**Aplicatie pentru**

**Gestionarea unei biblioteci - "BibliotecaATM”**



Proiect realizat de:

Moldovan Andrei-Gabriel

Grupa C 112-B

**Introducere**

Proiectul **BibliotecaATM** își propune să dezvolte o aplicație de gestionare a unei biblioteci moderne, adaptată nevoilor și cerințelor utilizatorilor din secolul XXI. Această aplicație va servi drept instrument esențial pentru organizarea și administrarea eficientă a resurselor bibliotecii, inclusiv a cărților, a membrilor și a proceselor de împrumut și returnare.

În centrul acestei aplicații stă o structură solidă de baze de date care să reflecte toate aspectele esențiale ale unei biblioteci, inclusiv detalii despre cărți, autori, genuri, membri, exemplare, împrumuturi și recenzii.

Pentru a asigura o organizare coerentă și eficientă, aplicația **BibliotecaATM** utilizează mai multe tabele interconectate:

Tabela "genuri" stochează informații despre diferitele genuri literare, facilitând clasificarea și căutarea cărților după categorii specifice.

Tabela "autori" conține detalii despre autorii cărților, inclusiv numele și naționalitatea acestora.

Tabela "carti" reprezintă o listă exhaustivă a cărților disponibile în bibliotecă, inclusiv titlul, autorul, genul, anul publicării, numărul de pagini și alte detalii relevante.

Tabela "exemplare" evidențiază fiecare exemplar din bibliotecă și starea sa de disponibilitate pentru împrumut.

Tabela "membri" conține detalii despre membrii bibliotecii, inclusiv numele, prenumele, adresa și numărul de telefon.

Tabela "imprumuturi\_membri" este o asociere între împrumuturile făcute de membri și detaliile acestora.

Tabela "recenzii" permite membriilor să ofere feedback cu privire la cărțile citite, inclusiv note și comentarii.

Tabelele "editori", "carti\_editori", "locatii", "rafturi" și "carti\_rafturi" sunt utilizate pentru gestionarea detaliilor suplimentare legate de editori, locații și rafturi în bibliotecă.

Prin implementarea acestei aplicații, se urmărește crearea unei soluții intuitive și eficiente pentru administrarea unei biblioteci moderne, în beneficiul atât al personalului bibliotecii, cât și al utilizatorilor săi.

**TABELE**

Conform celor prezentate mai sus in ceea ce priveste logica aplicatiei, s-a ajuns la urmatoarea structura de tabele:



**Diagrama Bazei De Date**



**CREAREA BAZEI DE DATE SI A GRUPULUI DE FISIERE AFERENTE**

CREATE DATABASE BibliotecaATM

ON PRIMARY

(

Name = Data1,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Data1.mdf',

size = 10MB, -- KB, Mb, GB, TB

maxsize = unlimited,

filegrowth = 1GB

),

FILEGROUP fgCurrent

(

Name = Data2,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Data2.ndf',

size = 10MB, -- KB, Mb, GB, TB

maxsize = unlimited,

filegrowth = 1GB

),

(

Name = Data3,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Data3.ndf',

size = 10MB, -- KB, Mb, GB, TB

maxsize = unlimited,

filegrowth = 1GB

),

FILEGROUP fgArchive

(

Name = Data4,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Data4.ndf',

size = 10MB, -- KB, Mb, GB, TB

maxsize = unlimited,

filegrowth = 1GB

)

LOG ON

(

Name = Log1,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Log1.ldf',

size = 10MB, -- KB, Mb, GB, TB

maxsize = unlimited,

filegrowth = 1024MB

),

(

Name = Log2,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Log2.ldf',

size = 10MB,

maxsize = unlimited,

filegrowth = 1024MB

),

(

Name = Log3,

FileName = 'C:\Users\Andrei\Desktop\ProiectBD\Log3.ldf',

size = 10MB,

maxsize = unlimited,

filegrowth = 1024MB

**CREAREA TABELELOR**

**1. Crearea tabelei cu genurile de carti, contine un ID si genul acesteia**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'genuri')

BEGIN

CREATE TABLE genuri (

gen\_id INT PRIMARY KEY,

nume\_gen NVARCHAR(100),

autor\_id INT,

carte\_id INT,

locatie\_id INT,

);

END;

**2. Crearea tabelei cu autorii cartilor, acestia contin un ID al autorului, un nume si o nationalitate**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'autori')

BEGIN

CREATE TABLE autori (

autor\_id INT PRIMARY KEY,

nume\_autor NVARCHAR(100),

nationalitate NVARCHAR(100),

carte\_id INT

);

END;

**3. Crearea tabelei ce contine cartile, acestea contin un ID unic, un titlu, ID ul autorului, al genului, anul publicatiei, numarul de pagini, numarul de capitole si culoarea copertii.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'carti')

BEGIN

CREATE TABLE carti (

carte\_id INT PRIMARY KEY,

titlu NVARCHAR(100),

autor\_id INT,

gen\_id INT,

an\_publicare INT,

numar\_pagini INT,

numar\_capitole INT,

culoare\_coperta NVARCHAR(100)

);

END;

**4. Crearea tabelei exemplare, acestea au un ID si un numar de carti disponibile.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'exemplare')

BEGIN

CREATE TABLE exemplare (

exemplar\_id INT PRIMARY KEY,

carte\_id INT,

disponibil BIT,

);

END;

**5. Crearea tabelei membrii, contine numele, prenumele, ID-ul, adresa si telefonul fiecarui membrului.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'membri')

BEGIN

CREATE TABLE membri (

membru\_id INT PRIMARY KEY,

nume NVARCHAR(100),

prenume NVARCHAR(100),

adresa NVARCHAR(255),

telefon NVARCHAR(20)

);

END;

**6. Crearea tabelei imprumut\_membri, tabela care face legatura intre membru si carte, pentru a se putea realiza cu succes imprumutul.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'imprumuturi\_membri')

BEGIN

CREATE TABLE imprumuturi\_membri (

imprumut\_id INT PRIMARY KEY,

membru\_id INT,

exemplar\_id INT,

data\_imprumut DATE,

data\_returnare DATE

);

END;

**7. Crearea tabelei recenzii, care retine recenzii date de membri cartilor.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'recenzii')

BEGIN

CREATE TABLE recenzii (

recenzie\_id INT PRIMARY KEY,

carte\_id INT,

membru\_id INT,

nota INT,

comentariu NVARCHAR(MAX)

);

END;

**8. Crearea tabelei editori, care are un ID si un nume, fiind asociat cartilor de editor.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'editori')

BEGIN

CREATE TABLE editori (

editor\_id INT PRIMARY KEY,

nume\_editor NVARCHAR(100)

);

END;

**9. Crearea tabelei carti\_editor, care face o asociere intre carti si editori.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'editori')

BEGIN

CREATE TABLE editori (

editor\_id INT PRIMARY KEY,

nume\_editor NVARCHAR(100)

);

END;

**10. Crearea tabelei locatii, care retine locatia pe raft a fiecarei carti.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'locatii')

BEGIN

CREATE TABLE locatii (

locatie\_id INT PRIMARY KEY,

nume\_locatie NVARCHAR(100),

adresa\_locatie NVARCHAR(255)

);

END;

**11. Crearea tabelei rafturi, care contine raftul pe care se afla cartile in biblioteca.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'rafturi')

BEGIN

CREATE TABLE rafturi (

raft\_id INT PRIMARY KEY,

locatie\_id INT,

nume\_raft NVARCHAR(100),

disponibil INT

);

END;

**12. Crearea tabelei carti\_rafturi, care face legatura dintre carti si rafturile pe care se afla.**

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'carti\_rafturi')

BEGIN

CREATE TABLE carti\_rafturi (

carte\_id INT,

raft\_id INT,

exemplar\_id INT

);

END;

**13. Crearea tabelei LogModificariCarti pentru a tine o evidenta a modificarilor realizate asupra cartilor**

IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'LogModificariCarti')

BEGIN

CREATE TABLE LogModificariCarti (

log\_id INT PRIMARY KEY IDENTITY(1,1),

carte\_id INT,

pagini\_vechi INT,

pagini\_noi INT,

data\_modificare DATETIME );END;

**Constrangeri**

ALTER TABLE genuri

ADD CONSTRAINT fk\_autor\_id FOREIGN KEY(autor\_id)

REFERENCES autori(autor\_id);

ALTER TABLE genuri

ADD CONSTRAINT fk\_carte\_id FOREIGN KEY(carte\_id)

REFERENCES carti(carte\_id);

ALTER TABLE carti\_rafturi

ADD CONSTRAINT fk\_exemplar\_ID FOREIGN KEY(exemplar\_id)

REFERENCES exemplare(exemplar\_id);

ALTER TABLE imprumuturi\_membri

ADD CONSTRAINT fk\_exemplar\_id\_membru FOREIGN KEY(exemplar\_id)

REFERENCES exemplare(exemplar\_id);

ALTER TABLE rafturi

ADD disponibli INT;

ALTER TABLE imprumuturi\_membri

ADD CONSTRAINT fk\_membru\_id FOREIGN KEY(membru\_id)

REFERENCES membri(membru\_id);

ALTER TABLE recenzii

ADD CONSTRAINT fk\_carte\_id\_rec FOREIGN KEY(carte\_id)

REFERENCES carti(carte\_id);

ALTER TABLE recenzii

ADD CONSTRAINT fk\_membru\_ID\_rec FOREIGN KEY(membru\_id)

REFERENCES membri(membru\_id);

ALTER TABLE carti\_editori

ADD CONSTRAINT fk\_carte\_id\_ce FOREIGN KEY(carte\_id)

REFERENCES carti(carte\_id);

ALTER TABLE carti\_editori

ADD CONSTRAINT fk\_editor\_id\_ce FOREIGN KEY(editor\_id)

REFERENCES editori(editor\_id);

ALTER TABLE genuri

ADD CONSTRAINT fk\_locatie\_id\_gen FOREIGN KEY(locatie\_id)

REFERENCES locatii(locatie\_id);

ALTER TABLE carti\_rafturi

ADD CONSTRAINT fk\_carte\_id\_carte FOREIGN KEY(carte\_id)

REFERENCES carti(carte\_id);

ALTER TABLE carti\_rafturi

ADD CONSTRAINT fk\_raft\_ID\_raft FOREIGN KEY(raft\_id)

REFERENCES rafturi(raft\_id);

ALTER TABLE autori

ADD CONSTRAINT fk\_carti\_id\_aut FOREIGN KEY(carte\_id)

REFERENCES carti(carte\_id);

**Inserarea Datelor**

USE BibliotecaATM

INSERT INTO genuri (gen\_id,nume\_gen)

VALUES

(1, 'Fictiune'),

(2, 'Non-fictiune'),

(3, 'Fantastic'),

(4, 'Mistic'),

(5, 'Drama'),

(6, 'Thriller'),

(7, 'SF'),

(8, 'Istoric'),

(9, 'Biografie'),

(10,'Poezie');

INSERT INTO autori (autor\_id,nume\_autor, nationalitate)

VALUES

(1,'George Orwell', 'Engleza'),

(2,'Fyodor Dostoevsky', 'Rusa'),

(3,'David Wroblewski', 'Americana'),

(4,'Lev Tolstoy', 'Rusa'),

(5,'J.K. Rowling', 'Engleza'),

(6,'Victor Hugo', 'Franceza'),

(7,'C.S. Lewis', 'Irlandeza'),

(8,'Leo Tolstoy', 'Rusa'),

(9,'Frances Hodgson Burnett', 'Britanica'),

(10,'John Steinbeck', 'Americana');

INSERT INTO carti (carte\_id,titlu, autor\_id, gen\_id, an\_publicare, numar\_pagini, numar\_capitole, culoare\_coperta)

VALUES

(1,'1984', 1, 1, 1949, 328, 23, 'gri'),

(2,'Crima si pedeapsa', 2, 5, 1866, 430, 40, 'maro'),

(3,'Povestea lui Edgar Sawtelle', 3, 3, 2008, 562, 45, 'albastra'),

(4,'Anna Karenina', 4, 7, 1877, 864, 80, 'rosie'),

(5,'Harry Potter si Piatra Filozofala', 5, 3, 1997, 332, 17, 'portocaliu'),

(6,'Mizerabilii', 6, 4, 1862, 1232, 100, 'neagra'),

(7,'Cronicile din Narnia', 7, 2, 1950, 767, 56, 'verde'),

(8,'Razboi si pace', 8, 5, 1869, 1225, 80, 'alba'),

(9,'Mica printesa', 9, 8, 1905, 210, 15, 'roz'),

(10,'Oameni si soareci', 10, 1, 1937, 187, 10, 'mov');

INSERT INTO exemplare (exemplar\_id,carte\_id, disponibil)

VALUES

(1,1, 1),

(2,2, 0),

(3,3, 1),

(4,4, 1),

(5,5, 0),

(6,6, 1),

(7,7, 1),

(8,8, 0),

(9,9, 1),

(10,10, 1);

INSERT INTO membri (membru\_id,nume, prenume, adresa, telefon)

VALUES

(1,'Popescu', 'Ion', 'Str. Principala nr. 10', '0722123456'),

(2,'Ionescu', 'Ana', 'Bd. Libertatii nr. 5', '0722567890'),

(3,'Radulescu', 'Maria', 'Aleea Florilor nr. 3', '0733123456'),

(4,'Georgescu', 'Mihai', 'Str. Soarelui nr. 15', '0711123456'),

(5, NULL, 'Elena', 'Bd. Independentei nr. 20', '0734123456'),

(6, NULL, 'Alexandru', 'Piata Unirii nr. 7', '0712123456'),

(7,'Gheorghiu', 'Andreea', 'Str. Victoriei nr. 12', '0723123456'),

(8,'Constantinescu', 'Cristina', 'Aleea Crizantemelor nr. 9', '0735123456'),

(9,'Marinescu', 'Adrian', 'Bd. Revolutiei nr. 8', '0713123456'),

(10,'Neacsu', 'Laura', 'Str. Bucuresti nr. 18', '0724123456');

INSERT INTO imprumuturi\_membri (imprumut\_id,membru\_id, exemplar\_id, data\_imprumut, data\_returnare)

VALUES

(1,1, 1, '2024-03-01', '2024-03-15'),

(2,2, 2, '2024-03-03', '2024-03-17'),

(3,3, 3, '2024-03-05', '2024-03-19'),

(4,4, 4, '2024-03-07', '2024-03-21'),

(5,5, 5, '2024-03-09', '2024-03-23'),

(6,6, 6, '2024-03-11', '2024-03-25'),

(7,7, 7, '2024-03-13', '2024-03-27'),

(8,8, 8, '2024-03-15', '2024-03-29'),

(9,9, 9, '2024-03-17', '2024-03-31'),

(10,10,10, '2024-03-19', '2024-04-02');

INSERT INTO recenzii (recenzie\_id,carte\_id, membru\_id, nota, comentariu)

VALUES

(1,1, 1, 5, 'O carte captivanta, cu un final surprinzator.'),

(2,2, 2, 4, 'O lectură interesanta, dar putin prea lunga.'),

(3,3, 3, 5, 'Un roman emotionant, cu personaje complexe.'),

(4,4, 4, 3, 'Nu m-a captivat la fel de mult ca alte opere ale autorului.'),

(5,5, 5, 5, 'O calatorie fantastica în lumea vrajitorilor.'),

(6,6, 6, 4, 'O poveste clasica, dar plina de invataminte.'),

(7,7, 7, 3, 'Un roman interesant, dar cu unele aspecte neclare.'),

(8,8, 8, 5, 'Una dintre cele mai bune carti pe care le-am citit.'),

(9,9, 9, 4, 'O lectura placuta, dar nu extraordinara.'),

(10,10, 10, 3, 'Un roman clasic, dar cu o actiune prea lenta.');

INSERT INTO editori (editor\_id,nume\_editor)

VALUES

(1,'Penguin Books'),

(2,'HarperCollins'),

(3,'Random House'),

(4,'Simon & Schuster'),

(5,'Macmillan Publishers'),

(6,'Hachette Livre'),

(7,'Oxford University Press'),

(8,'Bloomsbury Publishing'),

(9,'Wiley'),

(10,'Scholastic Corporation');

INSERT INTO carti\_editori (carte\_id, editor\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5),

(6, 6),

(7, 7),

(8, 8),

(9, 9),

(10, 10);

INSERT INTO locatii (locatie\_id,nume\_locatie, adresa\_locatie)

VALUES

(1,'Biblioteca Centrala', 'Str. Bibliotecii nr. 1'),

(2,'Biblioteca Municipala', 'Bd. Culturii nr. 2'),

(3,'Biblioteca Universitara', 'Str. Universității nr. 3'),

(4,'Biblioteca Copiilor', 'Bd. Copiilor nr. 4'),

(5,'Biblioteca Naționala', 'Str. Națională nr. 5'),

(6,'Biblioteca Școlara', 'Bd. Școlii nr. 6'),

(7,'Biblioteca Digitala', 'Str. Digitală nr. 7'),

(8,'Biblioteca Publica', 'Bd. Publică nr. 8'),

(9,'Biblioteca Virtuala', 'Str. Virtuală nr. 9'),

(10,'Biblioteca Raionala', 'Bd. Raională nr. 10');

INSERT INTO rafturi (raft\_id,locatie\_id, nume\_raft,disponibil)

VALUES

(1,1, 'Raft A',1),

(2,2, 'Raft B',1),

(3,3, 'Raft C',1),

(4,4, 'Raft D',1),

(5,5, 'Raft E',1),

(6,6, 'Raft F',1),

(7,7, 'Raft G',1),

(8,8, 'Raft H',1),

(9,9, 'Raft I',1),

(10,10, 'Raft J',1);

INSERT INTO carti\_rafturi (carte\_id, raft\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5),

(6, 6),

(7, 7),

(8, 8),

(9, 9),

(10, 10);

INSERT INTO genuri (gen\_id, nume\_gen)

VALUES

(11, 'Polițist'),

(12, 'Romantic'),

(13, 'Horror'),

(14, 'Biografic'),

(15, 'Filosofic');

INSERT INTO autori (autor\_id, nume\_autor, nationalitate)

VALUES

(11, 'Agatha Christie', 'Engleza'),

(12, 'Jane Austen', 'Engleza'),

(13, 'Stephen King', 'American'),

(14, 'Marie Curie', 'Poloneza'),

(15, 'Fyodor Dostoevsky', 'Rus');

INSERT INTO carti (carte\_id, titlu, autor\_id, gen\_id, an\_publicare, numar\_pagini, numar\_capitole, culoare\_coperta)

VALUES

(11, 'Murder on the Orient Express', 11, 11, 1934, 256, 18, 'verde'),

(12, 'Pride and Prejudice', 12, 12, 1813, 326, 61, 'alba'),

(13, 'The Shining', 13, 13, 1977, 447, 60, 'rosie'),

(14, 'Marie Curie: A Life', 14, 14, 1937, 312, 24, 'albastra'),

(15, 'Crime and Punishment', 15, 11, 1866, 430, 40, 'neagra');

INSERT INTO exemplare (exemplar\_id, carte\_id, disponibil)

VALUES

(11, 11, 1),

(12, 12, 1),

(13, 13, 1),

(14, 14, 1),

(15, 15, 0);

**Selectii**

USE BibliotecaATM

SELECT carte\_id from carti

SELECT \* FROM autori;

SELECT \* FROM membri;

SELECT \* FROM carti

SELECT c.titlu AS 'Title', a.nume\_autor AS 'Author', g.nume\_gen AS 'Genre'

FROM carti c, autori a, genuri g

WHERE c.autor\_id = a.autor\_id AND c.gen\_id = g.gen\_id;

SELECT c.titlu AS 'Title'

FROM carti c, exemplare e

WHERE c.carte\_id = e.carte\_id AND e.disponibil = 1;

SELECT DISTINCT m.nume, m.prenume

FROM membri m, imprumuturi\_membri i

SELECT \*

FROM carti

WHERE an\_publicare = 1905;

SELECT c.titlu AS 'Title', i.data\_imprumut AS 'Borrow Date', DATEDIFF(DAY, i.data\_imprumut, GETDATE()) AS 'Days Borrowed'

FROM carti c, exemplare e, imprumuturi\_membri i

WHERE c.carte\_id = e.carte\_id AND e.exemplar\_id = i.exemplar\_id;

SELECT c.titlu AS 'Title', i.data\_returnare AS 'Due Date', DATEDIFF(DAY, i.data\_returnare, GETDATE()) AS 'Days Overdue'

FROM carti c, exemplare e, imprumuturi\_membri i

WHERE c.carte\_id = e.carte\_id AND e.exemplar\_id = i.exemplar\_id AND i.data\_returnare < GETDATE();

SELECT c.titlu AS TitluCarte, a.nume\_autor AS NumeAutor

FROM carti c

INNER JOIN autori a ON c.autor\_id = a.autor\_id;

SELECT \*

FROM imprumuturi\_membri

WHERE data\_returnare IS NOT NULL;

SELECT \*

FROM membri

WHERE adresa LIKE '%Piata Unirii nr.7%';

SELECT \*

FROM recenzii

WHERE nota >= 4;

SELECT \*

FROM carti

WHERE an\_publicare > 1941

ORDER BY numar\_pagini DESC;

**Interogari de selectie**

-- Selectează titlurile cărților care au recenzii și exemplare disponibile

SELECT c.titlu

FROM carti c

JOIN recenzii ON c.carte\_id = recenzii.carte\_id

INTERSECT

SELECT c.titlu

FROM carti c

JOIN exemplare e ON c.carte\_id = e.carte\_id

WHERE e.disponibil = 1;

-- Selectează titlurile cărților care au recenzii dar nu au exemplare disponibile

SELECT c.titlu

FROM carti c

JOIN recenzii ON c.carte\_id = recenzii.carte\_id

EXCEPT

SELECT c.titlu

FROM carti c

JOIN exemplare e ON c.carte\_id = e.carte\_id

WHERE e.disponibil = 1;

-- Selectează numele distincte ale autorilor care au cărți în genurile "Fictiune" sau "Non-fictiune"

SELECT DISTINCT a.nume\_autor

FROM autori a

JOIN carti c ON a.autor\_id = c.autor\_id

JOIN genuri g ON c.gen\_id = g.gen\_id

WHERE g.nume\_gen IN ('Fictiune', 'Non-fictiune');

-- Calculează numărul de cărți pentru fiecare gen

SELECT g.nume\_gen, COUNT(\*) AS numar\_carti

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

GROUP BY g.nume\_gen;

-- Calculează media numărului de pagini pentru cărți în funcție de anul de publicare

SELECT an\_publicare, AVG(numar\_pagini) AS numar\_mediu\_pagini

FROM carti

GROUP BY an\_publicare

ORDER BY an\_publicare;

-- Calculează numărul de împrumuturi pentru fiecare membru, ordonat descrescător după numărul de împrumuturi

SELECT membru\_id, COUNT(\*) AS numar\_imprumuturi

FROM imprumuturi\_membri

GROUP BY membru\_id

ORDER BY numar\_imprumuturi DESC;

-- Calculează media evaluării pentru fiecare gen de carte

SELECT g.nume\_gen, AVG(r.nota) AS evaluare\_medie

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY g.nume\_gen;

-- Calculează numărul de cărți pentru fiecare gen cu mai mult de 200 de pagini

SELECT g.nume\_gen, COUNT(\*) AS numar\_carti

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

WHERE c.numar\_pagini > 200

GROUP BY g.nume\_gen;

-- Calculează media evaluării pentru fiecare gen de carte cu o evaluare medie de recenzie de cel puțin 4

SELECT g.nume\_gen, AVG(r.nota) AS evaluare\_medie

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY g.nume\_gen

HAVING AVG(r.nota) >= 4;

-- Calculează numărul de împrumuturi pentru fiecare membru cu cel puțin un împrumut

SELECT membru\_id, COUNT(\*) AS numar\_imprumuturi

FROM imprumuturi\_membri

GROUP BY membru\_id

HAVING COUNT(\*) >= 1;

-- Calculează numărul de exemplare disponibile pentru fiecare carte cu cel puțin o copie disponibilă

SELECT carte\_id, COUNT(\*) AS numar\_exemplare\_disponibile

FROM exemplare

WHERE disponibil = 1

GROUP BY carte\_id

HAVING COUNT(\*) >= 1;

--Selectează titlul cărții împreună cu numele autorului și genul cărții

SELECT c.titlu AS TitluCarte, a.nume\_autor AS NumeAutor, g.nume\_gen AS NumeGen

FROM carti c

JOIN autori a ON c.autor\_id = a.autor\_id

JOIN genuri g ON c.gen\_id = g.gen\_id;

-- Selectează numele și prenumele membrului împreună cu data împrumutului și disponibilitatea exemplarului

SELECT m.nume, m.prenume, i.data\_imprumut, e.disponibil

FROM membri m

JOIN imprumuturi\_membri i ON m.membru\_id = i.membru\_id

JOIN exemplare e ON i.exemplar\_id = e.exemplar\_id;

-- Selectează titlul cărții împreună cu numele editorului

SELECT c.titlu, e.nume\_editor

FROM carti c

JOIN carti\_editori ce ON c.carte\_id = ce.carte\_id

JOIN editori e ON ce.editor\_id = e.editor\_id;

-- Selectează numele și prenumele membrilor care au făcut imprumuturi sau au scris recenzii

SELECT nume, prenume

FROM membri

JOIN imprumuturi\_membri ON membri.membru\_id = imprumuturi\_membri.membru\_id

UNION

SELECT nume, prenume

FROM membri

JOIN recenzii ON membri.membru\_id = recenzii.membru\_id;

**CTE**

-- Common Table Expression (CTE): Creează o vedere temporară pentru a afișa recenziile cărților împreună cu numele autorilor

WITH CartiRecenzii AS (

SELECT c.titlu, r.nota, r.comentariu, a.nume\_autor

FROM carti c

JOIN recenzii r ON c.carte\_id = r.carte\_id

JOIN autori a ON c.autor\_id = a.autor\_id

)

SELECT \* FROM CartiRecenzii;

-- CTE: Creează o vedere temporară pentru a afișa numărul de exemplare disponibile pentru fiecare raft

WITH ExemplareRafturi AS (

SELECT r.raft\_id, r.nume\_raft, COUNT(e.exemplar\_id) AS numar\_exemplare\_disponibile

FROM rafturi r

LEFT JOIN carti\_rafturi cr ON r.raft\_id = cr.raft\_id

LEFT JOIN exemplare e ON cr.exemplar\_id = e.exemplar\_id

WHERE r.disponibil = 0

GROUP BY r.raft\_id, r.nume\_raft

)

SELECT \* FROM ExemplareRafturi;

-- CTE: Creează o vedere temporară pentru a afișa evaluarea medie pentru fiecare gen cu o evaluare medie de recenzie mai mare de 4

WITH EvaluareMedieGen AS (

SELECT g.nume\_gen, AVG(r.nota) AS evaluare\_medie

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY g.nume\_gen

HAVING AVG(r.nota) > 4

)

SELECT \* FROM EvaluareMedieGen;

-- CTE: Creează o vedere temporară pentru a afișa numărul de împrumuturi pentru fiecare membru

WITH NumarImprumuturiMembri AS (

SELECT membru\_id, COUNT(\*) AS numar\_imprumuturi

FROM imprumuturi\_membri

GROUP BY membru\_id

)

SELECT m.nume, m.prenume, nim.numar\_imprumuturi

FROM membri m

JOIN NumarImprumuturiMembri nim ON m.membru\_id = nim.membru\_id;

**VIEW-URI**

-- Vizualizare pentru a afișa detaliile împrumuturilor de cărți

CREATE VIEW View\_CartiImprumutate AS

SELECT im.imprumut\_id, c.titlu AS titlu\_carte, m.nume AS nume\_membru, m.prenume AS prenume\_membru, im.data\_imprumut, im.data\_returnare

FROM imprumuturi\_membri im

JOIN membri m ON im.membru\_id = m.membru\_id

JOIN exemplare e ON im.exemplar\_id = e.exemplar\_id

JOIN carti c ON e.carte\_id = c.carte\_id;

-- Vizualizare pentru a calcula evaluarea medie pentru fiecare gen de carte

CREATE VIEW View\_EvaluareMedieGen AS

SELECT g.nume\_gen, AVG(r.nota) AS evaluare\_medie

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY g.nume\_gen;

-- Vizualizare pentru a număra numărul de împrumuturi pentru fiecare membru

CREATE OR ALTER VIEW View\_MembriCuImprumuturi

AS

SELECT

m.membru\_id,

m.nume,

m.prenume,

m.adresa,

m.telefon,

COUNT(im.imprumut\_id) AS Numar\_Imprumuturi

FROM

membri m

JOIN

imprumuturi\_membri im ON m.membru\_id = im.membru\_id

WHERE

m.nume IS NOT NULL

AND m.prenume IS NOT NULL

GROUP BY

m.membru\_id,

m.nume,

m.prenume,

m.adresa,

m.telefon;

-- Vizualizare pentru a număra numărul de recenzii și a calcula media notelor pentru fiecare carte

CREATE VIEW View\_NumarRecenziiMedieNote AS

SELECT c.titlu AS titlu\_carte, COUNT(r.recenzie\_id) AS numar\_recenzii, AVG(r.nota) AS medie\_note

FROM carti c

LEFT JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY c.titlu;

-- Vizualizare pentru a afișa detalii despre cărți împreună cu autorii și genurile acestora

CREATE OR ALTER VIEW View\_CartiCuAutoriSiGenuri

AS

SELECT

c.titlu AS Titlu\_Carte,

a.nume\_autor AS Nume\_Autor,

a.nationalitate AS Nationalitate\_Autor,

g.nume\_gen AS Nume\_Gen

FROM

carti c

JOIN

autori a ON c.autor\_id = a.autor\_id

JOIN

genuri g ON c.gen\_id = g.gen\_id

--Vizualizare date:

SELECT \* FROM View\_CartiImprumutate

SELECT \* FROM View\_EvaluareMedieGen

SELECT \* FROM View\_MembriCuImprumuturi

SELECT \* FROM View\_NumarRecenziiMedieNote

SELECT \* FROM View\_CartiCuAutoriSiGenuri

**Proceduri Stocate**

-- Procedura stocată 1: Selectează titlurile cărților care au atât recenzii, cât și exemplare disponibile

CREATE PROCEDURE GetBooksByReviewAndAvailability

AS

BEGIN

SELECT c.titlu

FROM carti c

JOIN recenzii ON c.carte\_id = recenzii.carte\_id

INTERSECT

SELECT c.titlu

FROM carti c

JOIN exemplare e ON c.carte\_id = e.carte\_id

WHERE e.disponibil = 1;

END;

GO

-- Procedura stocată 2: Selectează titlurile cărților care au recenzii, dar nu au exemplare disponibile

CREATE PROCEDURE GetBooksByReviewsButNoAvailability

AS

BEGIN

SELECT c.titlu

FROM carti c

JOIN recenzii ON c.carte\_id = recenzii.carte\_id

EXCEPT

SELECT c.titlu

FROM carti c

JOIN exemplare e ON c.carte\_id = e.carte\_id

WHERE e.disponibil = 1;

END;

GO

-- Procedura stocată 3: Selectează numele distincte ale autorilor care au cărți în genurile "Fictiune" sau "Non-fictiune"

CREATE PROCEDURE GetAuthorsByGenres

AS

BEGIN

SELECT DISTINCT a.nume\_autor

FROM autori a

JOIN carti c ON a.autor\_id = c.autor\_id

JOIN genuri g ON c.gen\_id = g.gen\_id

WHERE g.nume\_gen IN ('Fictiune', 'Non-fictiune');

END;

GO

-- Procedura stocată 4: Calculează numărul de cărți pentru fiecare gen

CREATE PROCEDURE GetBookCountByGenre

AS

BEGIN

SELECT g.nume\_gen, COUNT(\*) AS numar\_carti

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

GROUP BY g.nume\_gen;

END;

GO

-- Procedura stocată 5: Calculează media numărului de pagini pentru cărți în funcție de anul de publicare

CREATE PROCEDURE GetAveragePagesByYear

AS

BEGIN

SELECT an\_publicare, AVG(numar\_pagini) AS numar\_mediu\_pagini

FROM carti

GROUP BY an\_publicare

ORDER BY an\_publicare;

END;

GO

-- Procedura stocată 6: Calculează numărul de împrumuturi pentru fiecare membru, ordonat descrescător după numărul de împrumuturi

CREATE PROCEDURE GetLoanCountsByMember

AS

BEGIN

SELECT membru\_id, COUNT(\*) AS numar\_imprumuturi

FROM imprumuturi\_membri

GROUP BY membru\_id

ORDER BY numar\_imprumuturi DESC;

END;

GO

-- Procedura stocată 7: Calculează media evaluării pentru fiecare gen de carte

CREATE PROCEDURE GetAverageRatingByGenre

AS

BEGIN

SELECT g.nume\_gen, AVG(r.nota) AS evaluare\_medie

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY g.nume\_gen;

END;

GO

-- Procedura stocată 8: Calculează numărul de cărți pentru fiecare gen cu mai mult de 200 de pagini

CREATE PROCEDURE GetBookCountByGenreOver200Pages

AS

BEGIN

SELECT g.nume\_gen, COUNT(\*) AS numar\_carti

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

WHERE c.numar\_pagini > 200

GROUP BY g.nume\_gen;

END;

GO

-- Procedura stocată 9: Calculează media evaluării pentru fiecare gen de carte cu o evaluare medie de recenzie de cel puțin 4

CREATE PROCEDURE GetAverageRatingByHighReviewGenre

AS

BEGIN

SELECT g.nume\_gen, AVG(r.nota) AS evaluare\_medie

FROM carti c

JOIN genuri g ON c.gen\_id = g.gen\_id

JOIN recenzii r ON c.carte\_id = r.carte\_id

GROUP BY g.nume\_gen

HAVING AVG(r.nota) >= 4;

END;

GO

-- Procedura stocată 10: Calculează numărul de exemplare disponibile pentru fiecare carte cu cel puțin o copie disponibilă

CREATE PROCEDURE GetAvailableCopiesCountByBook

AS

BEGIN

SELECT carte\_id, COUNT(\*) AS numar\_exemplare\_disponibile

FROM exemplare

WHERE disponibil = 1

GROUP BY carte\_id

HAVING COUNT(\*) >= 1;

END;

GO

EXEC GetBooksByReviewAndAvailability;

EXEC GetBooksByReviewsButNoAvailability;

EXEC GetAuthorsByGenres;

EXEC GetBookCountByGenre;

EXEC GetAveragePagesByYear;

EXEC GetLoanCountsByMember;

EXEC GetAverageRatingByGenre;

EXEC GetBookCountByGenreOver200Pages;

EXEC GetAverageRatingByHighReviewGenre;

EXEC GetAvailableCopiesCountByBook;

**Triggere**

-- Trigger 1: Trigger pentru înregistrarea returnării cărților într-un tabel de jurnal

CREATE TRIGGER RecordReturn

ON imprumuturi\_membri

AFTER UPDATE

AS

BEGIN

INSERT INTO LogReturnari (imprumut\_id, data\_returnare)

SELECT i.imprumut\_id, GETDATE()

FROM inserted i

JOIN deleted d ON i.imprumut\_id = d.imprumut\_id

WHERE i.data\_returnare IS NOT NULL

AND d.data\_returnare IS NULL;

END;

-- Trigger 2: Trigger pentru a actualiza stocul de exemplare după un împrumut

CREATE TRIGGER UpdateStockOnLoan

ON imprumuturi\_membri

AFTER INSERT

AS

BEGIN

UPDATE exemplare

SET disponibil = 0

WHERE exemplar\_id IN (SELECT exemplar\_id FROM inserted);

END;

GO

-- Trigger 3: Trigger pentru a înregistra recenzii noi

CREATE TRIGGER RecordNewReview

ON recenzii

AFTER INSERT

AS

BEGIN

INSERT INTO LogRecenzii (carte\_id, membru\_id, nota, comentariu, data\_recenzie)

SELECT carte\_id, membru\_id, nota, comentariu, GETDATE()

FROM inserted;

END;

GO

-- Trigger 4: Trigger pentru a gestiona stocul de exemplare pe baza returnărilor

CREATE TRIGGER UpdateStockOnReturn

ON imprumuturi\_membri

AFTER DELETE

AS

BEGIN

UPDATE exemplare

SET disponibil = 1

WHERE exemplar\_id IN (SELECT exemplar\_id FROM deleted);

END;

GO

-- Trigger 5: Trigger pentru a restricționa ștergerea de autori cu cărți asociate

CREATE TRIGGER PreventAuthorDeletion

ON autori

INSTEAD OF DELETE

AS

BEGIN

IF EXISTS (SELECT 1 FROM carti WHERE autor\_id IN (SELECT autor\_id FROM deleted))

BEGIN

PRINT ('Nu poți șterge acest autor deoarece are cărți asociate.');

END

ELSE

BEGIN

DELETE FROM autori WHERE autor\_id IN (SELECT autor\_id FROM deleted);

END

END;

GO

-- Trigger 6: Trigger pentru a înregistra modificările de pagini la cărți

CREATE TRIGGER LogPageChanges

ON carti

AFTER UPDATE

AS

BEGIN

INSERT INTO LogModificariCarti (carte\_id, pagini\_vechi, pagini\_noi, data\_modificare)

SELECT d.carte\_id, d.numar\_pagini AS pagini\_vechi, i.numar\_pagini AS pagini\_noi, GETDATE()

FROM deleted d

JOIN inserted i ON d.carte\_id = i.carte\_id;

END;

GO

-- Trigger 7: Trigger pentru a interzice actualizarea genurilor cărților

CREATE TRIGGER PreventGenreUpdate

ON genuri

INSTEAD OF UPDATE

AS

BEGIN

PRINT ('Nu se permite actualizarea genurilor cărților.');

END;

GO

-- Trigger 8: Trigger pentru a trimite notificări în funcție de modificările făcute la cărți

CREATE TRIGGER SendNotificationsOnBookChanges

ON carti

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

PRINT 'Modificare efectuată asupra cărților. Trimitere notificare...';

END;

GO

-- Trigger 9: Trigger pentru afișarea unui mesaj la inserarea unui membru

CREATE TRIGGER ShowMessageOnMemberInsert

ON membri

AFTER INSERT

AS

BEGIN

PRINT 'Un nou membru a fost inserat în tabelul membri.';

END;

-- Trigger 10: Trigger pentru afișarea unui mesaj după înregistrarea unui împrumut

CREATE TRIGGER ShowBookLoanMessage

ON imprumuturi\_membri

AFTER INSERT

AS

BEGIN

DECLARE @TitluCarte NVARCHAR(255);

SELECT @TitluCarte = c.titlu

FROM imprumuturi\_membri im

JOIN carti c ON im.exemplar\_id = c.carte\_id

WHERE im.imprumut\_id IN (SELECT imprumut\_id FROM inserted);

PRINT 'Cartea "' + @TitluCarte + '" a fost împrumutată din nou.';

END;

Update-uri

-- 1. Actualizarea titlului unei cărți

UPDATE carti

SET titlu = '1984 Revisited'

WHERE carte\_id = 1;

-- 2. Modificarea numărului de pagini pentru o altă carte

UPDATE carti

SET numar\_pagini = 450

WHERE carte\_id = 2;

-- 3. Schimbarea stării de disponibilitate a unui exemplar

UPDATE exemplare

SET disponibil = 1

WHERE exemplar\_id = 3;

-- 4. Actualizarea adresei unui membru

UPDATE membri

SET adresa = 'Str. Libertății nr. 20'

WHERE membru\_id = 2;

-- 5. Modificarea culorii copertei unei cărți

UPDATE carti

SET culoare\_coperta = 'verde'

WHERE titlu = 'Crima si pedeapsa';

-- 6. Actualizarea numelui editorului pentru o carte

UPDATE editori

SET nume\_editor = 'Penguin Books'

WHERE editor\_id = 1;

-- 7. Schimbarea datelor de contact ale unui membru

UPDATE membri

SET telefon = '0722333444'

WHERE nume = 'Radulescu' AND prenume = 'Maria';

-- 8. Actualizarea genului unei cărți

UPDATE carti

SET gen\_id = 6

WHERE titlu = 'Povestea lui Edgar Sawtelle';

-- 9. Modificarea numărului de capitole pentru o carte

UPDATE carti

SET numar\_capitole = 85

WHERE titlu = 'Anna Karenina';

-- 10. Schimbarea disponibilității unui alt exemplar

UPDATE exemplare

SET disponibil = 0

WHERE exemplar\_id = 5;

-- 11: Actualizare a genului cărților scrise de un autor specific

UPDATE carti

SET gen\_id = g.gen\_id

FROM carti c

JOIN autori a ON c.autor\_id = a.autor\_id

JOIN genuri g ON g.nume\_gen = 'Fictiune'

WHERE a.nume\_autor = 'George Orwell';

-- 12: Actualizare a stării de disponibilitate a exemplarelor cărții în funcție de locație

UPDATE e

SET e.disponibil = 0

FROM exemplare e

JOIN carti\_rafturi cr ON e.carte\_id = cr.carte\_id

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

JOIN carti c ON e.carte\_id = c.carte\_id

WHERE c.titlu = 'Mica printesa'

AND l.nume\_locatie = 'Biblioteca Copiilor';

-- 13: Actualizare a numărului de pagini pentru cărțile împrumutate de un anumit membru

UPDATE c

SET numar\_pagini = numar\_pagini + 50

FROM carti c

JOIN exemplare e ON c.carte\_id = e.carte\_id

JOIN imprumuturi\_membri im ON e.exemplar\_id = im.exemplar\_id

JOIN membri m ON im.membru\_id = m.membru\_id

WHERE m.nume = 'Popescu' AND m.prenume = 'Ion';

-- 14: Actualizare a locației cărților cu un număr mare de capitole

UPDATE r

SET r.nume\_raft = 'Raft A'

FROM carti\_rafturi cr

JOIN carti c ON cr.carte\_id = c.carte\_id

JOIN rafturi r ON cr.raft\_id = r.raft\_id

WHERE c.numar\_capitole > 80;

-- 15: Actualizare a comentariilor pentru recenziile cu nota maximă pentru cărți din genul "Fictiune"

UPDATE recenzii

SET comentariu = 'O carte foarte captivantă și bine scrisă!'

WHERE nota = 5 AND carte\_id IN (SELECT carte\_id FROM carti WHERE gen\_id = 1);

**Delete-uri**

-- Sterge membru cu ID 5 din tabela imprumuturi\_membri

DELETE FROM imprumuturi\_membri WHERE membru\_id = 5;

-- Sterge cartile de pe raftul 4

DELETE FROM carti\_rafturi

WHERE raft\_id = 4;

-- Sterge cartile editate de editorul id ID 5

DELETE FROM carti\_editori

WHERE editor\_id = 5;

-- Șterge un autor specific și cărțile scrise de el

DELETE FROM autori

WHERE nume\_autor = 'Frances Hodgson Burnett';

-- Șterge un gen de carte

DELETE FROM genuri

WHERE nume\_gen = 'Poezie';

-- Șterge o recenzie bazată pe ID-ul recenziei

DELETE FROM recenzii

WHERE recenzie\_id = 7;

-- Șterge o cartea dintr-un raft specific

DELETE FROM carti\_rafturi

WHERE carte\_id = 10 AND raft\_id = 10;

-- Șterge un editor după ID

DELETE FROM editori

WHERE editor\_id = 5;

-- Șterge o locație de raft

DELETE FROM rafturi

WHERE locatie\_id = 4;

-- Șterge o înregistrare din tabelul de împrumuturi membri

DELETE FROM imprumuturi\_membri

WHERE imprumut\_id = 3;

-- Șterge o carte specifică din tabelul principal de cărți

DELETE FROM carti

WHERE titlu = 'Marie Curie: A Life' AND an\_publicare = 1937;

-- Ștergere exemplare dintr-un anumit raft dintr-o anumită locație

DELETE cr

FROM carti\_rafturi cr

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE l.nume\_locatie = 'Biblioteca Centrala'

AND r.nume\_raft = 'Raft A';

DELETE r

FROM rafturi r

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE l.nume\_locatie = 'Biblioteca Centrala'

AND r.nume\_raft = 'Raft A';

DELETE e

FROM exemplare e

JOIN carti\_rafturi cr ON e.carte\_id = cr.carte\_id

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE l.nume\_locatie = 'Biblioteca Centrala'

AND r.nume\_raft = 'Raft A';

-- Ștergere recenzii pentru cărți dintr-un anumit gen și cu anul publicării mai mic de 2000

DELETE r

FROM recenzii r

JOIN carti c ON r.carte\_id = c.carte\_id

JOIN genuri g ON c.gen\_id = g.gen\_id

WHERE g.nume\_gen = 'Fictiune'

AND c.an\_publicare < 2000;

-- Ștergere editori care au publicat cărți cu mai mult de 500 de pagini

DELETE ce

FROM carti\_editori ce

JOIN carti c ON ce.carte\_id = c.carte\_id

WHERE c.numar\_pagini > 500;

-- Ștergere editori care nu mai sunt legați de nicio carte cu mai mult de 500 de pagini

DELETE e

FROM editori e

WHERE NOT EXISTS (

SELECT 1

FROM carti\_editori ce

WHERE ce.editor\_id = e.editor\_id

);

-- Ștergere cărți care nu sunt disponibile într-o anumită locație

SELECT e.\*

FROM exemplare e

JOIN carti\_rafturi cr ON e.carte\_id = cr.carte\_id

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE e.disponibil = 0

AND l.nume\_locatie = 'Biblioteca Municipala';

DELETE cr

FROM carti\_rafturi cr

JOIN exemplare e ON cr.carte\_id = e.carte\_id

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE e.disponibil = 0

AND l.nume\_locatie = 'Biblioteca Municipala';

DELETE c

FROM carti c

JOIN exemplare e ON c.carte\_id = e.carte\_id

JOIN carti\_rafturi cr ON c.carte\_id = cr.carte\_id

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE e.disponibil = 0

AND l.nume\_locatie = 'Biblioteca Municipala';

-- Ștergere rafturi dintr-o anumită locație care sunt disponibile

SELECT cr.\*

FROM carti\_rafturi cr

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE l.nume\_locatie = 'Biblioteca Universitara';

DELETE cr

FROM carti\_rafturi cr

JOIN rafturi r ON cr.raft\_id = r.raft\_id

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE l.nume\_locatie = 'Biblioteca Universitara';

DELETE r

FROM rafturi r

JOIN locatii l ON r.locatie\_id = l.locatie\_id

WHERE r.disponibil = 1

AND l.nume\_locatie = 'Biblioteca Universitara';

**Tranzactii**

--Creare tabelă pentru logul modificărilor recenziilor

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'LogRecenzii')

BEGIN

CREATE TABLE LogRecenzii (

log\_id INT PRIMARY KEY IDENTITY(1,1),

recenzie\_id UNIQUEIDENTIFIER,

carte\_id INT,

membru\_id INT,

nota INT,

comentariu VARCHAR(100),

data\_recenzie DATETIME DEFAULT GETDATE(),

data\_modificare DATETIME DEFAULT GETDATE()

);

END;

-- Tranzactie 1: Actualizare a unui câmp în tabelul carti (simulare)

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @carteID INT = 1;

DECLARE @paginiNoi INT = 250;

IF EXISTS (SELECT 1 FROM carti WHERE carte\_id = @carteID)

BEGIN

UPDATE carti

SET numar\_pagini = @paginiNoi

WHERE carte\_id = @carteID;

END

ELSE

BEGIN

THROW 50001, 'Cartea specificată nu a fost găsită.', 1;

END;

DECLARE @membruID INT = 1;

DECLARE @nota INT = 5;

DECLARE @comentariu NVARCHAR(MAX) = 'O carte minunată! Recomand cu căldură.';

DECLARE @recenzieID INT = 4;

INSERT INTO recenzii (recenzie\_id, carte\_id, membru\_id, nota, comentariu)

VALUES (@recenzieID, @carteID, @membruID, @nota, @comentariu);

INSERT INTO recenzii (recenzie\_id, carte\_id, membru\_id, nota, comentariu)

VALUES (@recenzieID, @carteID, @membruID, @nota, @comentariu);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

-- Tranzactie 2: Modificarea membrilor

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @nume NVARCHAR(100) = 'John';

DECLARE @prenume NVARCHAR(100) = 'Doe';

DECLARE @adresa NVARCHAR(255) = 'Str. Exemplu, Nr. 123';

DECLARE @telefon NVARCHAR(20) = '1234567890';

INSERT INTO membri (nume, prenume, adresa, telefon)

VALUES (@nume, @prenume, @adresa, @telefon);

DECLARE @membruID INT = SCOPE\_IDENTITY();

DECLARE @nouaAdresa NVARCHAR(255) = 'Str. Noua, Nr. 456';

UPDATE membri

SET adresa = @nouaAdresa

WHERE membru\_id = @membruID;

DECLARE @membruIDToDelete INT = @membruID;

DELETE FROM membri

WHERE membru\_id = @membruIDToDelete;

DELETE FROM membri WHERE membru\_id = @membruIDToDelete;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

-- Tranzactie 3 Inserare de autor nou

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @numeAutor NVARCHAR(100) = 'Jane Austen';

DECLARE @nationalitate NVARCHAR(100) = 'Engleză';

DECLARE @autor\_id INT = 31;

INSERT INTO autori (autor\_id,nume\_autor, nationalitate)

VALUES (@autor\_id, @numeAutor, @nationalitate);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

-- Tranzactie 4: Operatii pe carti

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @carteIDToDelete INT = 1;

DELETE FROM carti

WHERE carte\_id = @carteIDToDelete;

DECLARE @titlu NVARCHAR(100) = 'Noua carte';

DECLARE @autorID INT = 1;

DECLARE @genID INT = 1;

DECLARE @anPublicare INT = 2024;

DECLARE @numarPagini INT = 300;

DECLARE @numarCapitole INT = 20;

DECLARE @culoareCoperta NVARCHAR(100) = 'Albastră';

INSERT INTO carti (titlu, autor\_id, gen\_id, an\_publicare, numar\_pagini, numar\_capitole, culoare\_coperta)

VALUES (@titlu, @autorID, @genID, @anPublicare, @numarPagini, @numarCapitole, @culoareCoperta);

DECLARE @carteIDToUpdate INT = 2;

DECLARE @nouaCuloareCoperta NVARCHAR(100) = 'Verde';

UPDATE carti

SET culoare\_coperta = @nouaCuloareCoperta

WHERE carte\_id = @carteIDToUpdate;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

-- Tranzactie 5: Operatii cu recenzii si genuri

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @carteID INT = 2;

DECLARE @paginiNoi INT = 350;

IF EXISTS (SELECT 1 FROM carti WHERE carte\_id = @carteID)

BEGIN

UPDATE carti

SET numar\_pagini = @paginiNoi

WHERE carte\_id = @carteID;

END

ELSE

BEGIN

THROW 50001, 'Cartea specificată nu a fost găsită.', 1;

END;

DECLARE @membruID INT = 1;

DECLARE @nota INT = 4;

DECLARE @comentariu NVARCHAR(MAX) = 'O carte interesantă, cu un final neașteptat.';

INSERT INTO recenzii (carte\_id, membru\_id, nota, comentariu)

VALUES (@carteID, @membruID, @nota, @comentariu);

DECLARE @carteIDToDelete INT = 3;

DELETE FROM carti

WHERE carte\_id = @carteIDToDelete;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

-- Tranzație 6: Operatii pe membri si carti

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @nume NVARCHAR(100) = 'Alice';

DECLARE @prenume NVARCHAR(100) = 'Smith';

DECLARE @adresa NVARCHAR(255) = 'Str. Principala, Nr. 123';

DECLARE @telefon NVARCHAR(20) = '9876543210';

INSERT INTO membri (nume, prenume, adresa, telefon)

VALUES (@nume, @prenume, @adresa, @telefon);

DECLARE @membruID INT = SCOPE\_IDENTITY();

DECLARE @nouaAdresa NVARCHAR(255) = 'Str. Noua, Nr. 456';

UPDATE membri

SET adresa = @nouaAdresa

WHERE membru\_id = @membruID;

DECLARE @membruIDToDelete INT = @membruID;

DELETE FROM membri

WHERE membru\_id = @membruIDToDelete;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;