Facultatea de Matematică și Informatică, Universitatea din București

*PROIECT DE SEMESTRU*

BIBLIOTECA FACULTĂȚII

*Lungu Laura-Vanesa*

*Grupa 241*

**SISTEME DE GESTIUNE ALE BAZELOR DE DATE**

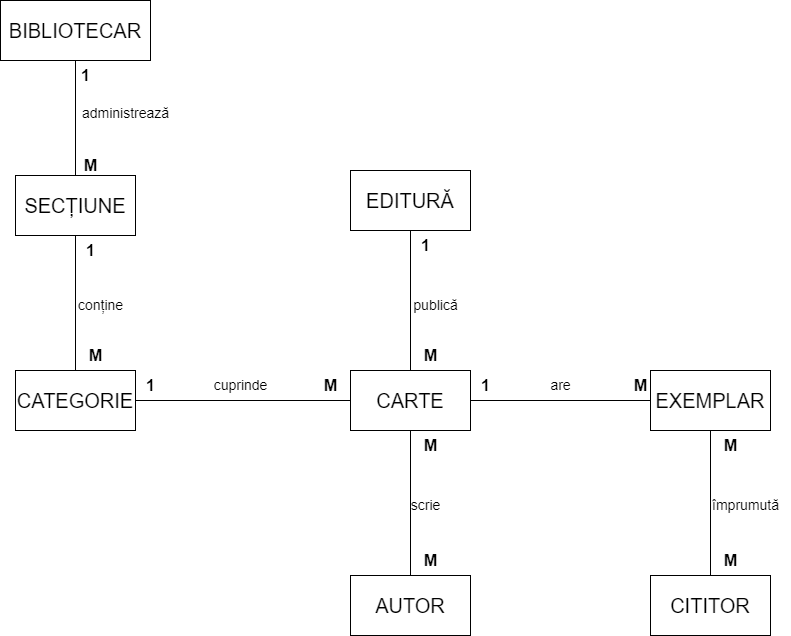
Ianuarie 2024

# Prezentarea Bazei de Date

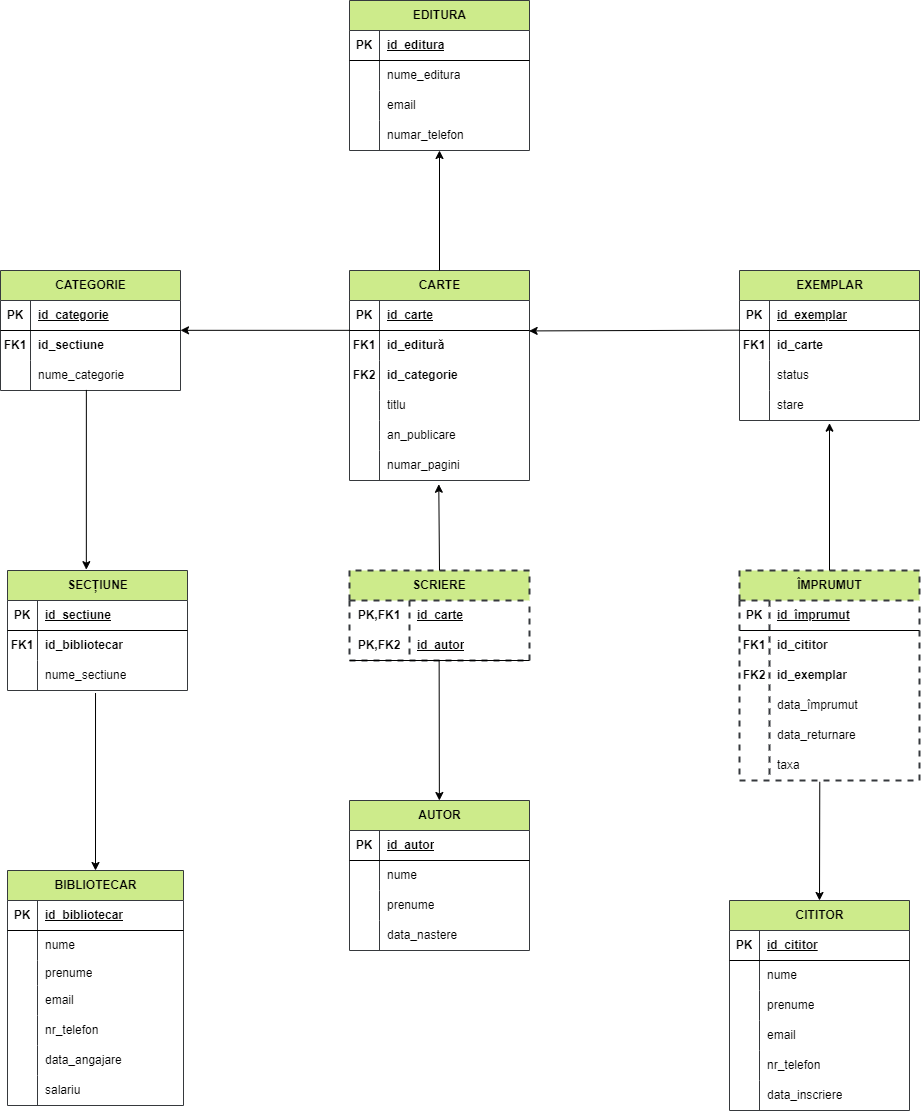
Pentru a facilita accesul la resursele bibliotecii, Facultatea de Matematică și Informatică decide implementarea unei baze de date în care să înregistreze toate informațiile necesare despre **cărțile** care se găsesc în bibliotecă: **exemplare**, **autori**, **categoriile** din care fac parte, precum și **editurile** care le-au publicat.

De asemenea, pentru a asigura o experință cât mai bună **cititorilor** bibliotecii, baza de date reține date și informații despre **secțiunile** bibliotecii în care se regăsesc cărțile, dar și informații despre **bibliotecarii** responsabili de administrarea acestor secțiuni.

# Diagrama entitate-relație (ERD)



# Diagrama Conceptuală



# Implementarea în Oracle a diagramei conceptuale

## Tabelul Bibliotecar

CREATE SEQUENCE seq\_bibliotecar

START WITH 1

INCREMENT BY 1

MAXVALUE 100

NOCYCLE;

CREATE TABLE BIBLIOTECAR (

id\_bibliotecar NUMBER(6) CONSTRAINT pk\_bibliotecar PRIMARY KEY,

nume VARCHAR2(100) NOT NULL,

prenume VARCHAR2(100) NOT NULL,

email VARCHAR2(100) NOT NULL,

nr\_telefon VARCHAR2(14),

data\_angajare DATE NOT NULL,

salariu NUMBER NOT NULL

);

## Tabelul Secțiune

CREATE TABLE SECTIUNE (

id\_sectiune NUMBER(6) CONSTRAINT pk\_sectiune PRIMARY KEY,

id\_bibliotecar NUMBER(6) NOT NULL,

nume\_sectiune VARCHAR2(100) NOT NULL,

CONSTRAINT fk\_sectiune\_bibliotecar FOREIGN KEY (id\_bibliotecar) REFERENCES BIBLIOTECAR(id\_bibliotecar)

);

## Tabelul Categorie

CREATE TABLE CATEGORIE (

id\_categorie NUMBER(6) CONSTRAINT pk\_categorie PRIMARY KEY,

id\_sectiune NUMBER(6) NOT NULL,

nume\_categorie VARCHAR2(100) NOT NULL,

CONSTRAINT fk\_categorie\_sectiune FOREIGN KEY (id\_sectiune) REFERENCES SECTIUNE(id\_sectiune)

);

## Tabelul Editură

CREATE TABLE EDITURA (

id\_editura NUMBER(6) CONSTRAINT pk\_editura PRIMARY KEY,

nume\_editura VARCHAR2(100) NOT NULL,

email VARCHAR2(100) NOT NULL,

nr\_telefon VARCHAR2(14)

);

## Tabelul Autor

CREATE SEQUENCE seq\_autor

START WITH 1

INCREMENT BY 1

MAXVALUE 10000

NOCACHE

NOCYCLE;

CREATE TABLE AUTOR (

id\_autor NUMBER(6) CONSTRAINT pk\_autor PRIMARY KEY,

nume VARCHAR2(100) NOT NULL,

prenume VARCHAR2(100) NOT NULL,

data\_nastere DATE

);

## Tabelul Carte

CREATE SEQUENCE seq\_carte

START WITH 1

INCREMENT BY 1

MAXVALUE 100000

NOCYCLE;

CREATE TABLE CARTE (

id\_carte NUMBER(6) CONSTRAINT pk\_carte PRIMARY KEY,

id\_editura NUMBER(6) NOT NULL,

id\_categorie NUMBER(6) NOT NULL,

titlu VARCHAR2(200) NOT NULL,

an\_publicare NUMBER(4),

numar\_pagini NUMBER(4),

CONSTRAINT fk\_carte\_editura FOREIGN KEY (id\_editura) REFERENCES EDITURA(id\_editura),

CONSTRAINT fk\_carte\_categorie FOREIGN KEY (id\_categorie) REFERENCES CATEGORIE(id\_categorie)

);

## Tabelul Scriere

CREATE TABLE SCRIERE (

id\_autor NUMBER(6) NOT NULL,

id\_carte NUMBER(6) NOT NULL,

CONSTRAINT fk\_autor FOREIGN KEY (id\_autor) REFERENCES AUTOR(id\_autor),

CONSTRAINT fk\_carte FOREIGN KEY (id\_carte) REFERENCES CARTE(id\_carte),

CONSTRAINT pk\_autor\_scrie\_carte PRIMARY KEY(id\_autor, id\_carte)

);

## Tabel Exemplar

CREATE SEQUENCE seq\_exemplar

START WITH 1

INCREMENT BY 1

MAXVALUE 10000000

NOCACHE

NOCYCLE;

CREATE TABLE EXEMPLAR (

id\_exemplar NUMBER(6) CONSTRAINT pk\_exemplar PRIMARY KEY,

id\_carte NUMBER(6) NOT NULL,

status VARCHAR2(30),

stare VARCHAR2(30),

CONSTRAINT fk\_exemplar\_carte FOREIGN KEY (id\_carte) REFERENCES CARTE(id\_carte)

);

ALTER TABLE EXEMPLAR

ADD CONSTRAINT check\_status

CHECK (status in ('IMPRUMUTAT', 'DISPONIBIL'));

ALTER TABLE EXEMPLAR

ADD CONSTRAINT check\_stare

CHECK (stare in ('NOUA','BUNA', 'DETERIORATA'));

## Tabelul Cititor

CREATE SEQUENCE seq\_cititor

START WITH 1

INCREMENT BY 1

MAXVALUE 100000

NOCACHE

NOCYCLE;

CREATE TABLE CITITOR (

id\_cititor NUMBER(6) CONSTRAINT pk\_cititor PRIMARY KEY,

nume VARCHAR2(100) NOT NULL,

prenume VARCHAR2(100) NOT NULL,

email VARCHAR2(100) NOT NULL,

nr\_telefon VARCHAR2(14) NOT NULL,

data\_inscriere DATE

);

## 10.Tabelul Împrumut

CREATE SEQUENCE seq\_imprumut

START WITH 1

INCREMENT BY 1

MAXVALUE 100000000

NOCACHE

NOCYCLE;

CREATE TABLE IMPRUMUT (

id\_imprumut NUMBER(6) CONSTRAINT pk\_imprumut PRIMARY KEY,

id\_exemplar NUMBER(6) NOT NULL,

id\_cititor NUMBER(6) NOT NULL,

data\_imprumut DATE NOT NULL,

data\_returnare DATE,

taxa NUMBER(10,2),

CONSTRAINT fk\_exemplar FOREIGN KEY (id\_exemplar) REFERENCES EXEMPLAR(id\_exemplar),

CONSTRAINT fk\_cititor FOREIGN KEY (id\_cititor) REFERENCES CITITOR(id\_cititor)

);

ALTER TABLE IMPRUMUT

ADD CONSTRAINT check\_data

CHECK (data\_returnare >= data\_imprumut);

# Inserarea datelor în tabele

## Tabelul Bibliotecar

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES(seq\_bibliotecar.NEXTVAL, 'Popescu', 'Ion', 'ion.popescu@yahoo.com', '0712345678', 2000,

TO\_DATE('01-01-1990', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Ionescu', 'Maria', 'maria.ionescu@gmail.com', '0723456789', 3500,

TO\_DATE('01-02-1995', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Popa', 'Mihai', 'mihai.popa@gmail.com', '0734567890', 3200,

TO\_DATE('01-03-2000', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Georgescu', 'Ana', 'ana.georgescu@gmail.com', '0745678901', 3100,

TO\_DATE('01-04-2005', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Radu', 'Adrian', 'adrian.radu@yahoo.com', '0756789012', 3300,

TO\_DATE('01-05-2010', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Pop', 'Andreea', 'andreea.pop@yahoo.com', '0767890123', 3400,

TO\_DATE('01-06-2015', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Badea', 'Mircea', 'mircea.badea@gmail.com', '0778901234', 2100,

TO\_DATE('01-07-2020', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Diaconu', 'Elena', 'elena.diaconu@gmail.com', '0789012345', 3200,

TO\_DATE('01-08-2021', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

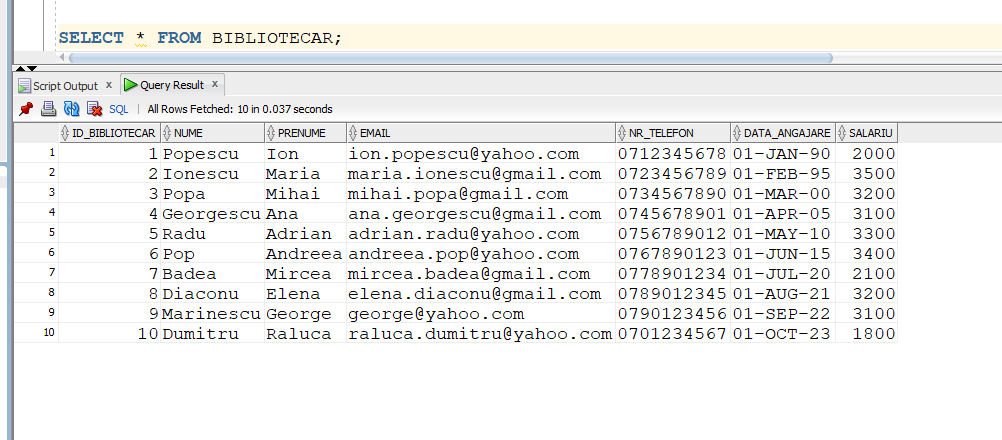
VALUES (seq\_bibliotecar.NEXTVAL, 'Marinescu', 'George', 'george@yahoo.com', '0790123456', 3100,

TO\_DATE('01-09-2022', 'DD-MM-YYYY'));

INSERT INTO BIBLIOTECAR (id\_bibliotecar, nume, prenume, email, nr\_telefon, salariu, data\_angajare)

VALUES (seq\_bibliotecar.NEXTVAL, 'Dumitru', 'Raluca', 'raluca.dumitru@yahoo.com', '0701234567', 1800,

TO\_DATE('01-10-2023', 'DD-MM-YYYY'));



## Tabelul Secțiune

INSERT INTO SECTIUNE (id\_sectiune, id\_bibliotecar, nume\_sectiune) VALUES (1, 1, 'Matematica');

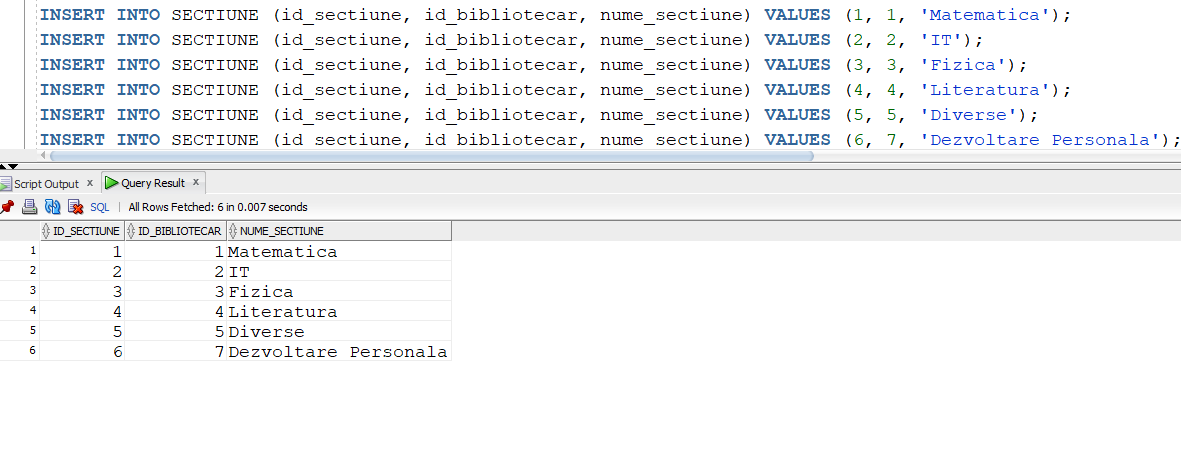
INSERT INTO SECTIUNE (id\_sectiune, id\_bibliotecar, nume\_sectiune) VALUES (2, 2, 'IT');

INSERT INTO SECTIUNE (id\_sectiune, id\_bibliotecar, nume\_sectiune) VALUES (3, 3, 'Fizica');

INSERT INTO SECTIUNE (id\_sectiune, id\_bibliotecar, nume\_sectiune) VALUES (4, 4, 'Literatura');

INSERT INTO SECTIUNE (id\_sectiune, id\_bibliotecar, nume\_sectiune) VALUES (5, 5, 'Diverse');

INSERT INTO SECTIUNE (id\_sectiune, id\_bibliotecar, nume\_sectiune) VALUES (6, 7, 'Dezvoltare Personala');



## Tabelul Categorie

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (1, 1, 'Algebra');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (2, 1, 'Geometrie');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (3, 1, 'Analiza');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (4, 2, 'Algoritmi');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (5, 2, 'Database');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (6, 2, 'Web\_dev');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (7, 2, 'Software');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (8, 3, 'Mecanica');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (9, 3, 'Optica');

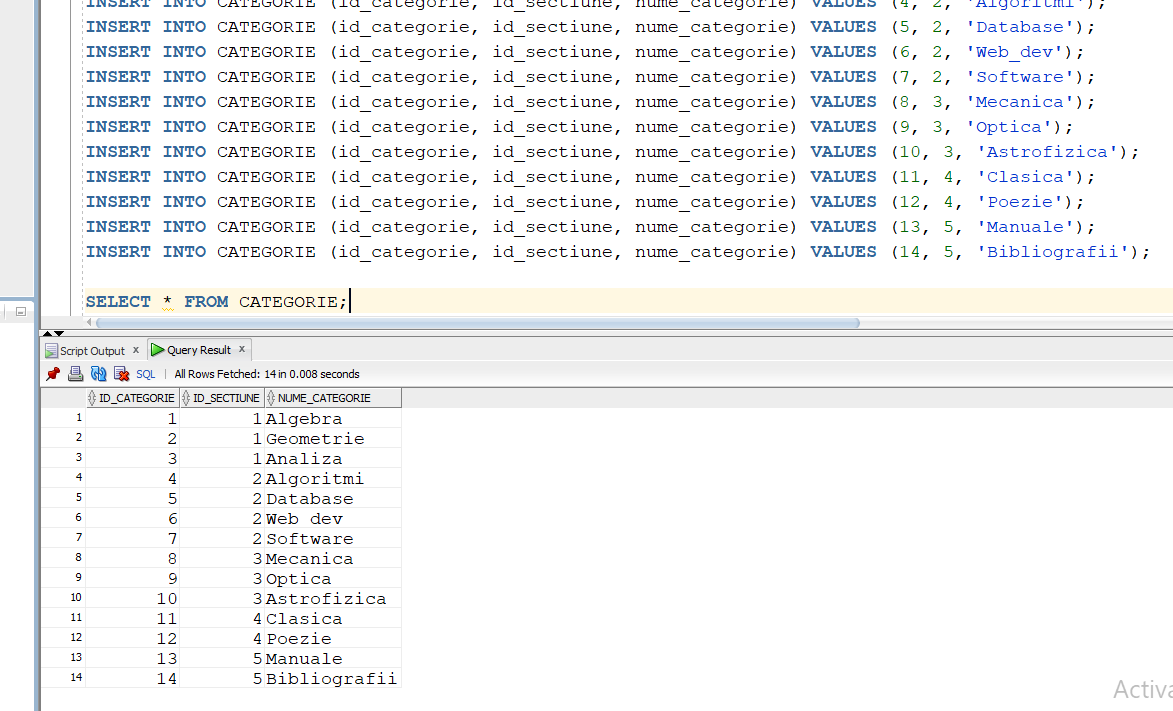
INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (10, 3, 'Astrofizica');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (11, 4, 'Clasica');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (12, 4, 'Poezie');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (13, 5, 'Manuale');

INSERT INTO CATEGORIE (id\_categorie, id\_sectiune, nume\_categorie) VALUES (14, 5, 'Bibliografii');



## Tabelul Editură

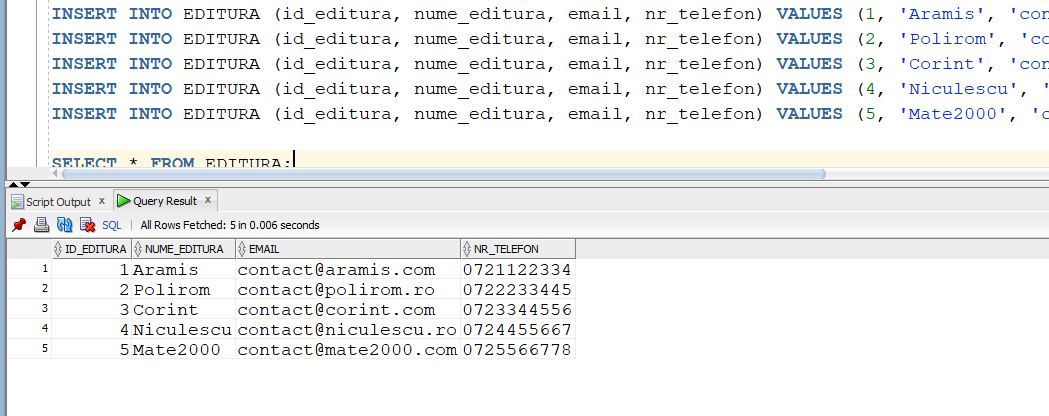
INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (1, 'Aramis', 'contact@aramis.com', '0721122334');

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (2, 'Polirom', 'contact@polirom.ro', '0722233445');

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (3, 'Corint', 'contact@corint.com', '0723344556');

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (4, 'Niculescu', 'contact@niculescu.ro', '0724455667');

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (5, 'Mate2000', 'contact@mate2000.com', '0725566778');



## Tabelul Autor

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Feynman', 'Richard P.', TO\_DATE('11-05-1918', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Watson', 'James D.', TO\_DATE('06-04-1928', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Hawking', 'Stephen', TO\_DATE('08-01-1942', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Darwin', 'Charles', TO\_DATE('12-02-1809', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Kuhn', 'Thomas S.', TO\_DATE('18-07-1922', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Austen', 'Jane', TO\_DATE('16-12-1775', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Shakespeare', 'William', TO\_DATE('26-04-1564', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Goethe', 'Johann Wolfgang', TO\_DATE('28-08-1749', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Eminescu', 'Mihai', TO\_DATE('15-01-1850', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Hugo', 'Victor', TO\_DATE('26-02-1802', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Dostoevsky', 'Fyodor', TO\_DATE('11-11-1821', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume) VALUES (seq\_autor.NEXTVAL, 'Popescu', 'Radu');

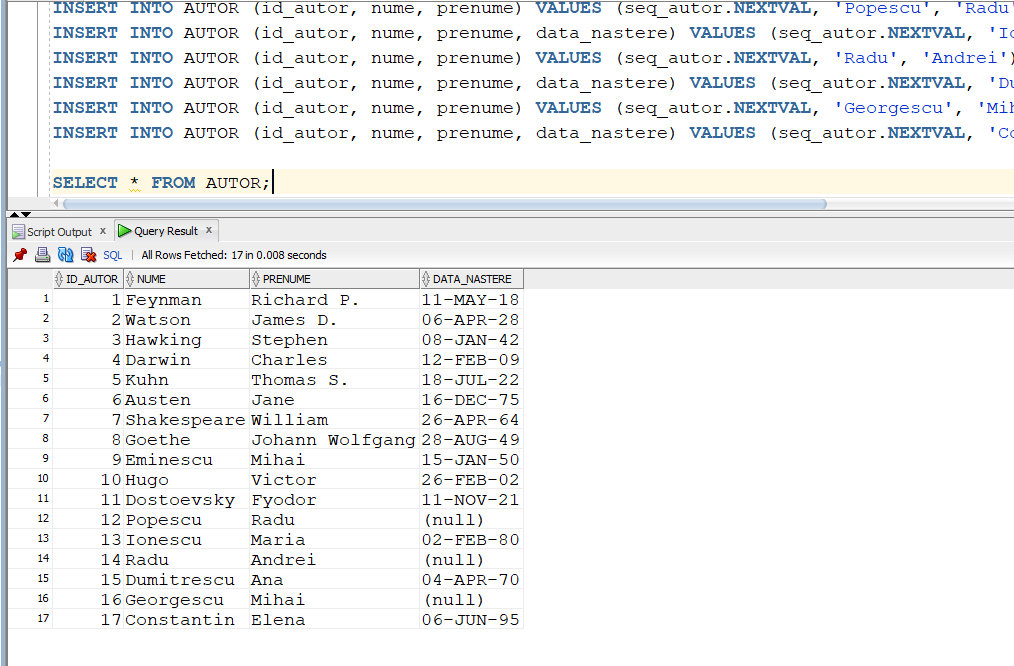
INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Ionescu', 'Maria', TO\_DATE('02-02-1980', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume) VALUES (seq\_autor.NEXTVAL, 'Radu', 'Andrei');

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Dumitrescu', 'Ana', TO\_DATE('04-04-1970', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume) VALUES (seq\_autor.NEXTVAL, 'Georgescu', 'Mihai');

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Constantin', 'Elena', TO\_DATE('06-06-1995', 'DD-MM-YYYY'));



## Tabelul Carte

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 1, 3, 'Introducere in analiza matematica', 350);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 3, 'Analiza - ecuatii diferentiale', 2010, 450);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare) VALUES (seq\_carte.NEXTVAL, 1, 2, 'Geometrie euclidiana', 2009);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 2, 'Introducere in geometria diferentiala', 2011, 450);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 5, 2, 'Geometrie computationala', 2007, 400);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 1, 'Teoria grupurilor si aplicatii', 400);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare) VALUES (seq\_carte.NEXTVAL, 1, 8, 'Mecanica: Fundamente si aplicatii', 2005);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 9, 'Introducere in optica geometrica', 2010, 250);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 3, 10, 'Astrofizica: Galaxii si stele', 2018, 400);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 4, 8, 'Mecanica cuantica si statistica', 2008, 280);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 5, 10, 'Originea si evolutia Universului', 2015, 350);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 1, 5, 'Baze de date relationale', 400);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 7, 'Dezvoltare software:tehnici moderne', 2016, 350);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu) VALUES (seq\_carte.NEXTVAL, 3, 6, 'Aplicatii web dinamice');

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 4, 4, 'Algoritmi si structuri de date', 2010, 450);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 5, 5, 'Introducere in bazele de date', 2018, 280);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 1, 4, 'Programare orientata pe obiecte', 2015, 400);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 6, 'Dezvoltare web cu HTML si CSS', 2013, 320);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 3, 4, 'Algoritmi si structuri de date avansate', 2017, 500);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 4, 7, 'Testare si calitatea software', 2022, 350);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 1, 11, 'Mândrie si prejudecata', 1813, 320);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 11, 'Romeo si Julieta', 1597, 256);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 3, 11, 'Faust', 1808, 480);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 4, 12, 'Luceafarul', 1883, 160);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 1, 11, 'Mizerabilii', 1862, 1488);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 3, 11, 'Crima si pedeapsa', 1866, 672);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 1, 13, 'The Feynman Lectures on Physics', 1963, 1552);

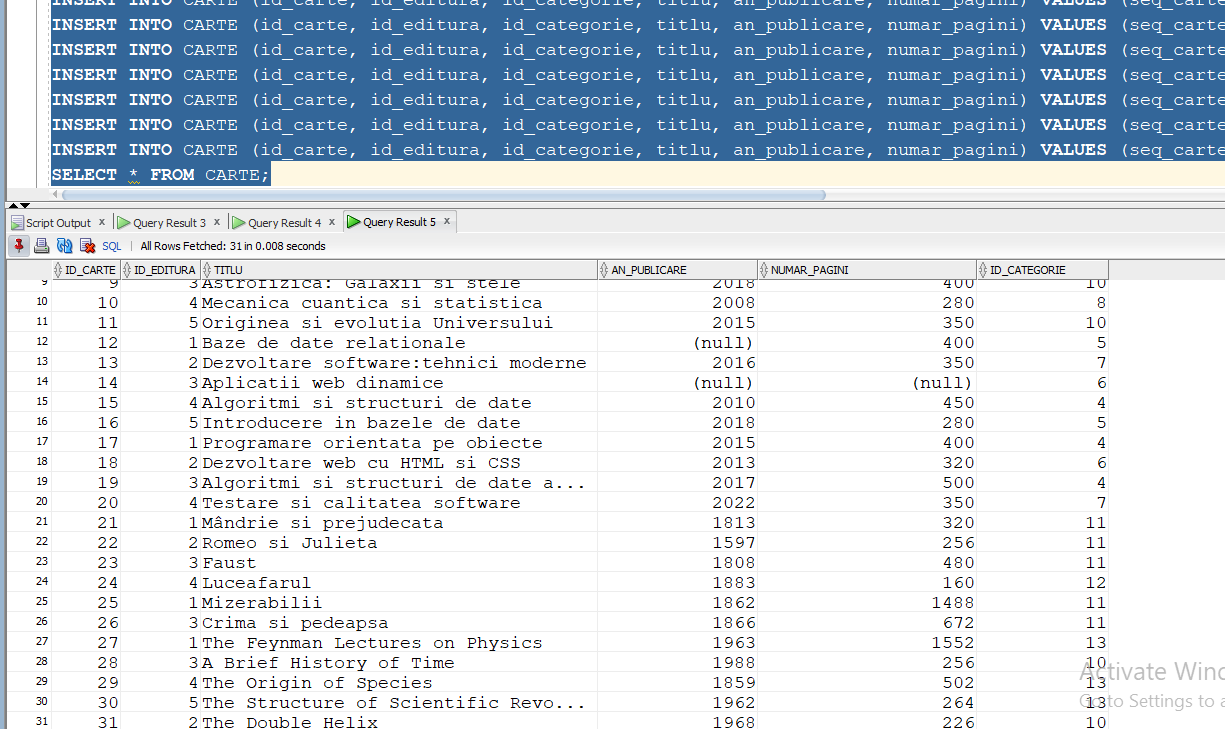
INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 3, 10, 'A Brief History of Time', 1988, 256);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 4, 13, 'The Origin of Species', 1859, 502);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 5, 13, 'The Structure of Scientific Revolutions', 1962, 264);

INSERT INTO CARTE (id\_carte, id\_editura, id\_categorie, titlu, an\_publicare, numar\_pagini) VALUES (seq\_carte.NEXTVAL, 2, 10, 'The Double Helix', 1968, 226);

SELECT \* FROM CARTE;



## Tabelul Scriere

INSERT INTO SCRIERE VALUES (12, 1);

INSERT INTO SCRIERE VALUES (13, 2);

INSERT INTO SCRIERE VALUES (14, 3);

INSERT INTO SCRIERE VALUES (15, 4);

INSERT INTO SCRIERE VALUES (16, 5);

INSERT INTO SCRIERE VALUES (13, 5);

INSERT INTO SCRIERE VALUES (17, 6);

INSERT INTO SCRIERE VALUES (12, 7);

INSERT INTO SCRIERE VALUES (13, 8);

INSERT INTO SCRIERE VALUES (14, 9);

INSERT INTO SCRIERE VALUES (15, 10);

INSERT INTO SCRIERE VALUES (16, 11);

INSERT INTO SCRIERE VALUES (17, 11);

INSERT INTO SCRIERE VALUES (12, 11);

INSERT INTO SCRIERE VALUES (17, 12);

INSERT INTO SCRIERE VALUES (12, 13);

INSERT INTO SCRIERE VALUES (13, 14);

INSERT INTO SCRIERE VALUES (14, 15);

INSERT INTO SCRIERE VALUES (15, 16);

INSERT INTO SCRIERE VALUES (17, 16);

INSERT INTO SCRIERE VALUES (16, 17);

INSERT INTO SCRIERE VALUES (17, 18);

INSERT INTO SCRIERE VALUES (12, 19);

INSERT INTO SCRIERE VALUES (13, 20);

INSERT INTO SCRIERE VALUES (12, 20);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Austen' AND prenume = 'Jane'), 21);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Shakespeare' AND prenume = 'William'), 22);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Goethe' AND prenume = 'Johann Wolfgang'), 23);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Eminescu' AND prenume = 'Mihai'), 24);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Hugo' AND prenume = 'Victor'),25);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Dostoevsky' AND prenume = 'Fyodor'),26);

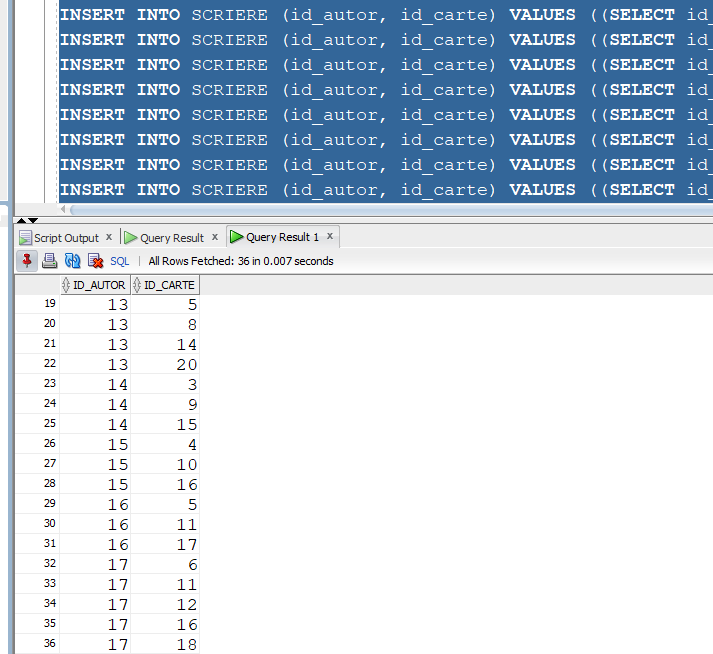
INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Feynman' AND prenume = 'Richard P.'), 27);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Hawking' AND prenume = 'Stephen'), 28);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Darwin' AND prenume = 'Charles'), 29);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Kuhn' AND prenume = 'Thomas S.'), 30);

INSERT INTO SCRIERE (id\_autor, id\_carte) VALUES ((SELECT id\_autor FROM AUTOR WHERE nume = 'Watson' AND prenume = 'James D.'), 31);



## Tabelul Exemplar

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 1, 'DISPONIBIL','NOUA' );

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 1, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 1, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 2, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 2, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 2, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 3, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 3, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 4, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 5, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 6, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 6, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 6, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 7, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 8, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 8, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 9, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 10, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 11, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 12, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 13, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 13, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 14, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 15, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 16, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 16, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 17, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 18, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 19, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 20, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 20, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 20, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 20, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 21, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 22, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 23, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 24, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 25, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 26, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 26, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 27, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 27, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 28, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 28, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 28, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 29, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 29, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 29, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 29, 'DISPONIBIL','NOUA');

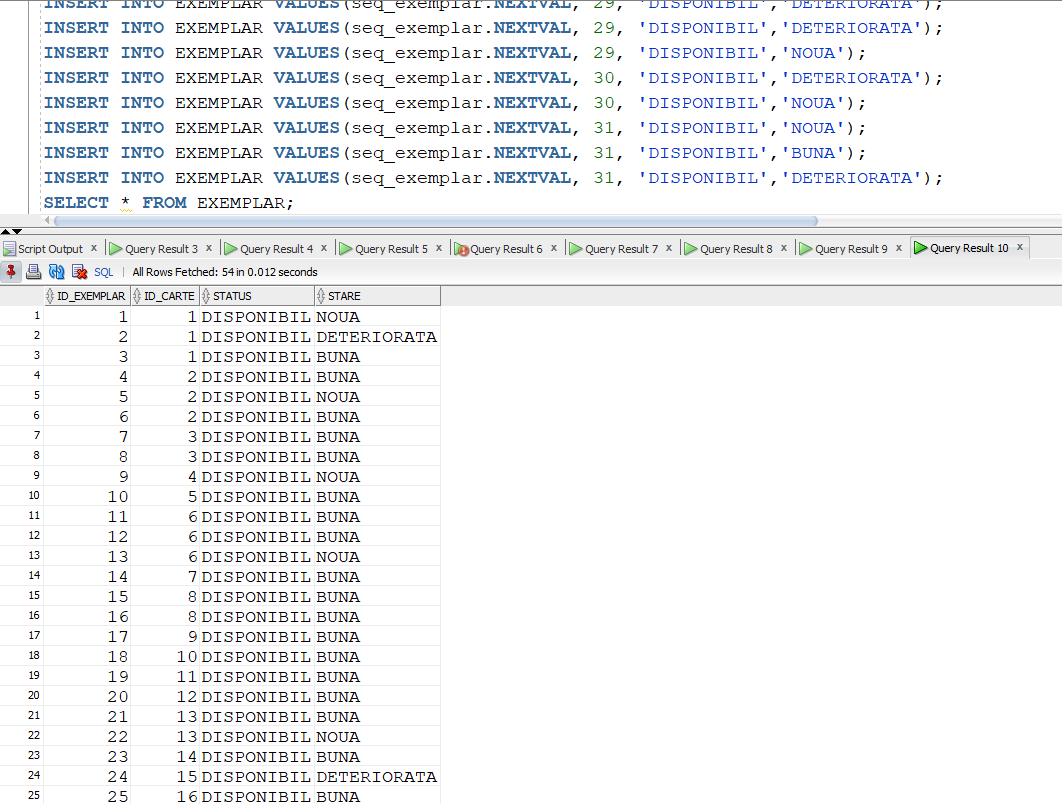
INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 30, 'DISPONIBIL','DETERIORATA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 30, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 31, 'DISPONIBIL','NOUA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 31, 'DISPONIBIL','BUNA');

INSERT INTO EXEMPLAR VALUES(seq\_exemplar.NEXTVAL, 31, 'DISPONIBIL','DETERIORATA');



## Tabelul Cititor

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Popovici', 'Elena', 'popovici.elena@gmail.com', '0721122334', TO\_DATE('2010-03-12', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Iorgulescu', 'Andrei', 'iorgu.andrei@yahoo.com', '0732233445', TO\_DATE('2011-06-25', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Popescu', 'Maria', 'popescu.maria@gmail.com', '0743344556', TO\_DATE('2012-08-10', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Georgescu', 'Alexandru', 'georgescu.alex@yahoo.com', '0754455667', TO\_DATE('2013-02-18', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Radu', 'Cristina', 'radu.cristina@gmail.com', '0765566778', TO\_DATE('2014-11-05', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Petrescu', 'Ionut', 'petrescu.ionut@yahoo.com', '0776677889', TO\_DATE('2015-09-20', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Constantin', 'Andreea', 'andreea.const@gmail.com', '0787788990', TO\_DATE('2016-07-03', 'YYYY-MM-DD'));

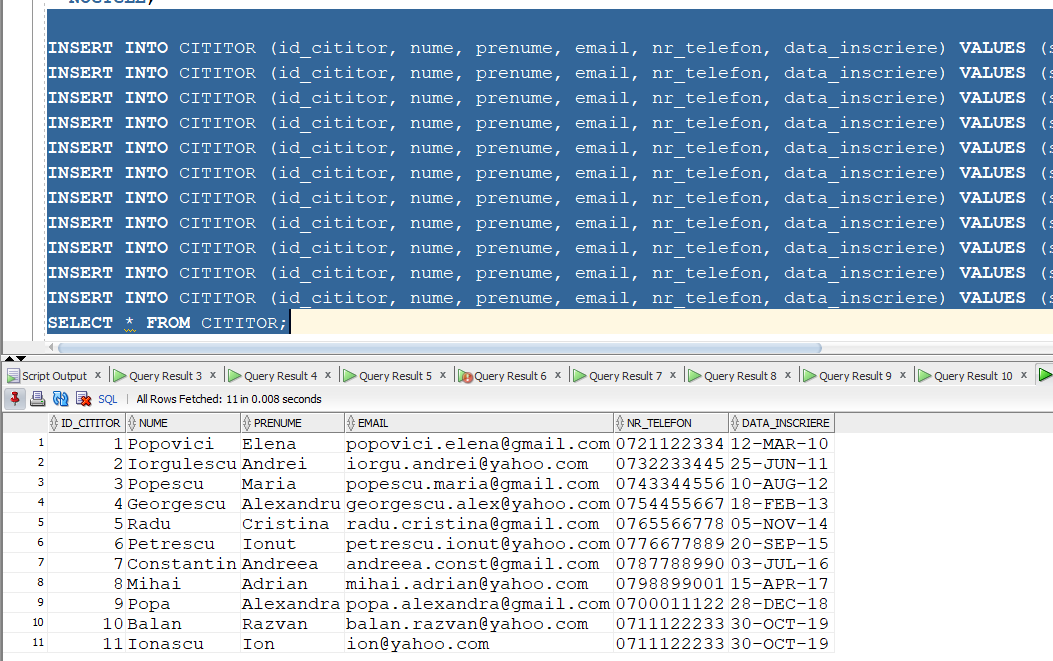
INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Mihai', 'Adrian', 'mihai.adrian@yahoo.com', '0798899001', TO\_DATE('2017-04-15', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Popa', 'Alexandra', 'popa.alexandra@gmail.com', '0700011122', TO\_DATE('2018-12-28', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Balan', 'Razvan', 'balan.razvan@yahoo.com', '0711122233', TO\_DATE('2019-10-30', 'YYYY-MM-DD'));

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Ionascu', 'Ion', 'ion@yahoo.com', '0711122233', TO\_DATE('2019-10-30', 'YYYY-MM-DD'));

SELECT \* FROM CITITOR;



## 10. Tabelul Împrumut

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 7, 1, TO\_DATE('2018-05-15', 'YYYY-MM-DD'), TO\_DATE('2018-06-10', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 3, 2, TO\_DATE('2019-02-20', 'YYYY-MM-DD'), TO\_DATE('2019-03-25', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 22, 2, TO\_DATE('2020-09-10', 'YYYY-MM-DD'), TO\_DATE('2020-10-20', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 22, 4, TO\_DATE('2021-07-05', 'YYYY-MM-DD'), TO\_DATE('2021-08-05', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 24, 5, TO\_DATE('2022-03-15', 'YYYY-MM-DD'), TO\_DATE('2022-05-15', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 33, 6, TO\_DATE('2022-08-10', 'YYYY-MM-DD'), TO\_DATE('2022-09-10', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut, data\_returnare)

VALUES (seq\_imprumut.NEXTVAL, 32, 6, TO\_DATE('2022-12-20', 'YYYY-MM-DD'), TO\_DATE('2023-01-20', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 29, 7, TO\_DATE('2023-04-05', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 26, 9, TO\_DATE('2023-03-01', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 5, 10, TO\_DATE('2022-12-30', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 32, 10, TO\_DATE('2023-04-05', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 6, 8, TO\_DATE('2023-04-15', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 42, 3, TO\_DATE('2023-02-21', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 2, 2, TO\_DATE('2023-03-05', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 10, 1, TO\_DATE('2023-05-10', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 27, 1, TO\_DATE('2023-05-20', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

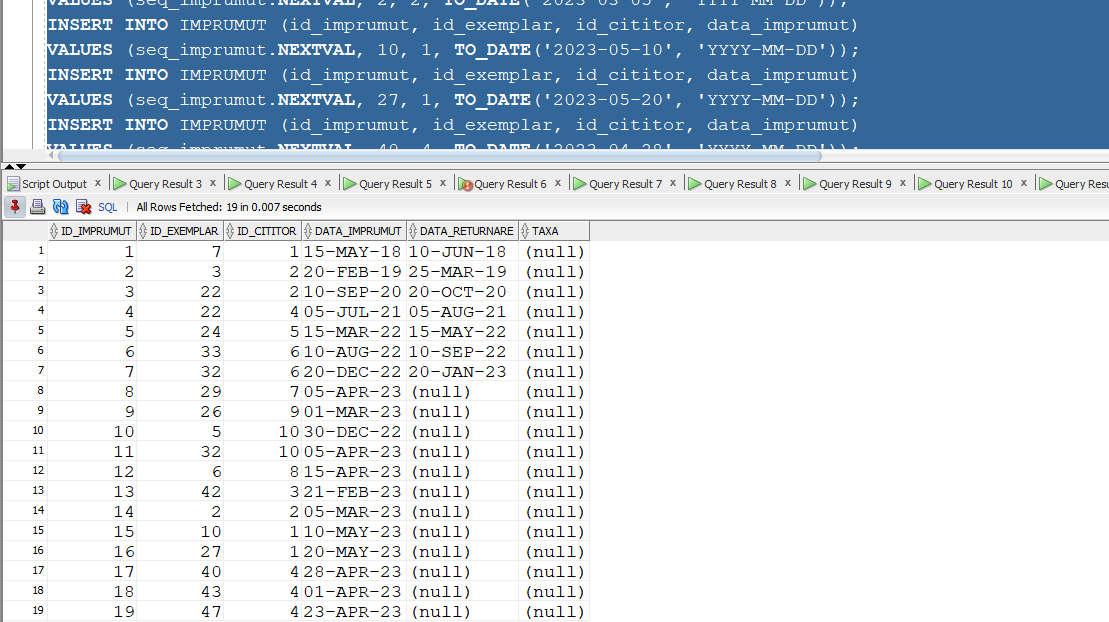
VALUES (seq\_imprumut.NEXTVAL, 40, 4, TO\_DATE('2023-04-28', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 43, 4, TO\_DATE('2023-04-01', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 47, 4, TO\_DATE('2023-04-23', 'YYYY-MM-DD'));



## EXTRA:

### Actualizarea datelor din tabelul Exemplar

UPDATE EXEMPLAR

SET status = (

SELECT

CASE

WHEN data\_returnare IS NOT NULL THEN 'DISPONIBIL'

ELSE 'IMPRUMUTAT'

END

FROM IMPRUMUT

WHERE id\_exemplar = EXEMPLAR.id\_exemplar

ORDER BY data\_imprumut DESC

FETCH FIRST 1 ROW ONLY

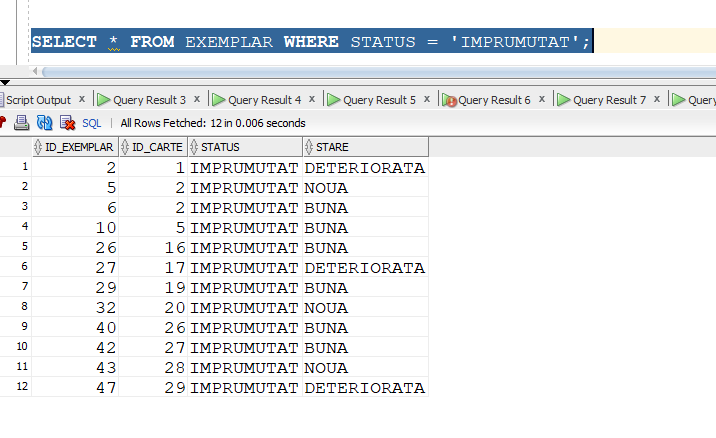
)

WHERE id\_exemplar IN (

SELECT id\_exemplar

FROM IMPRUMUT

);



### Actualizarea datelor din tabelul Împrumut

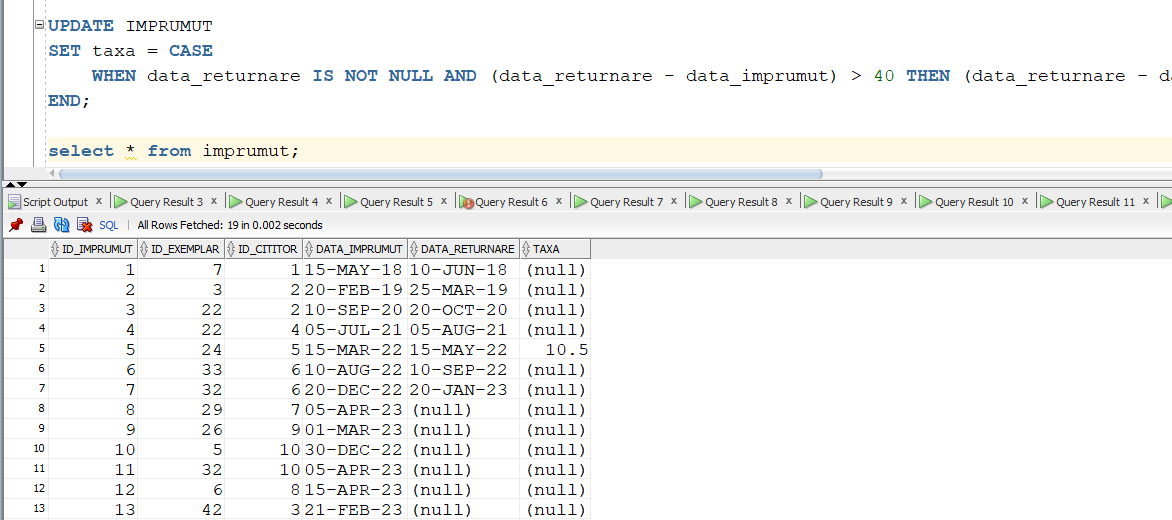
Pentru împrumuturile care au fost returnate, dar care au depășit 40 de zile de la ziua împrumutului am setat taxa ca fiind 0.5\*numarul de zile depasite.

UPDATE IMPRUMUT

SET taxa = CASE

WHEN data\_returnare IS NOT NULL AND (data\_returnare - data\_imprumut) > 40 THEN (data\_returnare - data\_imprumut) \*0.5

END;



# Subprogram stocat independent cu 3 tipuri de colecții.

**Cerință**: Scrieți un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Subprogramul va afișa pentru fiecare secțiune a bibliotecii numele secțiunii, numele categoriilor conținute de acea secțiune, precum și titlurile cărților care aparțin acelor categorii. În cazul în care secțiunea nu conține categorii sau o categorie nu conține niciun titlu, afișați un mesaj de informare. Apelați subprogramul.

CREATE OR REPLACE PROCEDURE INFO\_CARTI\_CATEGORII\_IN\_SECTIUNE IS

    -- TABLOU IMBRICAT cu numele sectiunilor

    TYPE tabel\_imbricat\_sectiune IS TABLE OF SECTIUNE%ROWTYPE;

    t\_sectiune tabel\_imbricat\_sectiune := tabel\_imbricat\_sectiune();

    -- VECTOR pentru numele categoriilor (estimativ stim ca nu ar putea fi mai mult de 20 de categorii intr-o sectiune)

    TYPE vector\_categorii IS VARRAY(20) OF CATEGORIE%ROWTYPE;

    v\_categorii vector\_categorii;

    -- TABLOU INDEXAT pentru titlurile cartilor

    TYPE tabel\_indexat\_carti IS TABLE OF CARTE.titlu%TYPE INDEX BY PLS\_INTEGER;

    t\_carti tabel\_indexat\_carti;

    BEGIN

        SELECT \*

        BULK COLLECT INTO t\_sectiune

        FROM SECTIUNE;

        FOR i IN 1..t\_sectiune.COUNT LOOP

            DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------------------------');

            SELECT \*

            BULK COLLECT INTO v\_categorii

            FROM CATEGORIE

            WHERE ID\_SECTIUNE = t\_sectiune(i).id\_sectiune;

            IF v\_categorii.COUNT > 0 THEN

                DBMS\_OUTPUT.PUT\_LINE(i||'. SECTIUNEA ' || t\_sectiune(i).nume\_sectiune || ' contine urmatoarele categorii: ');

                FOR j IN 1..v\_categorii.COUNT LOOP

                    SELECT titlu

                    BULK COLLECT INTO t\_carti

                    FROM CARTE

                    WHERE ID\_CATEGORIE = v\_categorii(j).id\_categorie;

                    IF t\_carti.COUNT > 0 THEN

                        DBMS\_OUTPUT.PUT\_LINE('      ' || j || '. CATEGORIA ' || UPPER( v\_categorii(j).nume\_categorie) || ' contine urmatoarele carti: ');

                        FOR k IN 1..t\_carti.COUNT LOOP

                            DBMS\_OUTPUT.PUT\_LINE('                  --> CARTE: ' || t\_carti(k));

                        END LOOP;

                    ELSE

                        DBMS\_OUTPUT.PUT\_LINE('      ' || j || '. CATEGORIA ' || UPPER(v\_categorii(j).nume\_categorie) || ' nu contine nicio carte inca! ');

                    END IF;

                END LOOP;

            ELSE

                DBMS\_OUTPUT.PUT\_LINE(i||'. SECTIUNEA '|| UPPER(t\_sectiune(i).nume\_sectiune) || ' nu contine nicio categorie inca! ');

            END IF;

        END LOOP;

        DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------------------------');

        DBMS\_OUTPUT.PUT\_LINE(' ');

    END INFO\_CARTI\_CATEGORII\_IN\_SECTIUNE;

/

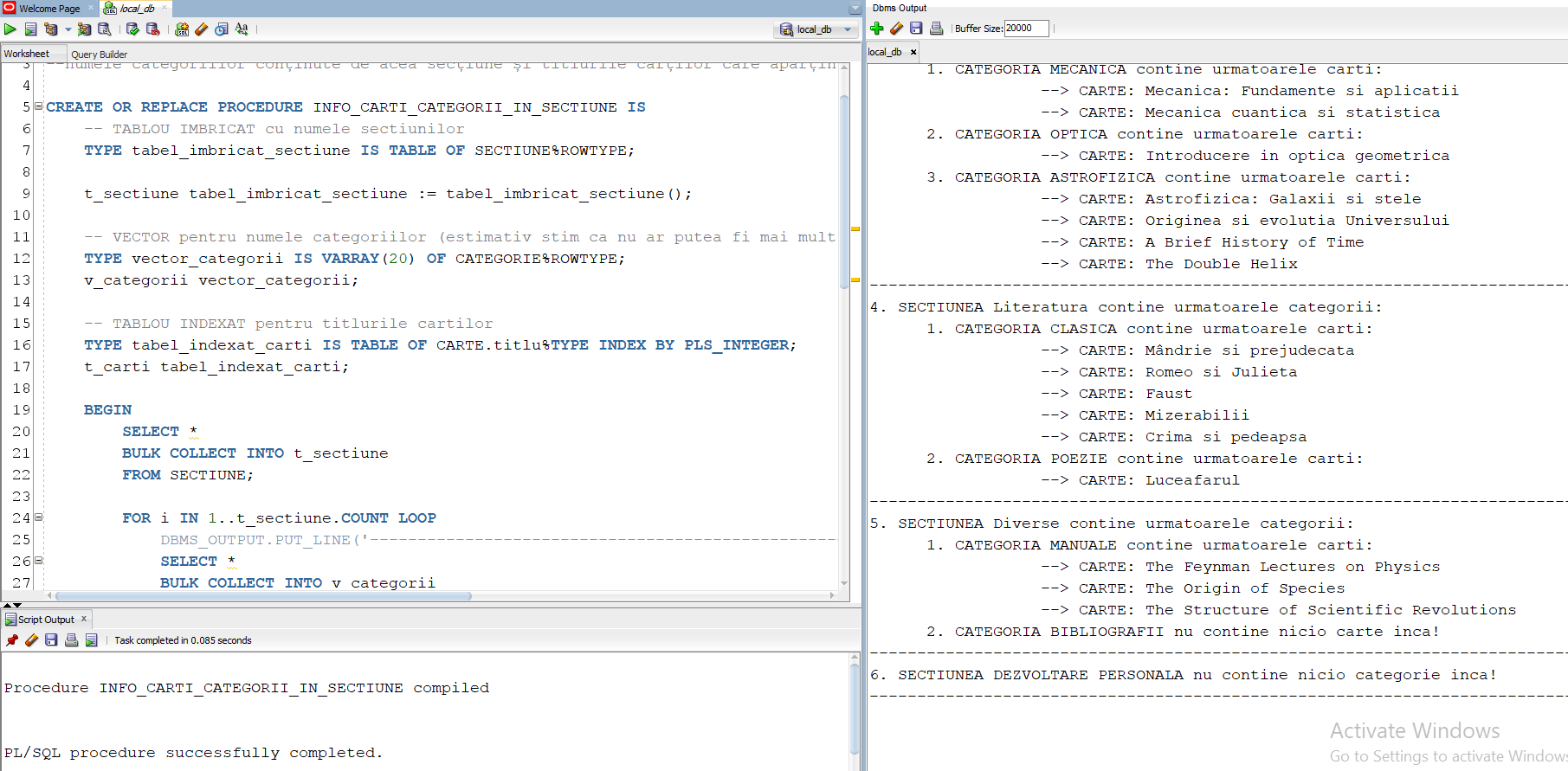
--Apelarea subprogramului

BEGIN

    INFO\_CARTI\_CATEGORII\_IN\_SECTIUNE;

END;

/



# Subprogram stocat independent cu 2 tipuri de cursoare

**Cerință**: Scrieți un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Acest subprogram va căuta în baza de date cititorii care au același nume ca cel dat ca parametru subprogramului și va afișa pentru fiecare numărul total de împrumuturi, alături de titlurile exemplarelor împrumutate, data împrumutului, data returnarii în cazul în care există și starea exemplarului. De asemenea, este tratat cazul în care nu se găsește niciun cititor în baza de date.

CREATE OR REPLACE PROCEDURE info\_imprumuturi\_cititor (v\_nume\_cititor IN cititor.nume%TYPE) IS

    -- Cursor neparametrizat care preia cititorii al caror nume se potriveste cu numele dat ca parametru subprogramului

    CURSOR c\_cititor IS

        SELECT c.nume, c.prenume, c.id\_cititor

        FROM CITITOR C

        WHERE UPPER(C.NUME) = UPPER(v\_nume\_cititor);

    -- Cursor parametrizat care preia informatii despre imprumuturile fiecarui cititor dat ca parametru

    CURSOR c\_imprumuturi(v\_id\_cititor cititor.id\_cititor%TYPE) IS

        SELECT i.id\_imprumut, c.titlu, i.data\_imprumut, i.data\_returnare, e.stare

        FROM IMPRUMUT i

        JOIN EXEMPLAR e ON i.id\_exemplar = e.id\_exemplar

        JOIN CARTE c ON e.id\_carte = c.id\_carte

        WHERE i.id\_cititor = v\_id\_cititor;

    v\_exista\_cititori BOOLEAN := FALSE;

    v\_nr\_imprumuturi NUMBER;

BEGIN

    FOR i IN c\_cititor LOOP

        v\_exista\_cititori := TRUE;

        DBMS\_OUTPUT.PUT\_LINE('------------------------');

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

        v\_nr\_imprumuturi := 0;

        FOR j IN c\_imprumuturi(i.id\_cititor) LOOP

            v\_nr\_imprumuturi := v\_nr\_imprumuturi + 1;

            DBMS\_OUTPUT.PUT\_LINE(v\_nr\_imprumuturi || '. Imprumutul cu ID-ul ' || j.id\_imprumut || ': ' || j.titlu || ', Data Imprumutului: ' || TO\_CHAR(j.data\_imprumut, 'DD-MON-YYYY') || ', Data Returnarii: ' || NVL(TO\_CHAR(j.data\_returnare, 'DD-MON-YYYY'), '-') || ', Starea: ' || j.stare);

        END LOOP;

        IF v\_nr\_imprumuturi > 0 THEN

            DBMS\_OUTPUT.PUT\_LINE('Total imprumuturi: ' || v\_nr\_imprumuturi);

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('Cititorul nu are imprumuturi!');

        END IF;

    END LOOP;

    IF NOT v\_exista\_cititori THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun cititor in baza de date cu acest nume!');

    END IF;

END info\_imprumuturi\_cititor;

/

--Am mai inserat niste date in tabele ca sa putem verifica mai multe variante

INSERT INTO CITITOR (id\_cititor, nume, prenume, email, nr\_telefon, data\_inscriere) VALUES (seq\_cititor.NEXTVAL, 'Iorgulescu', 'Alexandra', 'alexandra.iorg@gmail.com', '0721122334', TO\_DATE('2010-03-12', 'YYYY-MM-DD'));

INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 7, 12, TO\_DATE('2024-01-8', 'YYYY-MM-DD'));

/

--verificam pentru mai multi cititori cu acest nume

begin

    info\_imprumuturi\_cititor('Iorgulescu');

end;

/

--verificam pentru un cititor care nu are imprumuturi

begin

    info\_imprumuturi\_cititor('Ionascu');

end;

/

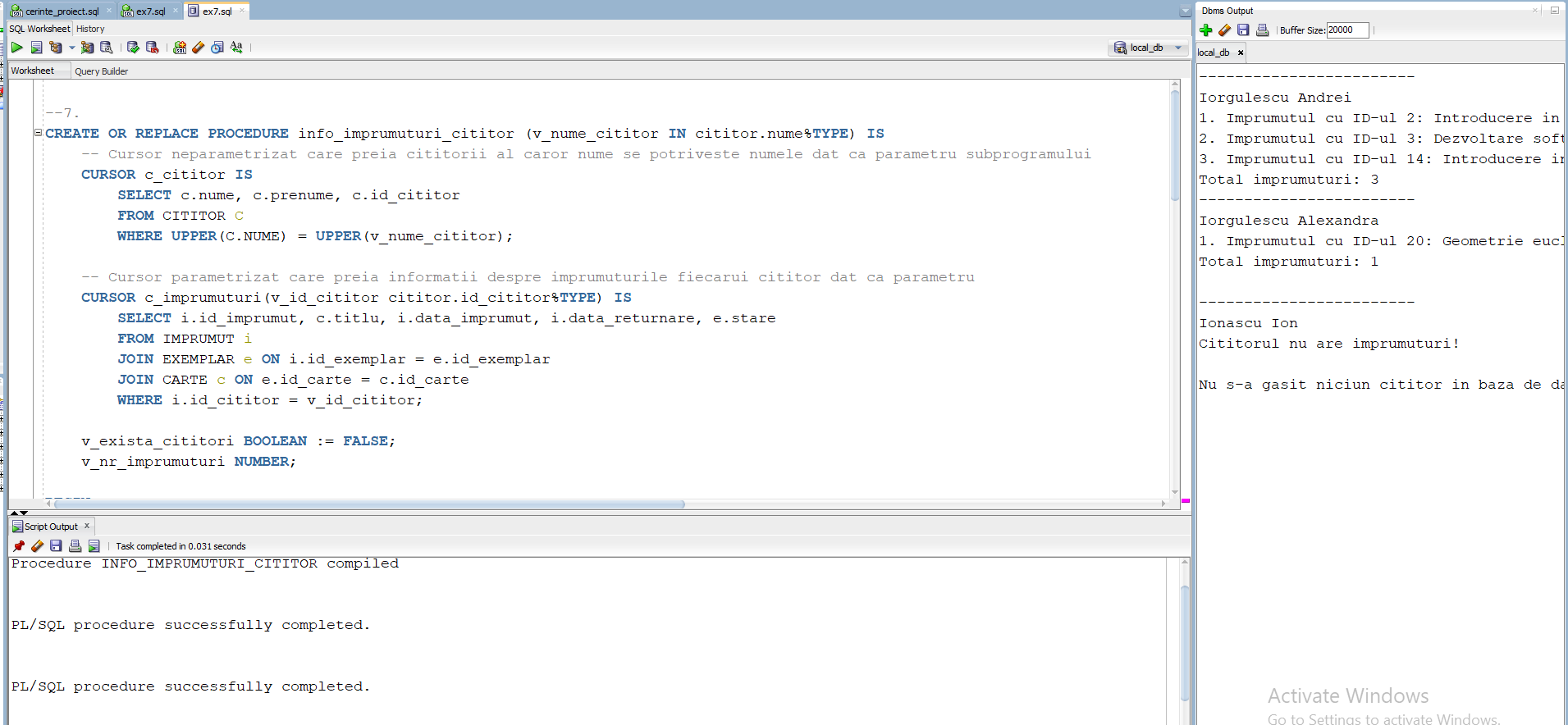
--verificam pentru un cititor care nu exista in baza de date

begin

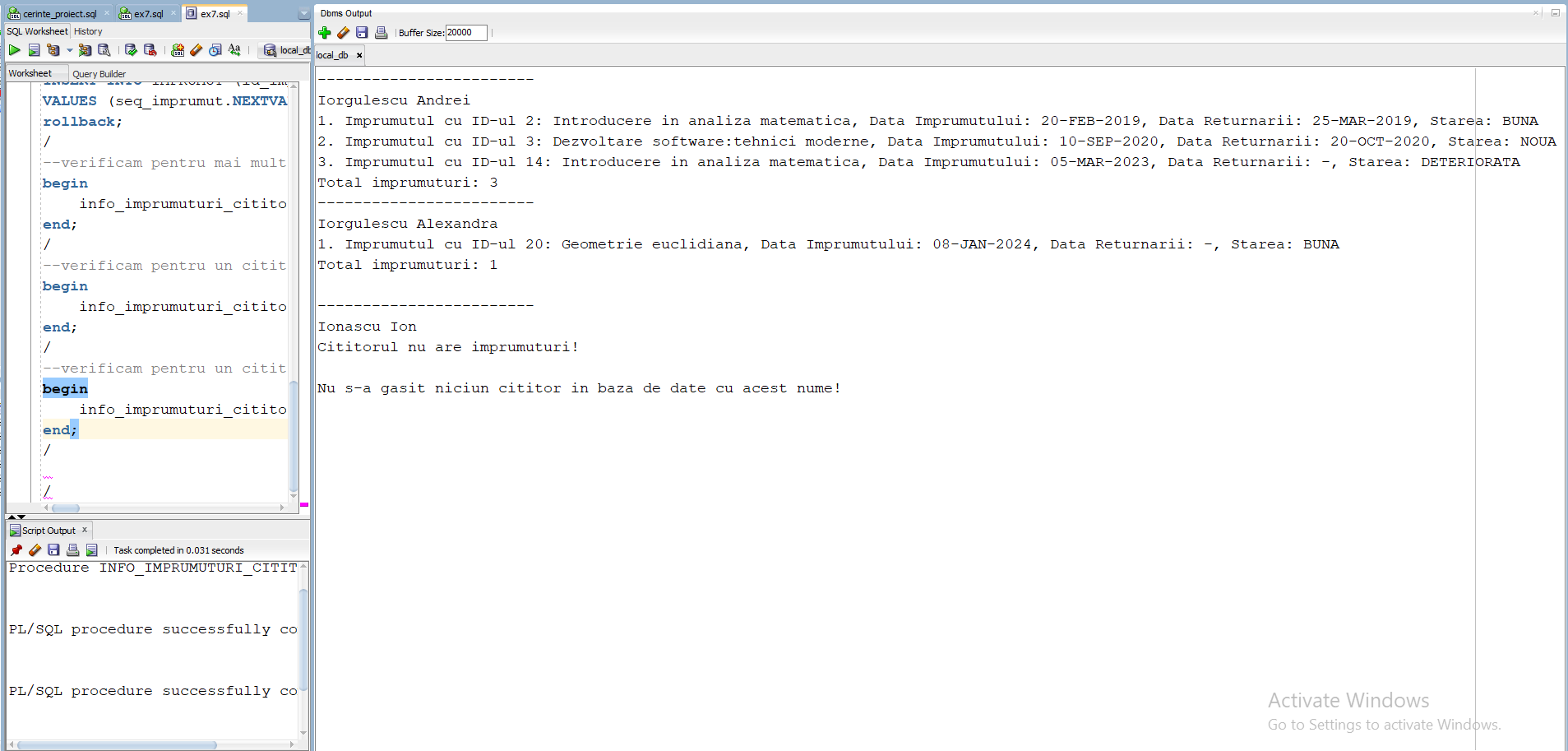
    info\_imprumuturi\_cititor('Voicu');

end;

În primul print-screen se observă codul compilat.



În al doilea print-screen se observă rezultatul.



# Funcție cu 3 tabele care să includă definirea a minim 2 excepții

**Cerință**: Biblioteca își dorește să înlocuiască exemplarele cărților care sunt deteriorate și vrea să contacteze editorii pentru a plasa o comandă cu numărul de exemplare noi de care are nevoie din fiecare carte. Pentru a face asta trebuie întâi să creeze o centralizare a situației cărților. Scrieți un subprogram stocat independent de tip funcție care să primească în parametru numele unei edituri și să afișeze pentru aceasta o listă formată din titlurile cărților, anul publicării și numărul de exemplare care au starea deteriorata. Tratați toate excepțiile care apar.

CREATE OR REPLACE FUNCTION NrExemplareDeteriorate(

    v\_nume\_editura editura.nume\_editura%TYPE

) RETURN VARCHAR2

IS

    v\_lista\_carti VARCHAR2(4000);

    v\_nr\_edituri NUMBER;

    custom\_no\_data\_found **exception**;

    custom\_too\_many\_rows **exception**;

    fara\_exemplare\_deteriorate **exception**;

BEGIN

        SELECT COUNT(\*)

        INTO v\_nr\_edituri

        FROM editura

        WHERE nume\_editura = v\_nume\_editura;

        IF v\_nr\_edituri = 0 THEN

**RAISE custom\_no\_data\_found;**

        ELSIF v\_nr\_edituri > 1 THEN

**RAISE custom\_too\_many\_rows;**

        END IF;

    FOR i IN (

        SELECT c.titlu, c.an\_publicare, COUNT(e.id\_exemplar) AS numar\_exemplare\_deteriorate

        FROM CARTE c

        JOIN EXEMPLAR e ON c.id\_carte = e.id\_carte

        JOIN EDITURA ed ON c.id\_editura = ed.id\_editura

        WHERE ed.nume\_editura = v\_nume\_editura AND e.stare = UPPER('deteriorata')

        GROUP BY c.titlu, c.an\_publicare

    )

    LOOP

        v\_lista\_carti := v\_lista\_carti || 'Titlu: ' || i.titlu || ', An Publicare: ' || i.an\_publicare || ', Numar Exemplare Deteriorate: ' || i.numar\_exemplare\_deteriorate || CHR(10);

    END LOOP;

    --verific daca lista returnata este goala

    IF v\_lista\_carti IS NULL THEN

**RAISE fara\_exemplare\_deteriorate;**

    END IF;

    RETURN v\_lista\_carti;

**EXCEPTION**

        WHEN custom\_no\_data\_found THEN

            DBMS\_OUTPUT.PUT\_LINE ('Nu exista editura in baza de date cu acest nume!');

            RETURN NULL;

        WHEN custom\_too\_many\_rows THEN

            DBMS\_OUTPUT.PUT\_LINE('Exista mai multe edituri in baza de date cu acest nume!');

            RETURN NULL;

        WHEN fara\_exemplare\_deteriorate THEN

            DBMS\_OUTPUT.PUT\_LINE('Nu exista carti deteriorate pentru editura specificata.');

            RETURN NULL;

        WHEN OTHERS THEN

            DBMS\_OUTPUT.PUT\_LINE('Alta Eroare!');

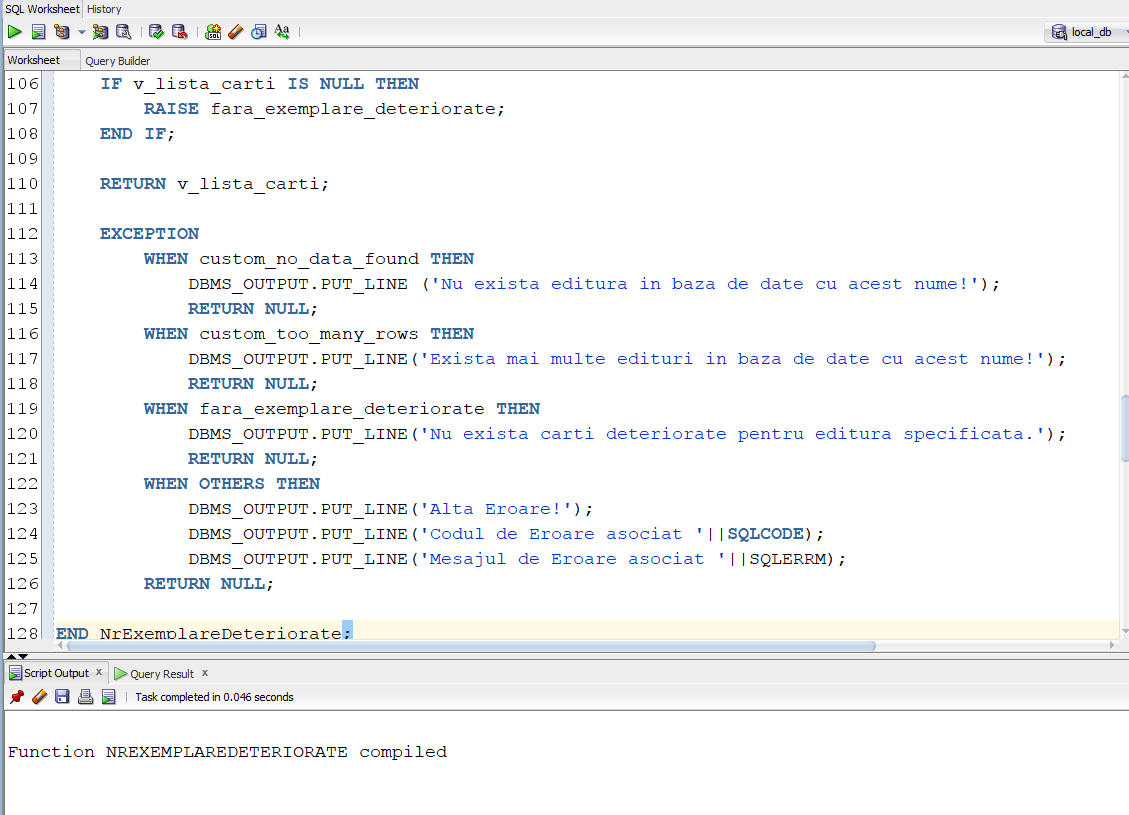
            DBMS\_OUTPUT.PUT\_LINE('Codul de Eroare asociat '||SQLCODE);

            DBMS\_OUTPUT.PUT\_LINE('Mesajul de Eroare asociat '||SQLERRM);

        RETURN NULL;

END NrExemplareDeteriorate;

Captura de ecran cu compilarea funcției:

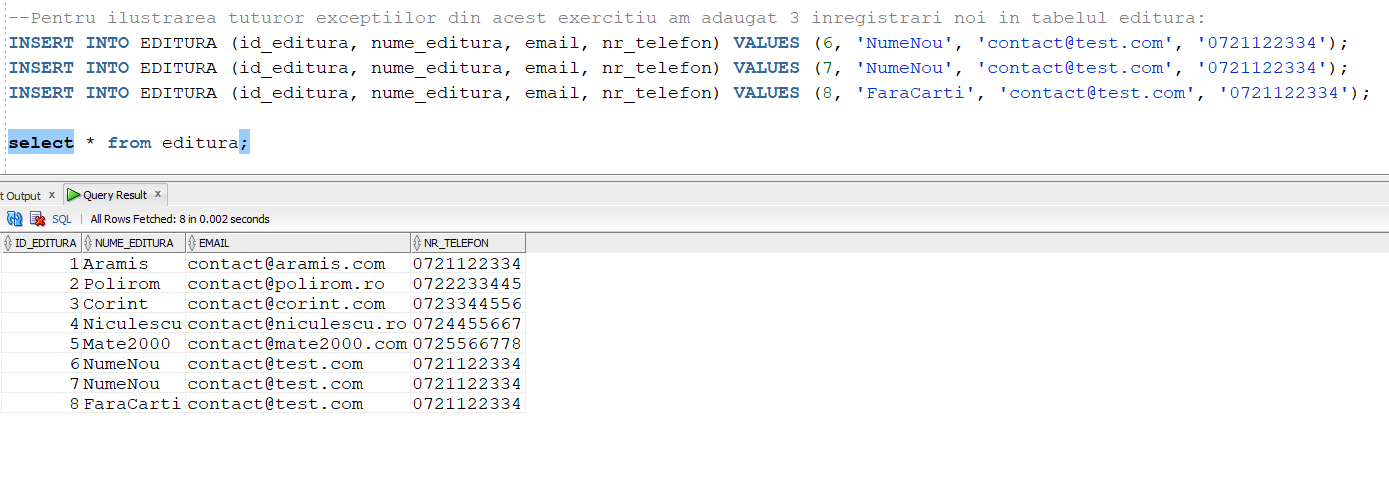


Pentru exemplificarea tuturor excepțiilor tratate în acest exercițiu am făcut 3 inserări noi în tabelul Editură, după cum se observă și în captura de ecran de mai jos:

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (6, 'NumeNou', 'contact@test.com', '0721122334');

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (7, 'NumeNou', 'contact@test.com', '0721122334');

INSERT INTO EDITURA (id\_editura, nume\_editura, email, nr\_telefon) VALUES (8, 'FaraCarti', 'contact@test.com', '0721122334');



Blocurile PL/SQL în care apelez funcția și tratez fiecare caz în parte:

--cazul fericit

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('EDITURA Aramis');

    DBMS\_OUTPUT.NEW\_LINE;

    DBMS\_OUTPUT.PUT\_LINE(NrExemplareDeteriorate('Aramis'));

END;

/

--custom\_no\_data\_found exception

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('EDITURA NUEXISTA');

    DBMS\_OUTPUT.NEW\_LINE;

    DBMS\_OUTPUT.PUT\_LINE(NrExemplareDeteriorate('NUEXISTA'));

END;

/

--custom\_too\_many\_rows exception

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('EDITURA NumeNou');

    DBMS\_OUTPUT.NEW\_LINE;

    DBMS\_OUTPUT.PUT\_LINE(NrExemplareDeteriorate('NumeNou'));

END;

/

--fara\_exemplare\_deteriorate exception

BEGIN

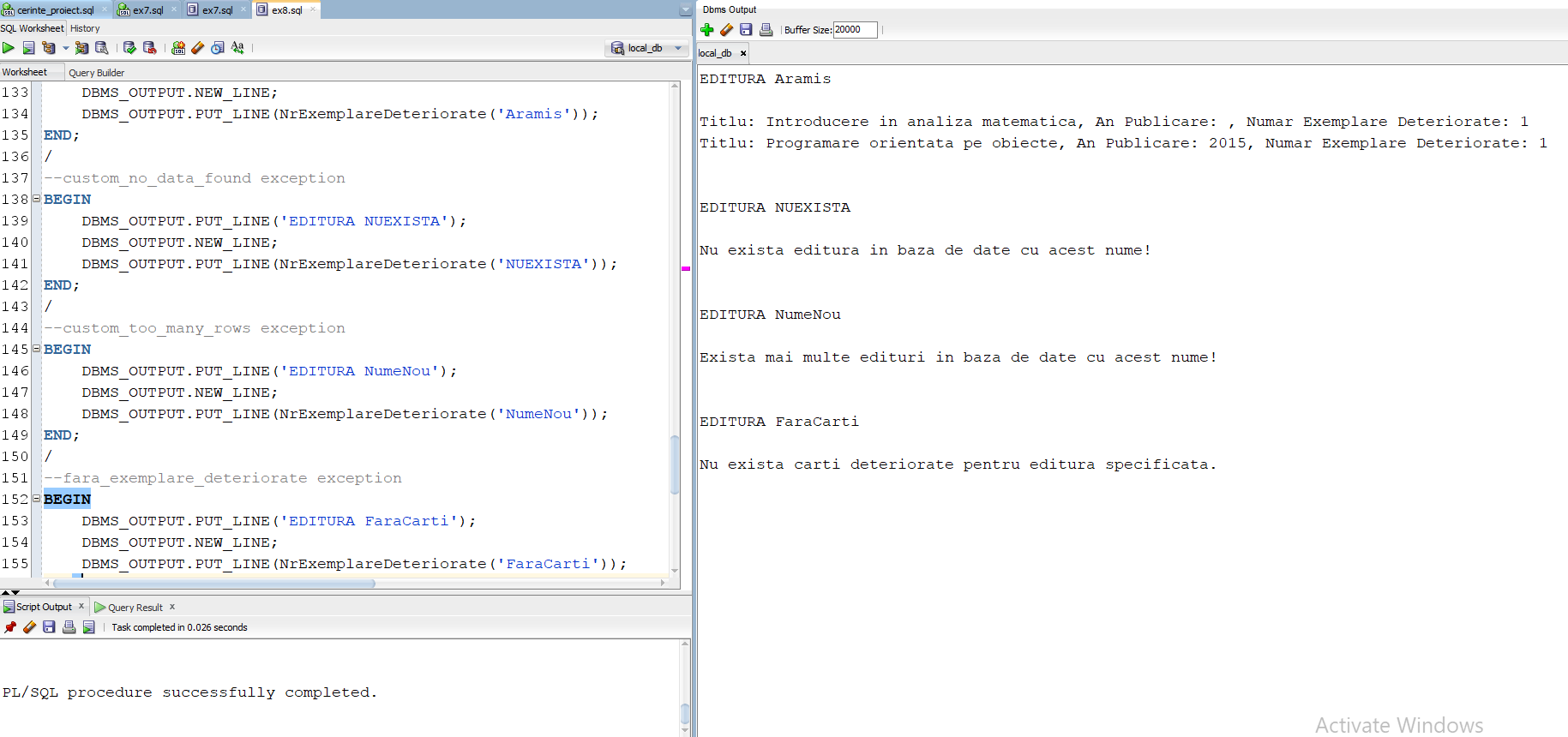
    DBMS\_OUTPUT.PUT\_LINE('EDITURA FaraCarti');

    DBMS\_OUTPUT.NEW\_LINE;

    DBMS\_OUTPUT.PUT\_LINE(NrExemplareDeteriorate('FaraCarti'));

END;

Exemplu cu tratarea tuturor cazurilor:



# Subprogram stocat independent de tip procedură cu 5 tabele

**Cerința**: Scrieți un subprogram stocat independent de tip procedură care să primească ca parametru numele unui autor și să afișeze o listă cu titlurile cărților lui existente în bibliotecă, alături de categoriile din care acestea fac parte, precum și numărul de exemplare, numărul de împrumuturi și un procentaj care să arate cât de căutate au fost cărțile acestui autor raportat la celelalte împrumuturi înregistrate, mai exact cât la sută din totalul împrumuturilor au aparținut unei cărți ce îi aparține.

CREATE OR REPLACE PROCEDURE RaportAutor(

    p\_nume\_autor VARCHAR2

) AS

    TYPE info\_carte IS RECORD (

        titlu VARCHAR2(100),

        categ VARCHAR2(100),

        nr\_exemplare NUMBER,

        nr\_imprumuturi NUMBER,

        popularitate NUMBER(10,2)

        --procentul pe care il reprezinta din nr total de imprumuturi

    );

    TYPE lista\_carti\_autor IS TABLE OF info\_carte;

    v\_id\_autor AUTOR.id\_autor%TYPE;

    v\_total\_imprumuturi NUMBER;

    v\_rez lista\_carti\_autor := lista\_carti\_autor();

    nu\_exista\_carti **exception**;

    PRAGMA EXCEPTION\_INIT (nu\_exista\_carti, -20002);

BEGIN

    SELECT id\_autor INTO v\_id\_autor

    FROM AUTOR

    WHERE UPPER(nume) = UPPER(p\_nume\_autor);

    SELECT COUNT(\*) INTO v\_total\_imprumuturi

    FROM IMPRUMUT;

    FOR rec IN (

        SELECT

            c.titlu,

            cg.nume\_categorie,

            COUNT(DISTINCT e.id\_exemplar) AS numar\_exemplare,

            COUNT(i.id\_imprumut) AS numar\_imprumuturi,

            ROUND(COUNT(i.id\_imprumut) \* 100 / v\_total\_imprumuturi, 2) AS procentaj

        FROM SCRIERE s

        JOIN CARTE c ON s.id\_carte = c.id\_carte

        JOIN CATEGORIE cg ON c.id\_categorie = cg.id\_categorie

        LEFT JOIN EXEMPLAR e ON c.id\_carte = e.id\_carte

        LEFT JOIN IMPRUMUT i ON e.id\_exemplar = i.id\_exemplar

        WHERE s.id\_autor = v\_id\_autor

        GROUP BY c.titlu, cg.nume\_categorie

    ) LOOP

        v\_rez.EXTEND;

        v\_rez(v\_rez.LAST).titlu := rec.titlu;

        v\_rez(v\_rez.LAST).categ := rec.nume\_categorie;

        v\_rez(v\_rez.LAST).nr\_exemplare := rec.numar\_exemplare;

        v\_rez(v\_rez.LAST).nr\_imprumuturi := rec.numar\_imprumuturi;

        v\_rez(v\_rez.LAST).popularitate := rec.procentaj;

    END LOOP;

    IF v\_rez.COUNT = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20002, 'Biblioteca nu detine nicio carte ce apartine acestui autor!');

    END IF;

    FOR i IN 1..v\_rez.COUNT LOOP

        DBMS\_OUTPUT.PUT\_LINE('Titlu: ' || v\_rez(i).titlu || ', Categorie: ' || v\_rez(i).categ || ', Numar Exemplare: ' || v\_rez(i).nr\_exemplare || ', Numar Imprumuturi: ' || v\_rez(i).nr\_imprumuturi || ', Procentaj Popularitate: ' || v\_rez(i).popularitate || '%');

    END LOOP;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun autor cu numele dat in baza de date!');

    WHEN TOO\_MANY\_ROWS THEN

        DBMS\_OUTPUT.PUT\_LINE ('In baza de date exista mai multi autori cu acest nume!');

    WHEN ZERO\_DIVIDE THEN

        DBMS\_OUTPUT.PUT\_LINE ('Nu exista imprumuturi inregistrate in baza de date! Procentajul nu poate fi calculat corect prin impartirea la 0!');

    WHEN nu\_exista\_carti THEN

        DBMS\_OUTPUT.PUT\_LINE('Codul de Eroare asociat '||SQLCODE);

        DBMS\_OUTPUT.PUT\_LINE('Mesajul de Eroare asociat '||SQLERRM);

    WHEN OTHERS THEN

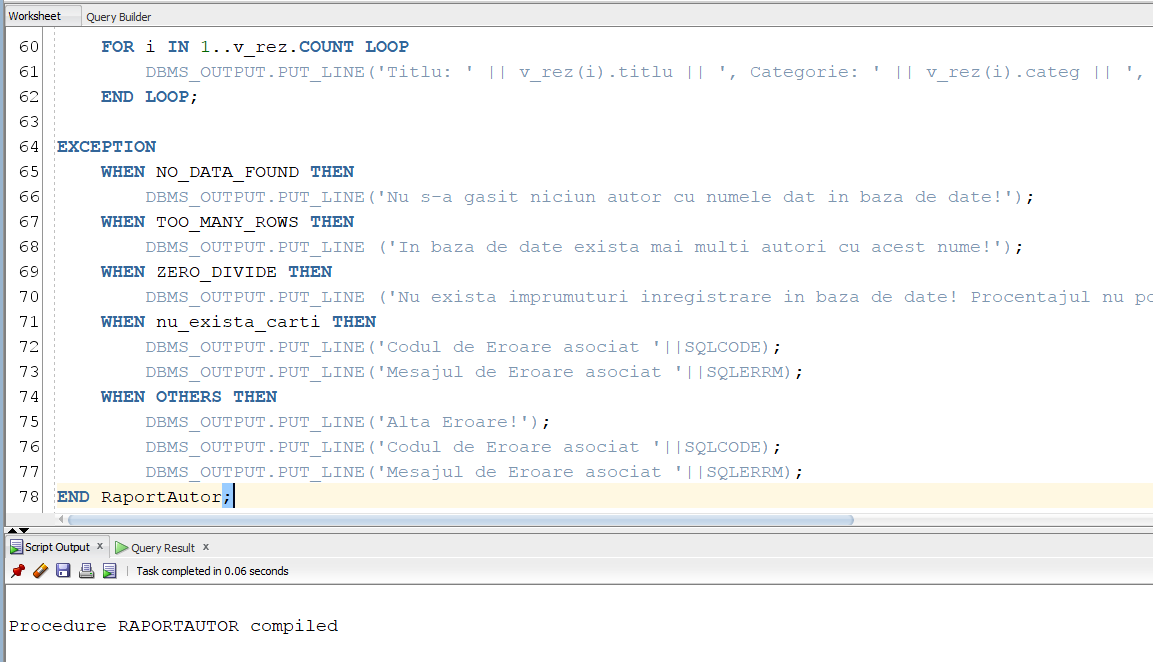
        DBMS\_OUTPUT.PUT\_LINE('Alta Eroare!');

        DBMS\_OUTPUT.PUT\_LINE('Codul de Eroare asociat '||SQLCODE);

        DBMS\_OUTPUT.PUT\_LINE('Mesajul de Eroare asociat '||SQLERRM);

END RaportAutor;

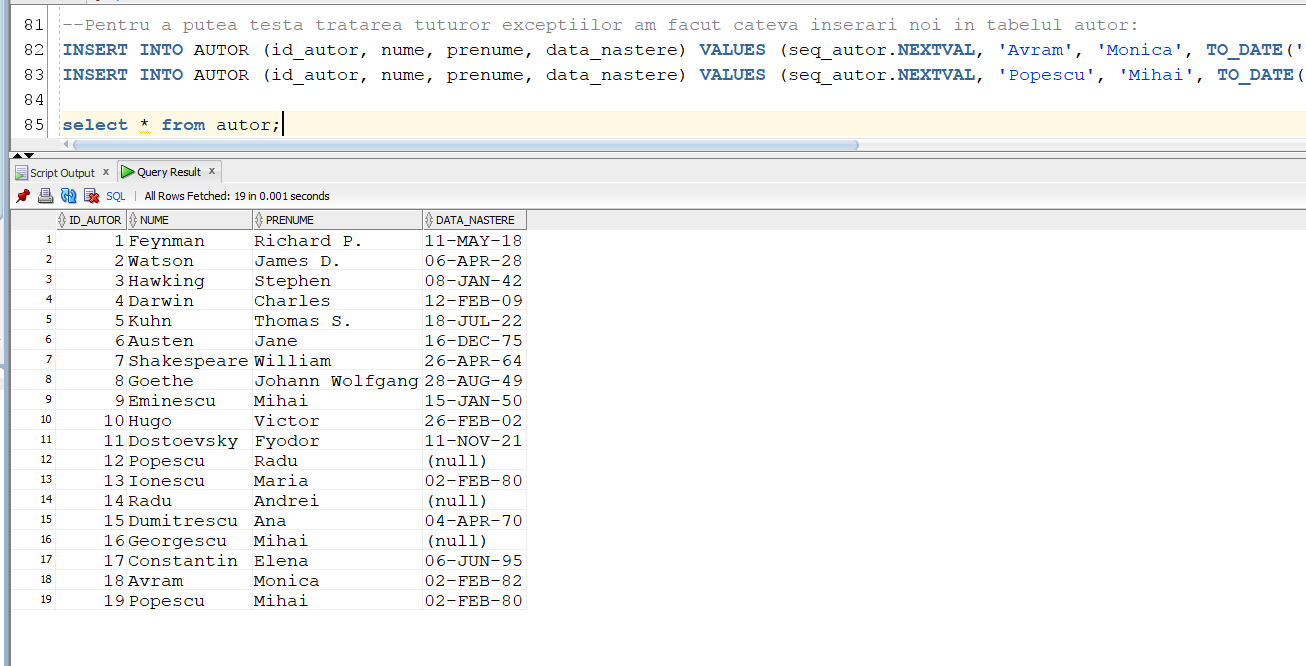
Captură de ecran cu subprogramul compilat:



Pentra a exemplifica toate excepțiile au fost inserate linii noi in tabelul AUTOR:

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Avram', 'Monica', TO\_DATE('02-02-1982', 'DD-MM-YYYY'));

INSERT INTO AUTOR (id\_autor, nume, prenume, data\_nastere) VALUES (seq\_autor.NEXTVAL, 'Popescu', 'Mihai', TO\_DATE('02-02-1980', 'DD-MM-YYYY'));



Exemplificarea tuturor excepțiilor:

BEGIN

    --cazul fericit

    DBMS\_OUTPUT.PUT\_LINE('NUMELE AUTORULUI: Ionescu');

    RaportAutor('Ionescu');

    DBMS\_OUTPUT.NEW\_LINE;

**--NO\_DATA\_FOUND**

    DBMS\_OUTPUT.PUT\_LINE('NUMELE AUTORULUI: Voicu');

    RaportAutor('Voicu');

    DBMS\_OUTPUT.NEW\_LINE;

**--TOO\_MANY\_ROWS**

    DBMS\_OUTPUT.PUT\_LINE('NUMELE AUTORULUI: Popescu');

    RaportAutor('Popescu');

    DBMS\_OUTPUT.NEW\_LINE;

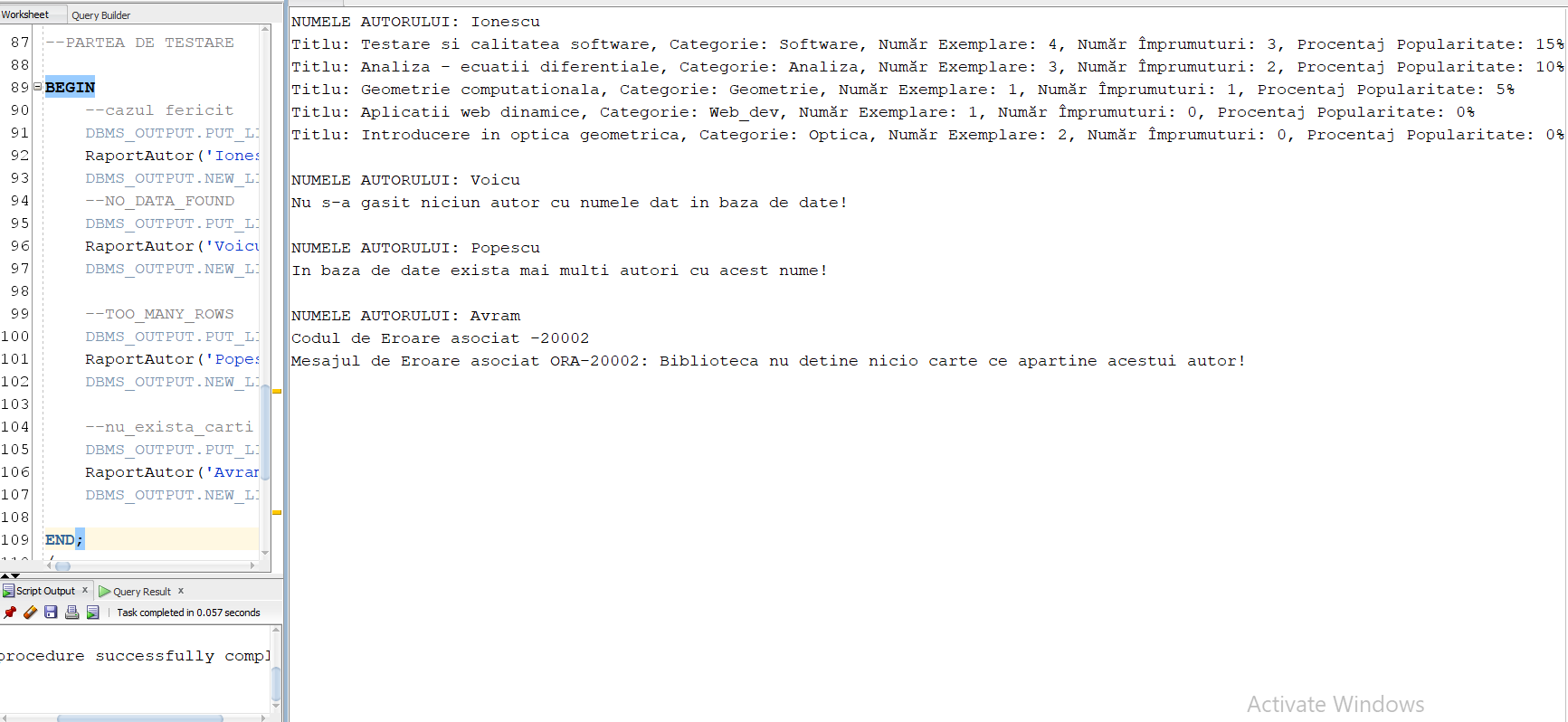
    --nu\_exista\_carti exception

    DBMS\_OUTPUT.PUT\_LINE('NUMELE AUTORULUI: Avram');

    RaportAutor('Avram');

    DBMS\_OUTPUT.NEW\_LINE;

END;

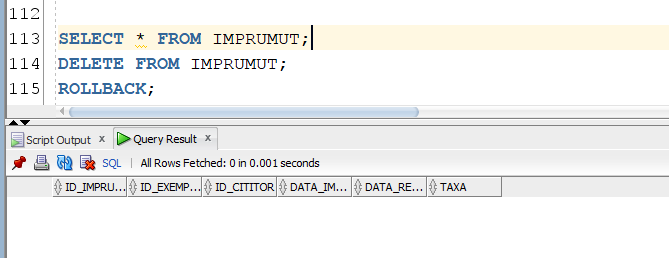


Testarea excepției zero\_divide am făcut-o după ce am mai realizat niște modificări tabelei Împrumut, după cum urmează:

SELECT \* FROM IMPRUMUT;

DELETE FROM IMPRUMUT;

ROLLBACK;



Testarea excepției **ZERO\_DIVIDE**:

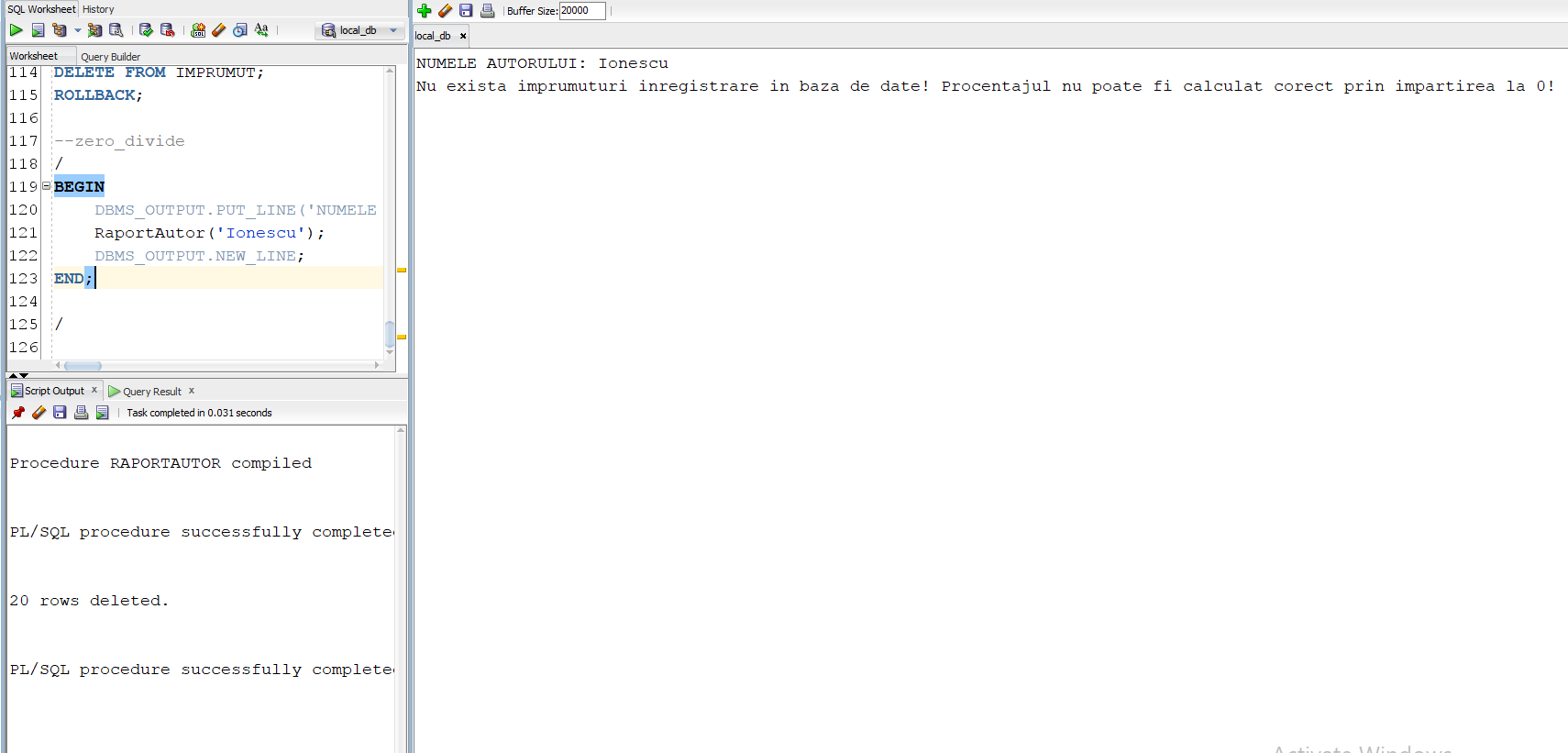
BEGIN

    DBMS\_OUTPUT.PUT\_LINE('NUMELE AUTORULUI: Ionescu');

    RaportAutor('Ionescu');

    DBMS\_OUTPUT.NEW\_LINE;

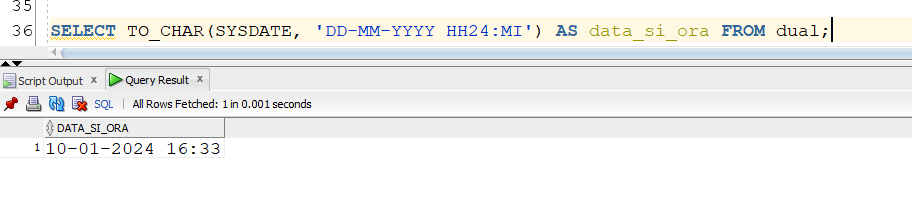
END;

­­

# Trigger de tip LMD la nivel de comandă

**Cerință**: Creați un trigger care să se declanșeze atunci când se actualizează, inserează sau șterge din tabela Împrumut în afara programului cu publicul al bibliotecii. Programul cu publicul al bibliotecii: luni-vineri 08.00-16.00, sâmbătă 09.00-13.00, duminică: închis.

Ora la care am făcut verificările:



CREATE OR REPLACE TRIGGER program\_biblioteca

    BEFORE INSERT OR DELETE OR UPDATE ON IMPRUMUT

BEGIN

    IF (TO\_CHAR(SYSDATE, 'D') = 1) OR

       ((TO\_CHAR(SYSDATE, 'D') = 7) AND TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 9 AND 12) OR

       (TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 8 AND 15)

    THEN

        DBMS\_OUTPUT.PUT\_LINE('PROGRAM CU PUBLICUL');

        DBMS\_OUTPUT.PUT\_LINE('Luni-Vineri: 08.00-16.00');

        DBMS\_OUTPUT.PUT\_LINE('Sambata: 09.00-13.00');

        DBMS\_OUTPUT.PUT\_LINE('Duminica: INCHIS!');

        IF INSERTING THEN

            RAISE\_APPLICATION\_ERROR(-20001, 'Nu se pot face imprumuturi in afara programului cu publicul!');

        ELSIF DELETING THEN

            RAISE\_APPLICATION\_ERROR(-20002, 'Nu se pot anula imprumuturi in afara programului cu publicul!');

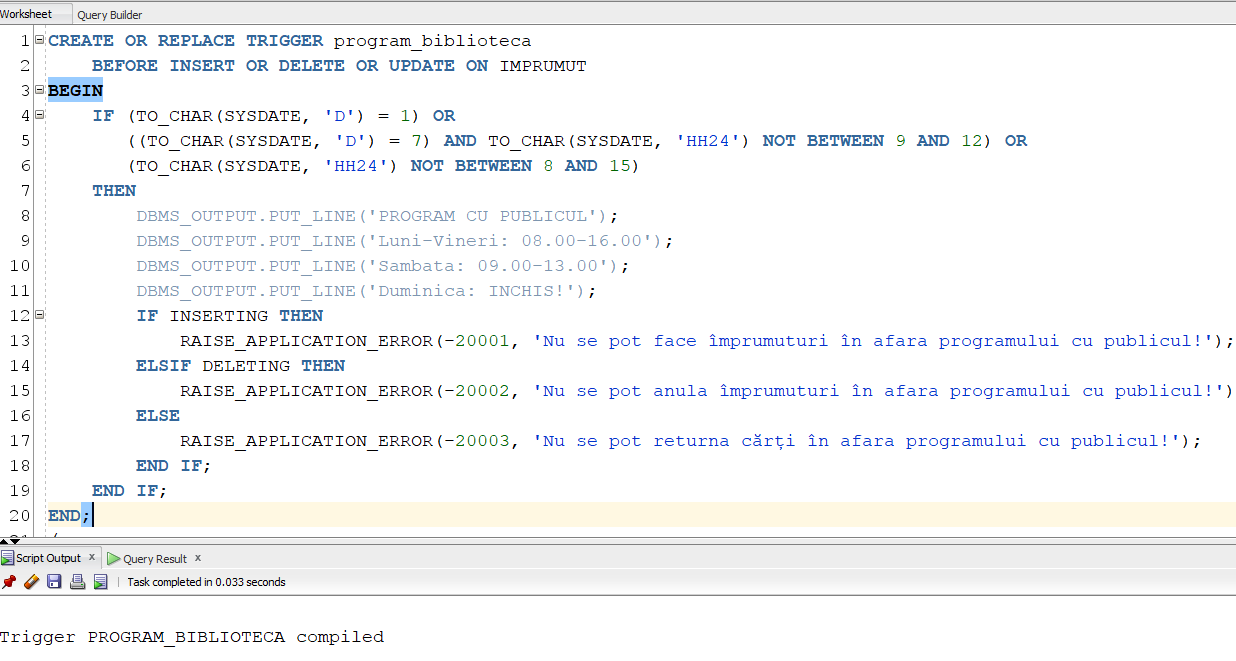
        ELSE

            RAISE\_APPLICATION\_ERROR(-20003, 'Nu se pot returna carti in afara programului cu publicul!');

        END IF;

    END IF;

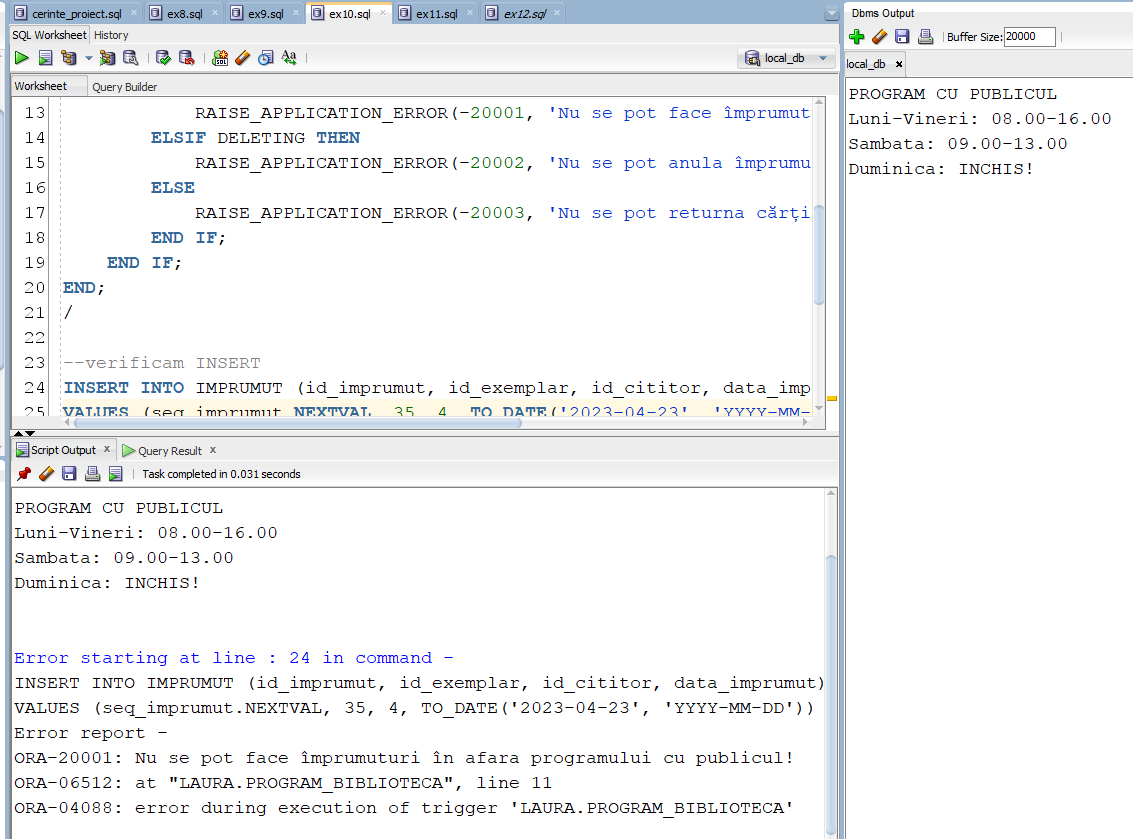
END;



**Verificăm pentru insert**

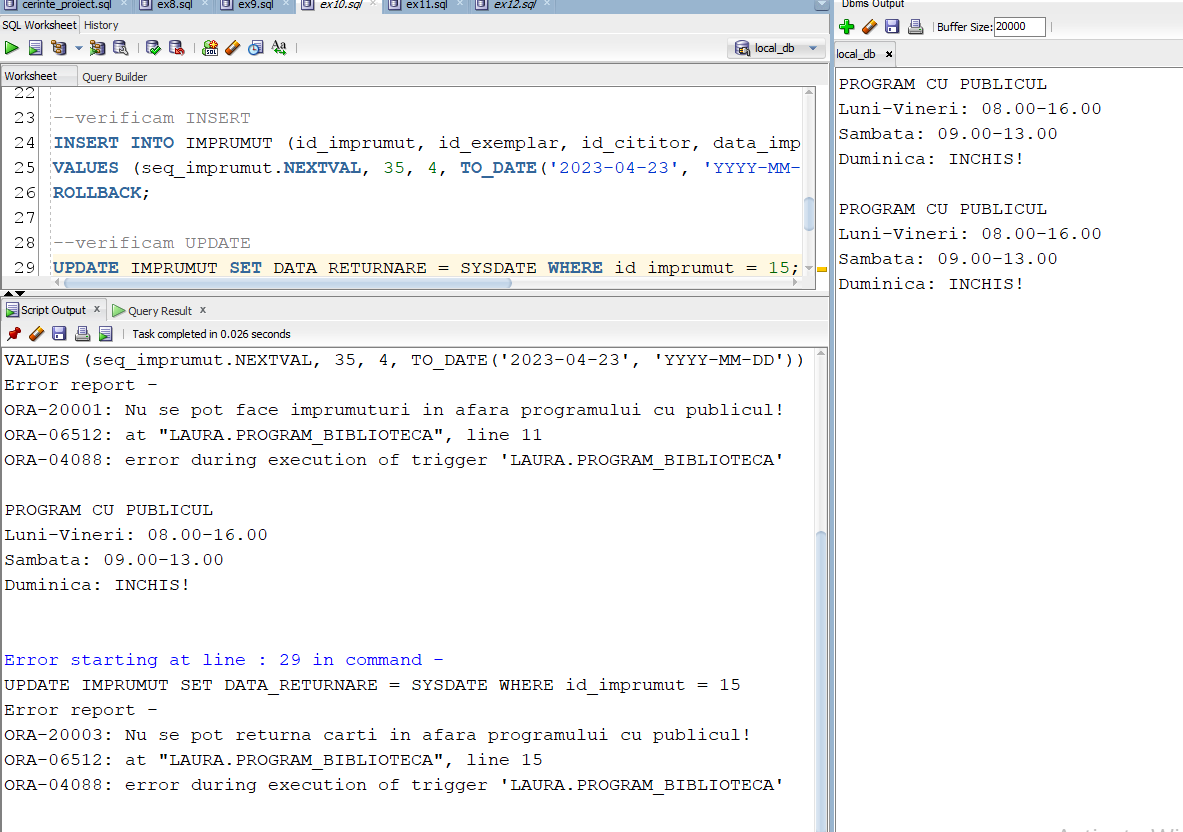
INSERT INTO IMPRUMUT (id\_imprumut, id\_exemplar, id\_cititor, data\_imprumut)

VALUES (seq\_imprumut.NEXTVAL, 35, 4, TO\_DATE('2023-04-23', 'YYYY-MM-DD'));



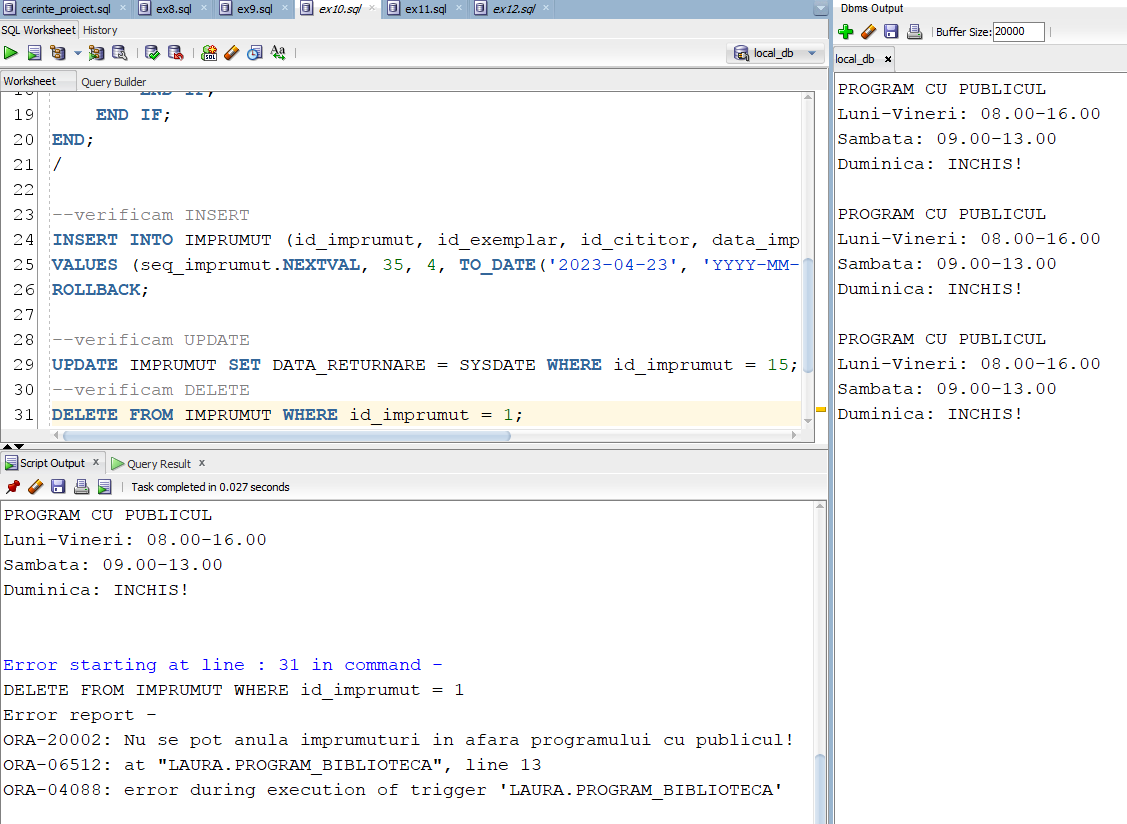
**Verificăm pentru update**

UPDATE IMPRUMUT SET DATA\_RETURNARE = SYSDATE WHERE id\_imprumut = 15;



**Verificăm pentru delete**

DELETE FROM IMPRUMUT WHERE id\_imprumut = 1;



# Trigger de tip LMD la nivel de linie

**Cerință**: Creați un trigger care să se declanșeze atunci când se încearcă modificarea coloanei data\_returnare din tabelul împrumut cu o dată mai mare decât cea din ziua curentă.

CREATE OR REPLACE TRIGGER data\_retur\_incorecta

BEFORE UPDATE OF data\_returnare ON imprumut

FOR EACH ROW

BEGIN

    IF(:NEW.data\_returnare > SYSDATE)

        THEN RAISE\_APPLICATION\_ERROR(-20000,

        'Ati introdus o valoare invalida pentru data returnarii!

        Aceasta nu poate fi mai mare decat data zilei curente.');

    END IF;

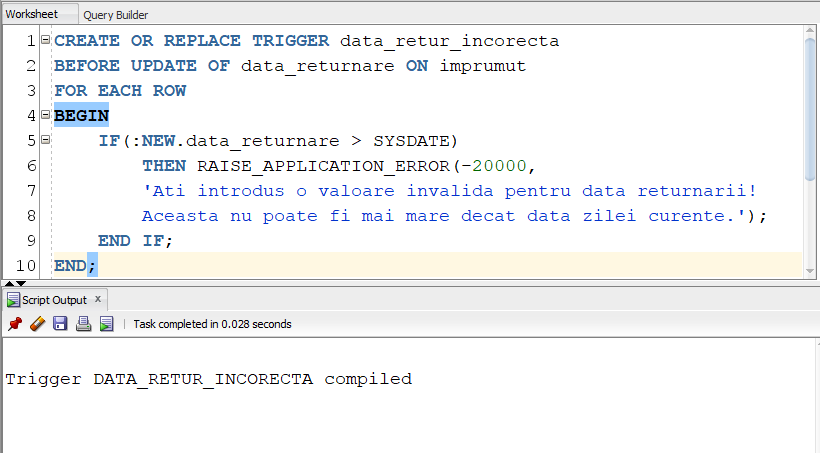
END;

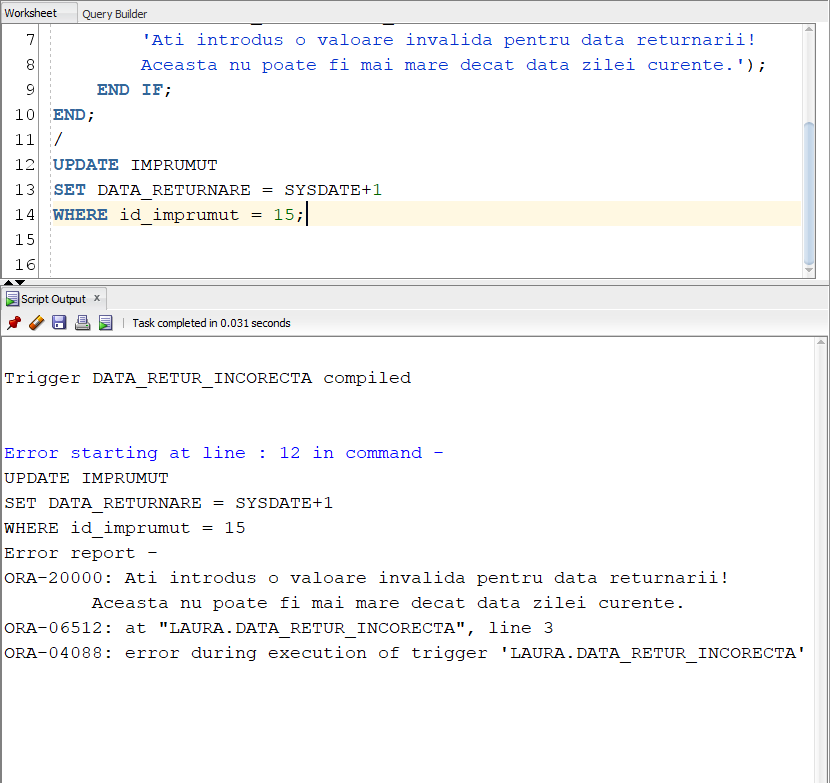
--verificare

UPDATE IMPRUMUT

SET DATA\_RETURNARE = SYSDATE+1

WHERE id\_imprumut = 15;

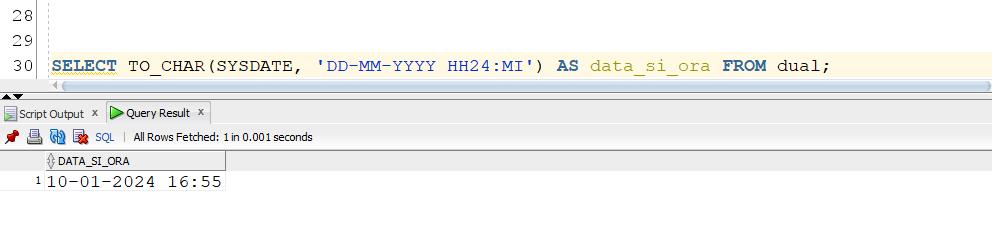




# Trigger de tip LDD

**Cerință**: Creați un trigger care să se declanșeze atunci când se încearcă crearea, ștergerea sau alterarea tabelelor din baza de date atunci când biblioteca nu se află în timpul programului de lucru. Program: luni-sâmbătă: 08.00-16.00, duminică: închis.

Ora la care s-a făcut verificarea:



CREATE OR REPLACE TRIGGER modificare\_BD

BEFORE CREATE OR ALTER OR DROP

ON SCHEMA

DECLARE

    ora\_inceput NUMBER := 8;

    ora\_sfarsit NUMBER := 15;

    zi\_nelucratoare NUMBER := 1; -- duminica

BEGIN

    IF TO\_NUMBER(TO\_CHAR(SYSDATE, 'D')) = zi\_nelucratoare OR

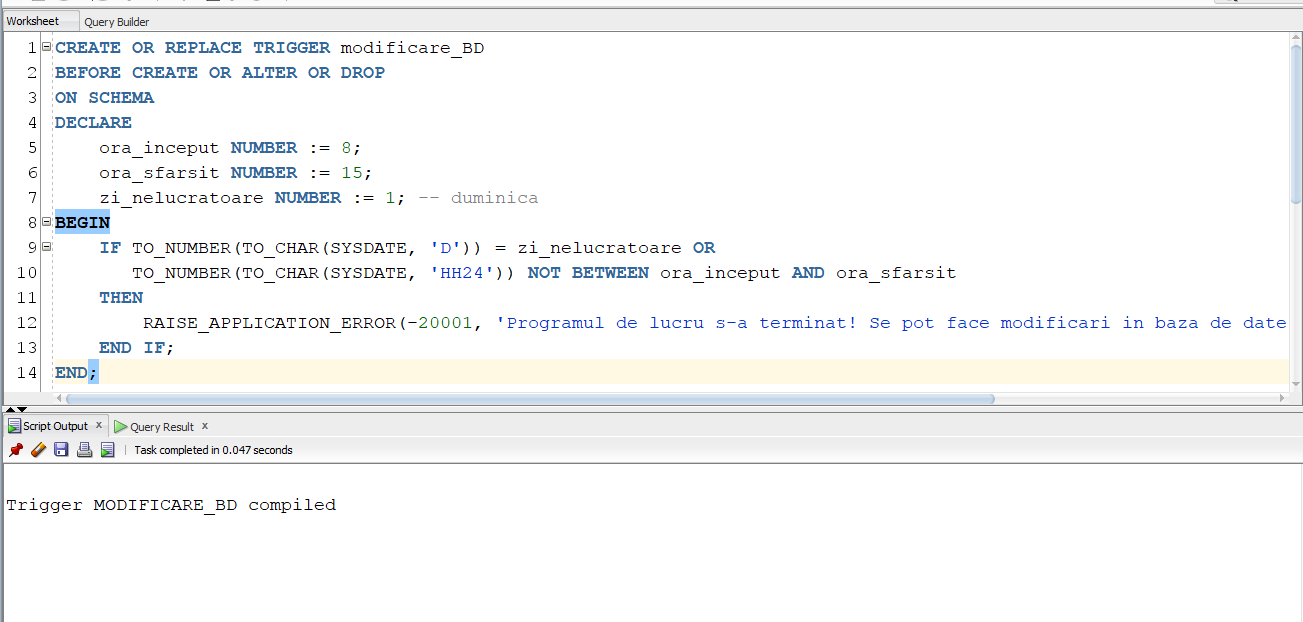
       TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) NOT BETWEEN ora\_inceput AND ora\_sfarsit

    THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Programul de lucru s-a terminat! Se pot face modificari in baza de date doar in intervalul 08.00-16.00');

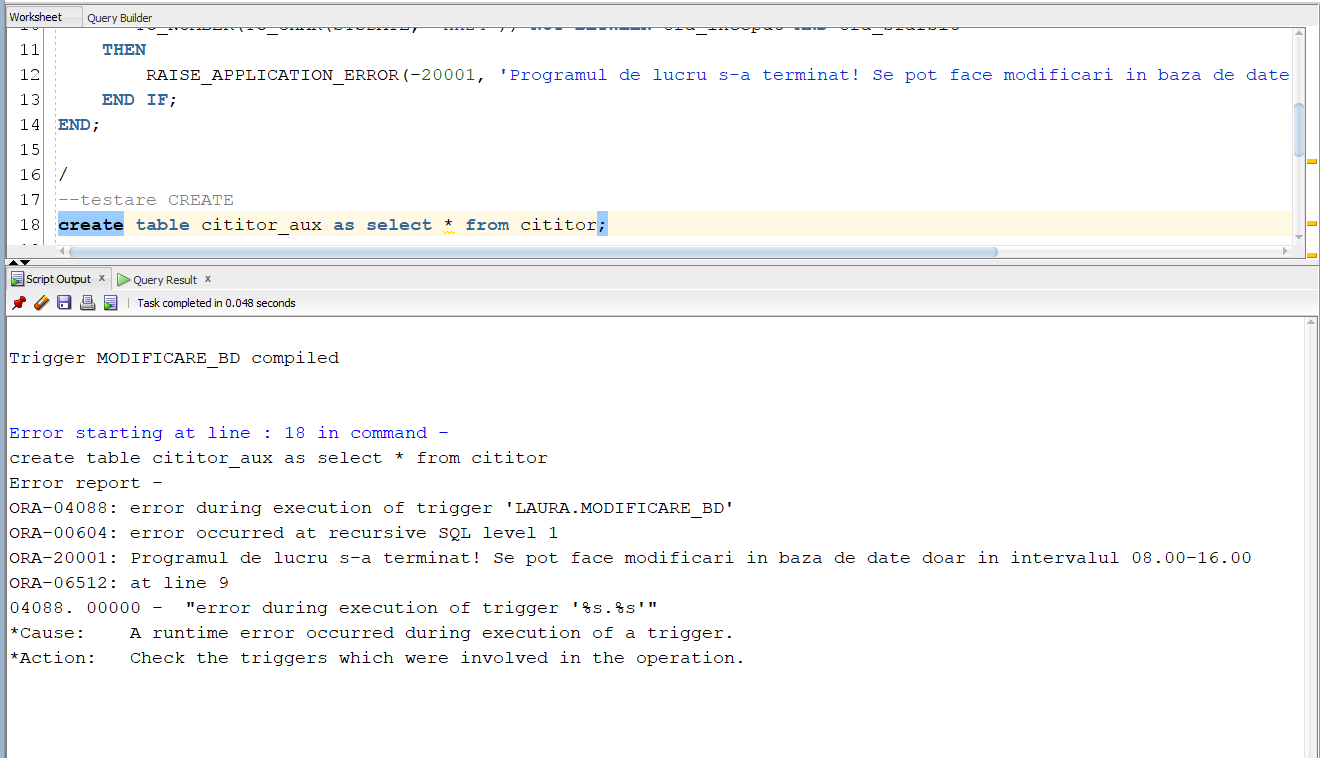
    END IF;

END;



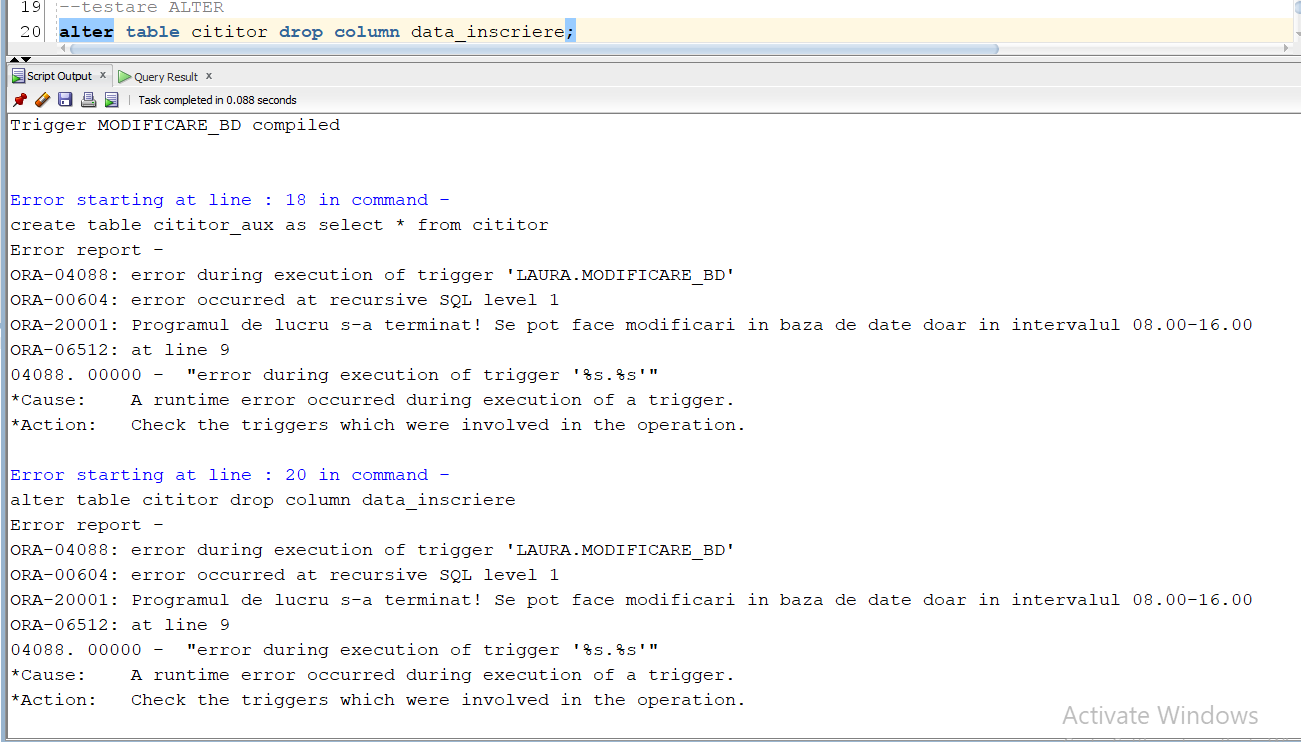
**Verificăm pentru CREATE**

create table cititor\_aux as select \* from cititor;



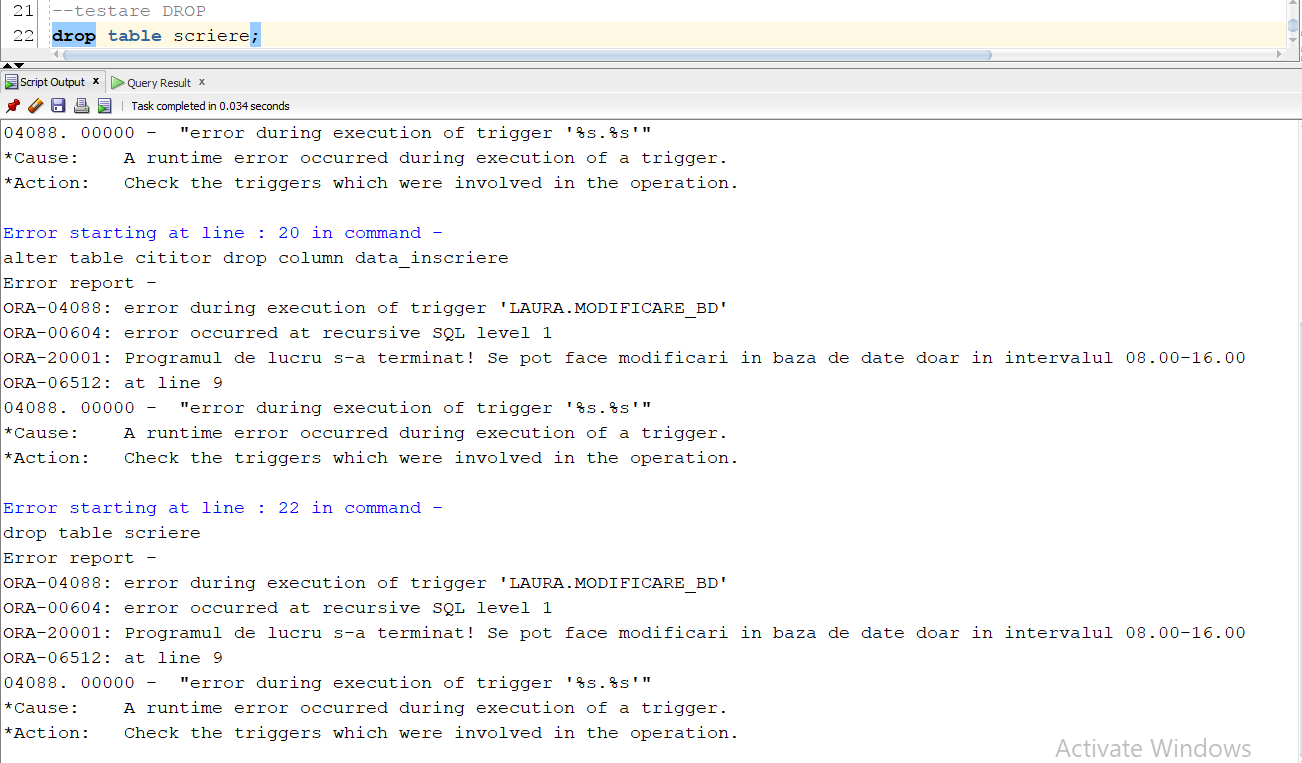
**Verificăm pentru ALTER**

alter table cititor drop column data\_inscriere;



**Verificăm pentru DELETE**

drop table scriere;



# Pachet care conține toate obiectele definite în proiect.

CREATE OR REPLACE PACKAGE proiectSGBD AS

    PROCEDURE exercitiul6;

    PROCEDURE exercitiul7(v\_nume\_cititor cititor.nume%TYPE);

    FUNCTION exercitiul8(v\_nume\_editura editura.nume\_editura%TYPE) RETURN VARCHAR2;

    PROCEDURE exercitiul9(p\_nume\_autor VARCHAR2);

END proiectSGBD;

CREATE OR REPLACE PACKAGE BODY proiectSGBD AS

    PROCEDURE exercitiul6 IS

    TYPE tabel\_imbricat\_sectiune IS TABLE OF SECTIUNE%ROWTYPE;

    t\_sectiune tabel\_imbricat\_sectiune := tabel\_imbricat\_sectiune();

    TYPE vector\_categorii IS VARRAY(20) OF CATEGORIE%ROWTYPE;

    v\_categorii vector\_categorii;

    TYPE tabel\_indexat\_carti IS TABLE OF CARTE.titlu%TYPE INDEX BY PLS\_INTEGER;

    t\_carti tabel\_indexat\_carti;

    BEGIN

        SELECT \*

        BULK COLLECT INTO t\_sectiune

        FROM SECTIUNE;

        FOR i IN 1..t\_sectiune.COUNT LOOP

            DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------------------------');

            SELECT \*

            BULK COLLECT INTO v\_categorii

            FROM CATEGORIE

            WHERE ID\_SECTIUNE = t\_sectiune(i).id\_sectiune;

            IF v\_categorii.COUNT > 0 THEN

                DBMS\_OUTPUT.PUT\_LINE(i||'. SECTIUNEA ' || t\_sectiune(i).nume\_sectiune || ' contine urmatoarele categorii: ');

                FOR j IN 1..v\_categorii.COUNT LOOP

                    SELECT titlu

                    BULK COLLECT INTO t\_carti

                    FROM CARTE

                    WHERE ID\_CATEGORIE = v\_categorii(j).id\_categorie;

                    IF t\_carti.COUNT > 0 THEN

                        DBMS\_OUTPUT.PUT\_LINE('      ' || j || '. CATEGORIA ' || UPPER( v\_categorii(j).nume\_categorie) || ' contine urmatoarele carti: ');

                        FOR k IN 1..t\_carti.COUNT LOOP

                            DBMS\_OUTPUT.PUT\_LINE('                  --> CARTE: ' || t\_carti(k));

                        END LOOP;

ELSE

                        DBMS\_OUTPUT.PUT\_LINE('      ' || j || '. CATEGORIA ' || UPPER(v\_categorii(j).nume\_categorie) || ' nu contine nicio carte inca! ');

                    END IF;

                END LOOP;

            ELSE

                DBMS\_OUTPUT.PUT\_LINE(i||'. SECTIUNEA '|| UPPER(t\_sectiune(i).nume\_sectiune) || ' nu contine nicio categorie inca! ');

            END IF;

        END LOOP;

        DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------------------------');

        DBMS\_OUTPUT.PUT\_LINE(' ');

    END exercitiul6;

    PROCEDURE exercitiul7 (v\_nume\_cititor cititor.nume%TYPE) IS

    -- Cursor neparametrizat care preia cititorii al caror nume se potriveste numele dat ca parametru subprogramului

    CURSOR c\_cititor IS

        SELECT c.nume, c.prenume, c.id\_cititor

        FROM CITITOR C

        WHERE UPPER(C.NUME) = UPPER(v\_nume\_cititor);

    -- Cursor parametrizat care preia informatii despre imprumuturile fiecarui cititor dat ca parametru

    CURSOR c\_imprumuturi(v\_id\_cititor cititor.id\_cititor%TYPE) IS

        SELECT i.id\_imprumut, c.titlu, i.data\_imprumut, i.data\_returnare, e.stare

        FROM IMPRUMUT i

        JOIN EXEMPLAR e ON i.id\_exemplar = e.id\_exemplar

        JOIN CARTE c ON e.id\_carte = c.id\_carte

        WHERE i.id\_cititor = v\_id\_cititor;

    v\_exista\_cititori BOOLEAN := FALSE;

    v\_nr\_imprumuturi NUMBER;

BEGIN

    FOR i IN c\_cititor LOOP

        v\_exista\_cititori := TRUE;

        DBMS\_OUTPUT.PUT\_LINE('------------------------');

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

        v\_nr\_imprumuturi := 0;

        FOR j IN c\_imprumuturi(i.id\_cititor) LOOP

            v\_nr\_imprumuturi := v\_nr\_imprumuturi + 1;

            DBMS\_OUTPUT.PUT\_LINE(v\_nr\_imprumuturi || '. Imprumutul cu ID-ul ' || j.id\_imprumut || ': ' || j.titlu

              || ', Data Imprumutului: ' || TO\_CHAR(j.data\_imprumut, 'DD-MON-YYYY')

              || ', Data Returnarii: ' || NVL(TO\_CHAR(j.data\_returnare, 'DD-MON-YYYY'), '-')

              || ', Starea: ' || j.stare);

        END LOOP;

        IF v\_nr\_imprumuturi > 0 THEN

            DBMS\_OUTPUT.PUT\_LINE('Total imprumuturi: ' || v\_nr\_imprumuturi);

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('Cititorul nu are imprumuturi!');

        END IF;

    END LOOP;

    IF NOT v\_exista\_cititori THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun cititor in baza de date cu acest nume!');

    END IF;

    END exercitiul7;

    FUNCTION exercitiul8

    (v\_nume\_editura editura.nume\_editura%TYPE)

    RETURN VARCHAR2

    IS

        v\_lista\_carti VARCHAR2(4000);

        v\_nr\_edituri NUMBER;

        custom\_no\_data\_found exception;

        custom\_too\_many\_rows exception;

        fara\_exemplare\_deteriorate exception;

    BEGIN

        SELECT COUNT(\*)

        INTO v\_nr\_edituri

        FROM editura

        WHERE nume\_editura = v\_nume\_editura;

        IF v\_nr\_edituri = 0 THEN

            RAISE custom\_no\_data\_found;

        ELSIF v\_nr\_edituri > 1 THEN

            RAISE custom\_too\_many\_rows;

        END IF;

    FOR i IN (

        SELECT c.titlu, c.an\_publicare, COUNT(e.id\_exemplar) AS numar\_exemplare\_deteriorate

        FROM CARTE c

        JOIN EXEMPLAR e ON c.id\_carte = e.id\_carte

        JOIN EDITURA ed ON c.id\_editura = ed.id\_editura

        WHERE ed.nume\_editura = v\_nume\_editura AND e.stare = UPPER('deteriorata')

        GROUP BY c.titlu, c.an\_publicare

    )

    LOOP

        v\_lista\_carti := v\_lista\_carti || 'Titlu: ' || i.titlu || ', An Publicare: ' || i.an\_publicare ||

                       ', Numar Exemplare Deteriorate: ' || i.numar\_exemplare\_deteriorate || CHR(10);

    END LOOP;

    --verific daca lista returnata este goala

    IF v\_lista\_carti IS NULL THEN

        RAISE fara\_exemplare\_deteriorate;

    END IF;

    RETURN v\_lista\_carti;

    EXCEPTION

        WHEN custom\_no\_data\_found THEN

            DBMS\_OUTPUT.PUT\_LINE ('Nu exista editura in baza de date cu acest nume!');

            RETURN NULL;

        WHEN custom\_too\_many\_rows THEN

            DBMS\_OUTPUT.PUT\_LINE('Exista mai multe edituri in baza de date cu acest nume!');

            RETURN NULL;

        WHEN fara\_exemplare\_deteriorate THEN

            DBMS\_OUTPUT.PUT\_LINE('Nu exista carti deteriorate pentru editura specificata.');

            RETURN NULL;

        WHEN OTHERS THEN

            DBMS\_OUTPUT.PUT\_LINE('Alta Eroare!');

            DBMS\_OUTPUT.PUT\_LINE('Codul de Eroare asociat '||SQLCODE);

            DBMS\_OUTPUT.PUT\_LINE('Mesajul de Eroare asociat '||SQLERRM);

        RETURN NULL;

    END exercitiul8;

    PROCEDURE exercitiul9(

    p\_nume\_autor VARCHAR2

    ) AS

    TYPE info\_carte IS RECORD (

        titlu VARCHAR2(100),

        categ VARCHAR2(100),

        nr\_exemplare NUMBER,

        nr\_imprumuturi NUMBER,

        popularitate NUMBER(10,2)

        --procentul pe care il reprezinta din nr total de imprumuturi

    );

    TYPE lista\_carti\_autor IS TABLE OF info\_carte;

    v\_id\_autor AUTOR.id\_autor%TYPE;

    v\_total\_imprumuturi NUMBER;

    v\_rez lista\_carti\_autor := lista\_carti\_autor();

    nu\_exista\_carti exception;

    PRAGMA EXCEPTION\_INIT (nu\_exista\_carti, -20002);

BEGIN

    SELECT id\_autor INTO v\_id\_autor

    FROM AUTOR

    WHERE UPPER(nume) = UPPER(p\_nume\_autor);

    SELECT COUNT(\*) INTO v\_total\_imprumuturi

    FROM IMPRUMUT;

    FOR rec IN (

        SELECT

            c.titlu,

            cg.nume\_categorie,

            COUNT(DISTINCT e.id\_exemplar) AS numar\_exemplare,

            COUNT(i.id\_imprumut) AS numar\_imprumuturi,

            ROUND(COUNT(i.id\_imprumut) \* 100 / v\_total\_imprumuturi, 2)AS procentaj

        FROM SCRIERE s

        JOIN CARTE c ON s.id\_carte = c.id\_carte

        JOIN CATEGORIE cg ON c.id\_categorie = cg.id\_categorie

        LEFT JOIN EXEMPLAR e ON c.id\_carte = e.id\_carte

        LEFT JOIN IMPRUMUT i ON e.id\_exemplar = i.id\_exemplar

        WHERE s.id\_autor = v\_id\_autor

        GROUP BY c.titlu, cg.nume\_categorie

    ) LOOP

        v\_rez.EXTEND;

        v\_rez(v\_rez.LAST).titlu := rec.titlu;

        v\_rez(v\_rez.LAST).categ := rec.nume\_categorie;

        v\_rez(v\_rez.LAST).nr\_exemplare := rec.numar\_exemplare;

        v\_rez(v\_rez.LAST).nr\_imprumuturi := rec.numar\_imprumuturi;

        v\_rez(v\_rez.LAST).popularitate := rec.procentaj;

    END LOOP;

    IF v\_rez.COUNT = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20002, 'Biblioteca nu detine nicio carte ce apartine acestui autor!');

    END IF;

    FOR i IN 1..v\_rez.COUNT LOOP

        DBMS\_OUTPUT.PUT\_LINE('Titlu: ' || v\_rez(i).titlu || ', Categorie: ' || v\_rez(i).categ || ', Num?r Exemplare: ' || v\_rez(i).nr\_exemplare || ', Numar Împrumuturi: ' || v\_rez(i).nr\_imprumuturi || ', Procentaj Popularitate: ' || v\_rez(i).popularitate || '%');

    END LOOP;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun autor cu numele dat in baza de date!');

    WHEN TOO\_MANY\_ROWS THEN

        DBMS\_OUTPUT.PUT\_LINE ('In baza de date exista mai multi autori cu acest nume!');

    WHEN ZERO\_DIVIDE THEN

        DBMS\_OUTPUT.PUT\_LINE ('Nu exista imprumuturi inregistrare in baza de date! Procentajul nu poate fi calculat corect prin impartirea la 0!');

    WHEN nu\_exista\_carti THEN

        DBMS\_OUTPUT.PUT\_LINE('Codul de Eroare asociat '||SQLCODE);

        DBMS\_OUTPUT.PUT\_LINE('Mesajul de Eroare asociat '||SQLERRM);

    WHEN OTHERS THEN

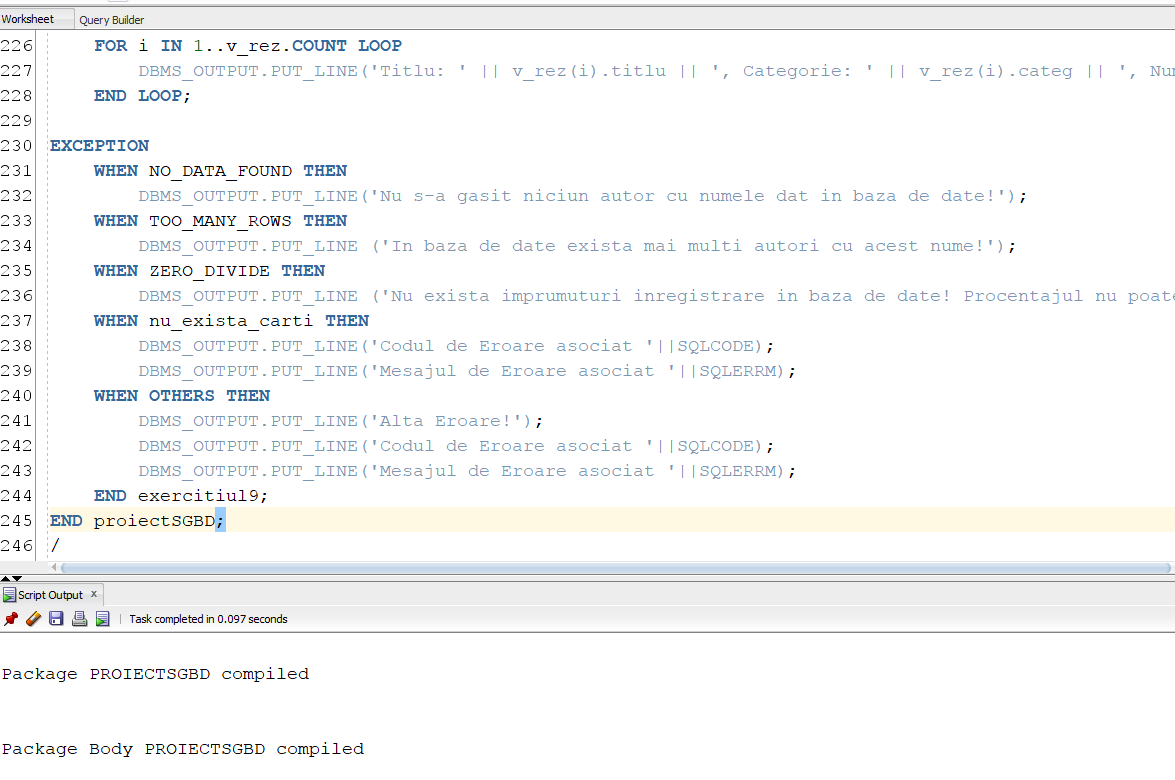
        DBMS\_OUTPUT.PUT\_LINE('Alta Eroare!');

        DBMS\_OUTPUT.PUT\_LINE('Codul de Eroare asociat '||SQLCODE);

        DBMS\_OUTPUT.PUT\_LINE('Mesajul de Eroare asociat '||SQLERRM);

    END exercitiul9;

END proiectSGBD;



BEGIN

    proiectSGBD.exercitiul6;

    proiectSGBD.exercitiul7('Iorgulescu');

    DBMS\_OUTPUT.PUT\_LINE('EDITURA Aramis');

    DBMS\_OUTPUT.NEW\_LINE;

    DBMS\_OUTPUT.PUT\_LINE(proiectSGBD.exercitiul8('Aramis'));

    DBMS\_OUTPUT.NEW\_LINE;

    DBMS\_OUTPUT.PUT\_LINE('NUMELE AUTORULUI: Ionescu');

    proiectSGBD.exercitiul9('Ionescu');

END;

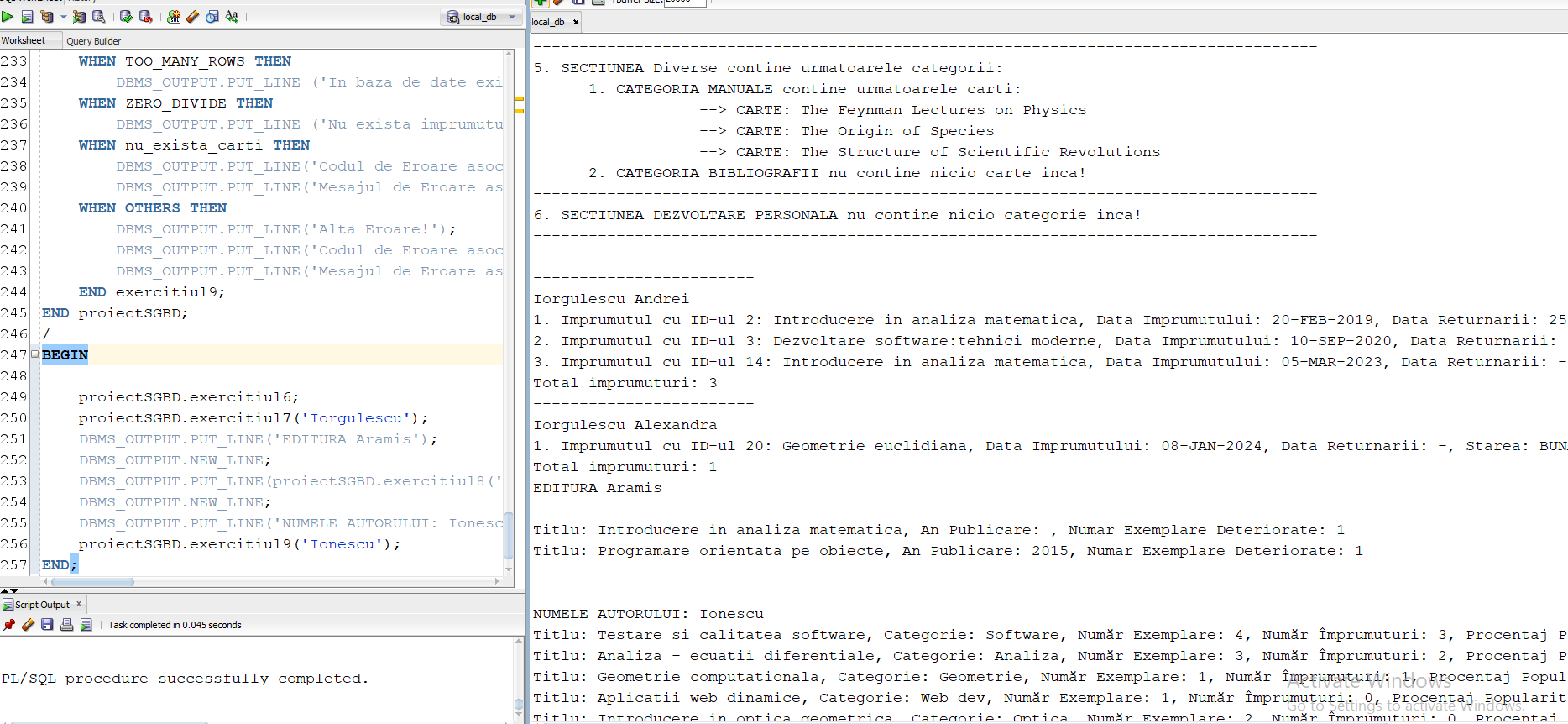


Table of Contents

[1. Prezentarea Bazei de Date 2](#_Toc155802655)

[2. Diagrama entitate-relație (ERD) 2](#_Toc155802656)

[3. Diagrama Conceptuală 3](#_Toc155802657)

[4. Implementarea în Oracle a diagramei conceptuale 4](#_Toc155802658)

[1. Tabelul Bibliotecar 4](#_Toc155802659)

[2. Tabelul Secțiune 4](#_Toc155802660)

[3. Tabelul Categorie 5](#_Toc155802661)

[4. Tabelul Editură 5](#_Toc155802662)

[5. Tabelul Autor 5](#_Toc155802663)

[6. Tabelul Carte 6](#_Toc155802664)

[7. Tabelul Scriere 6](#_Toc155802665)

[8. Tabel Exemplar 7](#_Toc155802666)

[9. Tabelul Cititor 8](#_Toc155802667)

[10.Tabelul Împrumut 8](#_Toc155802668)

[5. Inserarea datelor în tabele 9](#_Toc155802669)

[1. Tabelul Bibliotecar 9](#_Toc155802670)

[2. Tabelul Secțiune 10](#_Toc155802671)

[3. Tabelul Categorie 11](#_Toc155802672)

[4. Tabelul Editură 12](#_Toc155802673)

[5. Tabelul Autor 13](#_Toc155802674)

[6. Tabelul Carte 14](#_Toc155802675)

[7. Tabelul Scriere 16](#_Toc155802676)

[8. Tabelul Exemplar 18](#_Toc155802677)

[9. Tabelul Cititor 21](#_Toc155802678)

[10.Tabelul Împrumut 23](#_Toc155802679)

[Extra: 25](#_Toc155802680)

[1. Actualizarea datelor din tabelul Exemplar 25](#_Toc155802681)

[2. Actualizarea datelor din tabelul Împrumut 26](#_Toc155802682)

[6. Subprogram stocat independent cu 3 tipuri de colecții. 26](#_Toc155802683)

[7. Subprogram stocat independent cu 2 tipuri de cursoare 29](#_Toc155802684)

[8. Funcție cu 3 tabele care să includă definirea a minim 2 excepții 32](#_Toc155802685)

[9. Subprogram stocat independent de tip procedură cu 5 tabele 36](#_Toc155802686)

[10. Trigger de tip LMD la nivel de comandă 41](#_Toc155802687)

[11. Trigger de tip LMD la nivel de linie 44](#_Toc155802688)

[12. Trigger de tip LDD 45](#_Toc155802689)

[13. Pachet care conține toate obiectele definite în proiect. 48](#_Toc155802690)