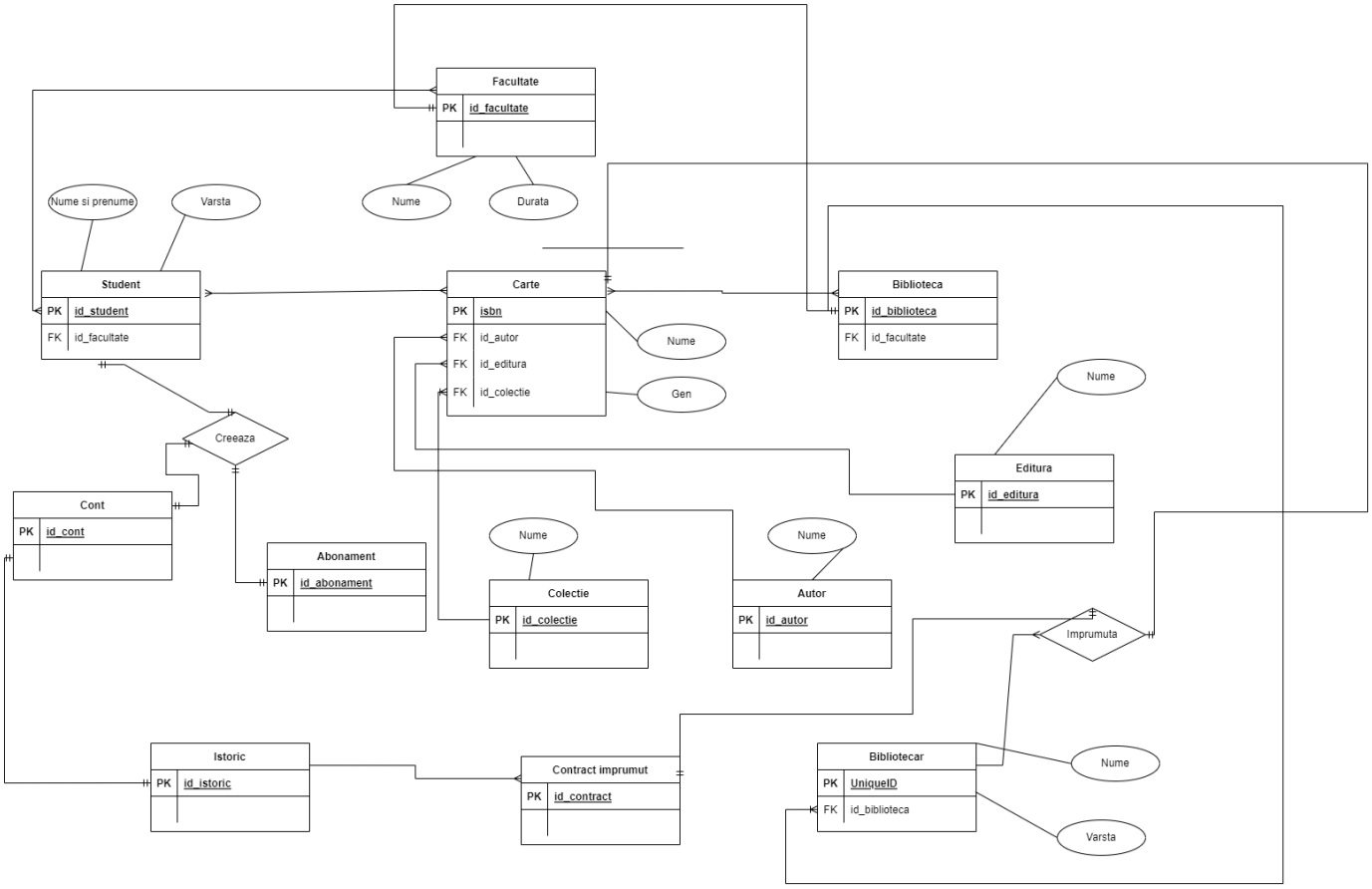
Proiect Sisteme de Gestiune a Bazelor de Date: University Library Management System

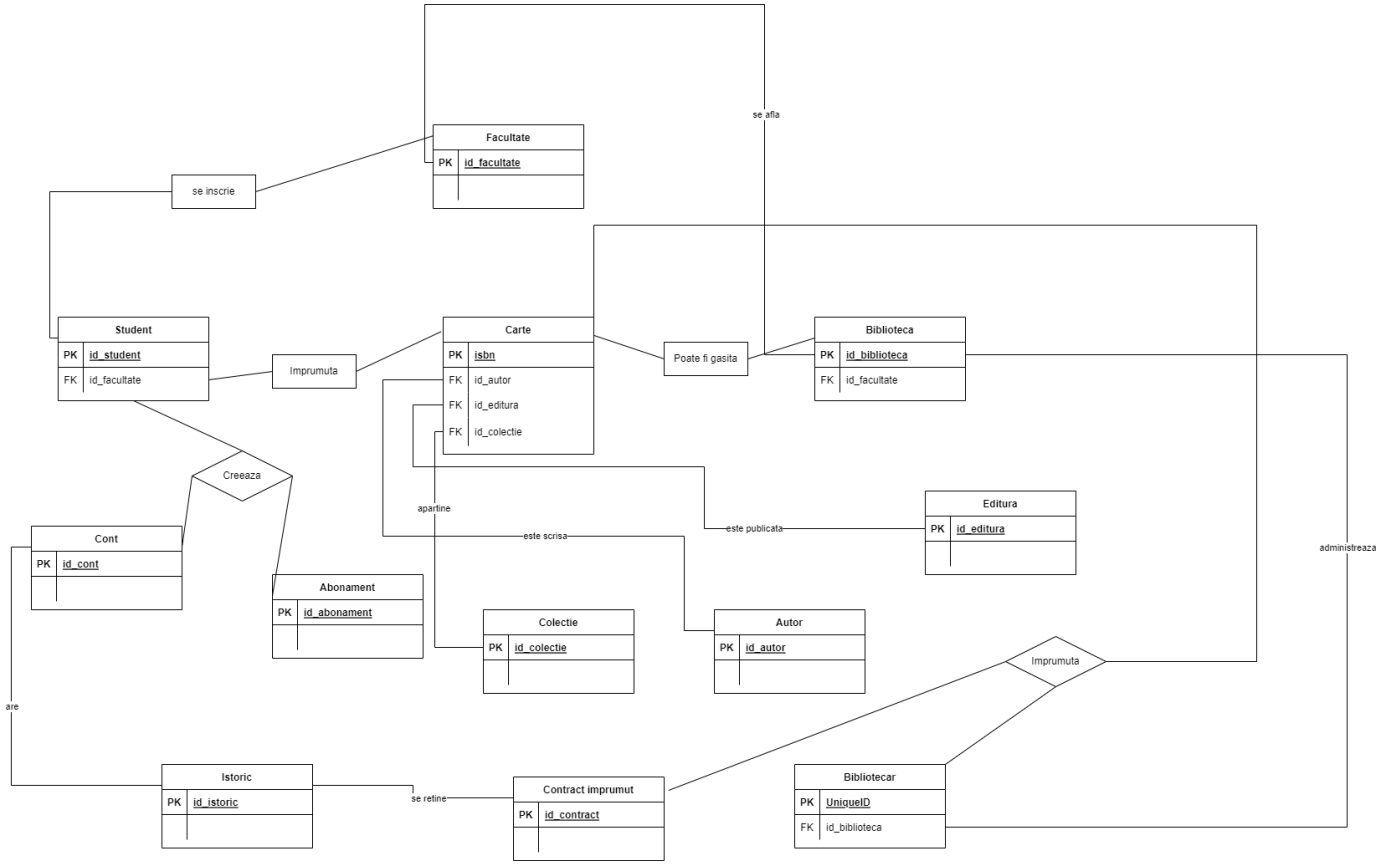
1)

Model Real: Se doreste implementarea unui sistem de gestiune pentru bibliotecile din facultatile din Bucuresti. Fiecare facultate are o biblioteca. Studentii (care pot fi inscrisi la una sau mai multe facultati) isi pot deschide un cont la bibilioteca facultatii. In cadrul acestui cont isi pot crea abonamente (acestea pot fi premium sau standard) cu ajutorul carora pot impurumuta carti. Pentru fiecare carte imprumutata se creeaza un contract de imprumut. Fiecare imprumut este trecut in istoricul contului utilizatorului. Fiecare carte are unul sau mai multi autori, exact o editura si poate face parte din una sau mai multe colectii. De asemenea, la fiecare biblioteca exista unul sau mai multi bibliotecati ce sunt responsabili de acestea si de emitera contractelor de imprumut.

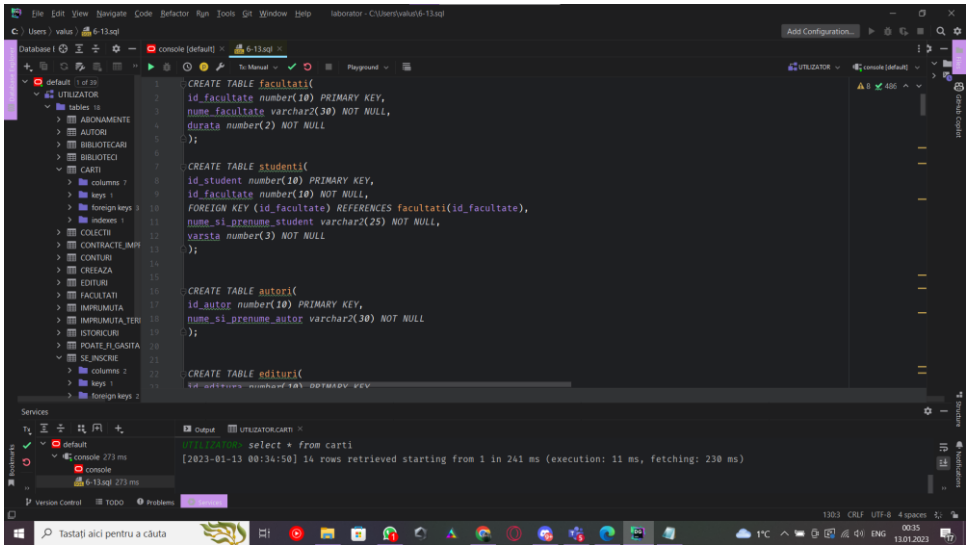
2) ERD:


Facultate
PK id_facultate
Nume si prenume
Varsta
Nume
Durata
Student
PK id_student
FK id_facultate
Carte
PK isbn
FK id_autor
FK id_editura
FK id_colectie
Nume
Gen
Biblioteca
PK id_biblioteca
FK id_facultate
Creeaza
Nume
Editura
PK id_editura
Cont
PK id_cont
Abonament
PK id_abonament
Nume
Colectie
PK id_colectie
Nume
Autor
PK id_autor
Imprumuta
Istoric
PK id_istoric
Contract imprumut
PK id_contract
Bibliotecar
PK UniqueID
FK id_biblioteca
Nume
Varsta

3) Diagrama Conceptuala:



4) Crearea tabelelor:




```
CREATE TABLE facultati(
id_facultate number(10) PRIMARY KEY,
nume_facultate varchar2(30) NOT NULL,
durata number(2) NOT NULL
);

CREATE TABLE studenti(
id_student number(10) PRIMARY KEY,
id_facultate number(10) NOT NULL,
FOREIGN KEY (id_facultate) REFERENCES facultati(id_facultate),
nume_si_prenume_student varchar2(25) NOT NULL,
varsta number(3) NOT NULL
);
```

```
CREATE TABLE autori(
id_autor number(10) PRIMARY KEY,
nume_si_prenume_autor varchar2(30) NOT NULL
);


CREATE TABLE edituri(
id_editura number(10) PRIMARY KEY,
nume_editura varchar2(30) NOT NULL,
tara varchar2(20) NOT NULL
);

CREATE TABLE colectii(
id_colectie number(10) PRIMARY KEY,
nume_colectie varchar2(30) NOT NULL
);

CREATE TABLE carti(
isbn number(13) PRIMARY KEY,
id_autor number(10) NOT NULL,
id_editura number(10) NOT NULL,
id_colectie number(10) NOT NULL,
FOREIGN KEY (id_autor) REFERENCES autori(id_autor),
FOREIGN KEY (id_editura) REFERENCES edituri(id_editura),
```

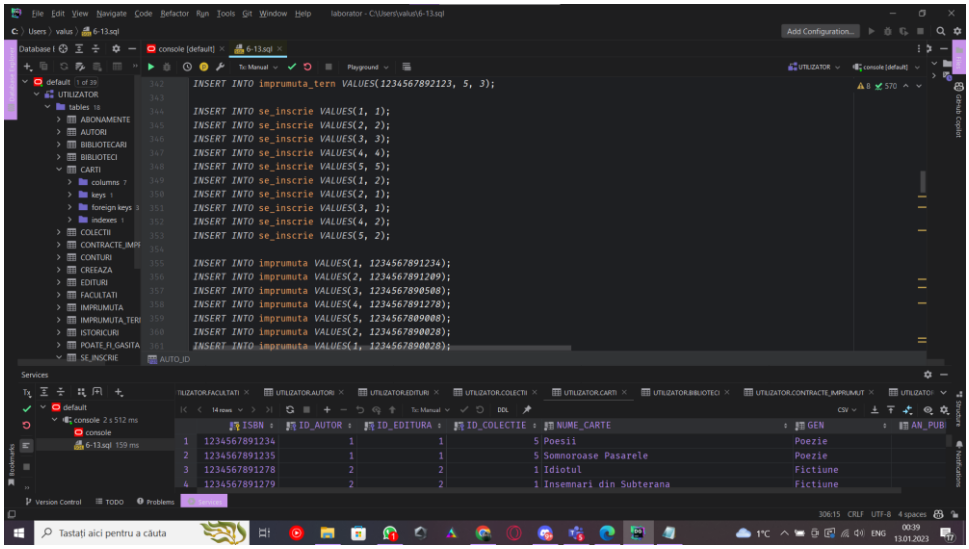
```
FOREIGN KEY (id_colectie) REFERENCES colectii(id_colectie),
nume_carte varchar2(100) NOT NULL,
gen varchar2(50) NOT NULL,
an_publicare number(4)
);


CREATE TABLE biblioteci(
id_biblioteca number(10) PRIMARY KEY,
id_facultate number(10) NOT NULL,
FOREIGN KEY (id_facultate) REFERENCES facultati(id_facultate),
nume_biblioteca varchar2(25)
);


CREATE TABLE conturi(
id_cont number(10) PRIMARY KEY,
username varchar2(25) NOT NULL
);


CREATE TABLE contracte_imprumut(
id_contract number(10) PRIMARY KEY,
taxa_intarziere number(3) NOT NULL,
durata number(3) NOT NULL
```

```
);


CREATE TABLE bibliotecari(
id_bibliotecar number(10) PRIMARY KEY,
id_biblioteca number(10) NOT NULL,
FOREIGN KEY (id_biblioteca) REFERENCES
biblioteci(id_biblioteca),
nume_si_prenume_bibliotecar varchar2(30) NOT NULL,
varsta number(3) NOT NULL
);


CREATE TABLE istoricuri(
id_istoric number(10) PRIMARY KEY,
nume_istoric varchar2(30)
);


CREATE TABLE abonamente(
id_abonament number(10) PRIMARY KEY,
tip_abonament varchar2(30) NOT NULL
);

CREATE TABLE creeaza(
```

```
id_cont number(10) NOT NULL,
id_student number(10) NOT NULL,
id_abonament number(10) NOT NULL,
CONSTRAINT pk_creeaza PRIMARY KEY (id_cont, id_student, id_abonament),
FOREIGN KEY (id_cont) REFERENCES conturi(id_cont),
FOREIGN KEY (id_student) REFERENCES studenti(id_student),
FOREIGN KEY (id_abonament) REFERENCES abonamente(id_abonament)
);

CREATE TABLE imprumuta_tern(
isbn number(13) NOT NULL,
id_contract number(10) NOT NULL,
id_bibliotecar number(10) NOT NULL,
CONSTRAINT pk_imprumuta_tern PRIMARY KEY (isbn, id_contract, id_bibliotecar),
FOREIGN KEY (isbn) REFERENCES carti(isbn),
FOREIGN KEY(id_contract) REFERENCES contracte_imprumut(id_contract),
FOREIGN KEY(id_bibliotecar) REFERENCES bibliotecari(id_bibliotecar)
);

CREATE TABLE se_inscrie(
id_student number(10) NOT NULL,
id_facultate number(10) NOT NULL,
```

```
FOREIGN KEY (id_student) REFERENCES studenti(id_student),
FOREIGN KEY(id_facultate) REFERENCES facultati(id_facultate),
CONSTRAINT pk_se_inscrie PRIMARY KEY (id_student,
id_facultate)
);

CREATE TABLE imprumuta(
id_student number(10) NOT NULL,
isbn number(13) NOT NULL,
FOREIGN KEY (id_student) REFERENCES studenti(id_student),
FOREIGN KEY(isbn) REFERENCES carti(isbn),
CONSTRAINT pk_imprumuta PRIMARY KEY (id_student, isbn)
);

CREATE TABLE poate_fi_gasita(
id_biblioteca number(10) NOT NULL,
isbn number(13) NOT NULL,
FOREIGN KEY (id_biblioteca) REFERENCES
biblioteci(id_biblioteca),
FOREIGN KEY(isbn) REFERENCES carti(isbn),
CONSTRAINT pk_poate_fi_gasita PRIMARY KEY (id_biblioteca,
isbn)
);
```

5) Inserare:



CREATE SEQUENCE AUTO_ID

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 1000

NOCYCLE;

INSERT INTO facultati(id_facultate, nume_facultate, durata)
VALUES (AUTO_ID.nextval, 'Facultatea de Matematica', 3);

INSERT INTO facultati(id_facultate, nume_facultate, durata)
VALUES (AUTO_ID.nextval, 'Facultatea de Medicina', 6);

INSERT INTO facultati(id_facultate, nume_facultate, durata)
VALUES (AUTO_ID.nextval, 'Facultatea Drobeta', 4);

INSERT INTO facultati(id_facultate, nume_facultate, durata)
VALUES (AUTO_ID.nextval, 'Facultatea de Filosofie', 5);

INSERT INTO facultati(id_facultate, nume_facultate, durata)
VALUES (AUTO_ID.nextval, 'Universitatea Craiova', 4);

```
DROP SEQUENCE AUTO_ID;

CREATE SEQUENCE AUTO_ID
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE;

INSERT INTO studenti(id_student, id_facultate,
nume_si_prenume_student, varsta) VALUES (AUTO_ID.nextval, 1,
'Mihai Popescu', 19);

INSERT INTO studenti(id_student, id_facultate,
nume_si_prenume_student, varsta) VALUES (AUTO_ID.nextval, 2,
'Ionut Nicu', 20);

INSERT INTO studenti(id_student, id_facultate,
nume_si_prenume_student, varsta) VALUES (AUTO_ID.nextval, 5,
'Valentin Ion', 20);

INSERT INTO studenti(id_student, id_facultate,
nume_si_prenume_student, varsta) VALUES (AUTO_ID.nextval, 1,
'Mihnea Mihali', 19);

INSERT INTO studenti(id_student, id_facultate,
nume_si_prenume_student, varsta) VALUES (AUTO_ID.nextval, 3,
'Marius Miopu', 21);

DROP SEQUENCE AUTO_ID;

CREATE SEQUENCE AUTO_ID
```

```
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE;

INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES
(AUTO_ID.nextval, 'Mihai Eminescu');
INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES
(AUTO_ID.nextval, 'Fyodor Dostoevsky');
INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES
(AUTO_ID.nextval, 'Albert Camus');
INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES
(AUTO_ID.nextval, 'Sylvia Plath');
INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES
(AUTO_ID.nextval, 'Ernest Hemingway');
INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES
(AUTO_ID.nextval, 'Nichita Stanescu');


DROP SEQUENCE AUTO_ID;

CREATE SEQUENCE AUTO_ID
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
```

```
NOCYCLE;


INSERT INTO edituri(id_editura, nume_editura, tara) VALUES
(AUTO_ID.nextval, 'Corint', 'Romania');

INSERT INTO edituri(id_editura, nume_editura, tara) VALUES
(AUTO_ID.nextval, 'Polirom', 'Romania');

INSERT INTO edituri(id_editura, nume_editura, tara) VALUES
(AUTO_ID.nextval, 'Nemira', 'Franta');

INSERT INTO edituri(id_editura, nume_editura, tara) VALUES
(AUTO_ID.nextval, 'Pinguin', 'Germania');

INSERT INTO edituri(id_editura, nume_editura, tara) VALUES
(AUTO_ID.nextval, 'Gold Fish', 'Anglia');


DROP SEQUENCE AUTO_ID;


CREATE SEQUENCE AUTO_ID

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 1000

NOCYCLE;


INSERT INTO colectii(id_colectie, nume_colectie) VALUES
(AUTO_ID.nextval, 'Carti de neuitat');

INSERT INTO colectii(id_colectie, nume_colectie) VALUES
(AUTO_ID.nextval, '1000 de carti de citit');
```

```
INSERT INTO colectii(id_colectie, nume_colectie) VALUES
(AUTO_ID.nextval, 'Fictiunea');

INSERT INTO colectii(id_colectie, nume_colectie) VALUES
(AUTO_ID.nextval, 'Poezii patrunzatoare');

INSERT INTO colectii(id_colectie, nume_colectie) VALUES
(AUTO_ID.nextval, 'Romanesti');


DROP SEQUENCE AUTO_ID;


INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567891234, 1, 1,
5, 'Poesii', 'Poezie', 1883);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567891235, 1, 1,
5, 'Somnoroase Pasarele', 'Poezie', 1883);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567891278, 2, 2,
1, 'Idiotul', 'Fictiune', 1867);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567891279, 2, 2,
1, 'Insemnari din Subterana', 'Fictiune', 1864);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567891200, 3, 2,
2, 'Mitul lui Sisif', 'Filosofie', 1942);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567891209, 3, 4,
2, 'Strainul', 'Fictiune', 1942);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567892122, 3, 1,
2, 'Ciuma', 'Fictiune', 1947);
```

```
INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567892123, 4, 2,
3, 'Clopotul de Sticla', 'Roman a clef', 1963);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567892198, 4, 2,
4, 'Ariel', 'Poezie', 1965);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567890028, 5, 5,
3, 'Batranul si Marea', 'Fictiune', 1952);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567890038, 5, 5,
1, 'Puhoaiele Primaverii', 'Parodie', 1926);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567890508, 6, 3,
4, 'Leoaica tanara, Iubirea: Poezii de dragoste', 'Poezie', 1964);

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie,
nume_carte, gen, an_publicare) VALUES (1234567809008, 6, 3,
4, 'Lectie despre cub', 'Poezie abstracta', 1979);



CREATE SEQUENCE AUTO_ID

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 1000

NOCYCLE;


INSERT INTO biblioteci(id_biblioteca, id_facultate,
nume_biblioteca) VALUES (AUTO_ID.nextval, 1, 'Gheorghe Titeica');
```

```
INSERT INTO biblioteci(id_biblioteca, id_facultate,
nume_biblioteca) VALUES (AUTO_ID.nextval, 2, 'Marius Emsi');

INSERT INTO biblioteci(id_biblioteca, id_facultate,
nume_biblioteca) VALUES (AUTO_ID.nextval, 3, 'Ilie Balaci');

INSERT INTO biblioteci(id_biblioteca, id_facultate,
nume_biblioteca) VALUES (AUTO_ID.nextval, 4, 'Lorin Andrei');

INSERT INTO biblioteci(id_biblioteca, id_facultate,
nume_biblioteca) VALUES (AUTO_ID.nextval, 5, 'Mihnea Mihail');



DROP SEQUENCE AUTO_ID;



CREATE SEQUENCE AUTO_ID

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 1000

NOCYCLE;


INSERT INTO conturi(id_cont, username) VALUES
(AUTO_ID.nextval, 'nicu123');

INSERT INTO conturi(id_cont, username) VALUES
(AUTO_ID.nextval, 'CititorulNebun');

INSERT INTO conturi(id_cont, username) VALUES
(AUTO_ID.nextval, 'Vasile_nelut');
```

```
INSERT INTO conturi(id_cont, username) VALUES
(AUTO_ID.nextval, 'MireIRA2');

INSERT INTO conturi(id_cont, username) VALUES
(AUTO_ID.nextval, 'ANTIFCU');


DROP SEQUENCE AUTO_ID;


CREATE SEQUENCE AUTO_ID

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 1000

NOCYCLE;


INSERT INTO contracte_imprumut(id_contract, taxa_intarziere,
durata) VALUES (AUTO_ID.nextval, 50, 30);

INSERT INTO contracte_imprumut(id_contract, taxa_intarziere,
durata) VALUES (AUTO_ID.nextval, 100, 60);

INSERT INTO contracte_imprumut(id_contract, taxa_intarziere,
durata) VALUES (AUTO_ID.nextval, 15, 20);

INSERT INTO contracte_imprumut(id_contract, taxa_intarziere,
durata) VALUES (AUTO_ID.nextval, 150, 60);

INSERT INTO contracte_imprumut(id_contract, taxa_intarziere,
durata) VALUES (AUTO_ID.nextval, 10, 5);


DROP SEQUENCE AUTO_ID;
```

```
CREATE SEQUENCE AUTO_ID
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE;

INSERT INTO bibliotecari VALUES (AUTO_ID.nextval, 1, 'Andrei Gheorghe', 40);
INSERT INTO bibliotecari VALUES (AUTO_ID.nextval, 2, 'Hagi Dobrin', 27);
INSERT INTO bibliotecari VALUES (AUTO_ID.nextval, 3, 'Tiberiu Iordache', 30);
INSERT INTO bibliotecari VALUES (AUTO_ID.nextval, 4, 'Andrei Ivan', 31);
INSERT INTO bibliotecari VALUES (AUTO_ID.nextval, 5, 'Vlad Marian', 38);

DROP SEQUENCE AUTO_ID;

CREATE SEQUENCE AUTO_ID
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE;
```

```
INSERT INTO istoricuri VALUES (AUTO_ID.nextval, 'istoric1');
INSERT INTO istoricuri VALUES (AUTO_ID.nextval, 'istoric2');
INSERT INTO istoricuri VALUES (AUTO_ID.nextval, 'istoric3');
INSERT INTO istoricuri VALUES (AUTO_ID.nextval, 'istoric4');
INSERT INTO istoricuri VALUES (AUTO_ID.nextval, 'istoric5');


DROP SEQUENCE AUTO_ID;


CREATE SEQUENCE AUTO_ID
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE;

INSERT INTO abonamente VALUES (AUTO_ID.nextval, 'premium');
INSERT INTO abonamente VALUES (AUTO_ID.nextval, 'standard');
INSERT INTO abonamente VALUES (AUTO_ID.nextval, 'bronz');
INSERT INTO abonamente VALUES (AUTO_ID.nextval, 'argint');
INSERT INTO abonamente VALUES (AUTO_ID.nextval, 'aur');
```

```
DROP SEQUENCE AUTO_ID;


INSERT INTO creeaza VALUES(1, 1, 1);
INSERT INTO creeaza VALUES(2, 2, 2);
INSERT INTO creeaza VALUES(3, 3, 3);
INSERT INTO creeaza VALUES(4, 4, 4);
INSERT INTO creeaza VALUES(5, 5, 5);
INSERT INTO creeaza VALUES(1, 2, 3);
INSERT INTO creeaza VALUES(2, 3, 1);
INSERT INTO creeaza VALUES(2, 4, 3);
INSERT INTO creeaza VALUES(4, 5, 1);
INSERT INTO creeaza VALUES(1, 2, 5);

INSERT INTO imprumuta_tern VALUES(1234567891234, 1, 1);
INSERT INTO imprumuta_tern VALUES(1234567891234, 2, 2);
INSERT INTO imprumuta_tern VALUES(1234567891234, 2, 5);
INSERT INTO imprumuta_tern VALUES(1234567891234, 3, 3);
INSERT INTO imprumuta_tern VALUES(1234567891234, 3, 4);
INSERT INTO imprumuta_tern VALUES(1234567892123, 3, 3);
INSERT INTO imprumuta_tern VALUES(1234567890508, 1, 2);
INSERT INTO imprumuta_tern VALUES(1234567890508, 1, 4);
INSERT INTO imprumuta_tern VALUES(1234567890508, 3, 1);
INSERT INTO imprumuta_tern VALUES(1234567892123, 5, 5);
INSERT INTO imprumuta_tern VALUES(1234567892123, 5, 3);
```

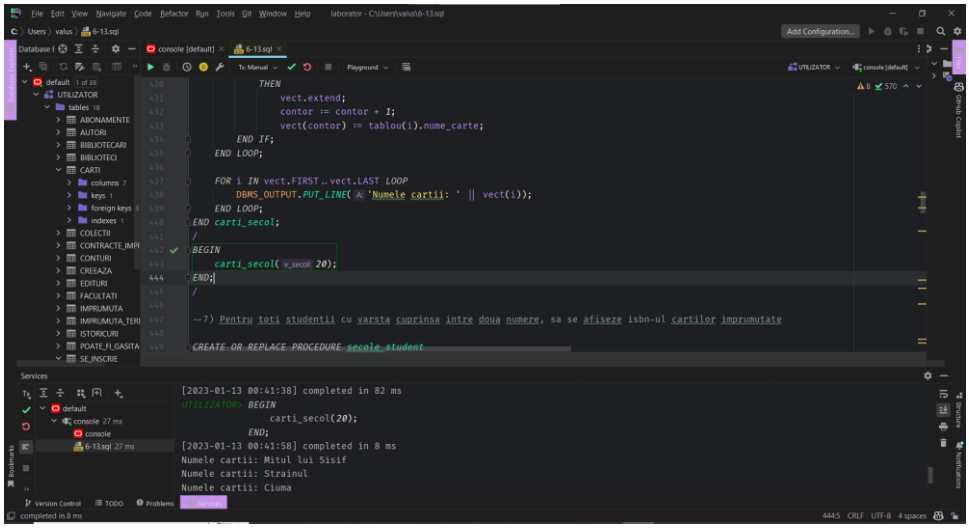
```
INSERT INTO se_inscrie VALUES(1, 1);
INSERT INTO se_inscrie VALUES(2, 2);
INSERT INTO se_inscrie VALUES(3, 3);
INSERT INTO se_inscrie VALUES(4, 4);
INSERT INTO se_inscrie VALUES(5, 5);
INSERT INTO se_inscrie VALUES(1, 2);
INSERT INTO se_inscrie VALUES(2, 1);
INSERT INTO se_inscrie VALUES(3, 1);
INSERT INTO se_inscrie VALUES(4, 2);
INSERT INTO se_inscrie VALUES(5, 2);

INSERT INTO imprumuta VALUES(1, 1234567891234);
INSERT INTO imprumuta VALUES(2, 1234567891209);
INSERT INTO imprumuta VALUES(3, 1234567890508);
INSERT INTO imprumuta VALUES(4, 1234567891278);
INSERT INTO imprumuta VALUES(5, 1234567809008);
INSERT INTO imprumuta VALUES(2, 1234567890028);
INSERT INTO imprumuta VALUES(1, 1234567890028);
INSERT INTO imprumuta VALUES(3, 1234567891235);
INSERT INTO imprumuta VALUES(3, 1234567892198);
INSERT INTO imprumuta VALUES(4, 1234567891234);

INSERT INTO poate_fi_gasita VALUES(1, 1234567891234);
INSERT INTO poate_fi_gasita VALUES(2, 1234567891234);
```

```
INSERT INTO poate_fi_gasita VALUES(3, 1234567809008);
INSERT INTO poate_fi_gasita VALUES(4, 1234567891235);
INSERT INTO poate_fi_gasita VALUES(5, 1234567809008);
INSERT INTO poate_fi_gasita VALUES(2, 1234567809008);
INSERT INTO poate_fi_gasita VALUES(1, 1234567890028);
INSERT INTO poate_fi_gasita VALUES(3, 1234567890028);
INSERT INTO poate_fi_gasita VALUES(4, 1234567890028);
INSERT INTO poate_fi_gasita VALUES(1, 1234567892198);

select * from studenti;

select * from facultati;

select * from autori;

select * from edituri;

select * from colectii;

select * from carti;

select * from biblioteci;

select * from contracte_imprumut;
```

```
select * from conturi;

select * from bibliotecari;

select * from istoricuri;

select * from abonamente;

select * from creeaza;

select * from imprumuta_tern;

select * from se_inscrie;

select * from imprumuta;

select * from poate_fi_gasita;
```

6) Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri de colecție studiate.

– Afisati si salvati titlurile cartilor care au fost publicate intr-un secol dat.



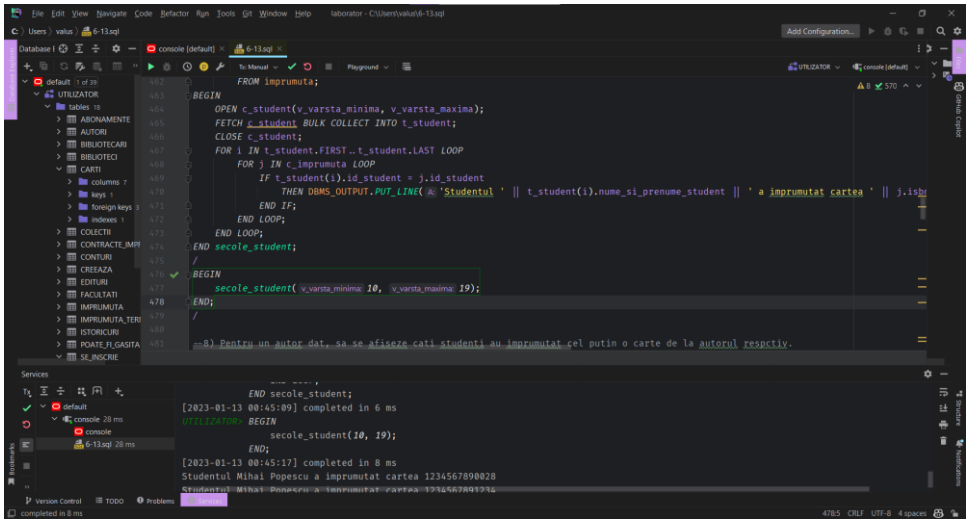
```
CREATE OR REPLACE PROCEDURE carti_secol
    (v_secol NUMBER)
AS
    TYPE tablou_indexat IS TABLE OF Carti%ROWTYPE INDEX BY PLS_INTEGER;
    TYPE vector IS VARRAY(90) OF VARCHAR2(200);
    tablou tablou_indexat;
    vect vector := vector();
    v_carte Carti%ROWTYPE;
    contor NUMBER(2);
BEGIN
    contor := 0;
    select * bulk collect into tablou from carti;
    FOR i IN tablou.first..tablou.last LOOP
        IF SUBSTR(TO_CHAR(tablou(i).an_publicare), 1, 2) =
(TO_CHAR(v_secol-1))
```

```
            THEN
               vect.extend;
               contor := contor + 1;
               vect(contor) := tablou(i).nume_carte;
         END IF;
      END LOOP;

      FOR i IN vect.FIRST..vect.LAST LOOP
         DBMS_OUTPUT.PUT_LINE('Numele cartii: ' || vect(i));
      END LOOP;
   END carti_secol;
   /
   BEGIN
      carti_secol(20);
   END;
   /
```

7) Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care

să utilizeze 2 tipuri de cursoare studiate, unul dintre acestea fiind cursor parametrizat.

– Pentru toti studentii cu varsta cuprinsa intre doua numere, sa se afiseze isbn-ul cartilor imprumutate

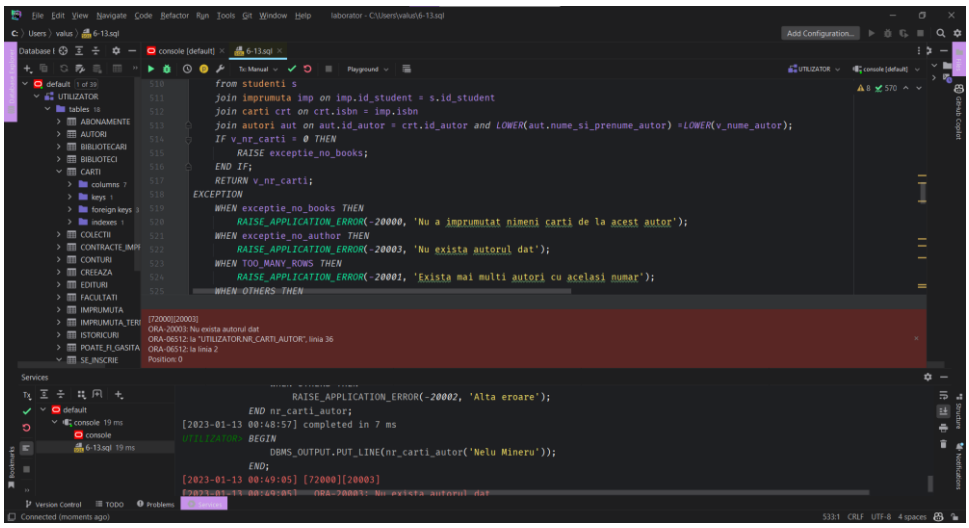


```
CREATE OR REPLACE PROCEDURE secole_student
    (v_varsta_minima NUMBER, v_varsta_maxima NUMBER)
AS
    TYPE tab_student IS TABLE OF studenti%ROWTYPE;
    t_student tab_student;
    cursor c_student(minn NUMBER, maxx NUMBER)
    IS
        select *
        from studenti
        where varsta between minn and maxx;

    cursor c_imprumuta IS
        SELECT *
```

```
        FROM imprumuta;
BEGIN
    OPEN c_student(v_varsta_minima, v_varsta_maxima);
    FETCH c_student BULK COLLECT INTO t_student;
    CLOSE c_student;
    FOR i IN t_student.FIRST..t_student.LAST LOOP
        FOR j IN c_imprumuta LOOP
            IF t_student(i).id_student = j.id_student
                THEN DBMS_OUTPUT.PUT_LINE('Studentul ' ||
t_student(i).nume_si_prenume_student || ' a imprumutat cartea '
|| j.isbn);
            END IF;
        END LOOP;
    END LOOP;
END secole_student;
/
BEGIN
    secole_student(10, 19);
END;
/
```

8) Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții.

– Pentru un autor dat, sa se afiseze cati studenti au imprumutat cel putin o carte de la autorul respctiv.



INSERT INTO autori(id_autor, nume_si_prenume_autor) VALUES (7, 'Jack Kerouac');

INSERT INTO carti(isbn, id_autor, id_editura, id_colectie, nume_carte, gen, an_publicare) VALUES (1234567809777, 7, 3, 4, 'Dharma Bums', 'Fictiune', 1958);

/

CREATE OR REPLACE FUNCTION nr_carti_autor

(v_nume_autor autori.nume_si_prenume_autor%TYPE)

RETURN NUMBER IS

v_nr_carti NUMBER;

ok BOOLEAN;

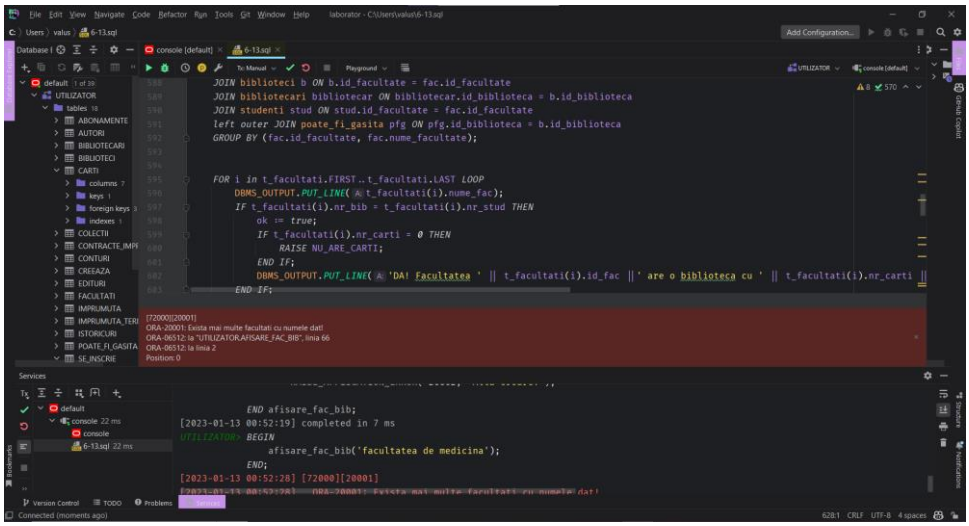
exceptie_no_author EXCEPTION;

```
    exceptie_no_books EXCEPTION;
    TYPE tab_autori IS TABLE OF autori%ROWTYPE INDEX BY
PLS_INTEGER;
    t_autori tab_autori;

BEGIN
    ok := FALSE;
    select * BULK COLLECT INTO t_autori from autori;
    FOR i IN t_autori.FIRST..t_autori.LAST LOOP
        IF t_autori(i).nume_si_prenume_autor = v_nume_autor
            THEN ok := TRUE;
        END IF;
    END LOOP;
    IF ok = FALSE THEN
        RAISE exceptie_no_author;
    END IF;

    select count(s.id_student) into v_nr_carti
    from studenti s
    join imprumuta imp on imp.id_student = s.id_student
    join carti crt on crt.isbn = imp.isbn
    join autori aut on aut.id_autor = crt.id_autor and
LOWER(aut.nume_si_prenume_autor) =LOWER(v_nume_autor);
    IF v_nr_carti = 0 THEN
        RAISE exceptie_no_books;
    END IF;
```

```
    RETURN v_nr_carti;
EXCEPTION
    WHEN exceptie_no_books THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu a imprumutat
nimeni carti de la acest autor');
    WHEN exceptie_no_author THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nu exista autorul dat');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi autori
cu acelasi numar');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END nr_carti_autor;
/ -- bun
BEGIN
    DBMS_OUTPUT.PUT_LINE(nr_carti_autor('Ernest Hemingway'));
END;
/ -- nu exista autorul
BEGIN
    DBMS_OUTPUT.PUT_LINE(nr_carti_autor('Nelu Mineru'));
END;
/ -- nu au fost imprumutate carti de la el
BEGIN
    DBMS_OUTPUT.PUT_LINE(nr_carti_autor('Jack Kerouac'));
END;
/
```

9) Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS.

– Sa se afiseze daca facultatea cu numele introdus are biblioteca cu numarul de bibliotecari egal cu numarul de studenti de la facultate si numarul de carti de la acea biblioteca.




```
CREATE OR REPLACE PROCEDURE afisare_fac_bib
    (v_nume_facultate facultati.nume_facultate%TYPE)
AS
    OK BOOLEAN;
    TYPE t_detalii_facultate IS RECORD(
        id_fac facultati.id_facultate%TYPE,
        nume_fac facultati.nume_facultate%TYPE,
```

```
    nr_bib NUMBER(5),
    nr_stud NUMBER(5),
    nr_carti NUMBER(5)
  );
  detalii_facultate t_detalii_facultate;
  TYPE tabel_facultati IS TABLE OF t_detalii_facultate INDEX BY
PLS_INTEGER;
  t_facultati tabel_facultati;

  TYPE tabel_fac_cu_id IS TABLE OF facultati%ROWTYPE INDEX
BY PLS_INTEGER;
  t_fac_cu_id tabel_fac_cu_id;

  NU_ARE_CARTI EXCEPTION;
  NU_SUNT_CONDITII EXCEPTION;



BEGIN
  ok := false;

  SELECT * BULK COLLECT INTO t_fac_cu_id FROM facultati
WHERE LOWER(nume_facultate) = LOWER(v_nume_facultate);

  IF t_fac_cu_id.count > 1 THEN
    RAISE TOO_MANY_ROWS;
```

```
    END IF;


    IF t_fac_cu_id.count = 0 THEN
        RAISE NO_DATA_FOUND;
    END IF;



    SELECT fac.id_facultate, fac.nume_facultate, COUNT(DISTINCT
bibliotecar.id_bibliotecar), COUNT(DISTINCT stud.id_student),
COUNT(DISTINCT pfg.ISBN) BULK COLLECT INTO t_facultati
    FROM (select * from facultati where LOWER(v_nume_facultate)
= LOWER(nume_facultate)) fac
    JOIN biblioteci b ON b.id_facultate = fac.id_facultate
    JOIN bibliotecari bibliotecar ON bibliotecar.id_biblioteca =
b.id_biblioteca
    JOIN studenti stud ON stud.id_facultate = fac.id_facultate
    left outer JOIN poate_fi_gasita pfg ON pfg.id_biblioteca =
b.id_biblioteca
    GROUP BY (fac.id_facultate, fac.nume_facultate);



    FOR i in t_facultati.FIRST..t_facultati.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(t_facultati(i).nume_fac);
        IF t_facultati(i).nr_bib = t_facultati(i).nr_stud THEN
            ok := true;
            IF t_facultati(i).nr_carti = 0 THEN
                RAISE NU_ARE_CARTI;
```

```
            END IF;
            DBMS_OUTPUT.PUT_LINE('DA! Facultatea ' ||
t_facultati(i).id_fac ||' are o biblioteca cu ' || t_facultati(i).nr_carti
|| ' titluri si are ' || t_facultati(i).nr_bib || ' bibliotecari si ' ||
t_facultati(i).nr_stud || ' studenti!');
        END IF;
    END LOOP;


     IF ok = false THEN
        RAISE NU_SUNT_CONDITII;
    END IF;


EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Facultatea nu exista in
baza noastra de date!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe
facultati cu numele dat!');
    WHEN NU_ARE_CARTI THEN
        RAISE_APPLICATION_ERROR(-20003, 'Facultatea nu are
carti!');
    WHEN NU_SUNT_CONDITII THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nu exista facultati care
sa indeplineasca conditiile!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
```

```
END afisare_fac_bib;
/
BEGIN
   afisare_fac_bib('facultatea de matematica');
END;
/
BEGIN
   afisare_fac_bib('facultatea de medicina');
END;
/
BEGIN
   afisare_fac_bib('facultatea drobeta');
END;
/
BEGIN
   afisare_fac_bib('Facultatea fara carti');
END;

/
BEGIN
   afisare_fac_bib('Facultatea care nu exista');
end;
/
```

INSERT INTO facultati(id_facultate, nume_facultate, durata) VALUES (6, 'Facultatea de Medicina', 6);

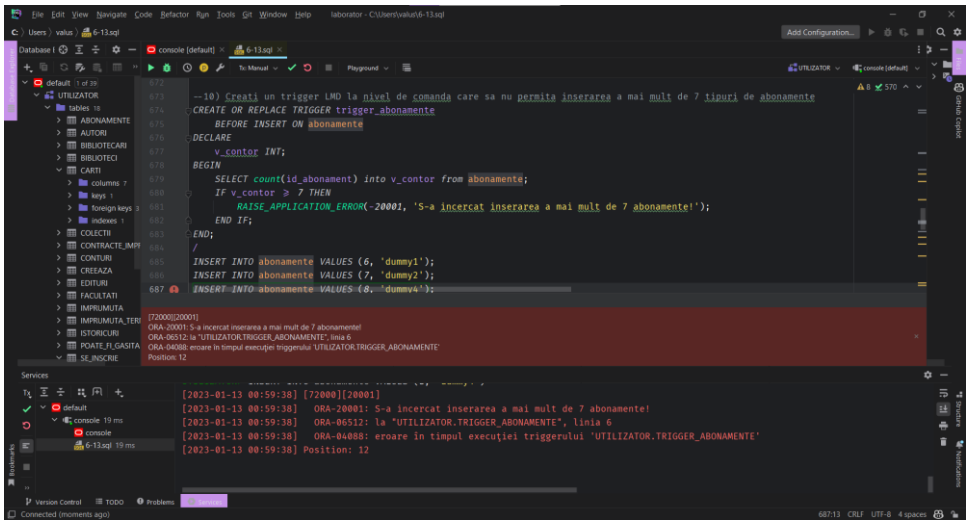
INSERT INTO facultati(id_facultate, nume_facultate, durata) VALUES (7, 'Facultatea fara carti', 6);

INSERT INTO biblioteci(id_biblioteca, id_facultate, nume_biblioteca) VALUES (7, 7, 'Biblioteca fara carti');

INSERT INTO bibliotecari VALUES (6, 7, 'Bib faracarti', 38);

INSERT INTO studenti(id_student, id_facultate, nume_si_prenume_student, varsta) VALUES (6, 3, 'Stud faracarti', 21);

10) Definiți un trigger de tip LMD la nivel de comandă.



-- Creati un trigger LMD la nivel de comanda care sa nu permita inserarea a mai mult de 7 tipuri de abonamente

CREATE OR REPLACE TRIGGER trigger_abonamente

BEFORE INSERT ON abonamente

DECLARE

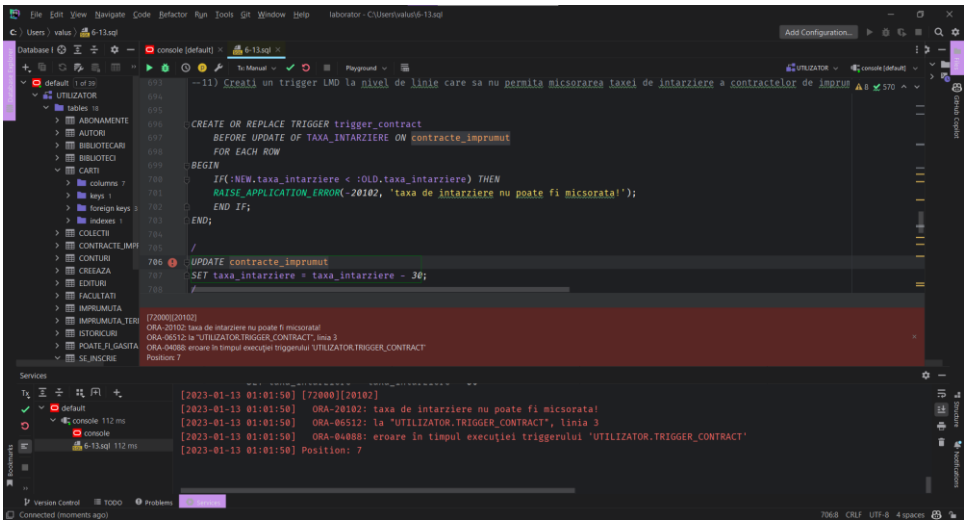
v_contor INT;

BEGIN

```
    SELECT count(id_abonament) into v_contor from abonamente;
    IF v_contor >= 7 THEN
        RAISE_APPLICATION_ERROR(-20001, 'S-a incercat inserarea
a mai mult de 7 abonamente!');
    END IF;
END;
/
INSERT INTO abonamente VALUES (6, 'dummy1');
INSERT INTO abonamente VALUES (7, 'dummy2');
INSERT INTO abonamente VALUES (8, 'dummy4');

delete from abonamente where id_abonament between 6 and 8;
/
```

11) Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

– Creati un trigger LMD la nivel de linie care sa nu permita micsorarea taxei de intarziere a contractelor de imprumut.

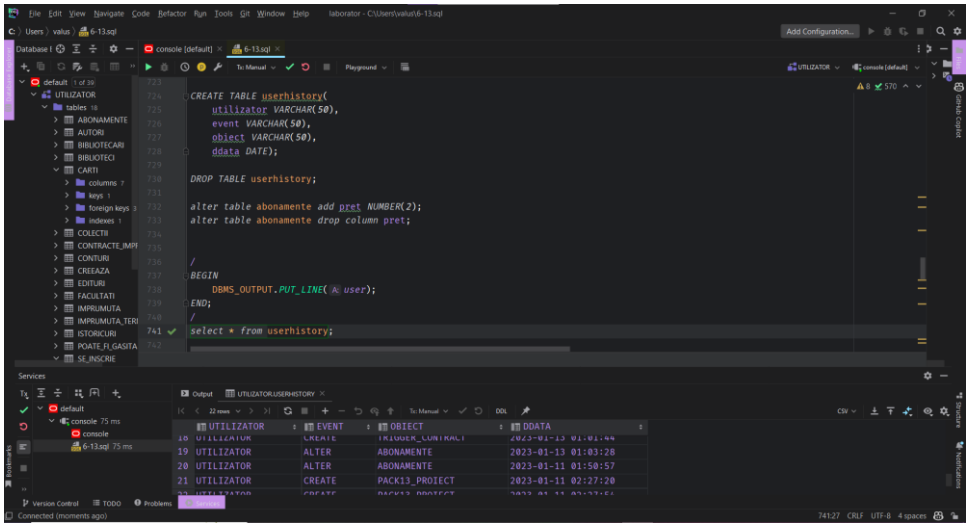


```
CREATE OR REPLACE TRIGGER trigger_contract
   BEFORE UPDATE OF TAXA_INTARZIERE ON contracte_imprumut
   FOR EACH ROW
BEGIN
   IF(:NEW.taxa_intarziere < :OLD.taxa_intarziere) THEN
   RAISE_APPLICATION_ERROR(-20102, 'taxa de intarziere nu
poate fi micsorata!');
   END IF;
END;

/
UPDATE contracte_imprumut
SET taxa_intarziere = taxa_intarziere - 30;
/
```

12) Definiți un trigger de tip LDD. Declanșați trigger-ul.

– Creati un trigger LDD care sa permita alterarea tabelelor doar de catre utilizatorul 'UTILIZATOR'

–Inserati modificarile in tabelul userhistory.




```
CREATE OR REPLACE TRIGGER trigger_user
    BEFORE CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    IF LOWER(USER) != LOWER('UTILIZATOR') THEN
        RAISE_APPLICATION_ERROR(-21000, 'Doar UTILIZATOR poate modifica tabelele!');
    END IF;
    INSERT INTO userhistory VALUES (user, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;
/

CREATE TABLE userhistory(
    utilizator VARCHAR(50),
    event VARCHAR(50),
    obiect VARCHAR(50),
    ddata DATE
```

);

DROP TABLE userhistory;

alter table abonamente add pret NUMBER(2);

/

BEGIN

DBMS_OUTPUT.PUT_LINE(user);

END;

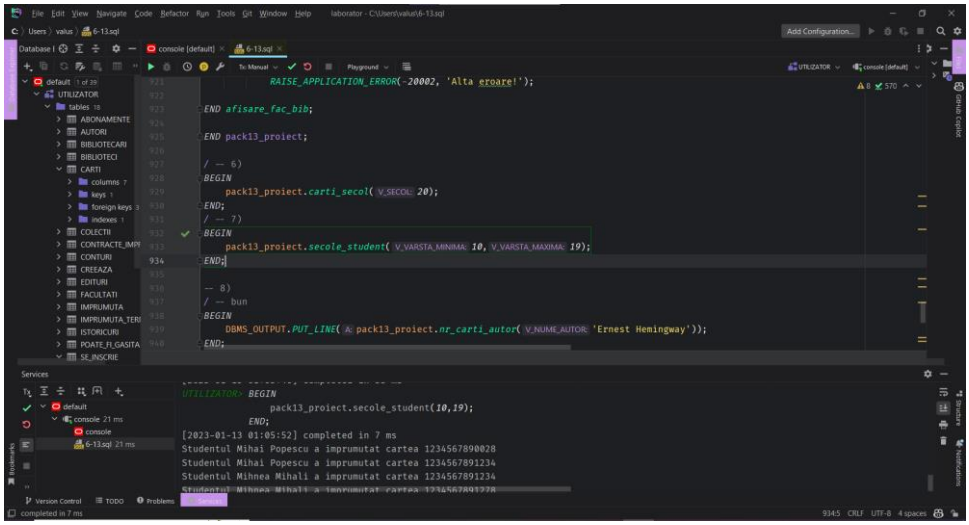
/

select * from userhistory;

/

13) Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

– Pachet 1



```
CREATE OR REPLACE PACKAGE pack13_proiect AS
   PROCEDURE carti_secol(v_secol NUMBER);
   PROCEDURE secole_student(v_varsta_minima NUMBER,
v_varsta_maxima NUMBER);
   FUNCTION nr_carti_autor(v_nume_autor
autori.nume_si_prenume_autor%TYPE) RETURN NUMBER;
   PROCEDURE afisare_fac_bib(v_nume_facultate
facultati.nume_facultate%TYPE);
END pack13_proiect;
/
CREATE OR REPLACE PACKAGE BODY pack13_proiect AS
   PROCEDURE carti_secol
      (v_secol NUMBER)
         AS
            TYPE tablou_indexat IS TABLE OF Carti%ROWTYPE INDEX
BY PLS_INTEGER;
            TYPE vector IS VARRAY(90) OF VARCHAR2(200);
            tablou tablou_indexat;
            vect vector := vector();
            v_carte Carti%ROWTYPE;
            contor NUMBER(2);
         BEGIN
            contor := 0;
            select * bulk collect into tablou from carti;
            FOR i IN tablou.first..tablou.last LOOP
```

```
                IF SUBSTR(TO_CHAR(tablou(i).an_publicare), 1, 2) =
(TO_CHAR(v_secol-1))
                  THEN
                    vect.extend;
                    contor := contor + 1;
                    vect(contor) := tablou(i).nume_carte;
                END IF;
              END LOOP;

              FOR i IN vect.FIRST..vect.LAST LOOP
                DBMS_OUTPUT.PUT_LINE('Numele cartii: ' || vect(i));
              END LOOP;
            END carti_secol;

            PROCEDURE secole_student
        (v_varsta_minima NUMBER, v_varsta_maxima NUMBER)
      AS
        TYPE tab_student IS TABLE OF studenti%ROWTYPE;
        t_student tab_student;
        cursor c_student(minn NUMBER, maxx NUMBER)
        IS
          select *
          from studenti
          where varsta between minn and maxx;

        cursor c_imprumuta IS
```

```
        SELECT *
        FROM imprumuta;
    BEGIN
      OPEN c_student(v_varsta_minima, v_varsta_maxima);
      FETCH c_student BULK COLLECT INTO t_student;
      CLOSE c_student;
      FOR i IN t_student.FIRST..t_student.LAST LOOP
        FOR j IN c_imprumuta LOOP
          IF t_student(i).id_student = j.id_student
            THEN DBMS_OUTPUT.PUT_LINE('Studentul ' ||
t_student(i).nume_si_prenume_student || ' a imprumutat cartea '
|| j.isbn);
          END IF;
        END LOOP;
      END LOOP;
    END secole_student;

    FUNCTION nr_carti_autor
      (v_nume_autor autori.nume_si_prenume_autor%TYPE)
    RETURN NUMBER IS
      v_nr_carti NUMBER;
      ok BOOLEAN;
      exceptie_no_author EXCEPTION;
      exceptie_no_books EXCEPTION;
      TYPE tab_autori IS TABLE OF autori%ROWTYPE INDEX BY
PLS_INTEGER;
```

```
    t_autori tab_autori;

  BEGIN
    ok := FALSE;
    select * BULK COLLECT INTO t_autori from autori;
    FOR i IN t_autori.FIRST..t_autori.LAST LOOP
      IF t_autori(i).nume_si_prenume_autor = v_nume_autor
        THEN ok := TRUE;
      END IF;
    END LOOP;
    IF ok = FALSE THEN
      RAISE exceptie_no_author;
    END IF;

    select count(s.id_student) into v_nr_carti
    from studenti s
    join imprumuta imp on imp.id_student = s.id_student
    join carti crt on crt.isbn = imp.isbn
    join autori aut on aut.id_autor = crt.id_autor and
LOWER(aut.nume_si_prenume_autor) =LOWER(v_nume_autor);
    IF v_nr_carti = 0 THEN
      RAISE exceptie_no_books;
    END IF;
    RETURN v_nr_carti;
  EXCEPTION
    WHEN exceptie_no_books THEN
```

```
            RAISE_APPLICATION_ERROR(-20000, 'Nu a imprumutat
nimeni carti de la acest autor');
        WHEN exceptie_no_author THEN
            RAISE_APPLICATION_ERROR(-20003, 'Nu exista autorul
dat');
        WHEN TOO_MANY_ROWS THEN
            RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi
autori cu acelasi numar');
        WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-20002, 'Alta eroare');
END nr_carti_autor;


    PROCEDURE afisare_fac_bib
        (v_nume_facultate facultati.nume_facultate%TYPE)
    AS
        OK BOOLEAN;
        TYPE t_detalii_facultate IS RECORD(
            id_fac facultati.id_facultate%TYPE,
            nume_fac facultati.nume_facultate%TYPE,
            nr_bib NUMBER(5),
            nr_stud NUMBER(5),
            nr_carti NUMBER(5)
        );
        detalii_facultate t_detalii_facultate;
        TYPE tabel_facultati IS TABLE OF t_detalii_facultate INDEX BY
PLS_INTEGER;
```

```
    t_facultati tabel_facultati;

    TYPE tabel_fac_cu_id IS TABLE OF facultati%ROWTYPE INDEX
BY PLS_INTEGER;
    t_fac_cu_id tabel_fac_cu_id;

    NU_ARE_CARTI EXCEPTION;
    NU_SUNT_CONDITII EXCEPTION;



  BEGIN
    ok := false;

    SELECT * BULK COLLECT INTO t_fac_cu_id FROM facultati
WHERE LOWER(nume_facultate) = LOWER(v_nume_facultate);

    IF t_fac_cu_id.count > 1 THEN
      RAISE TOO_MANY_ROWS;
    END IF;

    IF t_fac_cu_id.count = 0 THEN
      RAISE NO_DATA_FOUND;
    END IF;



    SELECT fac.id_facultate, fac.nume_facultate,
COUNT(DISTINCT bibliotecar.id_bibliotecar), COUNT(DISTINCT
```

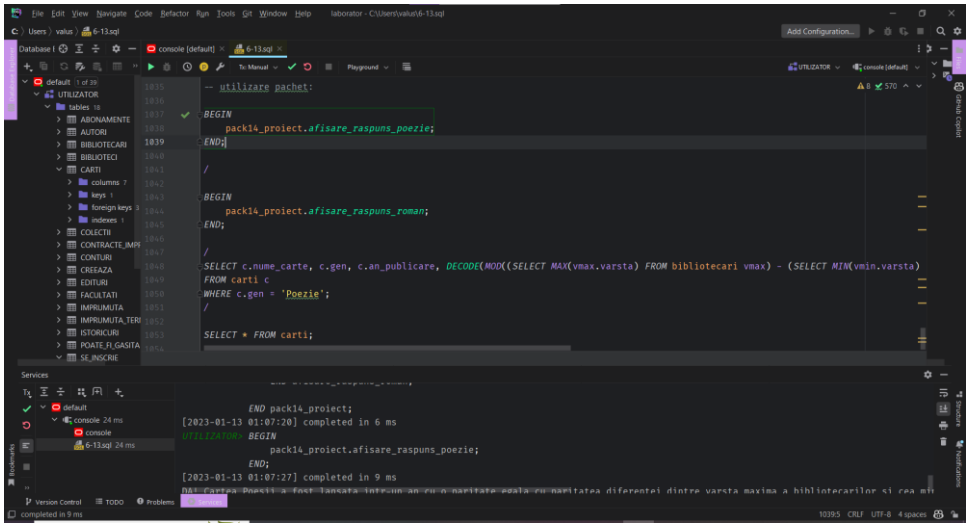
```
stud.id_student), COUNT(DISTINCT pfg.ISBN) BULK COLLECT INTO
t_facultati
        FROM (select * from facultati where
LOWER(v_nume_facultate) = LOWER(nume_facultate)) fac
        JOIN biblioteci b ON b.id_facultate = fac.id_facultate
        JOIN bibliotecari bibliotecar ON bibliotecar.id_biblioteca =
b.id_biblioteca
        JOIN studenti stud ON stud.id_facultate = fac.id_facultate
        left outer JOIN poate_fi_gasita pfg ON pfg.id_biblioteca =
b.id_biblioteca
        GROUP BY (fac.id_facultate, fac.nume_facultate);



        FOR i in t_facultati.FIRST..t_facultati.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(t_facultati(i).nume_fac);
            IF t_facultati(i).nr_bib = t_facultati(i).nr_stud THEN
                ok := true;
                IF t_facultati(i).nr_carti = 0 THEN
                    RAISE NU_ARE_CARTI;
                END IF;
                DBMS_OUTPUT.PUT_LINE('DA! Facultatea ' ||
t_facultati(i).id_fac ||' are o biblioteca cu ' || t_facultati(i).nr_carti
|| ' titluri si are ' || t_facultati(i).nr_bib || ' bibliotecari si ' ||
t_facultati(i).nr_stud || ' studenti!');
            END IF;
        END LOOP;


         IF ok = false THEN
```

```
            RAISE NU_SUNT_CONDITII;
        END IF;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20000, 'Facultatea nu exista
in baza noastra de date!');
        WHEN TOO_MANY_ROWS THEN
            RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe
facultati cu numele dat!');
        WHEN NU_ARE_CARTI THEN
            RAISE_APPLICATION_ERROR(-20003, 'Facultatea nu are
carti!');
        WHEN NU_SUNT_CONDITII THEN
            RAISE_APPLICATION_ERROR(-20004, 'Nu exista facultati
care sa indeplineasca conditiile!');
        WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');

END afisare_fac_bib;

END pack13_proiect;

/ -- 6)
BEGIN
    pack13_proiect.carti_secol(20);
END;
```

```
/ -- 7)
BEGIN
   pack13_proiect.secole_student(10,19);
END;

-- 8)
/ -- bun
BEGIN
   DBMS_OUTPUT.PUT_LINE(pack13_proiect.nr_carti_autor('Ernest
Hemingway'));
END;
/ -- nu exista autorul
BEGIN
   DBMS_OUTPUT.PUT_LINE(pack13_proiect.nr_carti_autor('Nelu
Mineru'));
END;
/ -- nu au fost imprumutate carti de la el
BEGIN
   DBMS_OUTPUT.PUT_LINE(pack13_proiect.nr_carti_autor('Jack
Kerouac'));
END;

-- 9)
/ -- Nu indeplineste conditiile
BEGIN
   pack13_proiect.afisare_fac_bib('facultatea de matematica');
```

```
END;
/ -- Mai multe facultati cu acelasi nume
BEGIN
   pack13_proiect.afisare_fac_bib('facultatea de medicina');
END;
/-- Bun
BEGIN
   pack13_proiect.afisare_fac_bib('facultatea drobeta');
END;
/ -- Facultatea nu are carti
BEGIN
   pack13_proiect.afisare_fac_bib('Facultatea fara carti');
END;

/ -- Facultatea nu exista in baza de date
BEGIN
   pack13_proiect.afisare_fac_bib('Facultatea care nu exista');
end;
/
```

14) Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).



-- Pentru cartile de gen 'Poezie' sa se afiseze 'da' daca cartea s-a lansat intr-un an cu o paritate egala cu paritatea diferentei dintre varsta maxima a bibliotecarilor si cea minima a studentilor si 'nu' daca nu e egala.

-- Iar pentru cele de gen 'Fictiune' sa se afiseze 'nu' daca cartea s-a lansat intr-un an cu o paritate egala cu paritatea diferentei dintre varsta maxima a bibliotecarilor si cea minima a studentilor si 'da' daca nu e egala.

```
CREATE OR REPLACE PACKAGE pack14_proiect AS
    paritate_diferenta BOOLEAN;
    TYPE tabel_poezii IS TABLE OF carti%ROWTYPE;
    CURSOR c_romane IS SELECT * FROM carti WHERE gen =
'Fictiune';

    FUNCTION paritate_diferenta_varsta RETURN NUMBER;
    FUNCTION paritate_an(v_carte carti%ROWTYPE) RETURN
BOOLEAN;
    PROCEDURE afisare_raspuns_poezie;
    PROCEDURE afisare_raspuns_roman;
```

```
END pack14_proiect;

CREATE OR REPLACE PACKAGE BODY pack14_proiect AS
   FUNCTION paritate_diferenta_varsta RETURN NUMBER IS
      v_max_bib NUMBER;
      v_min_stud NUMBER;
   BEGIN
      SELECT MAX(varsta) INTO v_max_bib FROM bibliotecari;
      SELECT MIN(varsta) INTO v_min_stud FROM studenti;
      RETURN MOD(v_max_bib - v_min_stud, 2);
   END paritate_diferenta_varsta;

   FUNCTION paritate_an(v_carte carti%ROWTYPE) RETURN
BOOLEAN IS
   BEGIN
      RETURN MOD(v_carte.AN_PUBLICARE, 2) =
paritate_diferenta_varsta;
   END paritate_an;

   PROCEDURE afisare_raspuns_poezie IS
      v_carte carti%ROWTYPE;
      t_poezii tabel_poezii;
   BEGIN
      SELECT * BULK COLLECT INTO t_poezii FROM carti WHERE
gen = 'Poezie';

      FOR i IN t_poezii.FIRST..t_poezii.LAST LOOP
```

```
        IF paritate_an(t_poezii(i)) THEN
            DBMS_OUTPUT.PUT_LINE('DA! Cartea ' ||
t_poezii(i).NUME_CARTE || ' a fost lansata intr-un an cu o paritate
egala cu paritatea diferentei dintre varsta maxima a bibliotecarilor
si cea minima a studentilor!');
        ELSE
            DBMS_OUTPUT.PUT_LINE('NU! Cartea ' ||
t_poezii(i).NUME_CARTE || ' nu a fost lansata intr-un an cu o
paritate egala cu paritatea diferentei dintre varsta maxima a
bibliotecarilor si cea minima a studentilor!');
        END IF;
    END LOOP;

END afisare_raspuns_poezie;

PROCEDURE afisare_raspuns_roman IS
    v_carte carti%ROWTYPE;

BEGIN
    FOR i IN c_romane LOOP
        IF paritate_an(i) = false THEN
            DBMS_OUTPUT.PUT_LINE('DA! Cartea ' || i.NUME_CARTE
|| ' nu a fost lansata intr-un an cu o paritate egala cu paritatea
diferentei dintre varsta maxima a bibliotecarilor si cea minima a
studentilor!');
        ELSE
            DBMS_OUTPUT.PUT_LINE('NU! Cartea ' || i.NUME_CARTE
|| ' a fost lansata intr-un an cu o paritate egala cu paritatea
```

```
diferentei dintre varsta maxima a bibliotecarilor si cea minima a
studentilor!');
            END IF;
        END LOOP;

    END afisare_raspuns_roman;

END pack14_proiect;
/
-- utilizare pachet:

BEGIN
    pack14_proiect.afisare_raspuns_poezie;
END;

/

BEGIN
    pack14_proiect.afisare_raspuns_roman;
END;

/
```

Repository pentru cod: https://github.com/valentinvale/ProiectSGBD