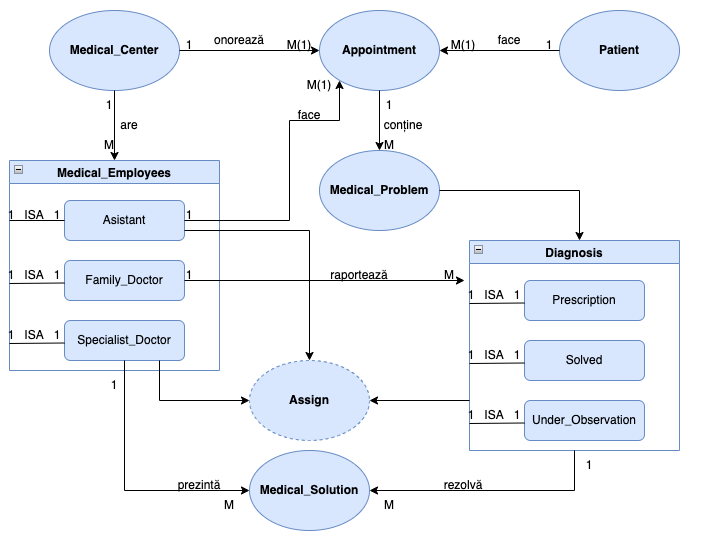
**PROIECT SISTEME DE GESTIUNE A BAZELOR DE DATE**

**GESTIUNEA ACTIVITĂȚILOR UNEI CLINICI MEDICALE**

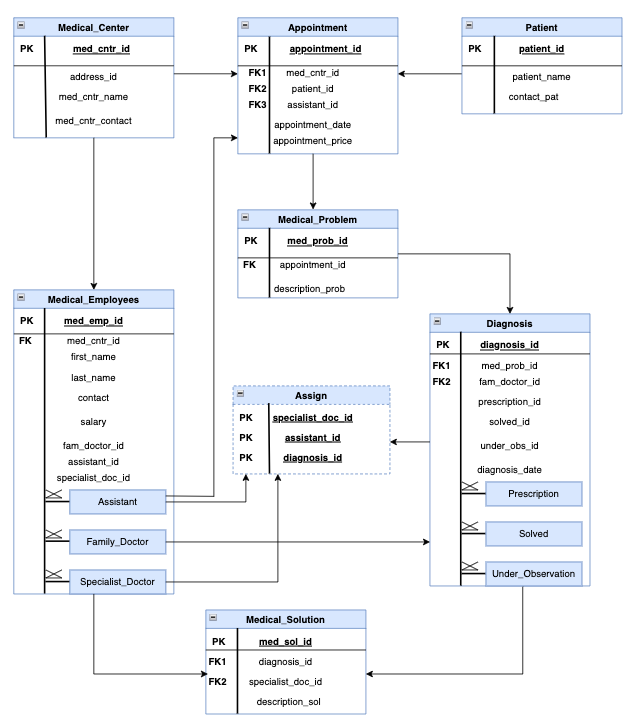
1. **Prezentarea bazei de date și utilitatea ei**

Pentru acest proiect am ales să realizez gestionarea unei clinici medicale. Clinica medicală are angajați de tipul: asistent, doctor de familie și doctor specialist. Pacienții pot face programări pentru a fi consultați în legătură cu anumite probleme de sănătate. Fiecare programare este făcută de către un asistent pentru o consultație la medicul de familie în urma căreia se va afla ce nevoi medicale are pacientul și starea sa de sănătate. Doctorul de familie raportează problema de sănătate a pacientului, iar asistentul îi asignează o programare către un doctor specialist. O problemă de sănătate poate fi declarată rezolvată, i se poate asigna o prescripție medicală sau poate fi pusă sub observație.

1. **Diagrama entitate-relație (ERD)**

****

1. **Diagrama conceptuală**

****

1. **Implementarea diagramei conceptuale (definirea tabelelor)**

-- Crearea tabelului MEDICAL\_CENTER

CREATE TABLE medical\_center

(

med\_cntr\_id VARCHAR2(6) NOT NULL,

address\_id VARCHAR2(25) NOT NULL,

med\_cntr\_name VARCHAR2(30) NOT NULL,

med\_cntr\_contact VARCHAR2(12) NOT NULL

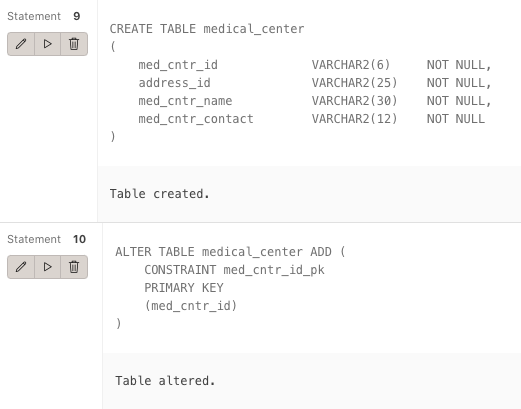
);

ALTER TABLE medical\_center ADD (

CONSTRAINT med\_cntr\_id\_pk

PRIMARY KEY

(med\_cntr\_id) );



-- Crearea tabelului PATIENT

CREATE TABLE patient

(

patient\_id VARCHAR2(6) NOT NULL,

patient\_name VARCHAR2(30) NOT NULL,

contact\_pat VARCHAR2(12) NOT NULL

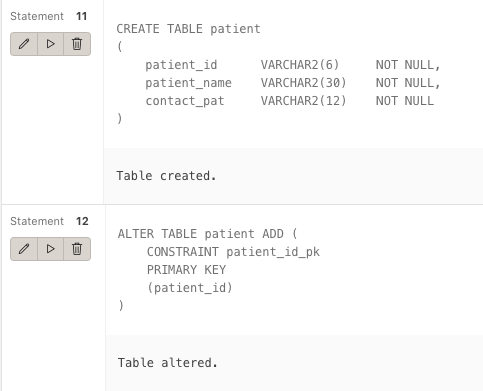
);

ALTER TABLE patient ADD (

CONSTRAINT patient\_id\_pk

PRIMARY KEY

(patient\_id) );



-- Crearea tabelului MEDICAL\_EMPLOYEES

CREATE TABLE medical\_employees

(

med\_emp\_id VARCHAR2(6) NOT NULL,

med\_cntr\_id VARCHAR2(6) NOT NULL,

first\_name VARCHAR2(30) NOT NULL,

last\_name VARCHAR2(30) NOT NULL,

contact VARCHAR2(12) NOT NULL,

salary VARCHAR2(6) NOT NULL,

fam\_doctor\_id VARCHAR2(6) NOT NULL,

assistant\_id VARCHAR2(6) NOT NULL,

specialist\_doc\_id VARCHAR2(6) NOT NULL

);

ALTER TABLE medical\_employees ADD (

CONSTRAINT med\_emp\_id\_pk

PRIMARY KEY

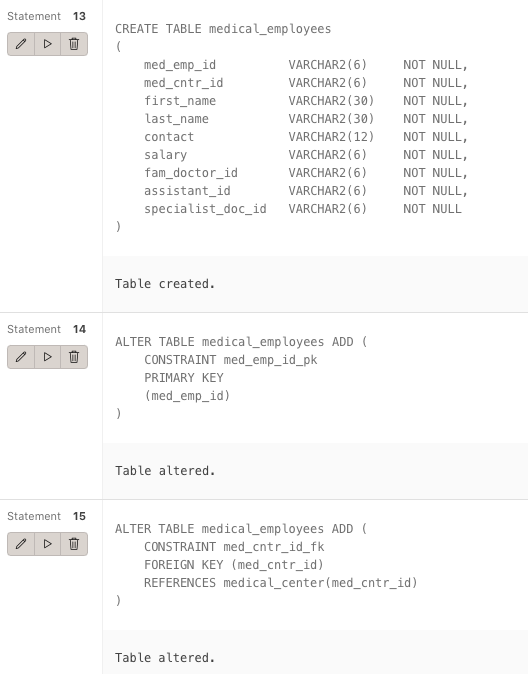
(med\_emp\_id) );

ALTER TABLE medical\_employees ADD (

CONSTRAINT med\_cntr\_id\_fk

FOREIGN KEY (med\_cntr\_id)

REFERENCES medical\_center(med\_cntr\_id) );



-- Crearea tabelului APPOINTMENT

CREATE TABLE appointment

(

appointment\_id VARCHAR2(6) NOT NULL,

med\_cntr\_id VARCHAR2(6) NOT NULL,

patient\_id VARCHAR2(6) NOT NULL,

assistant\_id VARCHAR2(6) NOT NULL,

appointment\_date DATE NOT NULL,

appointment\_price VARCHAR2(6) NOT NULL

);

ALTER TABLE appointment ADD (

CONSTRAINT appointment\_id\_pk

PRIMARY KEY

(appointment\_id) );

ALTER TABLE appointment ADD (

CONSTRAINT med\_cntr\_id\_app\_fk

FOREIGN KEY (med\_cntr\_id)

REFERENCES medical\_center(med\_cntr\_id) );

ALTER TABLE appointment ADD (

CONSTRAINT patient\_id\_app\_fk

FOREIGN KEY (patient\_id)

REFERENCES patient(patient\_id) );

ALTER TABLE appointment ADD (

CONSTRAINT assistant\_id\_app\_fk

FOREIGN KEY (assistant\_id)

REFERENCES medical\_employees(med\_emp\_id) );



-- Crearea tabelului MEDICAL\_PROBLEM

CREATE TABLE medical\_problem

(

med\_prob\_id VARCHAR2(6) NOT NULL,

appointment\_id VARCHAR2(6) NOT NULL,

description\_prob VARCHAR2(30)

);

ALTER TABLE medical\_problem ADD (

CONSTRAINT med\_prob\_id\_pk

PRIMARY KEY

(med\_prob\_id) );

ALTER TABLE medical\_problem ADD (

CONSTRAINT appointment\_id\_mp\_fk

FOREIGN KEY (appointment\_id)

REFERENCES appointment(appointment\_id) );



-- Crearea tabelului DIAGNOSIS

CREATE TABLE diagnosis

(

diagnosis\_id VARCHAR2(6) NOT NULL,

med\_prob\_id VARCHAR2(6) NOT NULL,

fam\_doctor\_id VARCHAR2(6) NOT NULL,

prescription\_id VARCHAR2(25) NOT NULL,

solved\_id VARCHAR2(25) NOT NULL,

under\_obs\_id VARCHAR2(25),

diagnosis\_date DATE NOT NULL

);

ALTER TABLE diagnosis ADD (

CONSTRAINT diagnosis\_id\_pk

PRIMARY KEY

(diagnosis\_id) );

ALTER TABLE diagnosis ADD (

CONSTRAINT med\_prob\_id\_dg\_fk

FOREIGN KEY (med\_prob\_id)

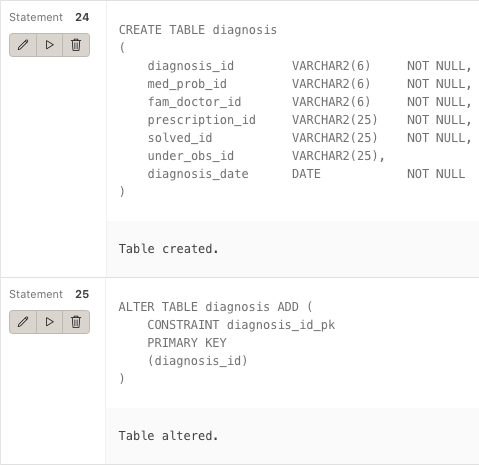
REFERENCES medical\_problem(med\_prob\_id) );

ALTER TABLE diagnosis ADD (

CONSTRAINT fam\_doctor\_id\_dg\_fk

FOREIGN KEY (fam\_doctor\_id)

REFERENCES medical\_employees(med\_emp\_id) );



-- Crearea tabelului MEDICAL\_SOLUTION

CREATE TABLE medical\_solution

(

med\_sol\_id VARCHAR2(25) NOT NULL,

diagnosis\_id VARCHAR2(6) NOT NULL,

specialist\_doc\_id VARCHAR2(6) NOT NULL,

description\_sol VARCHAR2(30)

);

ALTER TABLE medical\_solution ADD (

CONSTRAINT med\_sol\_id\_pk

PRIMARY KEY

(med\_sol\_id) );

ALTER TABLE medical\_solution ADD (

CONSTRAINT diagnosis\_id\_ms\_fk

FOREIGN KEY (diagnosis\_id)

REFERENCES diagnosis(diagnosis\_id) );

ALTER TABLE medical\_solution ADD (

CONSTRAINT specialist\_doc\_id\_ms\_fk

FOREIGN KEY (specialist\_doc\_id)

REFERENCES medical\_employees(med\_emp\_id) );



-- Crearea tabelului ASSIGN

CREATE TABLE assign

(

specialist\_doc\_id VARCHAR2(6) NOT NULL,

assistant\_id VARCHAR2(6) NOT NULL,

diagnosis\_id VARCHAR2(6) NOT NULL,

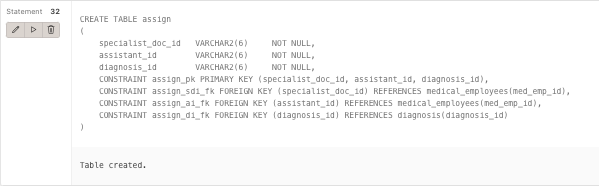
CONSTRAINT assign\_pk PRIMARY KEY (specialist\_doc\_id, assistant\_id, diagnosis\_id),

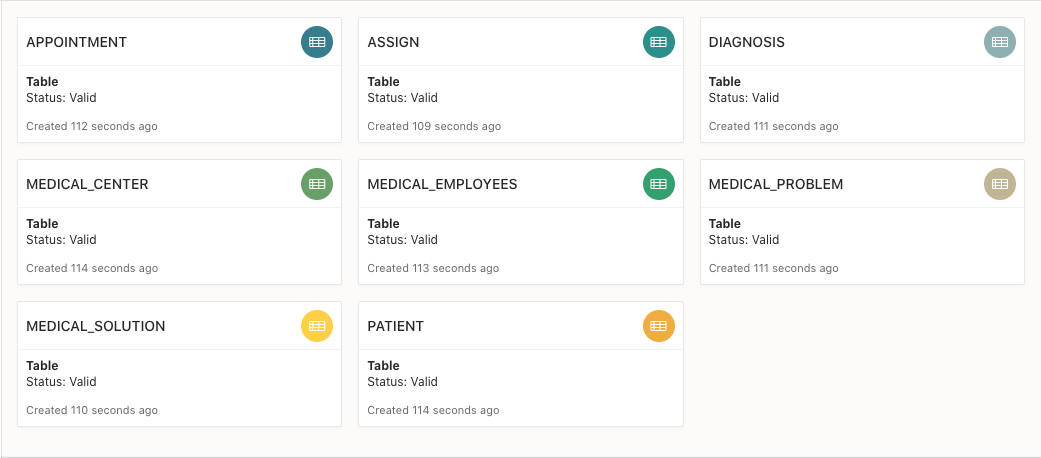
CONSTRAINT assign\_sdi\_fk FOREIGN KEY (specialist\_doc\_id) REFERENCES medical\_employees(med\_emp\_id),

CONSTRAINT assign\_ai\_fk FOREIGN KEY (assistant\_id) REFERENCES medical\_employees(med\_emp\_id),

CONSTRAINT assign\_di\_fk FOREIGN KEY (diagnosis\_id) REFERENCES diagnosis(diagnosis\_id)

);





1. **Adăugarea informațiilor în tabele**

-- INSERAREA IN TABELE --

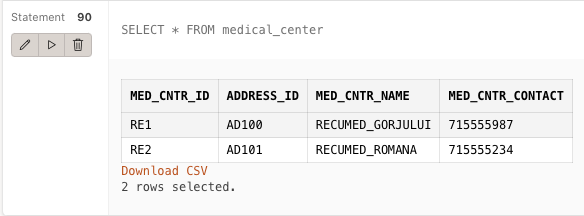
-- inserarea in tabelul MEDICAL\_CENTER

INSERT INTO medical\_center VALUES

('RE1', 'AD100', 'RECUMED\_GORJULUI', '715555987');

INSERT INTO medical\_center VALUES

('RE2', 'AD101', 'RECUMED\_ROMANA', '715555234');



-- inserarea in tabelul PATIENT

INSERT INTO patient VALUES

(1, 'Popa Ana', '715555811');

INSERT INTO patient VALUES

(2, 'Vasilescu Victor', '715555789');

INSERT INTO patient VALUES

(3, 'Atudorei Andrei', '715458699');

INSERT INTO patient VALUES

(4, 'Geman Sergiu', '715555811');

INSERT INTO patient VALUES

(5, 'Ion Cristian', '715115889');

INSERT INTO patient VALUES

(6, 'Pruteanu Gelu', '715335879');

INSERT INTO patient VALUES

(7, 'Geman Amalia', '715335879');

INSERT INTO patient VALUES

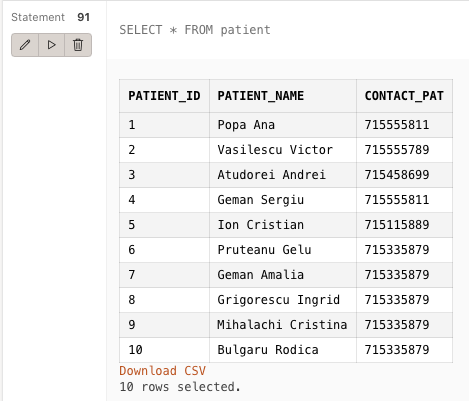
(8, 'Grigorescu Ingrid', '715335879');

INSERT INTO patient VALUES

(9, 'Mihalachi Cristina', '715335879');

INSERT INTO patient VALUES

(10, 'Bulgaru Rodica', '715335879');



-- inserarea in tabelul MEDICAL\_EMPLOYEES

INSERT INTO medical\_employees VALUES

(1, 'RE1', 'Ion', 'Florescu', '715110879', '2600', '1', '0', '0');

INSERT INTO medical\_employees VALUES

(2, 'RE1', 'Ana', 'Minciunescu', '715300879', '4000', '0', '0', '1');

INSERT INTO medical\_employees VALUES

(3, 'RE1', 'Florin', 'Mindrescu', '715125879', '5000', '0', '0', '1');

INSERT INTO medical\_employees VALUES

(4, 'RE1', 'Andrei', 'Florescu', '715665879', '2600', '1', '0', '0');

INSERT INTO medical\_employees VALUES

(5, 'RE1', 'Alex', 'Florea', '715337779', '2400', '1', '0', '0');

INSERT INTO medical\_employees VALUES

(6, 'RE1', 'Mihai', 'Neagoe', '715300879', '4000', '0', '0', '1');

INSERT INTO medical\_employees VALUES

(7, 'RE2', 'Miruna', 'Atudorei', '715115879', '3000', '1', '0', '0');

INSERT INTO medical\_employees VALUES

(8, 'RE2', 'Cristian', 'Pucean', '715225879', '3600', '1', '0', '0');

INSERT INTO medical\_employees VALUES

(9, 'RE2', 'Iulian', 'Hristea', '715388879', '2000', '0', '1', '0');

INSERT INTO medical\_employees VALUES

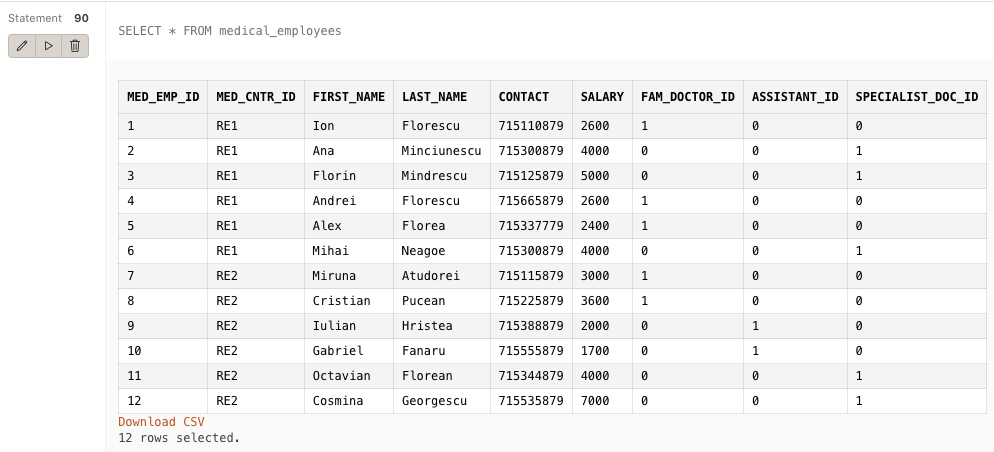
(10, 'RE2', 'Gabriel', 'Fanaru', '715555879', '1700', '0', '1', '0');

INSERT INTO medical\_employees VALUES

(11, 'RE2', 'Octavian', 'Florean', '715344879', '4000', '0', '0', '1');

INSERT INTO medical\_employees VALUES

(12, 'RE2', 'Cosmina', 'Georgescu', '715535879', '7000', '0', '0', '1');



-- inserarea in tabelul APPOINTMENT

INSERT INTO appointment VALUES

('APP1', 'RE1', 1, 9, TO\_DATE('08/10/2021', 'MM/DD/YYYY'), '110');

INSERT INTO appointment VALUES

('APP2', 'RE2', 2, 10, TO\_DATE('09/20/2021', 'MM/DD/YYYY'), '150');

INSERT INTO appointment VALUES

('APP3', 'RE1', 3, 9, TO\_DATE('10/01/2021', 'MM/DD/YYYY'), '200');

INSERT INTO appointment VALUES

('APP4', 'RE2', 4, 10, TO\_DATE('11/11/2020', 'MM/DD/YYYY'), '250');

INSERT INTO appointment VALUES

('APP5', 'RE1', 5, 9, TO\_DATE('02/20/2021', 'MM/DD/YYYY'), '110');

INSERT INTO appointment VALUES

('APP6', 'RE2', 6, 10, TO\_DATE('01/17/2019', 'MM/DD/YYYY'), '150');

INSERT INTO appointment VALUES

('APP7', 'RE1', 6, 9, TO\_DATE('08/10/2021', 'MM/DD/YYYY'), '220');

INSERT INTO appointment VALUES

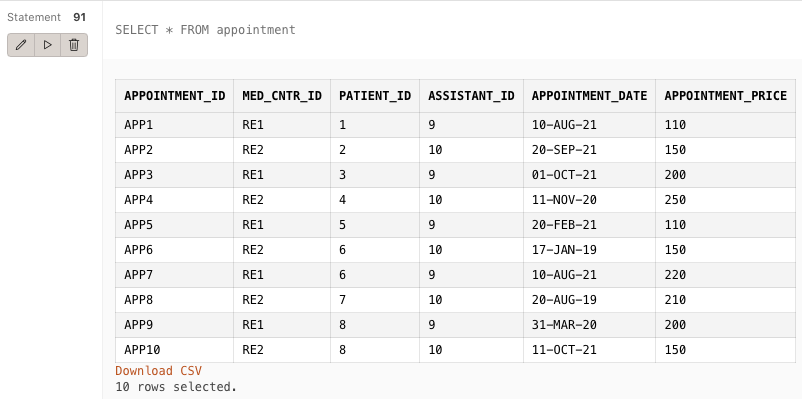
('APP8', 'RE2', 7, 10, TO\_DATE('08/20/2019', 'MM/DD/YYYY'), '210');

INSERT INTO appointment VALUES

('APP9', 'RE1', 8, 9, TO\_DATE('03/31/2020', 'MM/DD/YYYY'), '200');

INSERT INTO appointment VALUES

('APP10', 'RE2', 8, 10, TO\_DATE('10/11/2021', 'MM/DD/YYYY'), '150');



-- inserarea in tabelul MEDICAL\_PROBLEM

INSERT INTO medical\_problem VALUES

('MP1', 'APP1', 'GINECOLOGIE');

INSERT INTO medical\_problem VALUES

('MP2', 'APP2', 'DERMATOLOGIE');

INSERT INTO medical\_problem VALUES

('MP3', 'APP3', 'DERMATOLOGIE');

INSERT INTO medical\_problem VALUES

('MP4', 'APP4', 'NEUROLOGIE');

INSERT INTO medical\_problem VALUES

('MP5', 'APP5', 'STOMATOLOGIE');

INSERT INTO medical\_problem VALUES

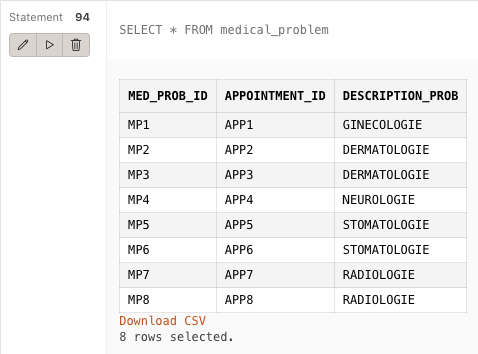
('MP6', 'APP6', 'STOMATOLOGIE');

INSERT INTO medical\_problem VALUES

('MP7', 'APP7', 'RADIOLOGIE');

INSERT INTO medical\_problem VALUES

('MP8', 'APP8', 'RADIOLOGIE');



-- inserarea in tabelul DIAGNOSIS

INSERT INTO diagnosis VALUES

('D1', 'MP1', 1, 'P100', '0', '0', TO\_DATE('08/10/2021', 'MM/DD/YYYY'));

INSERT INTO diagnosis VALUES

('D2', 'MP2', 4, 'P101', '0', '0', TO\_DATE('08/30/2021', 'MM/DD/YYYY'));

INSERT INTO diagnosis VALUES

('D4', 'MP4', 7, '0', '0', 'UO1', TO\_DATE('12/10/2021', 'MM/DD/YYYY'));

INSERT INTO diagnosis VALUES

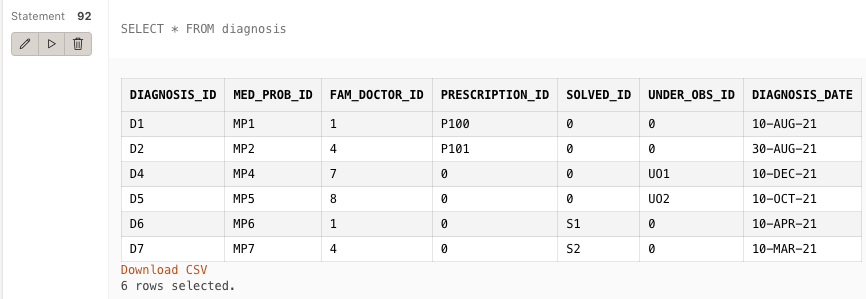
('D5', 'MP5', 8, '0', '0', 'UO2', TO\_DATE('10/10/2021', 'MM/DD/YYYY'));

INSERT INTO diagnosis VALUES

('D6', 'MP6', 1, '0', 'S1', '0', TO\_DATE('04/10/2021', 'MM/DD/YYYY'));

INSERT INTO diagnosis VALUES

('D7', 'MP7', 4, '0', 'S2', '0', TO\_DATE('03/10/2021', 'MM/DD/YYYY'));



-- insearea in tabelul MEDICAL\_SOLUTION

INSERT INTO medical\_solution VALUES

('MS1', 'D1', 12, 'Prescriptie medicala');

INSERT INTO medical\_solution VALUES

('MS2', 'D2', 11, 'Prescriptie medicala');

INSERT INTO medical\_solution VALUES

('MS3', 'D4', 6, 'Sub observatie');

INSERT INTO medical\_solution VALUES

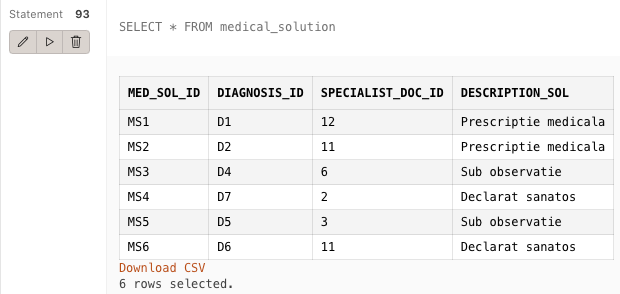
('MS4', 'D7', 2, 'Declarat sanatos');

INSERT INTO medical\_solution VALUES

('MS5', 'D5', 3, 'Sub observatie');

INSERT INTO medical\_solution VALUES

('MS6', 'D6', 11, 'Declarat sanatos');



---- inserarea in tabelul ASSIGN

INSERT INTO assign VALUES

(3, 9, 'D1');

INSERT INTO assign VALUES

(3, 10, 'D5');

INSERT INTO assign VALUES

(12, 9, 'D6');

INSERT INTO assign VALUES

(2, 10, 'D2');

INSERT INTO assign VALUES

(6, 10, 'D4');

INSERT INTO assign VALUES

(11, 9, 'D5');

INSERT INTO assign VALUES

(12, 10, 'D6');

INSERT INTO assign VALUES

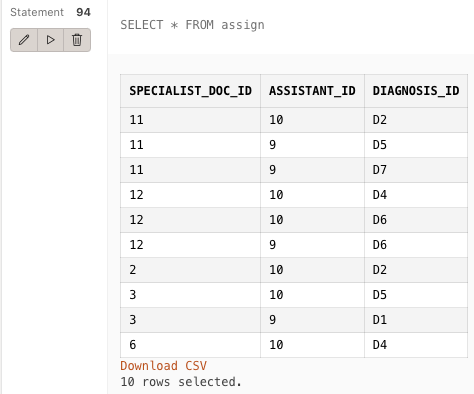
(11, 9, 'D7');

INSERT INTO assign VALUES

(11, 10, 'D2');

INSERT INTO assign VALUES

(12, 10, 'D4');



COMMIT;

1. **Folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate afișați salariile tuturor angajaților dintr-un centru medical dat ca parametru.**

CREATE OR REPLACE PROCEDURE Exercitiul6

( id\_mc medical\_center.med\_cntr\_id%TYPE)

IS

TYPE index\_table1 IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

v1 index\_table1;

v2 index\_table1;

v3 index\_table1;

TYPE medical\_employees IS TABLE OF NUMBER;

w medical\_employees;

BEGIN

SELECT med\_emp\_id, salary BULK COLLECT INTO v1, v2

from medical\_employees;

FOR i in v1.FIRST..v1.LAST LOOP

v3(v1(i)):=v2(i);

END LOOP;

SELECT med\_emp\_id BULK COLLECT INTO w

FROM medical\_employees

WHERE med\_cntr\_id = id\_mc;

FOR i IN 1..w.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Angajatul are salariul ' || v3(w(i)));

END LOOP;

END;

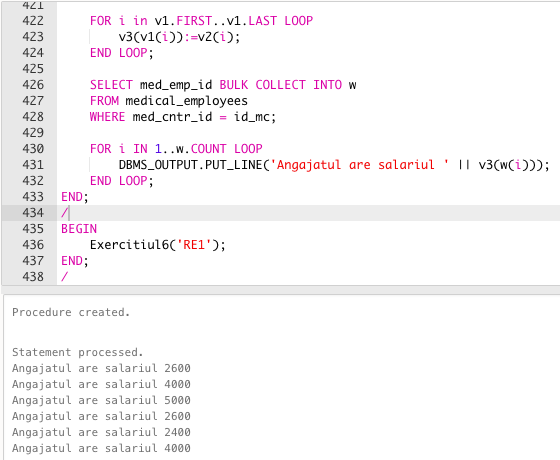
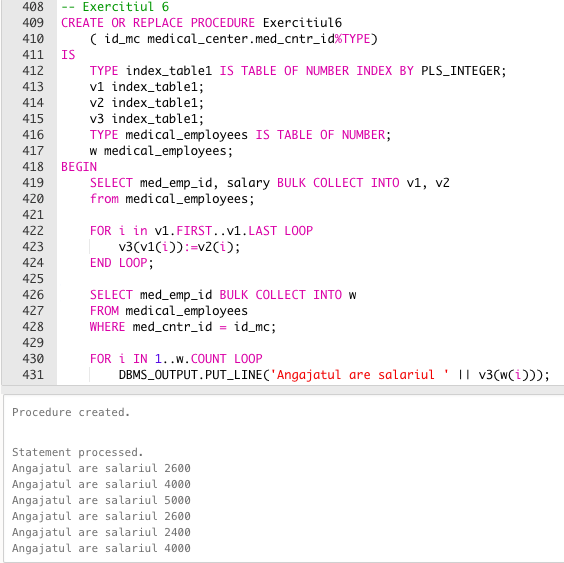
/

BEGIN

Exercitiul6('RE1');

END;

/

****

1. **Folosind o procedură stocată și un tip de cursor studiat afișați pentru fiecare clinică medicală numărul de angajați care lucreză în cadrul acesteia.**

-- cursor explicit

CREATE OR REPLACE PROCEDURE Exercitiul7

AS

name\_of\_med\_cntr medical\_center.med\_cntr\_name%TYPE;

number\_of\_employee NUMBER(4);

-- declarare cursor

CURSOR c IS

SELECT med\_cntr\_name, COUNT(med\_emp\_id)

FROM medical\_center m JOIN medical\_employees me ON (m.med\_cntr\_id = me.med\_cntr\_id)

GROUP BY med\_cntr\_name;

BEGIN

OPEN c; -- deschidere cursor

LOOP

FETCH c INTO name\_of\_med\_cntr, number\_of\_employee; -- incarcare cursor

EXIT WHEN c%NOTFOUND; -- verificare cursor

IF number\_of\_employee = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('In centrul medical ' || name\_of\_med\_cntr || ' nu lucreaza angajati');

ELSIF number\_of\_employee = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('In centrul medical ' || name\_of\_med\_cntr || ' lucreaza un angajat');

ELSE

DBMS\_OUTPUT.PUT\_LINE('In centrul medical ' || name\_of\_med\_cntr || ' lucreaza ' || number\_of\_employee || ' angajati');

END IF;

END LOOP;

CLOSE c; -- inchidere cursor

END;

/

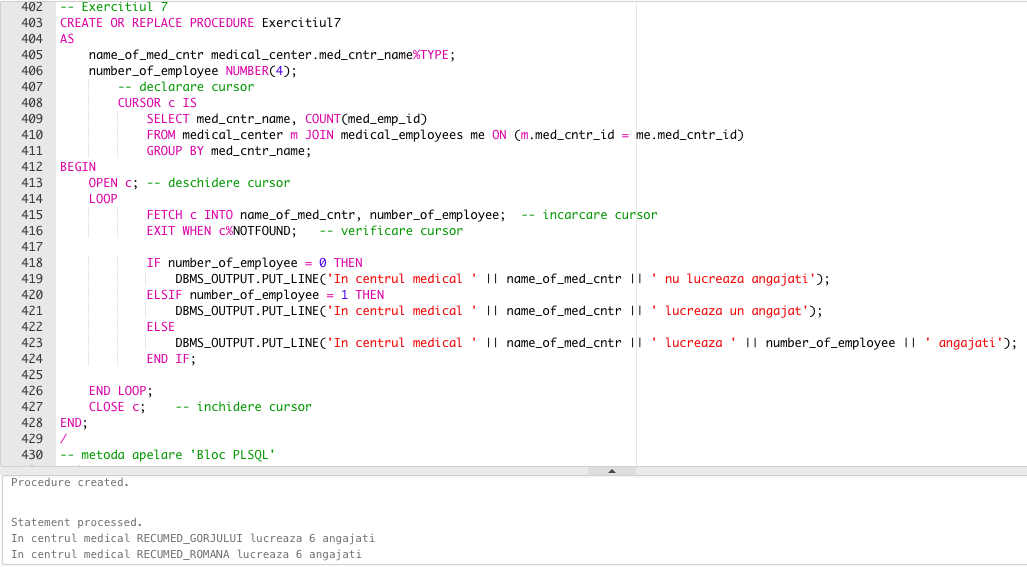
-- metoda apelare 'Bloc PLSQL'

BEGIN

Exercitiul7();

END;

/



1. **Folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite, afișați cel mai mare salariu al unui medic specialist care a asignat un diagnostic pentru o problemă medicală.**

CREATE OR REPLACE FUNCTION Exercitiul8

( med\_diagnosis diagnosis.diagnosis\_id%TYPE )

RETURN NUMBER

IS

maxim NUMBER;

BEGIN

SELECT MAX(salary) INTO maxim

FROM medical\_employees me

JOIN assign a ON ( me.med\_emp\_id = a.specialist\_doc\_id)

JOIN diagnosis d ON ( a.diagnosis\_id = d.diagnosis\_id)

WHERE d.diagnosis\_id = med\_diagnosis

GROUP BY d.diagnosis\_id;

RETURN maxim;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR( -20000, 'Nu exista diagnostic pentru acest id');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Alta eroare');

END Exercitiul8;

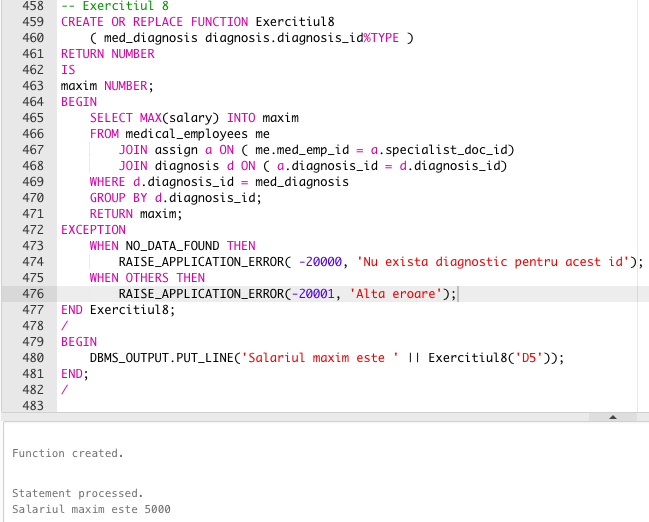
/

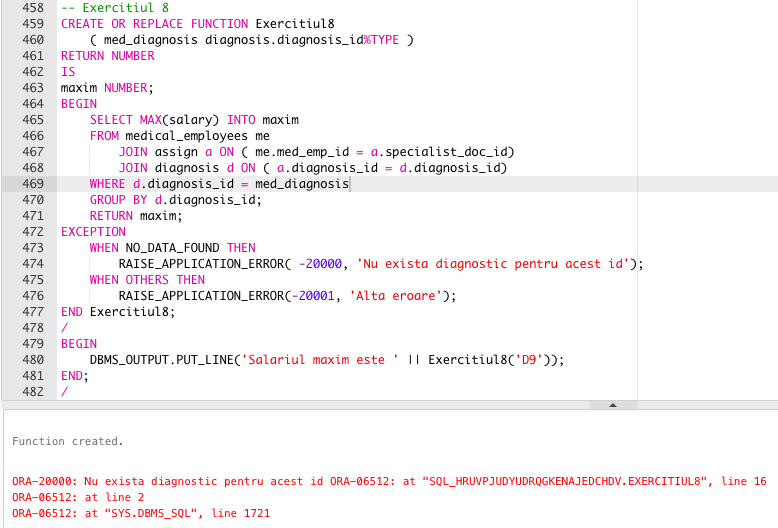
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Salariul maxim este ' || Exercitiul8('D9'));

END;

/

****

****

1. **Folosind un subprogram stocat de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite, afișați numele doctorului de familie care s-a ocupat de programarea pacientului dat.**

CREATE OR REPLACE PROCEDURE Exercitiul9

( name\_of\_patient patient.patient\_name%TYPE)

IS

name\_of\_doctor medical\_employees.first\_name%TYPE;

BEGIN

SELECT first\_name INTO name\_of\_doctor

FROM medical\_employees me JOIN diagnosis d ON ( me.med\_emp\_id = d.fam\_doctor\_id )

JOIN medical\_problem mp ON ( mp.med\_prob\_id = d.med\_prob\_id )

JOIN appointment a ON ( mp.appointment\_id = a.appointment\_id )

JOIN patient p ON ( a.patient\_id = p.patient\_id )

WHERE p.patient\_name = name\_of\_patient;

DBMS\_OUTPUT.PUT\_LINE( 'Numele doctorului de familie care s-a ocupat de programarea pacientului ' || name\_of\_patient || ' este ' || name\_of\_doctor );

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR( -20000, 'Nu exista pacient cu numele dat' );

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR( -20001, 'Exista mai multi doctori de familie care s-au ocupat de programare' );

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Alta eroare!' );

END Exercitiul9;

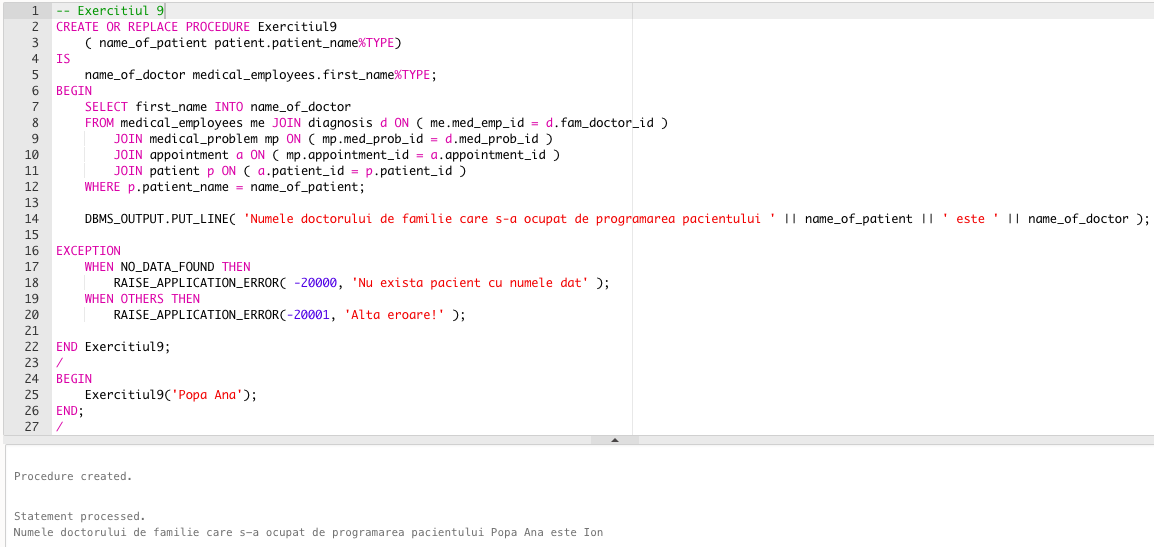
/

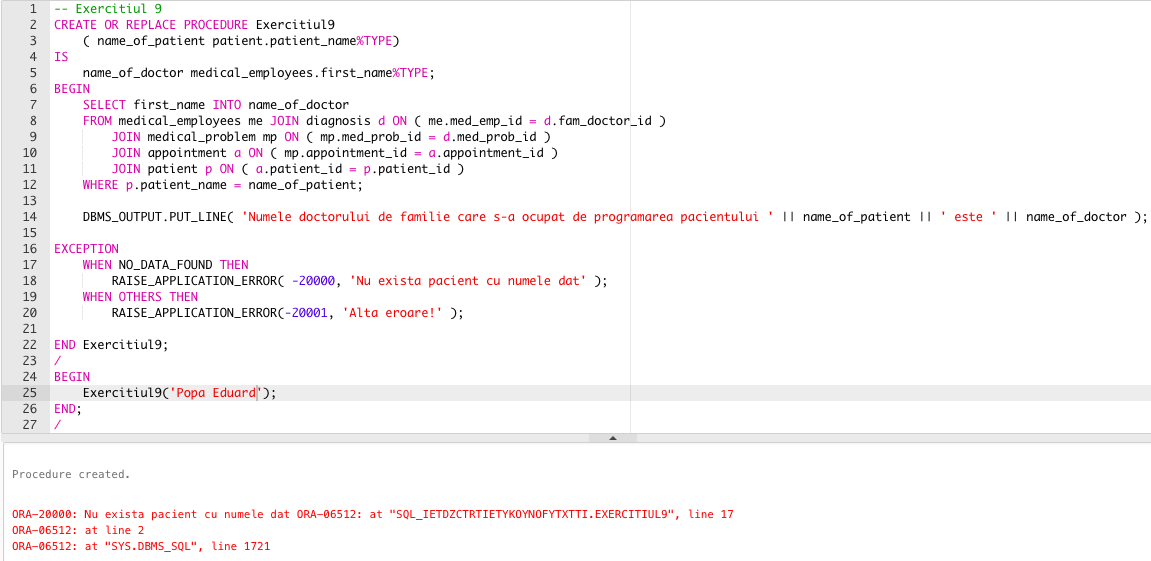
BEGIN

Exercitiul9('Popa Ana');

END;

/





1. **Definiți un declanșator care să permită lucrul asupra tabelului *medical\_employees* ( INSERT, UPDATE, DELETE ) decât în intervalul de ore 16:00-24:00, în zilele lucrătoare ( declanșator la nivel de comandă ).**

SELECT \* FROM medical\_employees;

CREATE OR REPLACE TRIGGER Exercitiul10

BEFORE INSERT OR UPDATE OR DELETE ON medical\_employees

BEGIN

IF ( TO\_CHAR ( SYSDATE, 'DD/MM') = '25/12' OR TO\_CHAR ( SYSDATE, 'DD/MM') = '01/05' )

OR ( TO\_CHAR ( SYSDATE, 'HH24' ) NOT BETWEEN 16 AND 24)

THEN

RAISE\_APPLICATION\_ERROR(-20001,'Tabelul nu poate fi actualizat');

END IF;

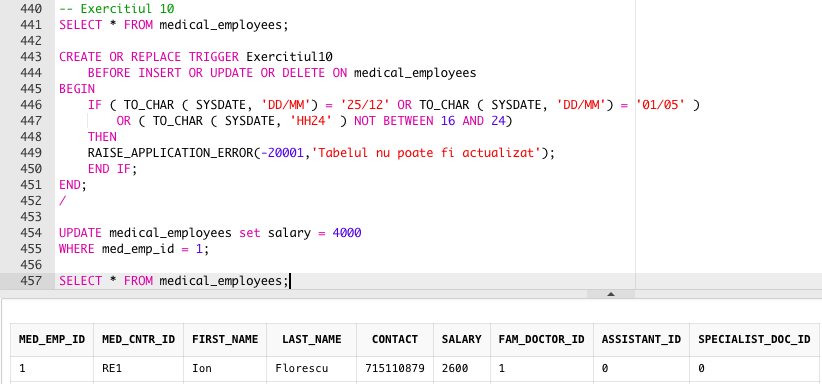
END;

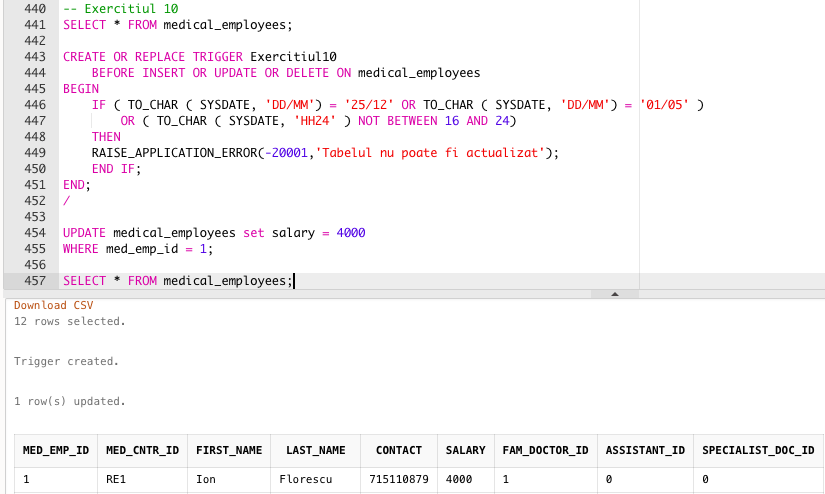
/

UPDATE medical\_employees set salary = 4000

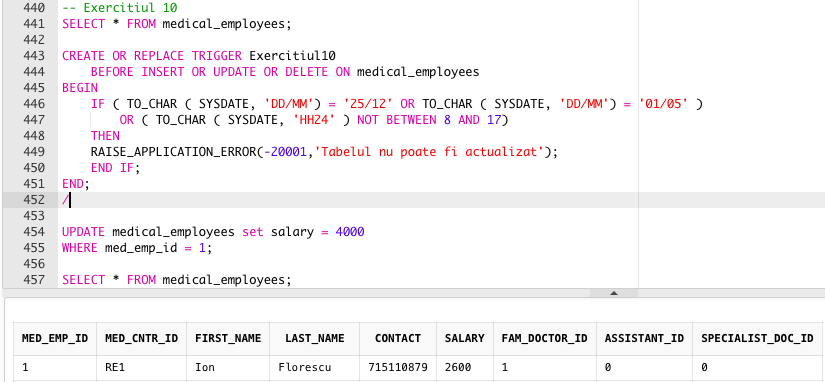
WHERE med\_emp\_id = 1;

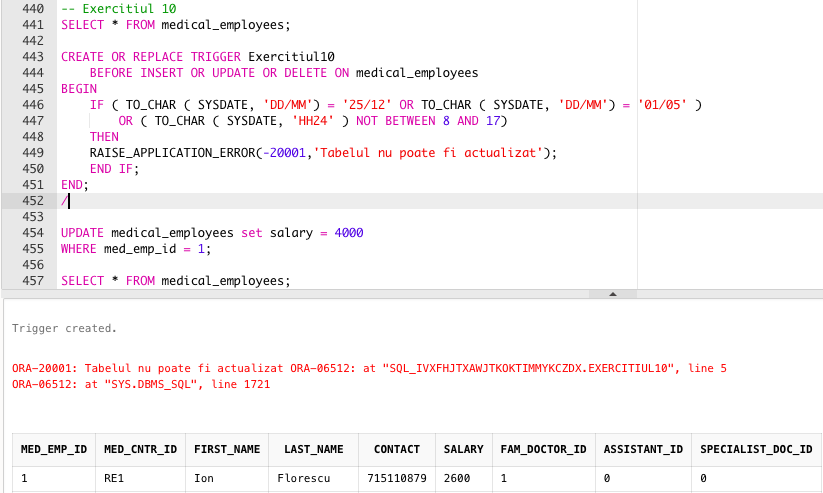
SELECT \* FROM medical\_employees;





Deoarece ora computer-ului meu este 23:30 am schimbat intervalul de ore la 08:00-17:00 pentru evidențierea cazului în care tabelul nu poate fi actualizat.





1. **Definiți un declanșator prin care să nu se permită micșorarea salariilor angajaților din tabelul *medical\_employees* ( declanșator la nivel de linie ).**

CREATE OR REPLACE TRIGGER Exercitiul11

BEFORE UPDATE OF salary ON medical\_employees

FOR EACH ROW

BEGIN

IF ( :NEW.salary < :OLD.salary ) THEN

RAISE\_APPLICATION\_ERROR(-20002,'Salariul nu poate fi micsorat');

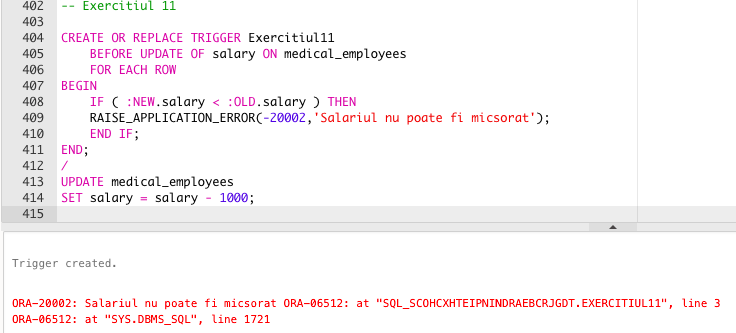
END IF;

END;

/

UPDATE medical\_employees

SET salary = salary - 1000;



1. **Creați tabelul *new\_event* cu următoarele câmpuri:**

* **eveniment ( evenimentul sistemului )**
* **nume\_obiect ( numele obiectului )**
* **tip\_obiect ( tipul obiectului )**
* **ora ( ora producerii evenimentului )**
* **data ( data producerii evenimentului )**

**Definiți un declanșator care să introducă date în acest tabel după ce utilizatorul a folosit o comandă LDD ( declanșator sistem – la nivel de schemă ).**

CREATE TABLE new\_event (

eveniment VARCHAR2(30),

nume\_obiect VARCHAR2(30),

tip\_obiect VARCHAR2(30),

time\_exec DATE,

data\_exec DATE

);

CREATE OR REPLACE TRIGGER Exercitiul12

AFTER CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

INSERT INTO new\_event

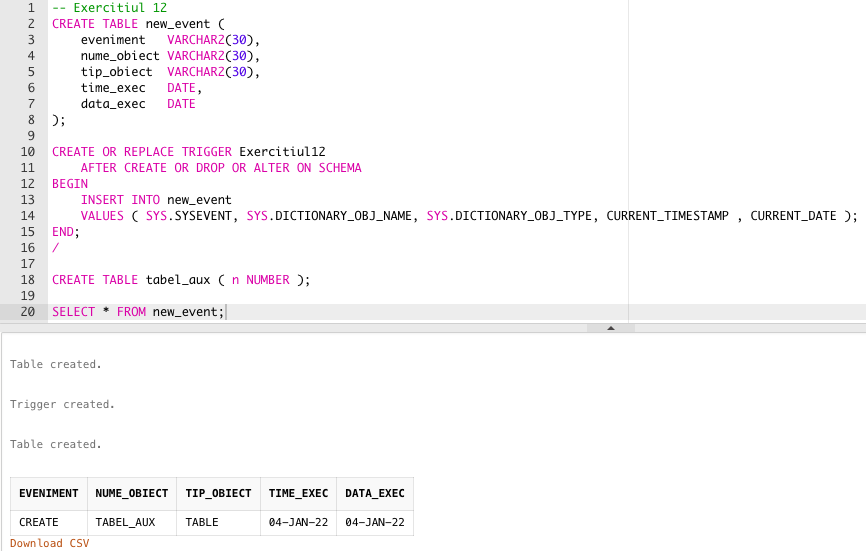
VALUES ( SYS.SYSEVENT, SYS.DICTIONARY\_OBJ\_NAME, SYS.DICTIONARY\_OBJ\_TYPE, CURRENT\_TIMESTAMP , CURRENT\_DATE );

END;

/

CREATE TABLE tabel\_aux ( n NUMBER );

SELECT \* FROM new\_event;



1. **Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.**

CREATE OR REPLACE PACKAGE proiect AS

PROCEDURE Exercitiul6 ( id\_mc medical\_center.med\_cntr\_id%TYPE);

PROCEDURE Exercitiul7;

FUNCTION Exercitiul8 ( med\_diagnosis diagnosis.diagnosis\_id%TYPE)

RETURN NUMBER;

PROCEDURE Exercitiul9 ( name\_of\_patient patient.patient\_name%TYPE );

END proiect;

/

CREATE OR REPLACE PACKAGE BODY proiect AS

FUNCTION Exercitiul8

( med\_diagnosis diagnosis.diagnosis\_id%TYPE )

RETURN NUMBER

IS

maxim NUMBER;

BEGIN

SELECT MAX(salary) INTO maxim

FROM medical\_employees me

JOIN assign a ON ( me.med\_emp\_id = a.specialist\_doc\_id)

JOIN diagnosis d ON ( a.diagnosis\_id = d.diagnosis\_id)

WHERE d.diagnosis\_id = med\_diagnosis

GROUP BY d.diagnosis\_id;

RETURN maxim;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR( -20000, 'Nu exista diagnostic pentru acest id');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Alta eroare');

END Exercitiul8;

PROCEDURE Exercitiul7

AS

name\_of\_med\_cntr medical\_center.med\_cntr\_name%TYPE;

number\_of\_employee NUMBER(4);

-- declarare cursor

CURSOR c IS

SELECT med\_cntr\_name, COUNT(med\_emp\_id)

FROM medical\_center m JOIN medical\_employees me ON (m.med\_cntr\_id = me.med\_cntr\_id)

GROUP BY med\_cntr\_name;

BEGIN

OPEN c; -- deschidere cursor

LOOP

FETCH c INTO name\_of\_med\_cntr, number\_of\_employee; -- incarcare cursor

EXIT WHEN c%NOTFOUND; -- verificare cursor

IF number\_of\_employee = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('In centrul medical ' || name\_of\_med\_cntr || ' nu lucreaza angajati');

ELSIF number\_of\_employee = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('In centrul medical ' || name\_of\_med\_cntr || ' lucreaza un angajat');

ELSE

DBMS\_OUTPUT.PUT\_LINE('In centrul medical ' || name\_of\_med\_cntr || ' lucreaza ' || number\_of\_employee || ' angajati');

END IF;

END LOOP;

CLOSE c; -- inchidere cursor

END Exercitiul7;

PROCEDURE Exercitiul9

( name\_of\_patient patient.patient\_name%TYPE)

IS

name\_of\_doctor medical\_employees.first\_name%TYPE;

BEGIN

SELECT first\_name INTO name\_of\_doctor

FROM medical\_employees me JOIN diagnosis d ON ( me.med\_emp\_id = d.fam\_doctor\_id )

JOIN medical\_problem mp ON ( mp.med\_prob\_id = d.med\_prob\_id )

JOIN appointment a ON ( mp.appointment\_id = a.appointment\_id )

JOIN patient p ON ( a.patient\_id = p.patient\_id )

WHERE p.patient\_name = name\_of\_patient;

DBMS\_OUTPUT.PUT\_LINE( 'Numele doctorului de familie care s-a ocupat de programarea pacientului ' || name\_of\_patient || ' este ' || name\_of\_doctor );

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR( -20000, 'Nu exista pacient cu numele dat' );

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR( -20001, 'Exista mai multi doctori de familie care s-au ocupat de programare' );

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Alta eroare!' );

END Exercitiul9;

PROCEDURE Exercitiul6

( id\_mc medical\_center.med\_cntr\_id%TYPE)

IS

TYPE index\_table1 IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

v1 index\_table1;

v2 index\_table1;

v3 index\_table1;

TYPE medical\_employees IS TABLE OF NUMBER;

w medical\_employees;

BEGIN

SELECT med\_emp\_id, salary BULK COLLECT INTO v1, v2

from medical\_employees;

FOR i in v1.FIRST..v1.LAST LOOP

v3(v1(i)):=v2(i);

END LOOP;

SELECT med\_emp\_id BULK COLLECT INTO w

FROM medical\_employees

WHERE med\_cntr\_id = id\_mc;

FOR i IN 1..w.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(v3(w(i)));

END LOOP;

END Exercitiul6;

END proiect;

/

BEGIN

proiect.Exercitiul9('Popa Ana');

END;

/

