**Temă**

E1. Adaptați cerința exercițiului 4 din fișierul Laborator8\_PLSQL\_optional.pdf pe schema aleasă pentru proiect.

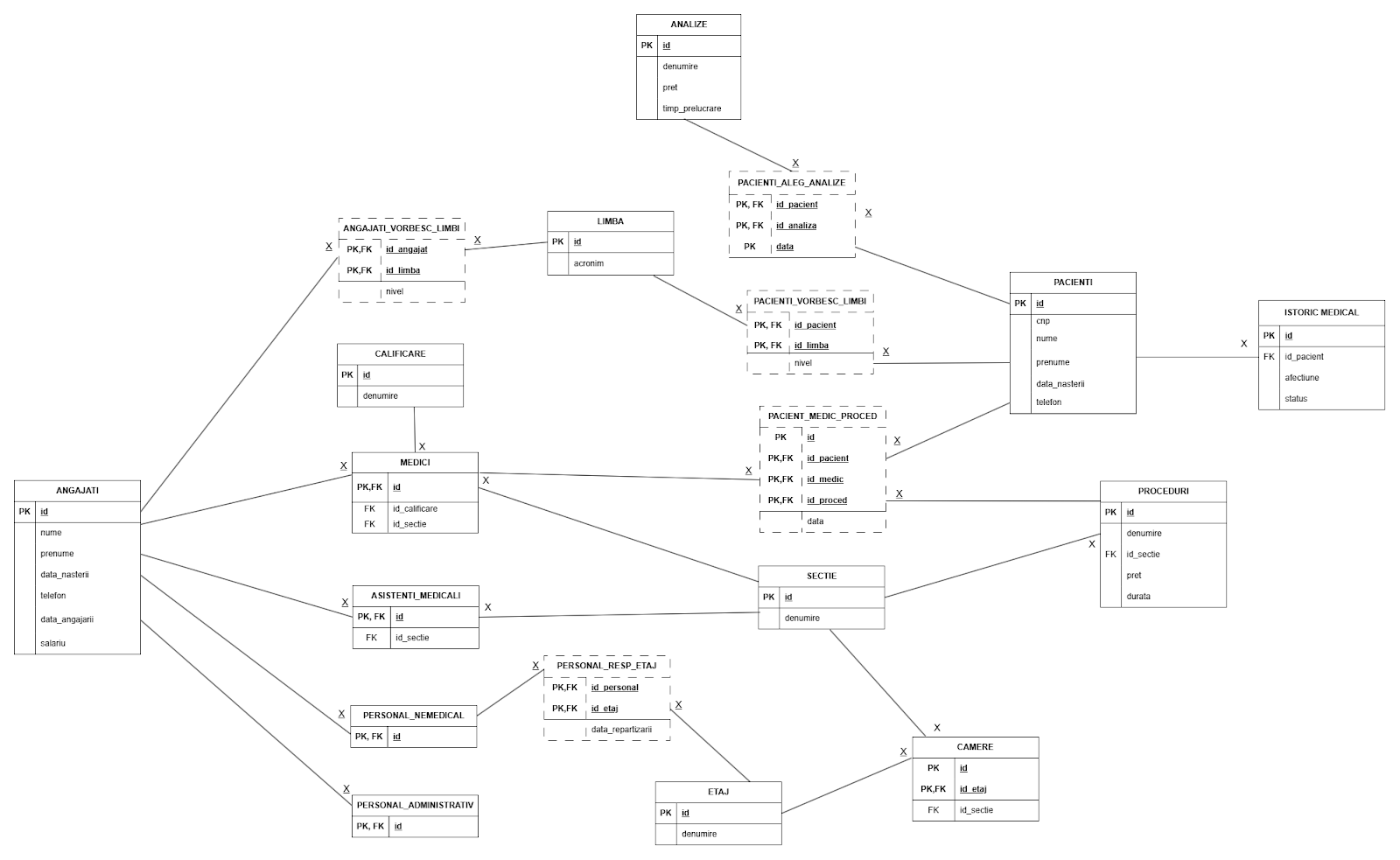
*Cerința:*

Să se creeze un pachet care să conţină:

- o funcţie prin care se vor returna toţi pacienții (nume și prenume) care îndeplinesc o anumită condiţie, dată ca parametru;

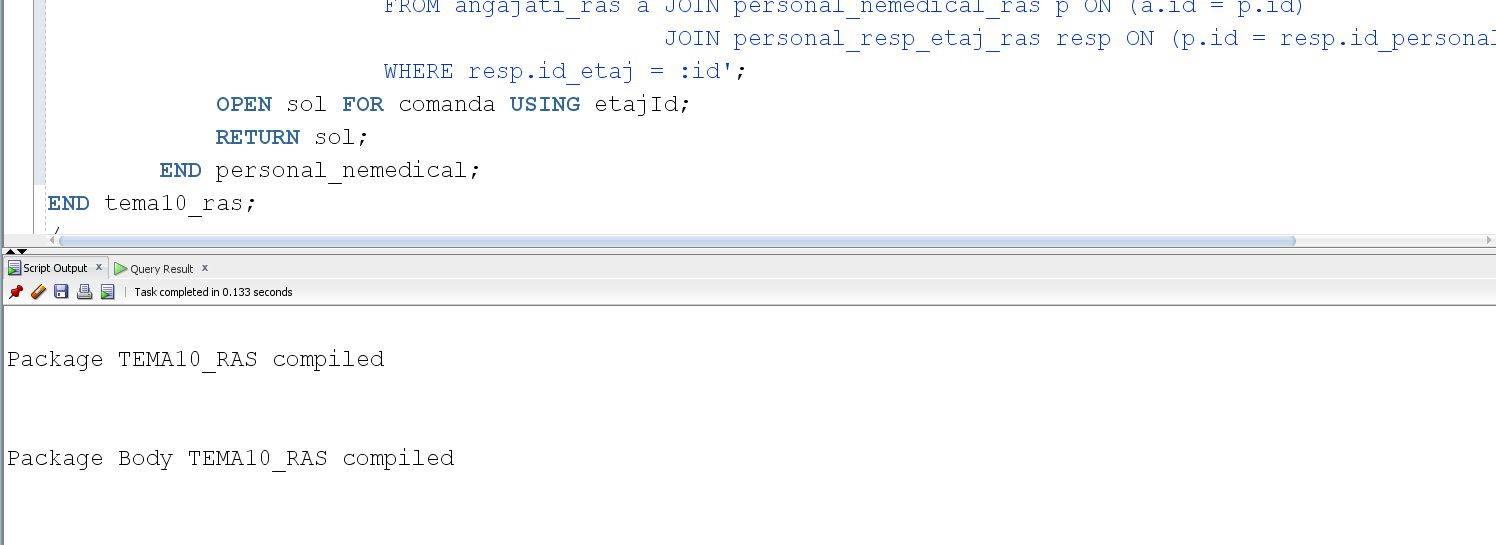
- o funcţie prin care se va returna tot personalul nemedical (nume + prenume) care lucrează pe un anumit etaj *etajId*, dat ca parametru;

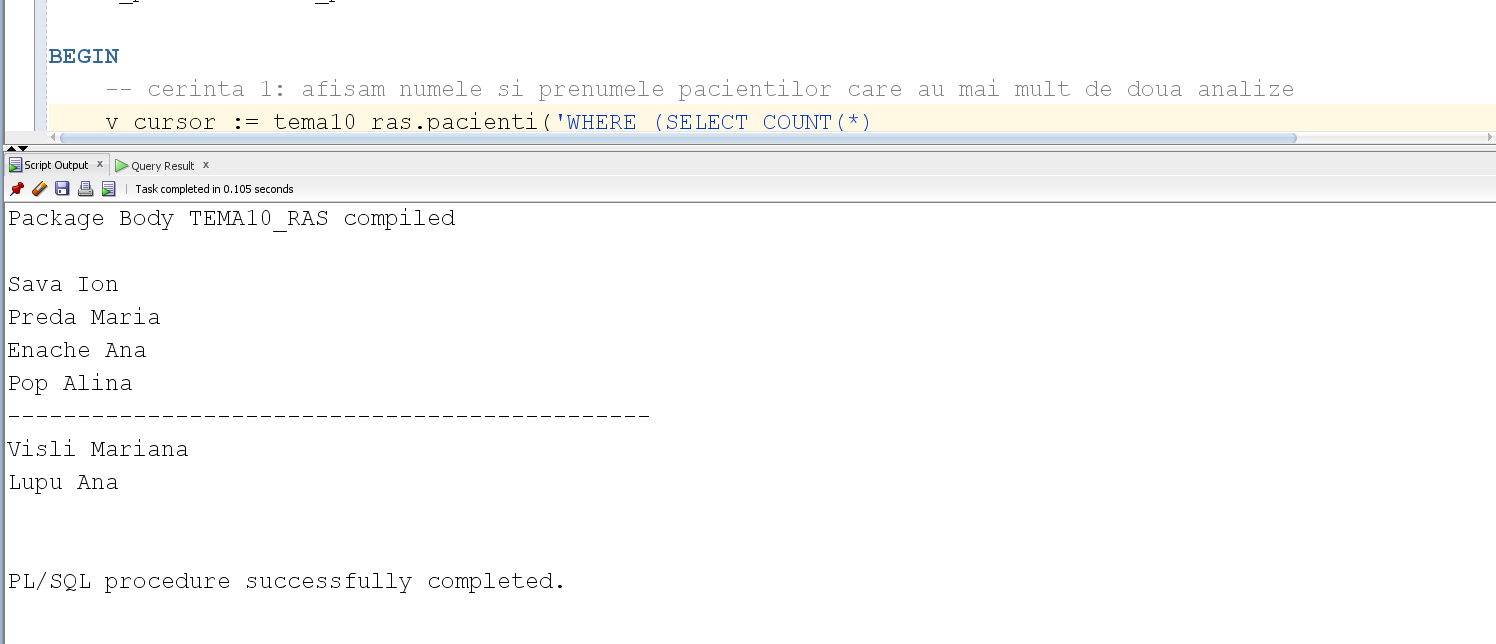
S-a folosit următoarea schemă:



*Rezolvare:*

| SET SERVEROUTPUT ON;  CREATE OR REPLACE PACKAGE tema10\_ras AS  TYPE refcursor IS REF CURSOR;  FUNCTION pacienti (conditie VARCHAR2) RETURN refcursor;  FUNCTION personal\_nemedical (etajId NUMBER) RETURN refcursor;  END tema10\_ras;  /  CREATE OR REPLACE PACKAGE BODY tema10\_ras AS  FUNCTION pacienti (conditie VARCHAR2) RETURN refcursor  IS  sol refcursor;  comanda VARCHAR2(300);  BEGIN  comanda := 'SELECT p.nume, p.prenume  FROM pacienti\_ras p ' || conditie;  OPEN sol FOR comanda;  RETURN sol;  END pacienti;    FUNCTION personal\_nemedical (etajId NUMBER) RETURN refcursor  IS  sol refcursor;  comanda VARCHAR2(300);  BEGIN  comanda := 'SELECT a.nume,a.prenume  FROM angajati\_ras a JOIN personal\_nemedical\_ras p ON (a.id = p.id)  JOIN personal\_resp\_etaj\_ras resp ON (p.id = resp.id\_personal)  WHERE resp.id\_etaj = :id';  OPEN sol FOR comanda USING etajId;  RETURN sol;  END personal\_nemedical;  END tema10\_ras;  /  DECLARE  TYPE date\_pacient IS RECORD (nume pacienti\_ras.nume%TYPE,  prenume pacienti\_ras.prenume%TYPE);  v\_pacient date\_pacient;  v\_cursor tema10\_ras.refcursor;    TYPE date\_personal IS RECORD (nume angajati\_ras.nume%TYPE,  prenume angajati\_ras.prenume%TYPE);  v\_personal date\_personal;    BEGIN  -- cerinta 1: afisam numele si prenumele pacientilor care au mai mult de doua analize  v\_cursor := tema10\_ras.pacienti('WHERE (SELECT COUNT(\*)  FROM pacienti\_aleg\_analize\_ras asoc  WHERE asoc.id\_pacient = p.id) > 2'  );  LOOP  FETCH v\_cursor INTO v\_pacient;  EXIT WHEN v\_cursor%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(v\_pacient.nume || ' ' || v\_pacient.prenume);  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------');  -- cerinta 2: afisam numele si prenumele personalului nemedical  -- de pe un anumit etaj  v\_cursor := tema10\_ras.personal\_nemedical('1');  LOOP  FETCH v\_cursor INTO v\_personal;  EXIT WHEN v\_cursor%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(v\_personal.nume || ' ' || v\_personal.prenume);  END LOOP;  END;  / |
| --- |





E2. Implementați o procedură care declanșează trigger-ul sistem creat pentru proiect.

*La proiect am implementat un trigger LDD care populează un tabel de tip istoric, unde se ține evidența comenzilor LDD.*

*Cerința*: Scrieți o procedură care primește numele unui tabel, numele unei coloane și tipul acesteia și adaugă coloana în tabel.

*Rezolvare:*

| -- triggerul LDD definit la proiect  -- cream tabelul pentru istoric  CREATE TABLE istoric\_ras (utilizator VARCHAR2(50),  nume\_bd VARCHAR2(50),  eveniment VARCHAR2(50),  nume\_obiect VARCHAR2(50),  tip\_obiect VARCHAR2(50),  data DATE);  -- cream triggerul  CREATE OR REPLACE TRIGGER update\_istoric  AFTER CREATE OR DROP OR ALTER ON SCHEMA  BEGIN  INSERT INTO istoric\_ras  VALUES (SYS.LOGIN\_USER, SYS.DATABASE\_NAME, SYS.SYSEVENT, SYS.DICTIONARY\_OBJ\_NAME,  SYS.DICTIONARY\_OBJ\_TYPE, SYSDATE);  END;  /  CREATE OR REPLACE PROCEDURE adauga\_coloana(tabel VARCHAR2, nume\_coloana VARCHAR2, tip\_coloana VARCHAR2)  IS  comanda VARCHAR2(300);  BEGIN  comanda := 'ALTER TABLE ' || tabel ||  ' ADD ' || nume\_coloana || ' ' || tip\_coloana;    EXECUTE IMMEDIATE comanda;    END adauga\_coloana;  /  BEGIN  adauga\_coloana('pacienti\_ras', 'nr\_proceduri', 'NUMBER');  END;  /  SELECT \*  FROM pacienti\_ras;  SELECT \*  FROM istoric\_ras; |
| --- |

