Baza de date a unei biblioteci

[1.Utilitate bazei de date](#_fo6dusfjsrfw)

[2.Diagrama entitate-relatie](#_93yi9nqfuga1)

[3.Diagrama conceptuala](#_sbvv38ki0o9t)

[4.Model implementat in Oracle (folosind DataGrip)](#_3sqvxaxzgog2)

[5.Popularea bazei de date](#_sh65qr8c1gr8)

[6.Subprogram + colectii](#_1uq018hwfpgw)

[7.Subprogram + cursoare](#_r499xw5a54xo)

[8.Functie + exceptii](#_3qm5fjwubi08)

[9.Procedura + exceptii + 5 tabele](#_98m2iyifk453)

[10.Trigger LMD la nivel de comanda](#_3j6r04bucwtq)

[11.Trigger LMD la nivel de linie](#_3ofzb5i6psd0)

[12.Trigger LDD](#_dmz7epwpme16)

[13. Definire Pachet](#_garftifak178)

[14. Definire Pachet pentru a facilitate functionalitate](#_78kuuzi4p7vi)

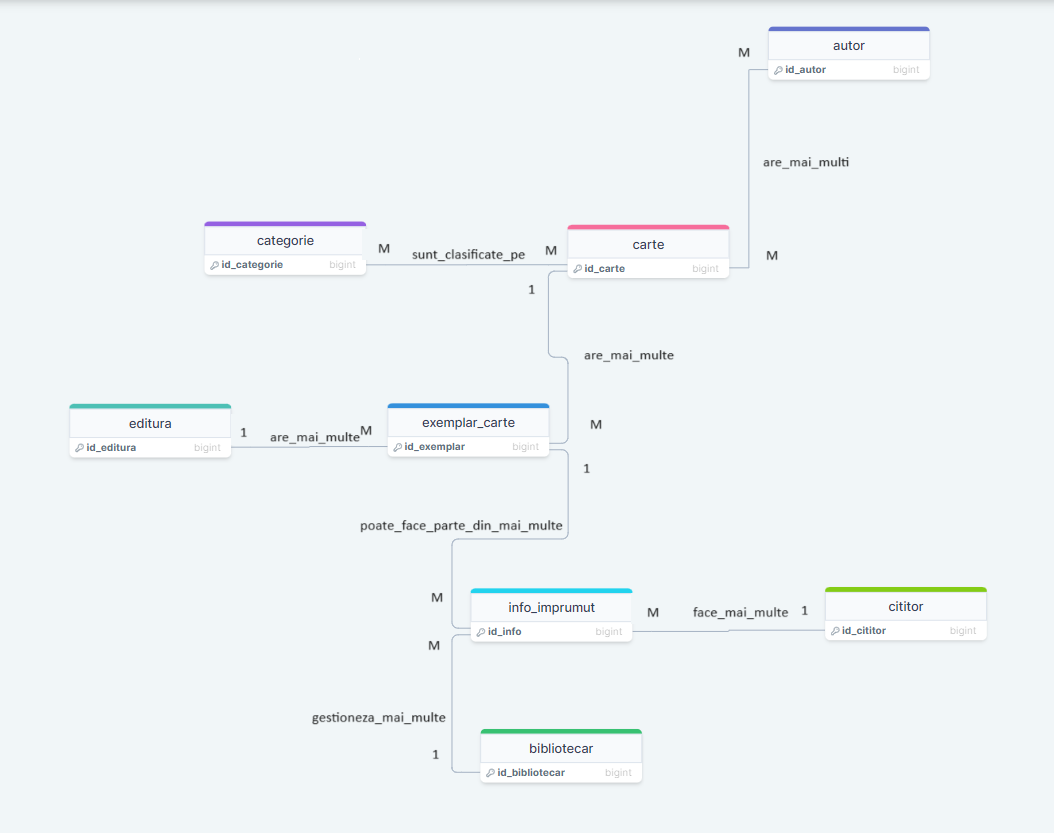
## **1.Utilitate bazei de date**

Biblioteca este o instituție care oferă acces la o varietate de surse de informații, cum ar fi cărți, reviste, publicații periodice, filme, muzică și alte materiale media. Scopul bibliotecii este de a permite utilizatorilor sa împrumute aceste resurse pentru a-si extinde cunoștințele și pentru a se bucura de lucrurile pe care le oferă.

Bibliotecile pot varia între mărime și scop, de la biblioteci locale la biblioteci universitare sau naționale. În general, bibliotecile îsi organizeaza colecțiile în funcție de subiect, autor sau editura.

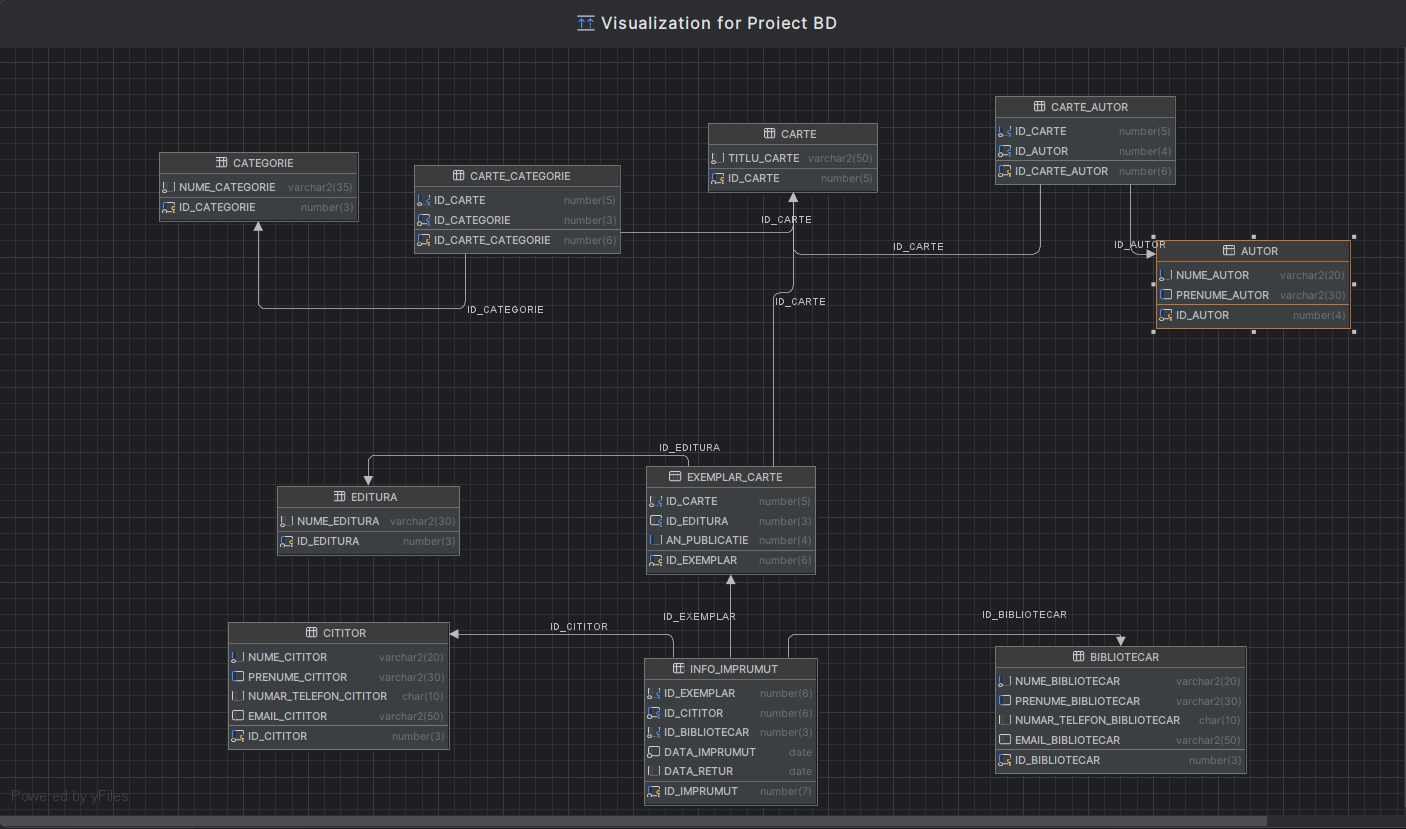
Scopul bazei de date este de a stoca eficient, fara redundanta datele necesare pentru buna functioanare a unei instituii precum o biblioteca. Aceasta implica stocarea cartilor si a particularatilor acestora(editura, an, autori) existente in biblioteca, cat si a imprumuturilor, citiorilor care sunt abonati la biblioteca si a bibliotecarilor.

## **2.Diagrama entitate-relatie**

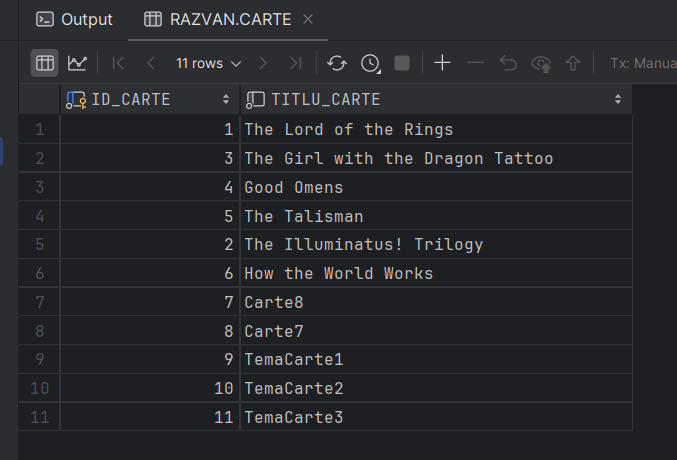


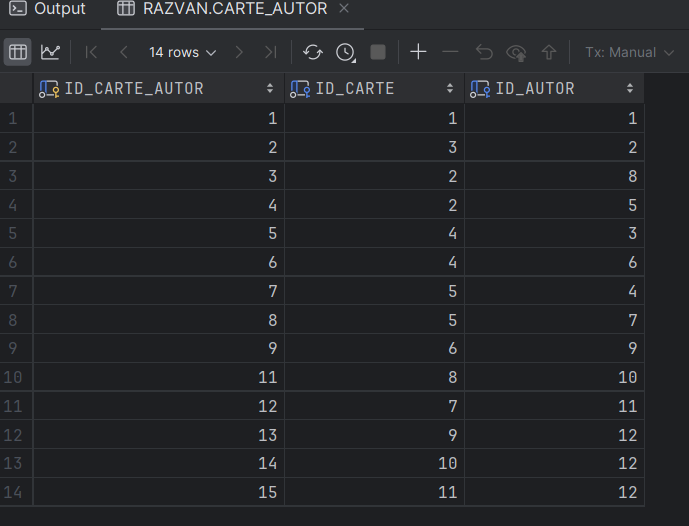
## **3.Diagrama conceptuala**

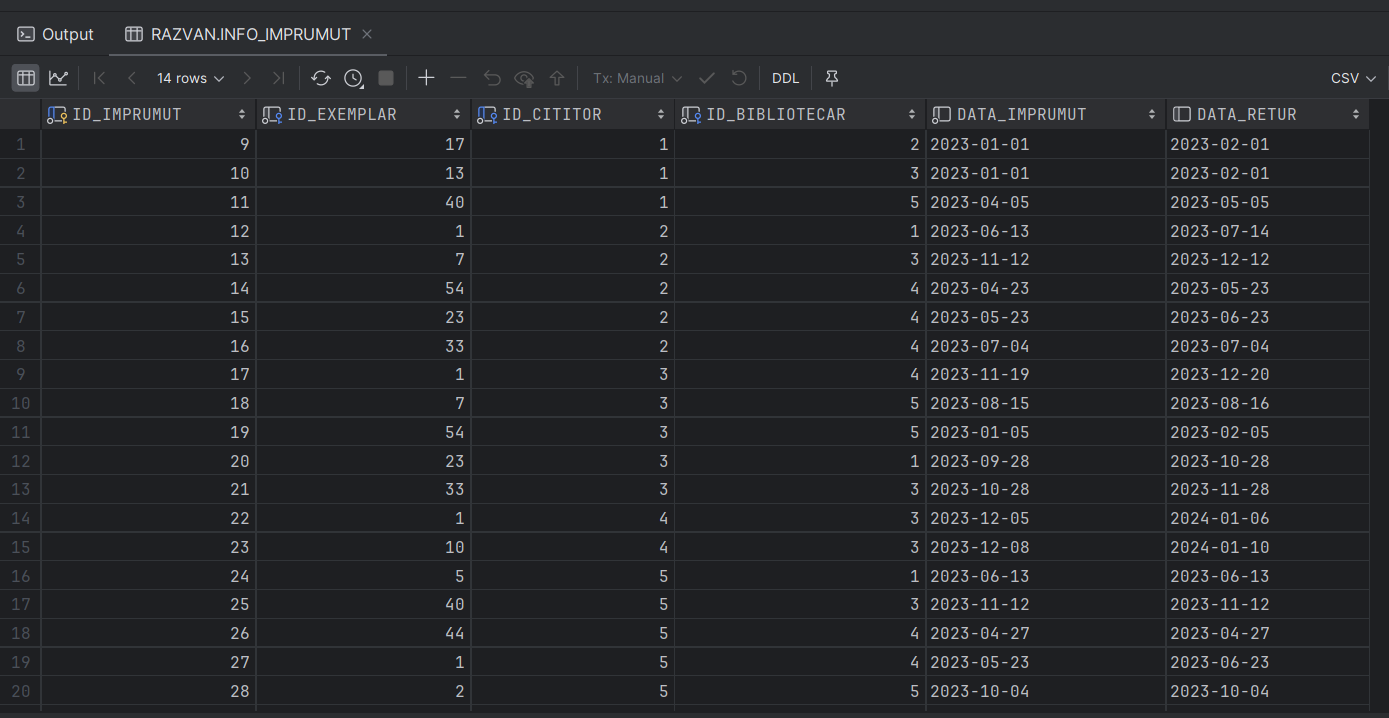
## **4.Model implementat in Oracle (folosind DataGrip)**



## **5.Popularea bazei de date**







## **6.Subprogram + colectii**

Se selecteaza primele 5 carti. Pentru fiecare carte se alege un exemplar. La final se va afisa pe ecran dupa urmatorul sablon:

titlu carte 1 : an exemplar ales

titlu carte 2 : an exemplar ales

.

.

.

titlu carte n : an exemplar ales

Rezolvare:

CREATE OR REPLACE PROCEDURE *afis\_carti*

IS

-- declarare colectii

TYPE nume\_exemplar IS TABLE OF EXEMPLAR\_CARTE%ROWTYPE INDEX BY VARCHAR2(50);

TYPE index\_exemplar IS TABLE OF EXEMPLAR\_CARTE%ROWTYPE;

TYPE id\_cinci IS VARRAY(5) OF CARTE.ID\_CARTE%TYPE;

-- declarare variabile

c\_carte\_exemplare nume\_exemplar;

c\_exemplare index\_exemplar := index\_exemplar();

c\_id\_carti id\_cinci := id\_cinci();

-- variabile intermediare

v\_titlu\_curent CARTE.TITLU\_CARTE%TYPE;

v\_iterator VARCHAR2(50);

BEGIN

-- se selcteaza primele 5 id uri

select ID\_CARTE

BULK COLLECT INTO c\_id\_carti

from CARTE

where ROWNUM<6

order by ID\_CARTE;

FOR i in c\_id\_carti.first..c\_id\_carti.last LOOP

c\_exemplare.extend;

-- se selecteaza un exemplar si se adauga in table

select \*

into c\_exemplare(c\_exemplare.last)

from EXEMPLAR\_CARTE

where ID\_CARTE = c\_id\_carti(i) AND ROWNUM<2;

-- se salveaza titlul cartii curente in variabila

select TITLU\_CARTE

into v\_titlu\_curent

from CARTE

where ID\_CARTE = c\_id\_carti(i);

-- se insereaza in dictionar

c\_carte\_exemplare(v\_titlu\_curent) := c\_exemplare(c\_exemplare.LAST);

END LOOP;

-- afisare

-- se itereaza folosind o variabila de tipul varchar

v\_iterator := c\_carte\_exemplare.FIRST;

WHILE v\_iterator IS NOT NULL LOOP

DBMS\_OUTPUT.*PUT\_LINE*(v\_iterator || ' : ' || c\_carte\_exemplare(v\_iterator).AN\_PUBLICATIE);

v\_iterator := c\_carte\_exemplare.*NEXT*(v\_iterator);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.*PUT\_LINE*('Nu exista date!');

WHEN OTHERS THEN DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');

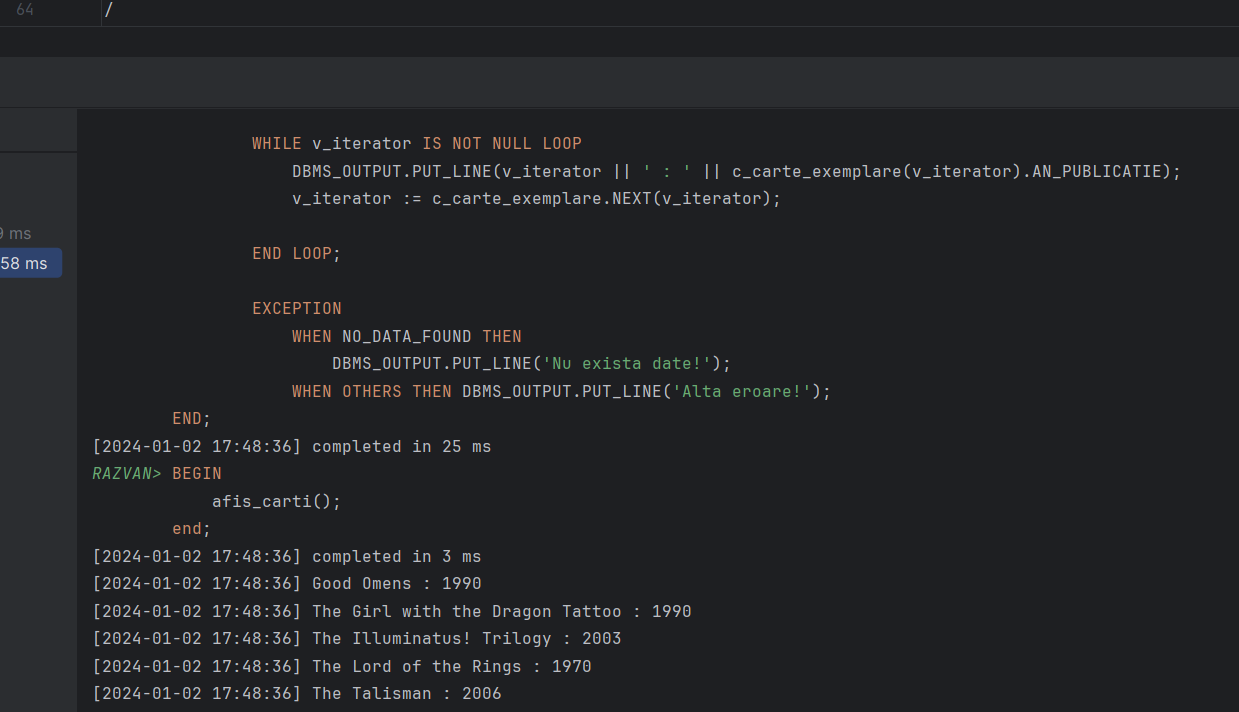
END;

/

BEGIN

*afis\_carti*();end;

Cod rulat:



## **7.Subprogram + cursoare**

Pentru o carte (nu exemplar) data prin id-ul sau sa se afiseze data de imprumut si data de retur a tuturor imprumuturilor care au data de expirare mai mica decat data curenta.

Rezolvare:

CREATE OR REPLACE PROCEDURE *imprumuturi\_carte* (v\_id\_carte\_curenta CARTE.ID\_CARTE%TYPE)

IS

-- folosit pentru a tine numararea de imprumuturi

v\_index NUMBER :=0;

-- cursor pentru a selecta exemplarele cu id\_ul de carte dat

CURSOR C\_CARTI (v\_id\_carte\_selectata CARTE.ID\_CARTE%TYPE) IS

SELECT ID\_EXEMPLAR

FROM EXEMPLAR\_CARTE

WHERE ID\_CARTE = v\_id\_carte\_selectata;

BEGIN

-- iterare prin cursor

FOR EXEMPLAR IN C\_CARTI(v\_id\_carte\_curenta)

LOOP

--Cursor cu subcerere pentru a obtine imprumuturile cerute

FOR IMPRUMUT IN (SELECT \*

FROM INFO\_IMPRUMUT I\_F

WHERE EXEMPLAR.ID\_EXEMPLAR = I\_F.ID\_EXEMPLAR AND I\_F.DATA\_IMPRUMUT < *SYSDATE*)

LOOP

v\_index:= v\_index+1;

DBMS\_OUTPUT.*PUT\_LINE*('Imprumut ' || v\_index);

DBMS\_OUTPUT.*PUT\_LINE*('Data imprumut: ' || IMPRUMUT.DATA\_IMPRUMUT);

DBMS\_OUTPUT.*PUT\_LINE*('Data retur: ' || IMPRUMUT.DATA\_RETUR);

END LOOP;

END LOOP;

END;

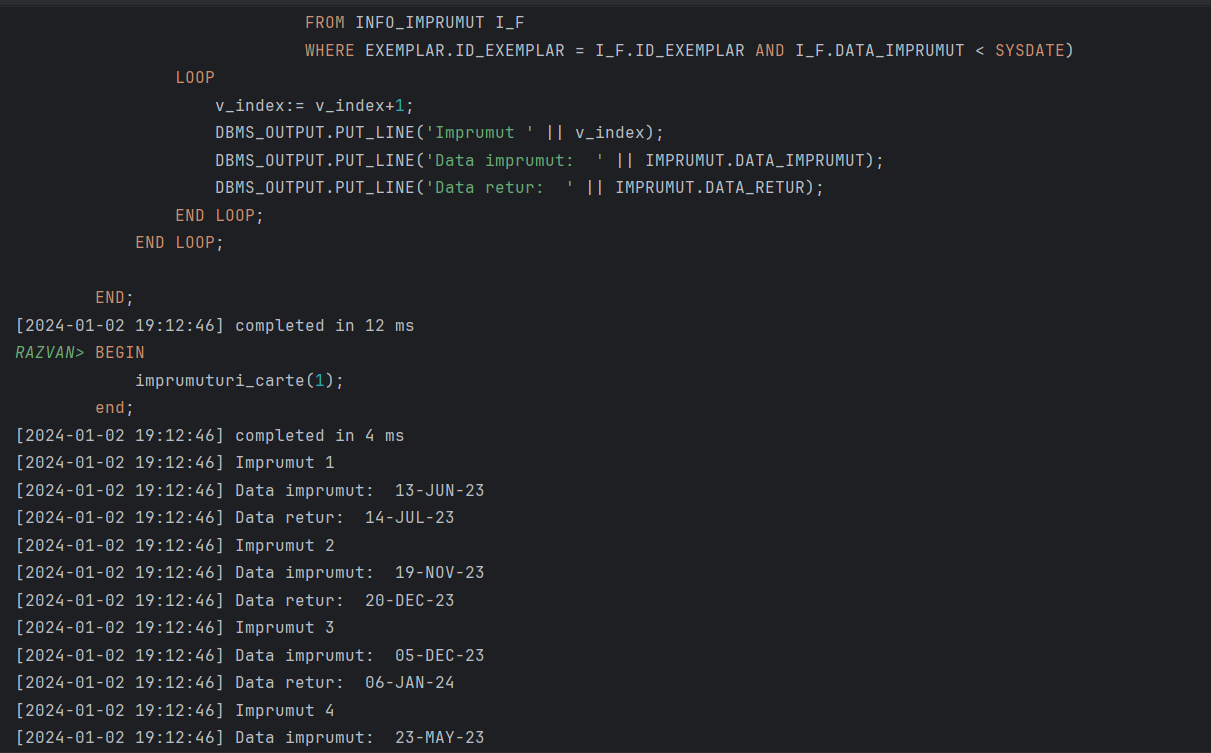
/

BEGIN

*imprumuturi\_carte*(1);

END;

Cod rulat:



## **8.Functie + exceptii**

Sa se creeze o functie care ia ca parametru 2 date : data de imprumut si data de retur. Functia returneaza numarul de carti care au fost imprumutate intre cele 2 dati si care trebuie returnate pana in data de retur. Se vor trata cu exceptii cazurile in care nu exista nicio carte , sau cand datile nu sunt valide. Prin numarul de carti ne referim la carti cu acelasi titlu nu la exemplare.

Rezolvare:

CREATE OR REPLACE FUNCTION *NUMAR\_CARTI*(string\_data\_imprumut VARCHAR2,string\_data\_retur VARCHAR2) RETURN NUMBER IS

INVALID\_DATA EXCEPTION;

WRONG\_DATA\_FORMAT EXCEPTION;

v\_data\_imprumut DATE;

v\_data\_retur DATE;

v\_numar\_carti NUMBER :=0;

BEGIN

-- Convertire date si tratare exceptii

IF *LENGTH*(string\_data\_imprumut) <> 10 OR *LENGTH*(string\_data\_retur) <> 10 THEN

RAISE WRONG\_DATA\_FORMAT;

END IF;

v\_data\_imprumut := *TRUNC*(*to\_date*(string\_data\_imprumut,'DD-MM-YYYY'));

v\_data\_retur := *TRUNC*(*to\_date*(string\_data\_retur,'DD-MM-YYYY'));

IF v\_data\_retur <= v\_data\_imprumut THEN

RAISE INVALID\_DATA;

END IF;

-- se selecteaza numarul de carti distincte

SELECT *COUNT*(DISTINCT(ID\_CARTE))

INTO v\_numar\_carti

FROM CARTE, (SELECT ID\_EXEMPLAR AS ID

FROM INFO\_IMPRUMUT

WHERE *TRUNC*(DATA\_RETUR)<= v\_data\_retur AND *TRUNC*(DATA\_IMPRUMUT)>= v\_data\_imprumut) DATA

WHERE DATA.ID IN ( SELECT C\_E.ID\_EXEMPLAR

FROM EXEMPLAR\_CARTE C\_E

WHERE ID\_CARTE = C\_E.ID\_CARTE);

return v\_numar\_carti;

EXCEPTION

WHEN INVALID\_DATA THEN

DBMS\_OUTPUT.*PUT\_LINE*('Invalid Data');

return -1;

WHEN WRONG\_DATA\_FORMAT THEN

DBMS\_OUTPUT.*PUT\_LINE*('Wrong format');

return -1;

END;

/

declare

i number;

begin

-- ca cand subprogramul ruleaza

i := *NUMAR\_CARTI*('01-01-2022','01-02-2024');

DBMS\_OUTPUT.*PUT\_LINE*(i);

-- caz data invalida

i:= *NUMAR\_CARTI*('01-01-2022','01-02-2021');

DBMS\_OUTPUT.*PUT\_LINE*(i);

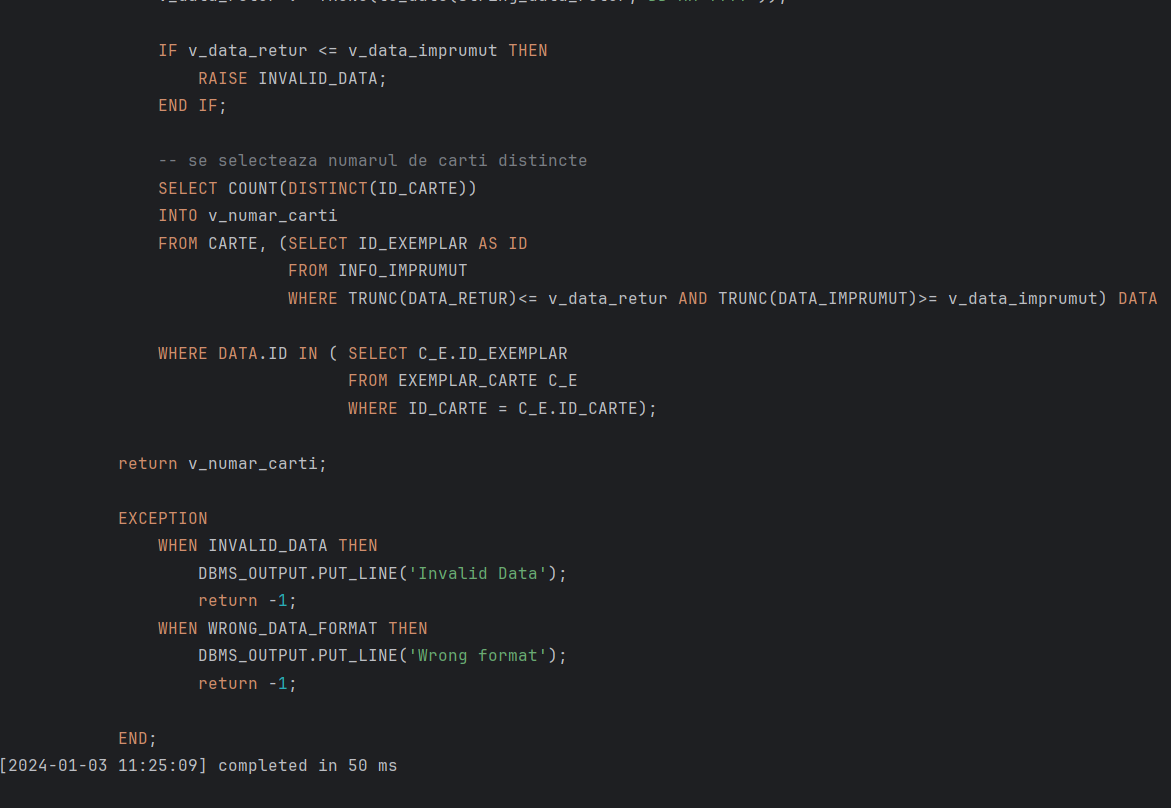
-- caz formatare gresita

i:= *NUMAR\_CARTI*('1-01-2022','01-02-20212');

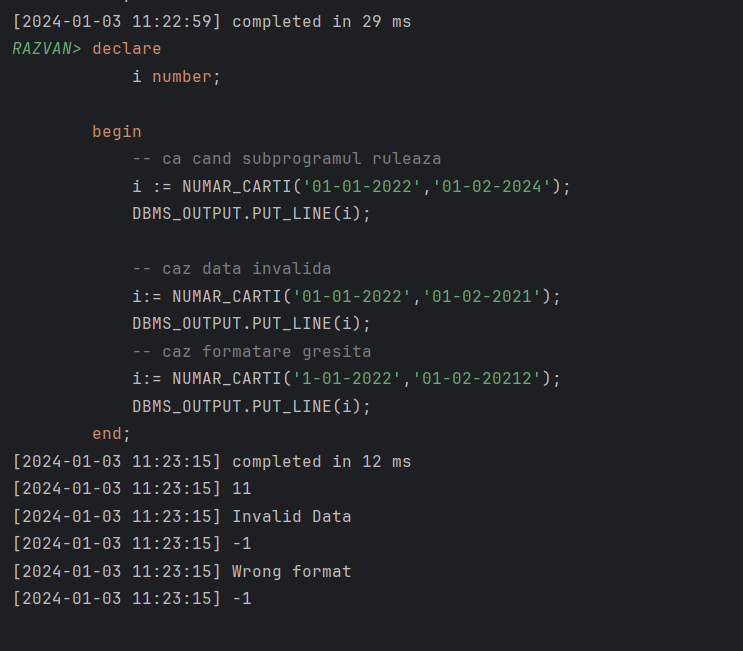
DBMS\_OUTPUT.*PUT\_LINE*(i);

end;

Rulare procedura:



Rulare cod:



## **9.Procedura + exceptii + 5 tabele**

Sa se afiseze categoria pentru care exista exact N carti imprumutate cu categoria respectiva. Se vor trata cazurile in care exista mai multe astfel de categorii sau daca nu exista o astfel de categorie.

Rezolvare:

CREATE OR REPLACE PROCEDURE *FIND\_CATEG* (v\_numar\_carti NUMBER) IS

-- declarare variabile

v\_denumire\_categorie CATEGORIE.NUME\_CATEGORIE%TYPE;

v\_id\_categorie CATEGORIE.ID\_CATEGORIE%TYPE;

v\_count NUMBER := 0;

BEGIN

-- pentru fiecare id de categorie se numara cate carti exista

-- se mai apeleaza count pentru a determina cate categorii cu numarul respectiv exista

SELECT *count*(*count*(\*))

INTO v\_count

FROM CATEGORIE JOIN CARTE\_CATEGORIE ON CATEGORIE.ID\_CATEGORIE = CARTE\_CATEGORIE.ID\_CATEGORIE

JOIN CARTE ON CARTE\_CATEGORIE.ID\_CARTE = CARTE.ID\_CARTE

JOIN EXEMPLAR\_CARTE ON CARTE.ID\_CARTE = EXEMPLAR\_CARTE.ID\_CARTE

JOIN INFO\_IMPRUMUT ON EXEMPLAR\_CARTE.ID\_EXEMPLAR = INFO\_IMPRUMUT.ID\_EXEMPLAR

GROUP BY CATEGORIE.ID\_CATEGORIE

HAVING *COUNT*(CATEGORIE.ID\_CATEGORIE) = v\_numar\_carti;

IF v\_count>1 THEN

RAISE TOO\_MANY\_ROWS;

END IF;

SELECT CATEGORIE.ID\_CATEGORIE,CATEGORIE.NUME\_CATEGORIE

INTO v\_id\_categorie,v\_denumire\_categorie

FROM CATEGORIE WHERE CATEGORIE.ID\_CATEGORIE = (

SELECT CATEGORIE.ID\_CATEGORIE

FROM CATEGORIE JOIN CARTE\_CATEGORIE ON CATEGORIE.ID\_CATEGORIE = CARTE\_CATEGORIE.ID\_CATEGORIE

JOIN CARTE ON CARTE\_CATEGORIE.ID\_CARTE = CARTE.ID\_CARTE

JOIN EXEMPLAR\_CARTE ON CARTE.ID\_CARTE = EXEMPLAR\_CARTE.ID\_CARTE

JOIN INFO\_IMPRUMUT ON EXEMPLAR\_CARTE.ID\_EXEMPLAR = INFO\_IMPRUMUT.ID\_EXEMPLAR

GROUP BY CATEGORIE.ID\_CATEGORIE

HAVING *COUNT*(CATEGORIE.ID\_CATEGORIE) = v\_numar\_carti

);

DBMS\_OUTPUT.*PUT\_LINE*('Categoria cu id-ul: ' || v\_id\_categorie);

DBMS\_OUTPUT.*PUT\_LINE*('Denumire categorie: ' || v\_denumire\_categorie);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.*PUT\_LINE*('Nu exista categorii cu proprietatea ceruta');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.*PUT\_LINE*('Exista mai multe categorii cu proprietatea ceruta');

END;

BEGIN

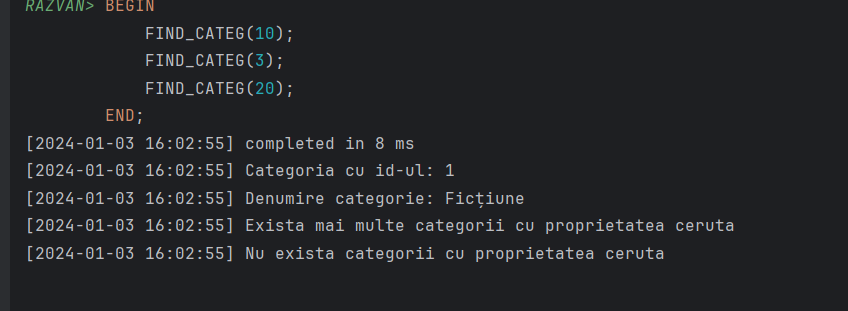
*FIND\_CATEG*(10);

*FIND\_CATEG*(3);

*FIND\_CATEG*(20);

END;

Cod rulat:



## **10.Trigger LMD la nivel de comanda**

Să se definească un trigger care sa nu permita modificarea imprumuturilor in zile pare.

Rezolvare:

CREATE OR REPLACE TRIGGER MODIFICA\_IMPRUMUT

BEFORE INSERT OR UPDATE OR DELETE ON INFO\_IMPRUMUT

DECLARE

V\_ZIUA NUMBER;

BEGIN

SELECT *EXTRACT*(DAY FROM *SYSDATE*) INTO V\_ZIUA FROM DUAL;

IF *MOD*(V\_ZIUA,2) = 0 THEN

*RAISE\_APPLICATION\_ERROR*(-20002,'Imprumuturile nu pot fi modificate in zile pare');

END IF;

END;

DECLARE

V\_ID\_IMPRUMUT NUMBER;

BEGIN

SELECT ID\_IMPRUMUT

INTO V\_ID\_IMPRUMUT

FROM INFO\_IMPRUMUT

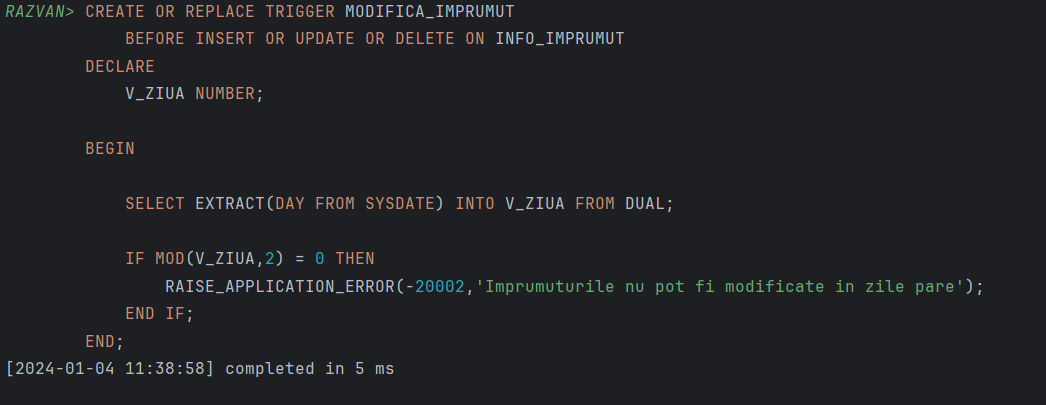
WHERE ROWNUM < 2;

DELETE FROM INFO\_IMPRUMUT WHERE ID\_IMPRUMUT = V\_ID\_IMPRUMUT;

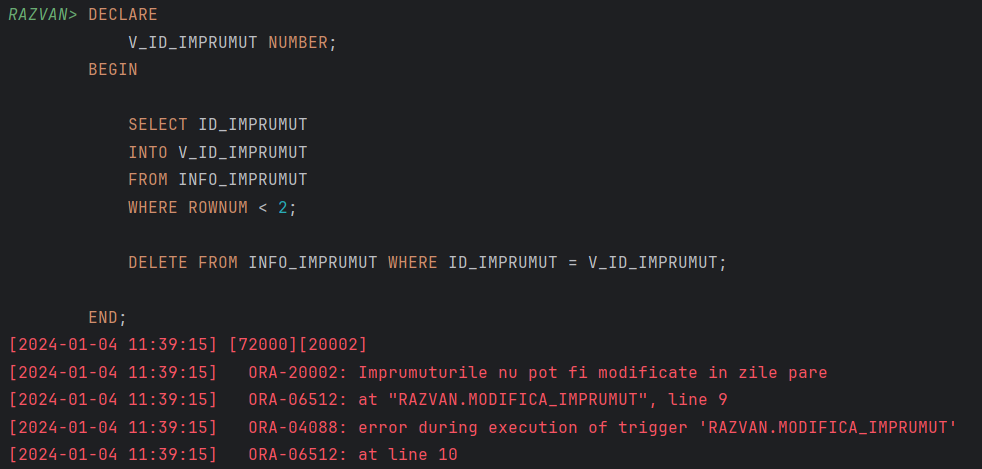
END;

DROP TRIGGER MODIFICA\_IMPRUMUT;

Rulare trigger:



Declansare trigger:



## **11.Trigger LMD la nivel de linie**

Sa se defineasca un trigger care sa nu permita modificarea anului de publicatie al unui exemplar de carte.

Rezolvare:

CREATE OR REPLACE TRIGGER AN\_EXEMPLAR

BEFORE UPDATE ON EXEMPLAR\_CARTE

FOR EACH ROW

BEGIN

IF :NEW.AN\_PUBLICATIE <> :OLD.AN\_PUBLICATIE THEN

*RAISE\_APPLICATION\_ERROR*(-20001,'Anul publicatiei unui exemplar nu poate fi modificat');

END IF;

END;

DECLARE

V\_ID\_EXEMPLAR EXEMPLAR\_CARTE.ID\_EXEMPLAR%TYPE;

BEGIN

select ID\_EXEMPLAR

into V\_ID\_EXEMPLAR

from EXEMPLAR\_CARTE

where ROWNUM <2;

UPDATE EXEMPLAR\_CARTE

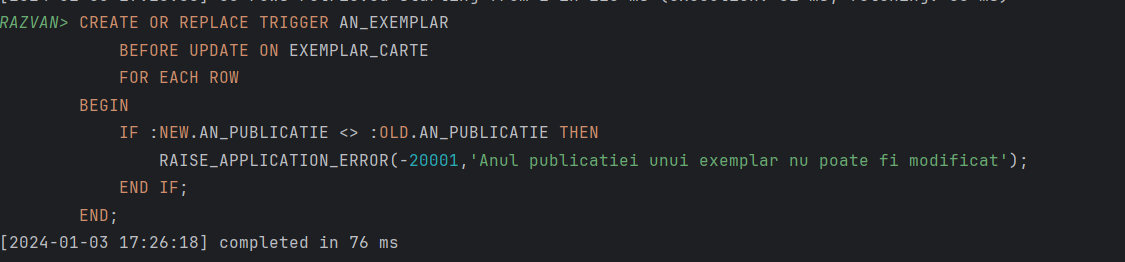
SET AN\_PUBLICATIE = 1900

WHERE ID\_EXEMPLAR = V\_ID\_EXEMPLAR;

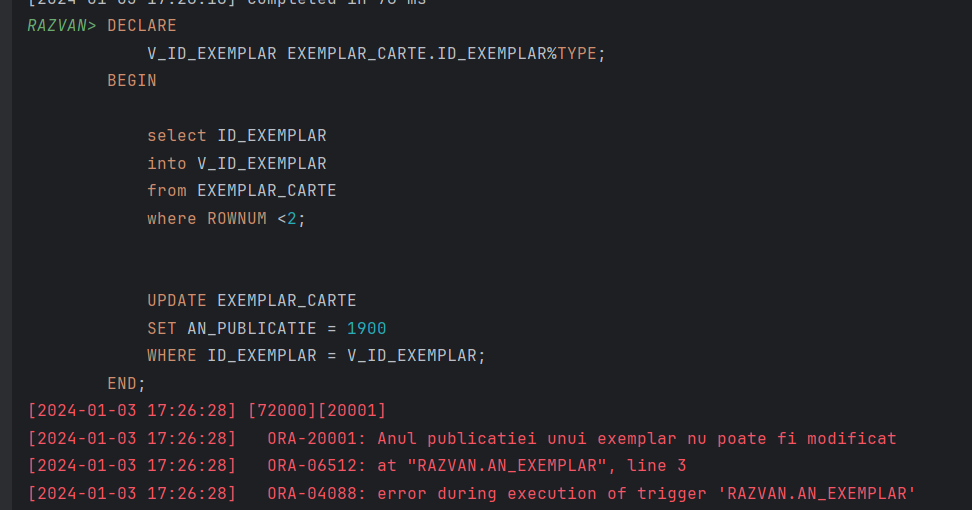
END;

DROP TRIGGER AN\_EXEMPLAR;

Rulare trigger:



Declansare trigger:



## **12.Trigger LDD**

Sa se creeze un trigger LDD care sa nu permita comenzile ALTER si DROP pe baza de date.

Rezolvare:

CREATE OR REPLACE TRIGGER NO\_ALTER\_TRIGGER

BEFORE ALTER OR DROP ON SCHEMA

BEGIN

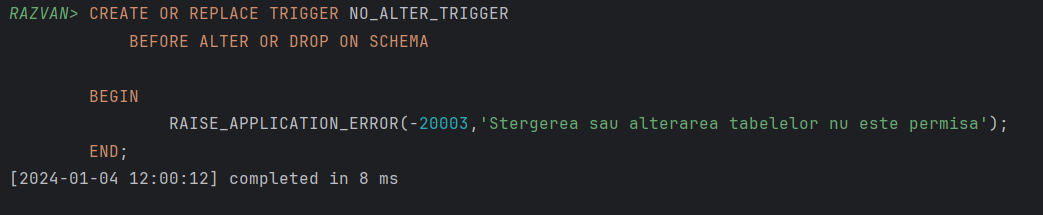
*RAISE\_APPLICATION\_ERROR*(-20003,'Stergerea sau alterarea tabelelor nu este permisa');

END;

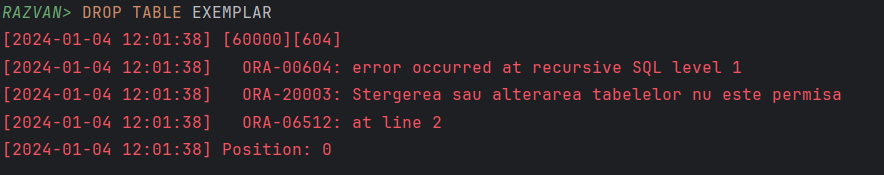
DROP TABLE EXEMPLAR;

DROP TRIGGER NO\_ALTER\_TRIGGER;

Rulare trigger:



Declansare trigger:



## **13. Definire Pachet**

Rezolvare:

CREATE OR REPLACE PACKAGE PACHET\_PROIECT IS

PROCEDURE *afis\_carti*;

PROCEDURE *imprumuturi\_carte* (v\_id\_carte\_curenta CARTE.ID\_CARTE%TYPE);

FUNCTION *NUMAR\_CARTI*(string\_data\_imprumut VARCHAR2,string\_data\_retur VARCHAR2) RETURN NUMBER;

PROCEDURE *FIND\_CATEG* (v\_numar\_carti NUMBER);

END PACHET\_PROIECT;

CREATE OR REPLACE PACKAGE BODY PACHET\_PROIECT IS

-- EX 6

PROCEDURE *afis\_carti*

IS

-- declarare colectii

TYPE nume\_exemplar IS TABLE OF EXEMPLAR\_CARTE%ROWTYPE INDEX BY VARCHAR2(50);

TYPE index\_exemplar IS TABLE OF EXEMPLAR\_CARTE%ROWTYPE;

TYPE id\_cinci IS VARRAY(5) OF CARTE.ID\_CARTE%TYPE;

-- declarare variabile

c\_carte\_exemplare nume\_exemplar;

c\_exemplare index\_exemplar := index\_exemplar();

c\_id\_carti id\_cinci := id\_cinci();

-- variabile intermediare

v\_titlu\_curent CARTE.TITLU\_CARTE%TYPE;

v\_iterator VARCHAR2(50);

BEGIN

-- se selcteaza primele 5 id uri

select ID\_CARTE

BULK COLLECT INTO c\_id\_carti

from CARTE

where ROWNUM<6

order by ID\_CARTE;

FOR i in c\_id\_carti.first..c\_id\_carti.last LOOP

c\_exemplare.extend;

-- se selecteaza un exemplar si se adauga in table

select \*

into c\_exemplare(c\_exemplare.last)

from EXEMPLAR\_CARTE

where ID\_CARTE = c\_id\_carti(i) AND ROWNUM<2;

-- se salveaza titlul cartii curente in variabila

select TITLU\_CARTE

into v\_titlu\_curent

from CARTE

where ID\_CARTE = c\_id\_carti(i);

-- se insereaza in dictionar

c\_carte\_exemplare(v\_titlu\_curent) := c\_exemplare(c\_exemplare.LAST);

END LOOP;

-- afisare

-- se itereaza folosind o variabila de tipul varchar

v\_iterator := c\_carte\_exemplare.FIRST;

WHILE v\_iterator IS NOT NULL LOOP

DBMS\_OUTPUT.*PUT\_LINE*(v\_iterator || ' : ' || c\_carte\_exemplare(v\_iterator).AN\_PUBLICATIE);

v\_iterator := c\_carte\_exemplare.*NEXT*(v\_iterator);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.*PUT\_LINE*('Nu exista date!');

WHEN OTHERS THEN DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');

END *afis\_carti*;

-- EX 7

PROCEDURE *imprumuturi\_carte* (v\_id\_carte\_curenta CARTE.ID\_CARTE%TYPE)

IS

-- folosit pentru a tine numararea de imprumuturi

v\_index NUMBER :=0;

-- cursor pentru a selecta exemplarele cu id\_ul de carte dat

CURSOR C\_CARTI (v\_id\_carte\_selectata CARTE.ID\_CARTE%TYPE) IS

SELECT ID\_EXEMPLAR

FROM EXEMPLAR\_CARTE

WHERE ID\_CARTE = v\_id\_carte\_selectata;

BEGIN

-- iterare prin cursor

FOR EXEMPLAR IN C\_CARTI(v\_id\_carte\_curenta)

LOOP

--Cursor cu subcerere pentru a obtine imprumuturile cerute

FOR IMPRUMUT IN (SELECT \*

FROM INFO\_IMPRUMUT I\_F

WHERE EXEMPLAR.ID\_EXEMPLAR = I\_F.ID\_EXEMPLAR AND I\_F.DATA\_IMPRUMUT < *SYSDATE*)

LOOP

v\_index:= v\_index+1;

DBMS\_OUTPUT.*PUT\_LINE*('Imprumut ' || v\_index);

DBMS\_OUTPUT.*PUT\_LINE*('Data imprumut: ' || IMPRUMUT.DATA\_IMPRUMUT);

DBMS\_OUTPUT.*PUT\_LINE*('Data retur: ' || IMPRUMUT.DATA\_RETUR);

END LOOP;

END LOOP;

END *imprumuturi\_carte*;

-- EX 8

FUNCTION *NUMAR\_CARTI*(string\_data\_imprumut VARCHAR2,string\_data\_retur VARCHAR2) RETURN NUMBER IS

INVALID\_DATA EXCEPTION;

WRONG\_DATA\_FORMAT EXCEPTION;

v\_data\_imprumut DATE;

v\_data\_retur DATE;

v\_numar\_carti NUMBER :=0;

BEGIN

-- Convertire date si tratare exceptii

IF *LENGTH*(string\_data\_imprumut) <> 10 OR *LENGTH*(string\_data\_retur) <> 10 THEN

RAISE WRONG\_DATA\_FORMAT;

END IF;

v\_data\_imprumut := *TRUNC*(*to\_date*(string\_data\_imprumut,'DD-MM-YYYY'));

v\_data\_retur := *TRUNC*(*to\_date*(string\_data\_retur,'DD-MM-YYYY'));

IF v\_data\_retur <= v\_data\_imprumut THEN

RAISE INVALID\_DATA;

END IF;

-- se selecteaza numarul de carti distincte

SELECT *COUNT*(DISTINCT(ID\_CARTE))

INTO v\_numar\_carti

FROM CARTE, (SELECT ID\_EXEMPLAR AS ID

FROM INFO\_IMPRUMUT

WHERE *TRUNC*(DATA\_RETUR)<= v\_data\_retur AND *TRUNC*(DATA\_IMPRUMUT)>= v\_data\_imprumut) DATA

WHERE DATA.ID IN ( SELECT C\_E.ID\_EXEMPLAR

FROM EXEMPLAR\_CARTE C\_E

WHERE ID\_CARTE = C\_E.ID\_CARTE);

return v\_numar\_carti;

EXCEPTION

WHEN INVALID\_DATA THEN

DBMS\_OUTPUT.*PUT\_LINE*('Invalid Data');

return -1;

WHEN WRONG\_DATA\_FORMAT THEN

DBMS\_OUTPUT.*PUT\_LINE*('Wrong format');

return -1;

END *NUMAR\_CARTI*;

-- EX 9

PROCEDURE *FIND\_CATEG* (v\_numar\_carti NUMBER) IS

-- declarare variabile

v\_denumire\_categorie CATEGORIE.NUME\_CATEGORIE%TYPE;

v\_id\_categorie CATEGORIE.ID\_CATEGORIE%TYPE;

v\_count NUMBER := 0;

BEGIN

-- pentru fiecare id de categorie se numara cate carti exista

-- se mai apeleaza count pentru a determina cate categorii cu numarul respectiv exista

SELECT *count*(*count*(\*))

INTO v\_count

FROM CATEGORIE JOIN CARTE\_CATEGORIE ON CATEGORIE.ID\_CATEGORIE = CARTE\_CATEGORIE.ID\_CATEGORIE

JOIN CARTE ON CARTE\_CATEGORIE.ID\_CARTE = CARTE.ID\_CARTE

JOIN EXEMPLAR\_CARTE ON CARTE.ID\_CARTE = EXEMPLAR\_CARTE.ID\_CARTE

JOIN INFO\_IMPRUMUT ON EXEMPLAR\_CARTE.ID\_EXEMPLAR = INFO\_IMPRUMUT.ID\_EXEMPLAR

GROUP BY CATEGORIE.ID\_CATEGORIE

HAVING *COUNT*(CATEGORIE.ID\_CATEGORIE) = v\_numar\_carti;

IF v\_count>1 THEN

RAISE TOO\_MANY\_ROWS;

END IF;

SELECT CATEGORIE.ID\_CATEGORIE,CATEGORIE.NUME\_CATEGORIE

INTO v\_id\_categorie,v\_denumire\_categorie

FROM CATEGORIE WHERE CATEGORIE.ID\_CATEGORIE = (

SELECT CATEGORIE.ID\_CATEGORIE

FROM CATEGORIE JOIN CARTE\_CATEGORIE ON CATEGORIE.ID\_CATEGORIE = CARTE\_CATEGORIE.ID\_CATEGORIE

JOIN CARTE ON CARTE\_CATEGORIE.ID\_CARTE = CARTE.ID\_CARTE

JOIN EXEMPLAR\_CARTE ON CARTE.ID\_CARTE = EXEMPLAR\_CARTE.ID\_CARTE

JOIN INFO\_IMPRUMUT ON EXEMPLAR\_CARTE.ID\_EXEMPLAR = INFO\_IMPRUMUT.ID\_EXEMPLAR

GROUP BY CATEGORIE.ID\_CATEGORIE

HAVING *COUNT*(CATEGORIE.ID\_CATEGORIE) = v\_numar\_carti

);

DBMS\_OUTPUT.*PUT\_LINE*('Categoria cu id-ul: ' || v\_id\_categorie);

DBMS\_OUTPUT.*PUT\_LINE*('Denumire categorie: ' || v\_denumire\_categorie);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.*PUT\_LINE*('Nu exista categorii cu proprietatea ceruta');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.*PUT\_LINE*('Exista mai multe categorii cu proprietatea ceruta');

END *FIND\_CATEG*;

END PACHET\_PROIECT;

DECLARE

i NUMBER;

BEGIN

--6

PACHET\_PROIECT.*afis\_carti*();

--7

PACHET\_PROIECT.*imprumuturi\_carte*(1);

--8

-- ca cand subprogramul ruleaza

i := PACHET\_PROIECT.*NUMAR\_CARTI*('01-01-2022','01-02-2024');

DBMS\_OUTPUT.*PUT\_LINE*(i);

-- caz data invalida

i:= PACHET\_PROIECT.*NUMAR\_CARTI*('01-01-2022','01-02-2021');

DBMS\_OUTPUT.*PUT\_LINE*(i);

-- caz formatare gresita

i:= PACHET\_PROIECT.*NUMAR\_CARTI*('1-01-2022','01-02-20212');

DBMS\_OUTPUT.*PUT\_LINE*(i);

--9

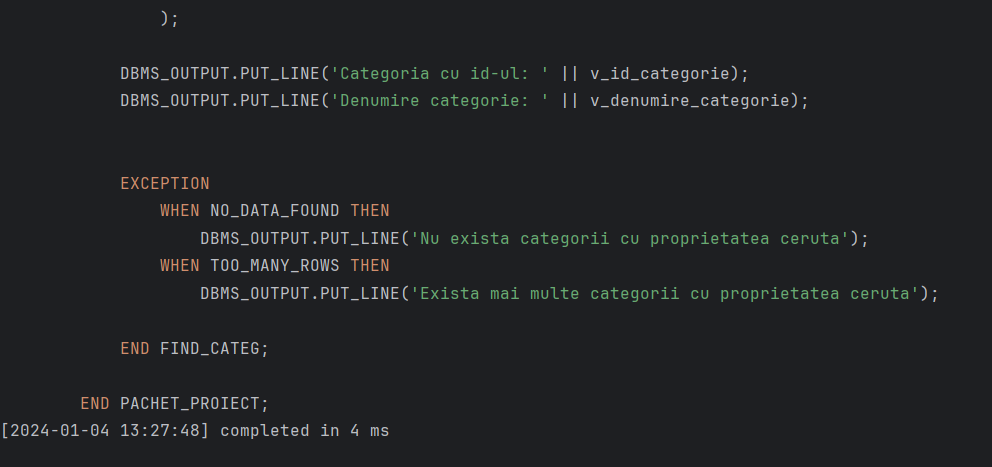
PACHET\_PROIECT.*FIND\_CATEG*(10);

PACHET\_PROIECT.*FIND\_CATEG*(3);

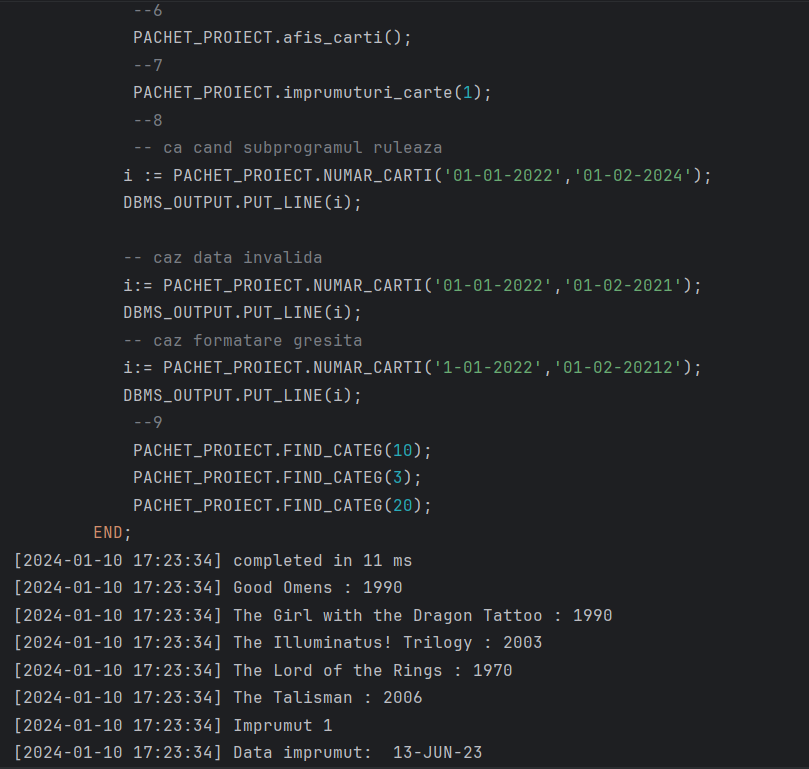
PACHET\_PROIECT.*FIND\_CATEG*(20);

END;

Rulare pachet:



Rulare functii din pachet:



## **14. Definire Pachet pentru a facilitate functionalitate**

Sa se defineasca un pachet care sa permita adaugarea unui nou exemplar de carte dupa:

* titlul cartii
* numele editurii
* anul de publicatie

Daca nu exista o editura cu numele specificat sau o carte cu titlul dat se vor adauga in baza de date editura/ cartea corespunatoare.

Pachetul dispune de:

* 2 varibile globale folosite pentru a extrage id-urile cartii si editurii
* 2 functii folosite pentru a gasi id-urile cartii si editurii
* 3 proceduri folosite pentru a adauga o noua carte, o noua editura si un nou exemplar dupa nume.

Rezolvare:

CREATE OR REPLACE PACKAGE EXEMPLAR\_NOU IS

FUNCTION *FIND\_EDITURA*(v\_nume\_editura EDITURA.NUME\_EDITURA%TYPE) RETURN EDITURA.ID\_EDITURA%TYPE;

FUNCTION *FIND\_CARTE*(v\_titlu\_carte CARTE.TITLU\_CARTE%TYPE) RETURN CARTE.ID\_CARTE%TYPE;

PROCEDURE *ADAUGA\_CARTE*(v\_titlu\_carte CARTE.TITLU\_CARTE%TYPE);

PROCEDURE *ADAUGA\_EDITURA*(v\_nume\_editura EDITURA.NUME\_EDITURA%TYPE);

PROCEDURE *ADAUGA\_EXEMPLAR*(v\_titlu\_carte CARTE.TITLU\_CARTE%TYPE,v\_nume\_editura EDITURA.NUME\_EDITURA%TYPE,v\_an\_publicatie EXEMPLAR\_CARTE.AN\_PUBLICATIE%TYPE);

END EXEMPLAR\_NOU;

CREATE OR REPLACE PACKAGE BODY EXEMPLAR\_NOU IS

v\_id\_carte\_global CARTE.ID\_CARTE%TYPE;

v\_id\_editura\_global EDITURA.ID\_EDITURA%TYPE;

FUNCTION *FIND\_EDITURA*(v\_nume\_editura EDITURA.NUME\_EDITURA%TYPE) RETURN EDITURA.ID\_EDITURA%TYPE IS

v\_id\_editura EDITURA.ID\_EDITURA%TYPE;

BEGIN

SELECT ID\_EDITURA

INTO v\_id\_editura

FROM EDITURA

WHERE *UPPER*(NUME\_EDITURA) = *TRIM*(*UPPER*(v\_nume\_editura));

RETURN v\_id\_editura;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN -1;

END *FIND\_EDITURA*;

FUNCTION *FIND\_CARTE*(v\_titlu\_carte CARTE.TITLU\_CARTE%TYPE) RETURN CARTE.ID\_CARTE%TYPE IS

v\_id\_carte CARTE.ID\_CARTE%TYPE;

BEGIN

SELECT ID\_CARTE

INTO v\_id\_carte

FROM CARTE

WHERE *UPPER*(TITLU\_CARTE) = *TRIM*(*UPPER*(v\_titlu\_carte));

RETURN V\_ID\_CARTE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN -1;

END *FIND\_CARTE*;

PROCEDURE *ADAUGA\_CARTE*(v\_titlu\_carte CARTE.TITLU\_CARTE%TYPE) IS

BEGIN

INSERT INTO CARTE

VALUES(SEQ\_CARTE.NEXTVAL,v\_titlu\_carte);

END *ADAUGA\_CARTE*;

PROCEDURE *ADAUGA\_EDITURA*(v\_nume\_editura EDITURA.NUME\_EDITURA%TYPE) IS

BEGIN

INSERT INTO editura (id\_editura, nume\_editura)

VALUES (SEQ\_EDITURA.NEXTVAL, v\_nume\_editura);

END *ADAUGA\_EDITURA*;

PROCEDURE *ADAUGA\_EXEMPLAR*(v\_titlu\_carte CARTE.TITLU\_CARTE%TYPE,v\_nume\_editura EDITURA.NUME\_EDITURA%TYPE,v\_an\_publicatie EXEMPLAR\_CARTE.AN\_PUBLICATIE%TYPE) IS

BEGIN

v\_id\_carte\_global := *FIND\_CARTE*(v\_titlu\_carte);

v\_id\_editura\_global := *FIND\_EDITURA*(v\_nume\_editura);

IF v\_id\_carte\_global = -1 THEN

*ADAUGA\_CARTE*(v\_titlu\_carte);

v\_id\_carte\_global := *FIND\_CARTE*(v\_titlu\_carte);

END IF;

IF v\_id\_editura\_global = -1 THEN

*ADAUGA\_EDITURA*(v\_nume\_editura);

v\_id\_editura\_global := *FIND\_EDITURA*(v\_nume\_editura);

END IF;

INSERT INTO EXEMPLAR\_CARTE

VALUES(SEQ\_EXEMPLAR\_CARTE.NEXTVAL,v\_id\_carte\_global,v\_id\_editura\_global,v\_an\_publicatie);

END *ADAUGA\_EXEMPLAR*;

END EXEMPLAR\_NOU;

BEGIN

EXEMPLAR\_NOU.*ADAUGA\_EXEMPLAR*('CustomCarte','All',2000);

EXEMPLAR\_NOU.*ADAUGA\_EXEMPLAR*('CustomCarte','CustomEdit',2000);

EXEMPLAR\_NOU.*ADAUGA\_EXEMPLAR*('CustomCarte','all',2000);

end;

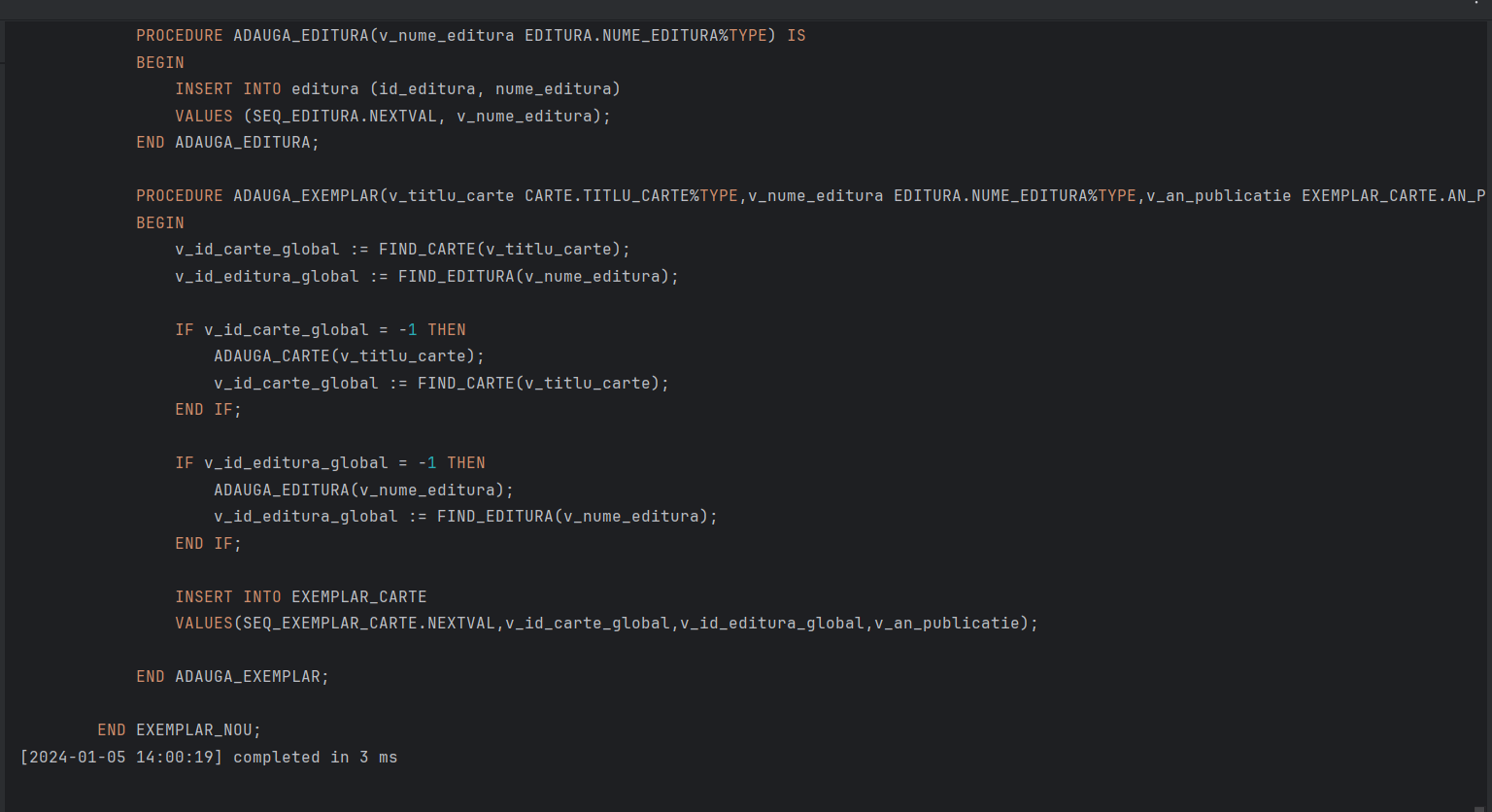
ROLLBACK;

SELECT \* FROM EDITURA;

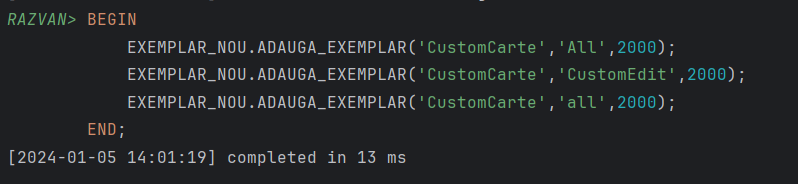
SELECT \* FROM CARTE;

select \* from EXEMPLAR\_CARTE;

Rulare Pachet:



Exemplu Pachet:



Noile inserari in tabel:

