

Proiect baze de date

*Gestiunea bibliotecilor școlare
din Romania*

Mai 2022

Lung Alexandra
Grupa 143

Facultatea de Matematică și Informatică
Universitatea din București

Cuprins

1. *Descrierea modelului real și a regulilor de funcționare*
2. *Prezentare constrângeri*
3. *Descrierea entităților*
4. *Descrierea relațiilor*
5. *Descrierea atributelor*
6. *Diagrama entitate-relație*
7. *Diagrama conceptuală*
8. *Scheme relaționale*
9. *Normalizare(FN1-FN3)*
 - a. *Forma normală 1(FN1)*
 - b. *Forma normală 2(FN2)*
 - c. *Forma normală 3(FN3)*
10. *Crearea tabelelor in SQL, inserarea datelor și adăugarea constrângerilor pentru:*
 - a. *Tabela Editura*
 - b. *Tabela Oras*
 - c. *Tabela Biblioteca*
 - d. *Tabela Elev*
 - e. *Tabela Legitimatie*
 - f. *Tabela Autor*
 - g. *Tabela Carte*
 - h. *Tabela FisaDeLectura*
 - i. *Tabela Domeniu*

- j. Tabela *Angajat*
- k. Tabela *Imprumut*
- l. Tabela *Scrisă*
- m. Tabela *Apartține*

11. Cereri complexe SQL

- a. Prima cerere
- b. A doua cerere
- c. A treia cerere
- d. A patra cerere
- e. A cincea cerere

12. Trei cereri de actualizare și suprimare a datelor

13. Crearea unei secvențe SQL folosite pentru inserarea înregistrărilor în tabele

14. Crearea unei vizualizări compuse

- a. Crearea vizualizării
- b. Operații LMD permise
- c. Operații LMD nepermise

15. Crearea unui index care optimizează cereri de căutare pe 2 criterii

16. Cereri SQL care utilizează outer-join și division

- a. Outer-join
- b. Division

17. Optimizarea unei cereri

- a. Cerere SQL
- b. Expresii algebrice
- c. Optimizare

d. Arbore algebric

18. Normalizare și denormalizare

- a. Forma normală Boyce-Codd (BCNF)*
- b. Forma normală 4 (FN4)*
- c. Forma normală 5 (FN5)*
- d. Aplicare denormalizare*

19. Bibliografie

1. Descrierea modelului real și a regulilor de funcționare

Trebuie menționat faptul că lucrarea de față își propune înainte de toate să identifice cele mai importante aspecte și probleme ale biblioteconomiei precum și descrierea implementării unui sistem digital de gestiune a unei biblioteci.

Modelul de date pe care l-am creat își propune să gestioneze informații ale bibliotecilor școlare din România. Astfel, pot exista mai multe biblioteci în cadrul unităților de învățământ, aflate în diverse orașe. În cadrul bibliotecilor școlare lucrează angajați care pot ocupa următoarele funcții: manager, bibliotecar, operator de introducere date, îngrijitor sau paznic. În plus, în cadrul bazei de date se va ține evidența elevilor, a împrumuturilor pe care aceștia le efectuează, precum și a cărților disponibile, autorii acestora, editura și domeniile din care fac parte. Pentru un elev se rețin și date despre legitimația acestuia, dar și fișa de lectură.

Modelul de date este util deoarece consider că este dificil de ținut evidența bibliotecilor școlare din întreaga țară, iar, astfel, o bază de date bine proiectată ar fi utilă atât pentru utilizatori, cât și pentru administratori.

Modelul de față respectă anumite restricții de funcționare:

- O bibliotecă poate avea unul sau mai mulți angajați, sau niciunul, în cazul în care aceasta nu are personal calificat, dar încă există baza de date
- Un angajat lucrează în cadrul unei singure biblioteci
- O bibliotecă școlară poate avea mai multe cărți, cel puțin una
- O carte este înregistrată la o singură bibliotecă

- O carte poate să aparțină mai multor domenii, dar trebuie să aparțină cel puțin unui domeniu
- Un domeniu poate avea mai multe cărți sau niciuna
- O editură poate avea mai multe cărți publicate, însă trebuie să aibă măcar o carte publicată, iar cartea poate fi publicată de o singură editură.
- Un autor poate publica mai multe cărți, dar trebuie să aibă cel puțin o carte publicată
- O carte poate fi scrisă de cel puțin un autor
- O bibliotecă școlară se poate afla într-un singur oraș
- Un oraș poate avea mai multe biblioteci școlare
- O carte poate fi trecută pe cel puțin o fișă de lectură sau niciuna dacă nu a fost împrumutată
- Un elev poate împrumuta mai multe cărți din biblioteci diferite

2. Prezentare constrângeri

Modelului de date i-au fost impuse următoarele constrângeri:

- O bibliotecă poate avea mai mulți angajați sau niciunul.
- Un angajat lucrează în cadrul unei singure biblioteci.
- O bibliotecă poate avea mai multe cărți și cel puțin una.
- O carte este înregistrată la o singură bibliotecă.
- O carte poate să aparțină mai multor domenii sau cel puțin unuia.
- Un domeniu poate avea mai multe cărți sau niciuna.
- O carte poate fi publicată de o singură editură.
- O editură poate avea mai multe cărți publicate și cel puțin una.
- Un autor poate publica mai multe cărți și cel puțin una.
- O carte poate fi scrisă de cel puțin un autor.
- O bibliotecă se poate afla într-un singur oraș.
- Un oraș poate avea mai multe biblioteci școlare
- O carte poate fi trecută pe cel puțin o fișă de lectură sau niciuna.
- O fișă de lectură are cel puțin o carte
- O fișă de lectură are un singur elev.
- Un elev poate avea mai multe fise de lectură sau niciuna.
- O legitimație aparține unui singur elev.
- Un elev poate avea mai multe legitimații sau niciuna.
- Un elev poate împrumuta mai multe cărți din biblioteci școlare diferite.

3. Descrierea entităților

Modelul de date proiectat are ca entități structurile:

- **Autor** = entitate care reține date despre o persoană care a publicat cel puțin o carte care există în cel o bibliotecă, cheia primară a entității fiind *id_autor#*
- **Editură** = entitate care reține date despre o editură (instituție) editează cărți, cheia primară a entității fiind *id_editura#*
- **Domeniu** = entitate care reține date despre un anumit domeniu în care se încadrează cărțile dintr-o bibliotecă, cheia primară a entității fiind *id_domeniu#*
- **Oraș** = entitate care reține orașele din România în care se găsesc bibliotecile din unitățile de învățământ, cheia primară a entității fiind *id_oras#*
- **Elev** = entitate care reține date despre un elev care frecventează o unitate de învățământ, cheia primară a entității fiind *id_elev#*
- **Carte** = entitate care reține date despre o carte care există într-o bibliotecă, cheia primară a entității fiind *id_carte#*
- **Bibliotecă** = entitate care reține date despre o bibliotecă aflată într-o unitate de învățământ, cheia primară a entității fiind *id_biblioteca#*
- **Angajat** = entitate care reține date relevante despre un angajat al unei biblioteci (nume, prenume, adresa, mail, telefon, salariu, funcție, data angajării), cheia primară a entității fiind *id_angajat#*
- **Legitimatie** = entitate care reține date despre un elev înscris la bibliotecă (id elev și data de expirare), cheia primară a entității fiind *id_legitimatie#*
- **Fisa de lectură** = entitate care reține date referitoare la istoricul unei cărți: de câte ori a fost împrumutată și în ce condiții se află cartea, cheia primară a entității fiind *id_fisa_de_lectura#*

4. Descrierea relațiilor

Descrierea relațiilor modelului de date precizând cardinalitatea minimă și maximă.

Orașul_are_Biblioteca = relația care leagă entitățile Oraș și Biblioteca refectionând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:1 și maximă de 1:n

Biblioteca_are_Angajat = relația care leagă entitățile Biblioteca și Angajat refectionând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:0 și maximă de 1:n

Cartea_apartine_Domeniului = relația care leagă entitățile Carte și Domeniu refectionând legătura dintre acestea (o carte apartine unui domeniu). Aceasta are cardinalitatea minimă de 0:1 și maximă de m:n

Editura_publică_Cartea = relația care leagă entitățile Editură și Carte refectionând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:1 și maximă de 1:n

Autorul_scrie_Carte = relația care leagă entitățile Autor și Carte refectionând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:1 și maximă de m:n

Cartea_se_găsește_în_FisaDeLectură = relația care leagă entitățile Carte și Fisă de lectură refectionând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:0 și maximă de 1:n

Elevul_are_FisaDeLectură = relația care leagă entitățile Elev și Fisă de lectură refectionând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:0 și maximă de 1:n

Relația de tip 3 dintre entitățile Elev, Biblioteca și Carte = relația care leagă entitățile Elev, Biblioteca și Carte refectionând legătura dintre cele patru tabele (un elev poate împrumuta mai multe cărți din diferite biblioteci).

5. Descrierea atributelor

Entitatea **Autor** are ca atribute:

- id_autor = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui autor
- nume = variabilă de tip șir de caractere de lungime maximă 55, care nu poate fi nulă, reprezentând numele unui autor
- prenume = variabilă de tip șir de caractere de lungime maximă 55, care nu poate fi nulă, reprezentând prenumele unui autor

Entitatea **Editură** are ca atribute:

- id_editură = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei edituri
- nume = variabilă de tip șir de caractere de lungime maximă 55, care nu poate fi nulă, reprezentând numele unei edituri
- adresă = variabilă de tip șir de caractere de lungime maximă 100, reprezentând adresa unde se găsește editura
- email = variabilă de tip șir de caractere de lungime maximă 55, reprezentând email-ul prin care se poate contacta editura

Entitatea **Domeniu** are ca atribute:

- id_domeniu = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unui domeniu
- nume = variabilă de tip șir de caractere de lungime maximă 30, care nu poate fi nulă, reprezentând numele unui domeniu

Entitatea **Oraș** are ca atribute:

- id_oras = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unui oraș
- nume = variabilă de tip șir de caractere de lungime maximă 30, care nu poate fi nulă, reprezentând numele unui oraș

Entitatea ***Elev*** are ca atribute:

- id_elev = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui elev
- nume = variabilă de tip șir de caractere de lungime maximă 55, care nu poate fi nulă, reprezentând numele unui elev
- prenume = variabilă de tip șir de caractere de lungime maximă 55, care nu poate fi nulă, reprezentând prenumele unui elev
- adresă = variabilă de tip șir de caractere de lungime maximă 100, reprezentând adresa unde locuiește elevul
- email = variabilă de tip șir de caractere de lungime maximă 55, reprezentând email-ul prin care se poate contacta elevul
- telefon = variabilă de tip șir de caractere de lungime maximă 11, reprezentând numărul de telefon prin care se poate contacta elevul

Entitatea ***Bibliotecă*** are ca atribute:

- id_bibliotecă = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei biblioteci
- nume = variabilă de tip șir de caractere de lungime maximă 200, care nu poate fi nulă, reprezentând numele unei biblioteci școlare
- adresă = variabilă de tip șir de caractere de lungime maximă 60, reprezentând adresa unde se găsește biblioteca
- id_oras = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul orașului în care se găsește biblioteca. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Oraș***

Entitatea **Legitimație** are ca attribute:

- id_legitimație = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei legitimații
- id_elev = id_legitimație = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul elevului care îi corespunde legitimația. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul **Elev**
- data_expirării = variabilă de tip dată calendaristică, reprezentând data expirării legitimației

Entitatea **Angajat** are ca attribute:

- id_angajat = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui angajat
- id_biblioteca = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul bibliotecii la care lucrează angajatul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul **Biblioteca**
- nume = variabilă de tip șir de caractere de lungime maximă 20, care nu poate fi nulă, reprezentând numele unui angajat
- prenume = variabilă de tip șir de caractere de lungime maximă 20, care nu poate fi nulă, reprezentând prenumele unui angajat
- adresă = variabilă de tip șir de caractere de lungime maximă 100, reprezentând adresa unde locuiește angajatul
- email = variabilă de tip șir de caractere de lungime maximă 55, reprezentând email-ul prin care se poate contacta angajatul
- telefon = variabilă de tip șir de caractere de lungime maximă 11, reprezentând numărul de telefon prin care se poate contacta angajatul
- salariu = variabilă de tip real, reprezentând salariul angajatului
- functie = variabilă de tip șir de caractere de lungime maximă 30, reprezentând funcția ocupată de un angajat. Atributul poate lua valorile : manager, bibliotecar, operator de introducere date, îngrijitor sau paznic
- data_angajării = variabilă de tip dată calendaristică, reprezentând data angajării unui angajat

Entitatea **Carte** are ca attribute:

- id_carte = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei cărți
- id_editură = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei edituri care a publicat cartea. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul **Editură**
- titlu = variabilă de tip șir de caractere de lungime maximă 1000, care nu poate fi nulă, reprezentând titlul unei cărți
- serie = variabilă de tip șir de caractere de lungime maximă 200, reprezentând seria din care face parte o carte
- număr_volum = variabilă de tip număr real, reprezentând numărul de ordine al unei cărți dintr-o serie
- preț = variabilă de tip număr real, care nu poate fi nulă, reprezentând prețul cărții
- limba = variabilă de tip șir de caractere de lungime maximă 200, reprezentând limba în care e scrisă cartea
- număr_pagini = variabilă de tip întreg, reprezentând numărul de pagini cărții
- ISBN = variabilă de tip șir de caractere de lungime maximă 21, reprezentând numărul standardizat al cărții

Entitatea **FisaDeLectura** are ca attribute:

- id_fisa_de_lectura = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei fișă de lectură
- id_carte = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei cărți care se găsește într-o fișă de lectură. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul **Carte**
- id_elev = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui elev care se găsește într-o fișă de lectură. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul **Elev**
- data_împrumutului = variabilă de tip dată calendaristică, reprezentând data la care a fost efectuat împrumutul
- data_limita = variabilă de tip dată calendaristică, reprezentând data limită la care trebuie restituită cartea
- data_restituire = variabilă de tip dată calendaristică, reprezentând data la care a fost restituită cartea împrumutată

Relația ***Autor_scrie_Carte*** are ca attribute:

- **id_carte** = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei cărți. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Carte***
- **id_autor** = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui autor. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Autor***
- ***Tuplul (id_carte, id_autor)*** este cheia primară a tabelului asociativ ***Scrisa***
- **data_publicare** = variabilă de tip dată calendaristică, reprezentând data la care a fost publicată cartea

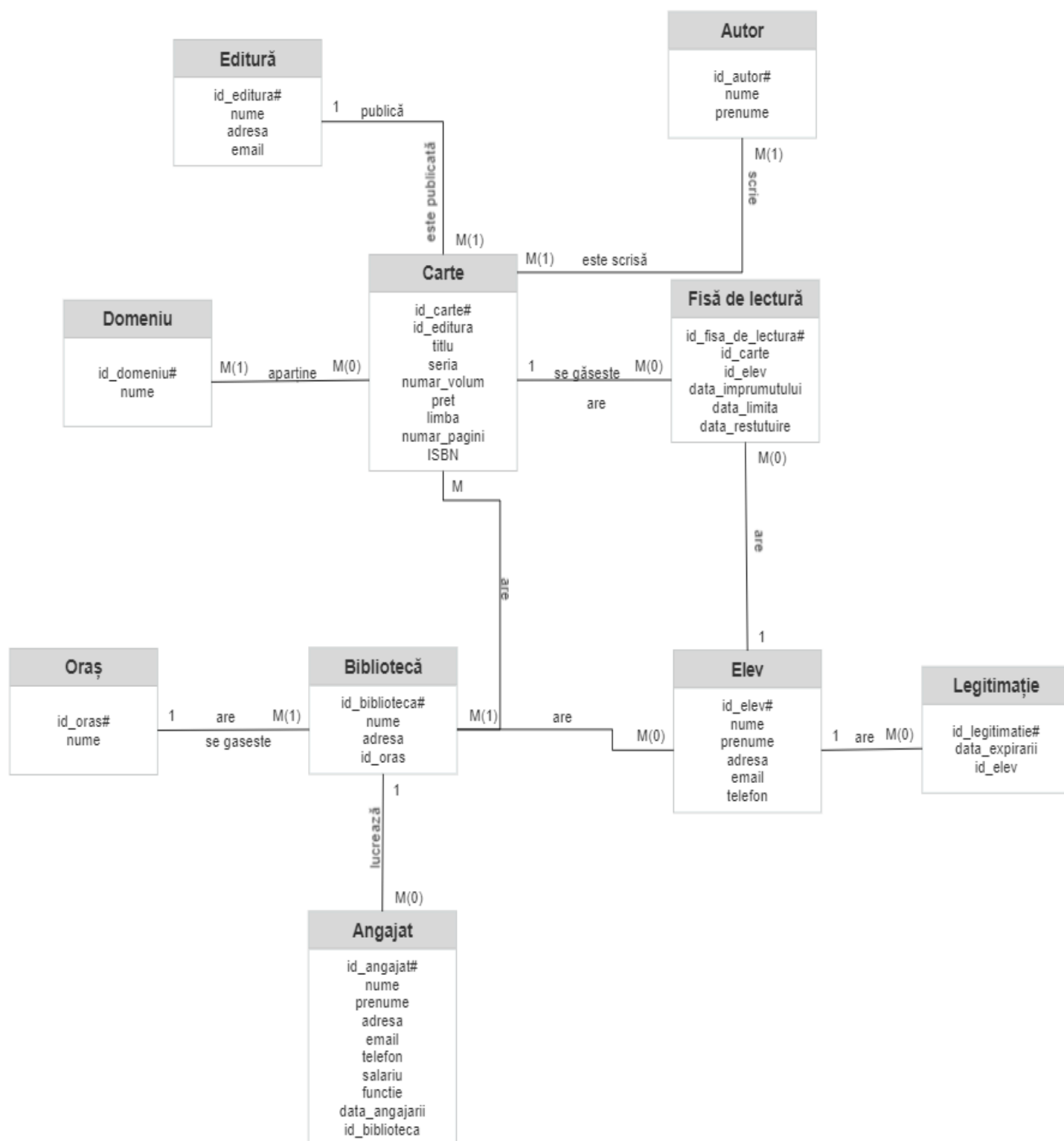
Relația ***Carte_apartine_Domeniu*** are ca attribute:

- **id_apartinere** = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unei apartineri, fiind cheia primară a tabelului asociativ ***Apartine***
- **id_carte** = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei cărți. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Carte***
- **id_domeniu** = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unui domeniu. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Domeniu***

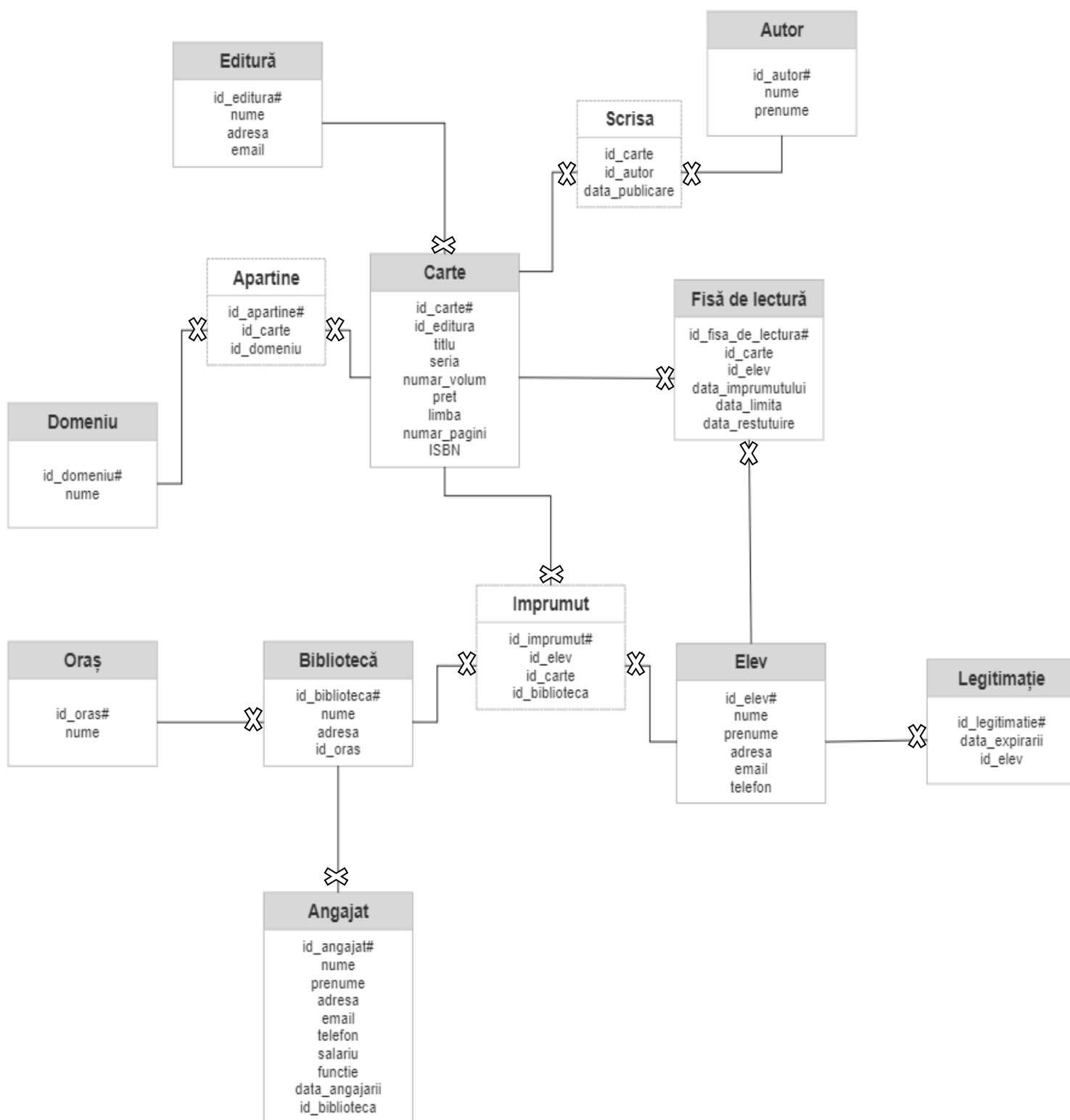
Relația de tip 3 dintre entitățile Elev, Bibliotecă și Carte are ca attribute:

- **id_imprumut** = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui împrumut, fiind cheia primară a tabelului asociativ ***Imprumut***
- **id_elev** = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui elev. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Elev***
- **id_carte** = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei cărți. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Carte***
- **id_bibliotecă** = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul bibliotecii. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Bibliotecă***

6. Diagrama entitate-relație



7. Diagrama conceptuală



8. Scheme relaționale

Schemele relaționale corespunzătoare diagramei conceptuale sunt următoarele:

Autor(id_autor#, nume, prenume)

Editură(id_editura#, nume, adresa, email)

Domeniu(id_domeniu#, nume)

Oraș(id_oras#, nume)

Elev(id_elev#, nume, prenume, adresa, email, telefon)

Carte(id_carte#, titlu, serie, numar_volum, preț, limba, numar_pagini, ISBN, id_editura)

Biblioteca(id_biblioteca#, nume, adresa, id_oras)

Apartține(id_apartinere #,id_carte ,id_domeniu)

Scrisa(id_carte#,id_autor#,data_publicare)

Angajat(id_Angajat#, nume, prenume, adresa, email, telefon, salariu, functie, data_anagajarii,id_biblioteca)

Legitimatie(id_legitimatie#, data_expirarii , id_elev)

Imprumut(id_imprumut#,id _elev, id_carte,id _biblioteca)

FisaDeLectura(id_fisa_de_lectura#,id_carte, id _elev, data_imprumutului, data_limita,data_restituire)

9. Normalizare(FN1-FN3)

Normalizarea este o tehnică de proiectare a bazelor de date prin care se elimină (sau se evită) anumite anomalii și inconsistențe ale datelor. O bază de date bine proiectată nu permite ca datele să fie redundante, adică aceeași informație să se găsească în locuri diferite sau să se memoreze în baza de date informații care se pot deduce pe baza altor informații memorate în aceeași bază de date.

Formele normale se aplică fiecărei entități în parte. O bază de date (sau un ERD) se găsește într-o anumită formă normală doar dacă toate entitățile se găsesc în acea formă normală.

Edgar Codd a definit primele trei forme normale 1NF, 2NF și 3NF. Ulterior s-au mai definit formele normale 4NF, 5NF, 6NF care însă sunt rar folosite în proiectarea bazelor de date.

a. Forma normală 1(FN1)

O entitate se găsește în prima formă normală dacă și numai dacă:

- nu există attribute cu valori multiple*
- nu există attribute sau grupuri de attribute care se repetă.*

Cu alte cuvinte toate attributele trebuie să fie atomice, adică să conțină o singură informație. De asemenea, prima formă normală cere și ca fiecare înregistrare să fie definită astfel încât să fie identificată în mod unic prin intermediul unei chei primare.

În modelul implementat se respectă toate cerințele primei forme normale: nu există grupuri repetitive și orice înregistrare aleasă este unică, aceasta fiind identificată prin intermediul cheii primare.

Pentru a exemplifica normalizarea se consideră următorul exemplu:

<i>Autor</i>	<i>Carte</i>
<i>AUT 1</i>	<i>C1,C2,C3</i>
<i>AUT 2</i>	<i>C1,C4</i>

Exemplu non-FN1

<i>Autor</i>	<i>Carte</i>
<i>AUT 1</i>	<i>C1</i>
<i>AUT1</i>	<i>C2</i>
<i>AUT1</i>	<i>C3</i>
<i>AUT 2</i>	<i>C1</i>
<i>AUT 2</i>	<i>C4</i>

Exemplu FN1

b. Forma normală 2(FN2)

O entitate se găsește în a doua formă normală dacă și numai dacă se găsește în prima formă normală și în plus, orice atribut care nu face parte din UID (Unique IDentifier) va depinde de întregul UID nu doar de o parte a acestuia.

În modelul implementat se respectă toate cerințele celei de a doua forme normale: toate relațiile sunt în FN1 și fiecare atribut, care nu este cheie primară, din fiecare entitate este depedndent de aceasta.

Se consideră următorul exemplu pentru normalizare:

<i>id_autor#</i>	<i>nume</i>	<i>prenume</i>	<i>id_carte#</i>	<i>data_publicare</i>
<i>1</i>	<i>Armentrout</i>	<i>Jennifer</i>	<i>C1</i>	<i>2018-02-19</i>
<i>2</i>	<i>Armentrout</i>	<i>Jennifer</i>	<i>C2</i>	<i>2011-07-29</i>
<i>3</i>	<i>Morgan</i>	<i>Richard</i>	<i>C3</i>	<i>2002-03-11</i>

Exemplu non-FN2

Vom aplica regula Casey-Delobel pentru FN2. Observăm următoarele dependențe:

$\{id_autor\} \rightarrow \{nume, prenume\} \rightarrow \text{determină funcțional autor}$

$\{id_autor\}, \{id_carte\} \rightarrow \{data_publicare\}$

Vom obține următoarele tabele:

$id_autor\#$	$nume$	$prenume$
1	Armentrout	Jennifer
2	Armentrout	Jennifer
3	Morgan	Richard

$id_autor\#$	$id_carte\#$	$data_publicare$
1	C1	2018-02-19
2	C2	2011-07-29
3	C3	2002-03-11

Exemplu FN2

c. Forma normală 3(FN3)

O entitate se găsește în a treia formă normală dacă și numai dacă se găsește în a doua formă normală și în plus niciun atribut care nu este parte a UID-ului nu depinde de un alt atribut non-UID. Cu alte cuvinte, nu se acceptă dependențe tranzitive, adică un atribut să depindă de UID în mod indirect.

În modelul implementat se respectă toate cerințele celei de a treia forme normale.

Se consideră următorul exemplu pentru normalizare:

<i>id_carte#</i>	<i>id_editura#</i>	<i>nume</i>	<i>adresa</i>	<i>email</i>
<i>C1</i>	<i>1</i>	<i>Leda</i>	<i>adr1</i>	<i>email1</i>
<i>C2</i>	<i>2</i>	<i>Litera</i>	<i>adr2</i>	<i>email2</i>
<i>C3</i>	<i>3</i>	<i>Paladin</i>	<i>adr3</i>	<i>email3</i>

Exemplu non-FN3

Vom aplica regula Casey-Delobel pentru FN2. Observăm următoarea dependență tranzitivă:

id_carte# -> *nume(editura)* -> *adresa, email(editura)*

Vom obține următoarele tabele:

<i>id_editura#</i>	<i>nume</i>	<i>adresa</i>	<i>email</i>
<i>1</i>	<i>Leda</i>	<i>adr1</i>	<i>email1</i>
<i>2</i>	<i>Litera</i>	<i>adr2</i>	<i>email2</i>
<i>3</i>	<i>Paladin</i>	<i>adr3</i>	<i>email3</i>

<i>id_carte#</i>	<i>id_editura#</i>
<i>C1</i>	<i>1</i>
<i>C2</i>	<i>2</i>
<i>C3</i>	<i>3</i>

Exemplu FN3

10. Crearea tabelelor în SQL, inserarea datelor și adăugarea constrângerilor

Secvențele corespunzătoare creerii și inserării datelor în SQL (constrângerile au fost adăugate o dată cu crearea tabelelor) sunt:

a. Tabela Editura

```
create table Editura(  
id_editura    int identity(1,1)    not null    primary key,  
nume          varchar(55)          not null,  
adresa        varchar(100),  
email         varchar(55)  
);
```

```
INSERT INTO Editura(nume,adresa,email)  
VALUES  
('Leda', 'Strada Mihai Eminescu Nr. 54A, Bucuresti 030167',  
'leda@ledascholars.org.'),  
('Leda Edge', 'Strada Mihai Eminescu Nr. 54A, Bucuresti 030167',  
'leda@ledascholars.org.' ),  
('Litera', ' Strada Moeciu Nr. 7A, Bucuresti 077190','contact@litera.ro'),  
('Paladin', 'Soseaua Unirii 216, Comuna Corbeanca', 'comenzi@editura-  
art.ro'),  
('Hodder & Stoughton General Division',  
null,'hukdcustomerservices@hachette.co.uk' ),  
('All', ' Bulevardul Constructorilor 20A, Bucuresti 260512','info@all.ro' ),  
('Art', 'Splaiul Independentei, Bucuresti 060043','manuscris@editura-  
art.ro.' );
```

b. Tabela Oras

```
CREATE TABLE Oras(  
id_oras      int identity    constraint pk_Oras primary key,  
nume         varchar(20)     not null,  
);
```

```
CREATE SEQUENCE SECV_ORAS
```

```
As int
```

```
INCREMENT by 1
```

```
START WITH 1
```

```
MAXVALUE 500
```

```
NO CYCLE;
```

```
SET IDENTITY_INSERT Oras ON
```

```
INSERT INTO Oras(id_oras,nume) VALUES
```

```
(NEXT VALUE FOR SECV_ORAS,'Bucuresti'),
```

```
(NEXT VALUE FOR SECV_ORAS,'Iasi'),
```

```
(NEXT VALUE FOR SECV_ORAS,'Pitesti'),
```

```
(NEXT VALUE FOR SECV_ORAS,'Timisoara'),
```

```
(NEXT VALUE FOR SECV_ORAS,'Cluj-Napoca'),
```

```
(NEXT VALUE FOR SECV_ORAS,'Ploiesti');
```

```
SET IDENTITY_INSERT Oras OFF
```

c. Tabela Biblioteca

```
CREATE TABLE Biblioteca(
```

```
id_biblioteca int identity(1,1) constraint pk_Biblioteca primary key,
```

```
nume          varchar(200) not null,
```

```
adresa        varchar(60) not null,
```

```
id_oras        int not null,
```

```
constraint fk_Biblioteca foreign key (id_oras) references Oras(id_oras),  
);
```

```
INSERT INTO Biblioteca VALUES
```

```
('Biblioteca Liceului Teoretic „Al. I. Cuza”', 'Strada Ion Creanga 37, Iasi 700317', 2),
```

```
('Biblioteca Liceului „Grigore Moisil” Timisoara', 'Strada Ghirlandei 4, Timisoara  
300231', 4),
```

```
('Biblioteca Liceului Teoretic „Ion Barbu”', 'Strada Transilvania 6, Pitesti', 3),
```

```
('Biblioteca Colegiului National „Spiru Haret” din Bucuresti', 'Italiana 17, Bucuresti  
021021', 1),
```

```
('Biblioteca Liceului Teoretic „Avram Iancu” Cluj-Napoca', 'Strada Onisifor Ghibu 33,  
Cluj-Napoca 400394', 5),
```

```
('Biblioteca Colegiului National „Ion Luca Caragiale” Ploiesti', 'Strada Gheorghe Doja  
98, Ploiesti 100164', 6);
```

d. Tabela Elev

```
CREATE TABLE Elev(  
    id_elev      int identity(1,1)      not null primary key,  
    nume         varchar(55)            not null,  
    prenume      varchar(55)            not null,  
    adresa       varchar(100),  
    email        varchar(55),  
    telefon      varchar(11)  
);  
  
INSERT INTO Elev VALUES  
    ('Stefan', 'Ovidiu', 'Calea Victoriei 63,Bucuresti',  
    'stefanovidiu@gmail.com', '0738586317'),  
    ('Ilie', 'Ovidiu', 'Strada Dristorului 10,Bucuresti', 'ilieovidiu@gmail.com',  
    '0726081740'),  
    ('Popescu', 'Lucian', 'Strada Lipscani 43,Bucuresti',  
    'popesculucian@gmail.com', '0723735665'),  
    ('Ionescu', 'Bogdan', 'Soseaua Mihai Bravu 25,Bucuresti',  
    'ionescubogdan@gmail.com', '0723503235'),  
    ('Mihail', 'Dan', 'Strada Grigore Alexandrescu 7,Bucuresti',  
    'mihaildan@gmail.com', '0735518610'),  
    ('Popa', 'Alin', 'Strada Agricultori 21,Bucuresti', 'popaalin@gmail.com',  
    '0742624114'),  
    ('Negoitescu', 'Alin', 'Calea Victoriei 63,Bucuresti',  
    'negoitescualin@gmail.com', '0700535017'),  
    ('Dumitrescu', 'Mircea', 'Str. Calea Vitan 55,Bucuresti',  
    'dumitrescumircea@gmail.com', '0778123420'),  
    ('Vlad', 'Lucian', 'Str. Calea Vitan 55,Bucuresti', 'vladlucian@gmail.com',  
    '0708432251'),  
    ('Mocanu', 'Gheorghe', 'Calea Mosilor 13,Bucuresti',  
    'mocanugheorghe@gmail.com', '0716055030'),  
    ('Rusu', 'Mircea', 'Calea Mosilor 13,Bucuresti', 'rusumircea@gmail.com',  
    '0776704550'),  
    ('Barbu', 'Mircea', 'Calea Aurel Vlaicu 15,Bucuresti',  
    'barbumircea@gmail.com', '0754885133'),
```



```
('Dobre', 'Manuel', 'Str. Calea Vitan 55,Bucuresti',  
'dobremanuel@gmail.com', '0735731517'),  
('Dinescu', 'Viorel', 'Soseaua Mihai Bravu 25,Bucuresti',  
'dinescuviorel@gmail.com', '0731476668'),  
('Codreanu', 'Horia', 'Strada Berzei 2,Bucuresti',  
'codreanuhoria@gmail.com', '0746327686'),  
('Draghici', 'Lucian', 'Strada Grigore Alexandrescu 7,Bucuresti',  
'draghicolucian@gmail.com', '0727556710'),  
('Pop', 'Manuel', 'Calea Mosilor 13,Bucuresti', 'popmanuel@gmail.com',  
'0773263442'),  
('Nicolescu', 'Ovidiu', 'Strada Berzei 2,Bucuresti',  
'nicolescuvidiu@gmail.com', '0706216628'),  
('Tudor', 'Horia', 'Calea Aurel Vlaicu 15,Bucuresti',  
'tudorhoria@gmail.com', '0728563567'),  
('Marin', 'Manuel', 'Str. Calea Vitan 55,Bucuresti',  
'marinmanuel@gmail.com', '0705142207'),  
('Tanase', 'Gabriel', 'Soseaua Oltenitei 12,Bucuresti',  
'tanasegabriel@gmail.com', '0722812520');
```

e. Tabela Legitimatie

```
CREATE TABLE Legitimatie(  
    id_legitimatie int identity(1,1) constraint pk_Legitimatie primary key,  
    data_expirarii date,  
    id_elev int not null,  
    constraint fk_Legitimatie foreign key(id_elev) references Elev(id_elev)  
);
```

```
INSERT INTO Legitimatie VALUES  
('2022-03-15',11),  
('2021-07-29',3),  
('2021-05-18',15),  
('2021-11-18',10),  
('2021-05-27',3),
```

```
('2020-10-22',10),  
( '2020-06-14',20),  
( '2022-02-19',19),  
( '2022-03-08',11),  
( '2020-08-30',13),  
( '2021-03-06',7),  
( '2021-02-16',16),  
( '2022-05-03',18),  
( '2022-03-30',6),  
( '2021-03-03',20);
```

f. Tabela Autor

```
create table Autor(  
    id_autor      int identity(1,1)      not null,  
    nume          varchar(55)            not null,  
    prenume       varchar(55)            not null,  
    constraint PK_Autor primary key (id_autor)  
);
```

```
INSERT INTO Autor VALUES  
    ('Armentrout','Jennifer'),  
    ('Morgan','Richard'),  
    ('Eminescu','Mihai'),  
    ('Kinney','Jeff'),  
    ('Twain','Mark'),  
    ('Petrescu','Cezar'),  
    ('Ahern','Cecilia');
```

g. Tabela Carte

```

CREATE TABLE Carte(
  id_carte          int identity(1,1) constraint pk_Carte primary key,
  titlu            varchar(1000)          not null,
  seria            varchar(200),
  numar_volum      float,
  pret             float          not null,
  limba            varchar(200),
  numar_pagini     int,
  ISBN             varchar(21),
  id_editura        int          not null,
  constraint fk_Carte foreign key (id_editura) references
  Editura(id_editura)
);

INSERT INTO Carte VALUES
  ('Like the First Time', 'Origin ',0.5,42.50,'engleza',500, 9780487989169,
  2),
  ('The Darkest Star ', 'Origin ',1,50,'engleza',400, 9780262989169, 2),
  ('The Burning Shadow ', 'Origin ',2,60.90,'engleza',300, 9780556989169,
  2),
  ('The Brightest Night ', 'Origin ',3,60,'engleza',450, 9780253989169, 2),
  ('Daimon', ' Covenant',0.5,30.50,'engleza',100, 9780301989169, 5),
  ('Half-Blood', ' Covenant',1,40.50,'engleza',200, 9780325989169, 5),
  ('Pure', ' Covenant',2,45,'engleza',100, 9780562989169, 5),
  ('Deity', ' Covenant',3,46,'engleza',130, 9780328989169, 5),
  ('Elixir', ' Covenant',3.5,46.90,'engleza',300, 9780373989169, 5),
  ('Apollyon', ' Covenant',4,50,'engleza',160, 9780397989169, 5),
  ('The One & Only', ' Covenant',4.5,52.50,'engleza',100, 9780430989169,
  5),
  ('Sentinel', ' Covenant',5,60,'engleza',100, 9780424989169, 5),
  ('Umbre', 'Lux',0.5,30,'romana',50, 9780328989169, 2),
  ('Obsidian', 'Lux',1,30,'romana',100, 9780424989169, 1),
  ('Onix', 'Lux',2,30,'romana',150, 9780547989169, 1),
  ('Opal', 'Lux',3,30,'romana',150, 9780283989169, 1),
  ('Origin', 'Lux',4,30,'romana',200, 9780571989169, 2),

```

```
( 'Opozitie', 'Lux',5,30,'romana',250, 9780250989169, 2),
( 'Carbon modificat ', null, null,29.30,'romana',500, 9780367989169, 4),
( 'Portalul Ingerilor ', null, null,33.30,'romana',900, 9780553989169, 4),
( 'Poesii ', null, null,19.30,'romana',60, 9780367989169, 3),
( 'Jurnalul unui pusti#1', 'Jurnalul unui pusti', 1,19,'romana',50,
9780367987169, 7),
( 'Jurnalul unui pusti#2', 'Jurnalul unui pusti', 2,29,'romana',50,
9780367979169, 7),
( 'Print si cersetor', null, null,19.30,'romana',60, 9780367984169, 6),
( 'Fram ursul polar', null, null,19.30,'romana',120, 9780367489169, 6),
( 'Jocul trecutului', null, null,55.30,'romana',336, 9780567489169, 6);
```

h. Tabela FisaDeLectura

```
CREATE TABLE FisaDeLectura(
  id_fisa_de_lectura    int identity(1,1)    constraint pk_FisaDeLectura
primary key,
  id_carte              int                  not null,
  id_elev               int                  not null,
  data_imprumutului     date,
  data_limita           date,
  data_restituire       date,
  constraint fk_istoric_carte foreign key(id_carte) references
CARTE(id_carte),
  constraint fk_istoric_elev foreign key(id_elev) references Elev(id_elev),
  constraint check_data check(data_limita>data_imprumutului)
);
```

```
INSERT INTO FisaDeLectura VALUES
(26,19,'2021-04-11','2021-12-26','2021-05-23'),
(18,18,'2021-10-20','2022-05-22','2022-03-25'),
(16,15,'2021-09-15','2022-03-27','2021-10-25'),
(4,13,'2021-03-31','2021-12-12','2021-08-01'),
(9,15,'2021-04-18','2021-08-18','2021-08-18'),
(23,20,'2021-09-13','2021-12-05','2021-12-03'),
(10,7,'2021-01-04','2021-04-18','2021-01-25'),
```

```
(8,3,'2021-07-26','2022-02-10','2021-11-11'),
(1,11,'2021-02-11','2021-03-01','2021-03-01'),
(9,15,'2021-08-09','2021-10-13','2021-09-22'),
(11,20,'2021-10-13','2022-03-11','2021-12-06'),
(9,10,'2021-06-20','2022-03-16','2021-12-04'),
(9,19,'2021-12-30','2022-05-21','2022-02-17'),
(8,10,'2021-09-24','2022-05-12','2021-11-18'),
(2,11,'2022-04-23','2022-05-03','2022-05-01'),
(13,20,'2022-02-13','2022-07-09',NULL),
(5,13,'2021-06-26','2022-04-11','2021-10-02'),
(6,6,'2021-02-06','2021-08-07','2021-06-29'),
(24,20,'2021-04-03','2021-09-09','2021-05-22'),
(14,18,'2021-10-24','2022-06-18',NULL),
(14,3,'2022-05-27','2022-06-17','2022-06-03'),
(13,10,'2021-12-09','2022-06-30',NULL),
(12,10,'2021-09-15','2022-04-18','2022-03-31'),
(14,11,'2021-10-10','2022-04-22','2022-03-26'),
(25,7,'2022-05-17','2022-05-27','2022-05-20'),
(24,18,'2022-01-31','2022-06-15','2022-06-13'),
(7,20,'2021-04-28','2021-08-31','2021-07-02'),
(3,13,'2021-06-13','2021-08-20','2021-06-13'),
(4,16,'2021-12-07','2022-01-03','2021-12-09'),
(23,19,'2021-01-03','2021-11-01','2021-05-28');
```

i. Tabela Domeniu

```
CREATE TABLE Domeniu(
  id_domeniu int identity(1,1)  constraint pk_Domeniu primary key,
  nume        varchar(30)      not null
);
```

```
INSERT INTO Domeniu VALUES
```

```
('Fictiune'),  
( 'Adolescenti'),  
( 'Stiintifico-fantastic'),  
( 'Fantasy'),  
( 'Mister'),  
( 'Politiste'),  
( 'Psihologice'),  
( 'Filozofice'),  
( 'Dezvoltare personala');
```

j. Tabela Angajat

```
CREATE TABLE Angajat(  
  id_angajat    int identity(1,1)  constraint pk_Angajat primary key,  
  nume          varchar(20)        not null,  
  prenume       varchar(20)        not null,  
  adresa        varchar(100),  
  email         varchar(55),  
  telefon       varchar(11),  
  salariu       float,  
  functie       varchar(30),  
  data_angajarii date,  
  id_biblioteca int                not null,  
  constraint fk_Angajat foreign key (id_biblioteca) references  
  Biblioteca(id_biblioteca),  
  constraint check_functie_angajat check (upper(functie) in  
  ('MANAGER','BIBLIOTECAR','PAZNIC','INGRIJITOR','OPERATOR DE  
  INTRODUCERE DATE'))  
);
```

```
INSERT INTO Angajat VALUES
```

```
('Lungu', 'Gabriel', 'Soseaua Mihai Bravu 25', 'lungugabriel@gmail.com',  
'0700235234', 6210, 'INGRIJITOR', '2018-07-07', 1),
```

('Ilie', 'Constantin', 'Strada Agricultori 21', 'ilieconstantin@gmail.com', '0778061180', 8352, 'PAZNIC', '2019-04-24', 2),
('Popescu', 'Constantin', 'Bulevardul Timisoara 8', 'popescuconstantin@gmail.com', '0756057671', 7444, 'BIBLIOTECAR', '2017-10-22', 2),
('Ionescu', 'Adrian', 'Strada Berzei 2', 'ionescuadrian@gmail.com', '0710824566', 9095, 'INGRIJITOR', '2012-07-25', 1),
('Mihail', 'Viorel', 'Strada Barbu Vacarescu 20', 'mihailviorel@gmail.com', '0736357704', 8739, 'OPERATOR DE INTRODUCERE DATE', '2019-04-03', 3),
('Popa', 'Gabriel', 'Calea Victoriei 26', 'popagabriel@gmail.com', '0703144330', 2309, 'INGRIJITOR', '2012-07-12', 4),
('Negoitescu', 'Constantin', 'Calea Victoriei 63', 'negoitescuconstantin@gmail.com', '0734024300', 8960, 'PAZNIC', '2019-06-12', 4),
('Dumitrescu', 'Bogdan', 'Calea Victoriei 63', 'dumitrescubogdan@gmail.com', '0777005444', 9040, 'PAZNIC', '2016-12-24', 4),
('Vlad', 'Marius', 'Strada Dimitrie Bolintineanu 9', 'vladmarius@gmail.com', '0740775375', 5044, 'INGRIJITOR', '2013-01-11', 2),
('Mocanu', 'Horia', 'Strada Lipscani 43', 'mocanuhoria@gmail.com', '0763551170', 2534, 'PAZNIC', '2018-09-04', 2),
('Rusu', 'Mircea', 'Strada Lipscani 43', 'rusumircea@gmail.com', '0711120362', 3346, 'BIBLIOTECAR', '2018-02-22', 4),
('Barbu', 'Gabriel', 'Calea Victoriei 63', 'barbugabriel@gmail.com', '0713644081', 4664, 'OPERATOR DE INTRODUCERE DATE', '2012-08-14', 3),
('Dobre', 'Ovidiu', 'Bulevardul Ion Gheorghe Duca 8', 'dobreovidiu@gmail.com', '0738485151', 7943, 'INGRIJITOR', '2020-07-12', 4),
('Dinescu', 'Manuel', 'Soseaua Stefan cel Mare 56', 'dinescumanuel@gmail.com', '0772552424', 3588, 'BIBLIOTECAR', '2013-11-22', 4),
('Codreanu', 'Dan', 'Calea Victoriei 63', 'codreanudand@gmail.com', '0774216267', 6956, 'INGRIJITOR', '2013-06-05', 2),

('Draghici', 'Bogdan', 'Bulevardul Ion Gheorghe Duca 8',
'draghicibogdan@gmail.com', '0733371865', 2657, 'OPERATOR DE
INTRODUCERE DATE', '2014-05-27', 4),
('Paun', 'Mihai', 'Soseaua Mihai Bravu 25', 'paunmihai@gmail.com',
'0758485170', 8321, 'PAZNIC', '2016-11-25', 3),
('Niculescu', 'Marius', 'Strada Armand Calinescu 17',
'niculescumarius@gmail.com', '0735827817', 7774, 'OPERATOR DE
INTRODUCERE DATE', '2020-09-16', 1),
('Tudor', 'Razvan', 'Calea Aurel Vlaicu 15', 'tudorrazvan@gmail.com',
'0757260662', 4874, 'OPERATOR DE INTRODUCERE DATE', '2021-12-31',
4),
('Marin', 'Horia', 'Str. Calea Vitian 55', 'marinhoria@gmail.com',
'0734646068', 2648, 'PAZNIC', '2016-04-17', 3),
('Lungu', 'Razvan', 'Strada Agricultori 21', 'lungurazvan@gmail.com',
'0774122440', 8532, 'MANAGER', '2013-08-24', 3),
('Ilie', 'Lucian', 'Bulevardul Timisoara 8', 'ionesculucian@gmail.com',
'0782825456', 6253, 'PAZNIC', '2020-03-22', 5),
('Popescu', 'Ovidiu', 'Strada Lipscani 43', 'marinovidui@gmail.com',
'0781603477', 2978, 'BIBLIOTECAR', '2016-01-20', 3),
('Ionescu', 'Adrian', 'Bulevardul Aerogarii 21',
'mocanuadrian@gmail.com', '0724864670', 8427, 'INGRIJITOR', '2018-11-
05', 5),
('Mihail', 'Ovidiu', 'Bulevardul Ion Gheorghe Duca 8',
'ilieovidui@gmail.com', '0740226353', 4009, 'PAZNIC', '2020-06-22', 1),
('Popa', 'Lucian', 'Strada Grigore Alexandrescu 7',
'dobrelucian@gmail.com', '0701125810', 6149, 'MANAGER', '2017-08-17',
5),
('Negoitescu', 'Dan', 'Strada Barbu Vacarescu 20',
'dinescudan@gmail.com', '0708513280', 2569, 'MANAGER', '2018-06-23',
1),
('Dumitrescu', 'Bogdan', 'Strada Cornul Luncii 2',
'dinescubogdan@gmail.com', '0774557068', 2777, 'OPERATOR DE
INTRODUCERE DATE', '2012-06-09', 5),
('Vlad', 'Mircea', 'Strada Armand Calinescu 17', 'tudormircea@gmail.com',
'0774648405', 6185, 'MANAGER', '2018-02-16', 6),

('Mocanu', 'Manuel', 'Soseaua Stefan cel Mare 56',
'marinmanuel@gmail.com', '0727655544', 7083, 'BIBLIOTECAR', '2018-03-20', 5),
('Rusu', 'Lucian', 'Strada Cornul Luncii 2', 'popalucian@gmail.com',
'0728184602', 5771, 'INGRIJITOR', '2020-04-08', 2),
('Barbu', 'Gabriel', 'Strada Armand Calinescu 17',
'paungabriel@gmail.com', '0705142272', 5102, 'OPERATOR DE
INTRODUCERE DATE', '2021-04-15', 3),
('Dobre', 'Viorel', 'Strada Cornul Luncii 2', 'codreanuviorel@gmail.com',
'0787822872', 9704, 'BIBLIOTECAR', '2013-09-28', 5),
('Dinescu', 'Manuel', 'Soseaua Mihai Bravu 25', 'iliemanuel@gmail.com',
'0731762787', 8144, 'BIBLIOTECAR', '2015-09-07', 3),
('Codreanu', 'Ionel', 'Soseaua Oltenitei 12', 'barbuionel@gmail.com',
'0752338436', 8029, 'BIBLIOTECAR', '2019-05-03', 2),
('Draghici', 'Mihai', 'Strada Agricultori 21', 'mocanumihai@gmail.com',
'0757455860', 4972, 'INGRIJITOR', '2021-05-07', 1),
('Paun', 'Mircea', 'Calea Mosilor 13', 'rusumircea@gmail.com',
'0741021121', 6275, 'PAZNIC', '2014-11-04', 6),
('Niculescu', 'Mihai', 'Strada Dristorului 10', 'popescumihai@gmail.com',
'0764022426', 3828, 'INGRIJITOR', '2021-09-23', 5),
('Tudor', 'Bogdan', 'Calea Mosilor 13', 'marinbogdan@gmail.com',
'0760846430', 7091, 'MANAGER', '2019-04-11', 4),
('Marin', 'Gheorghe', 'Strada Barbu Vacarescu 20',
'popagheorghe@gmail.com', '0735157823', 2465, 'PAZNIC', '2019-08-25',
4)

k. Tabela Imprumut

```
CREATE TABLE Imprumut(  
    id_imprumut    int identity(1,1)  constraint pk_Imprumut primary key,  
    id_elev        int,  
    id_carte       int,  
    id_biblioteca  int,  
    constraint fk_imprumut_elev foreign key(id_elev) references  
Elev(id_elev),  
    constraint fk_imprumut_carte foreign key(id_carte) references  
Carte(id_carte),  
    constraint fk_imprumut_biblioteca foreign key(id_biblioteca) references  
Biblioteca(id_biblioteca)  
);
```

```
INSERT INTO Imprumut VALUES
```

```
(19,26,1),  
(18,18,6),  
(15,16,3),  
(13,4,2),  
(15,9,6),  
(20,23,5),  
(7,10,1),  
(3,8,5),  
(11,1,5),  
(15,9,3),  
(20,11,2),  
(10,9,5),  
(19,9,2),  
(10,8,5),  
(11,2,6),  
(20,13,4),  
(13,5,1),  
(6,6,5),  
(20,24,2),
```

(18,14,6),
(3,14,1),
(10,13,3),
(10,12,4),
(11,14,6),
(7,25,1),
(18,24,2),
(20,7,5),
(13,3,4),
(16,4,4),
(19,23,3);

1. Tabela Scrisă

```
create table Scrisa(  
    id_carte      int    not null references Carte(id_carte),  
    id_autor      int    not null references Autor(id_autor),  
    data_publicare date,  
    constraint PK_Scrisa primary key (id_carte, id_autor)  
);
```

INSERT INTO Scrisa VALUES

(1,1,'2018-02-19'),
(2,1,'2019-01-21'),
(3,1,'2019-02-11'),
(4,1,'2020-03-22'),
(5,1,'2011-07-29'),
(6,1,'2011-07-25'),
(7,1,'2012-08-23'),
(8,1,'2012-10-24'),
(9,1,'2012-11-01'),
(10,1,'2013-06-01'),
(11,1,'2013-06-01'),
(12,1,'2012-07-11'),
(13,1,'2011-08-14'),
(15,1,'2012-09-17'),

```
(16,1,'2012-07-14'),  
(17,1,'2013-06-20'),  
(18,1,'2014-05-22'),  
(19,2,'2002-03-11'),  
(20,2,'2003-05-21'),  
(21,3,'1884-02-12'),  
(22,4,'2007-04-01'),  
(23,4,'2007-09-19'),  
(24,5,'1881-06-01'),  
(25,6,'1931-05-09'),  
(26,7,'2022-03-25');
```

m.Tabela Apartine

```
CREATE TABLE Apartine(  
    id_apartinere int identity(1,1) constraint pk_Apartinere primary key,  
    id_carte      int,  
    id_domeniu    int,  
    constraint fk_apartine_domeniu foreign key (id_domeniu) references  
DOMENIU(id_domeniu),  
    constraint fk_apartine_carte foreign key (id_carte) references  
CARTE(id_carte)  
);
```

```
INSERT INTO Apartine VALUES
```

```
(1,3),  
(2,3),  
(3,3),  
(4,3),  
(5,1),  
(6,1),  
(7,1),  
(8,1),  
(9,1),  
(10,1),  
(11,1),
```

(12,1),
(13,4),
(15,4),
(16,4),
(17,4),
(18,4),
(19,5),
(20,5),
(19,3),
(20,3),
(19,1),
(20,1),
(21,2),
(22,2),
(23,2),
(24,2),
(25,2),
(26,2),
(1,1),
(2,1),
(3,1),
(4,1),
(22,1),
(23,1);

11. Cereri complexe SQL

a. Prima cerere

Să se afișeze angajații bibliotecilor (nume, prenume, telefon, funcție și salariu inițial) care se află în orașul Cluj-Napoca, precum și salariul acestora, știind că li se acordă o majorare de salariu de 25%.

În cadrul acestei cereri am utilizat următoarele elemente:

- *NVL(Oracle) => ISNULL (SQL server)*
- *filtrare la nivel de linii*
- *subcereri nesincronizate*

*Prima oară am selectat id-ul orașului cu numele **Cluj-Napoca**, apoi am selectat bibliotecile al căror cod de oraș este egal cu id-ul găsit anterior, iar apoi am selectat datele referitoare la angajații care lucrează în acele biblioteci școlare.*

Comanda în Oracle

```
SELECT nume, prenume, telefon, functie, salariu as "Salariu",  
NVL(salariu+0.25*salariu,0) "Salariu Majorat"  
FROM Angajat  
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca  
WHERE id_oras=(SELECT id_oras FROM Oras WHERE  
upper(nume)='Cluj-Napoca'));
```

Comanda în SSMS

```
SELECT nume, prenume, telefon, functie, salariu as "Salariu",  
ISNULL(salariu+0.25*salariu,0) "Salariu Majorat"  
FROM Angajat  
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca  
WHERE id_oras=(SELECT id_oras FROM Oras WHERE upper(nume)='Cluj-  
Napoca'));
```

Lung_Alexandra_e...re (Proiect (66)) X create+inserare d...re (Proiect (67))

--Exercitiul 11.

--1. Sa se afiseze angajatii bibliotecilor(ume, prenume,telefon,functie si salariu initial) care se afla in orasul Cluj-Napoca, precum s
--salariul acestora, stiind ca li se acorda o majorare de salariu de 25%.

--In cadrul acestei cereri am utilizat urmatoarele elemente:

- NVL(Oracle) => ISNULL (SQL server)
- filtrare la nivel de linii
- subcereri nesincronizate

SELECT nume, prenume,telefon, functie, salariu as "Salariu", ISNULL(salariu+0.25*salariu,0) "Salariu Majorat"
FROM Angajat
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca WHERE id_oras=
(SELECT id_oras FROM Oras WHERE upper(nume)='Cluj-Napoca'));

150 %

Results Messages

	nume	prenume	telefon	functie	Salariu	Salariu Majorat
1	Ilie	Lucian	0782825456	PAZNIC	6253	7816,25
2	Ionescu	Adrian	0724864670	INGRIUITOR	8427	10533,75
3	Popa	Lucian	0701125810	MANAGER	6149	7686,25
4	Dumitrescu	Bogdan	0774557068	OPERATOR DE INTRODUCERE DATE	2777	3471,25
5	Mocanu	Manuel	0727655544	BIBLIOTECAR	7083	8853,75
6	Dobre	Viorel	0787822872	BIBLIOTECAR	9704	12130
7	Niculescu	Mihai	0764022426	INGRIUITOR	3828	4785

Query executed successfully. gestiune-biblioteci-scolare... Proiect (66) gestiune_biblioteci_s... 00:00:00 7 rows

b. A doua cerere

Să se afișeze elevii care au împrumutat cărți din domeniile Fantasy, Fictiune sau cărți scrise de Mark Twain și care au împrumutat cărți în luna a patra sau în ziua a șasea a oricărei și le-au adus la timp, ordonați alfabetic după numele de familie.

În cadrul acestei cereri am folosit join pe 4 tabele și am utilizat următoarele elemente:

- *filtrare la nivel de linii*
- *subcereri nesincronizate pe trei tabele*
- *ordonare*
- *date calendaristice*

Prima oară am selectat cărțile care aparțin celor două domenii sau cele scrise de Mark Twain, ulterior am selectat din elevii care au împrumutat aceste cărți în luna a patra sau a șasea, făcând joinuri pentru a le afla numele și prenumele.

```
SELECT distinct(e.id_elev),e.num ,e.prenume,ca.titlu
FROM Elev e, FisaDeLectura f, Domeniu d, Carte ca
WHERE e.id_elev=f.id_elev and f.data_limita>=f.data_restituire
and
f.id_carte=ca.id_carte and (MONTH(f.data_imprumutului) = 4 or
DAY(f.data_imprumutului) = 6)
and ca.id_carte in (SELECT id_carte FROM Apartine WHERE
id_domeniu in
(SELECT id_domeniu FROM Domeniu WHERE nume='Fantasy' or
nume='Fictiune')
or ca.id_carte in (SELECT id_carte FROM Scrisa WHERE
id_autor=(SELECT id_autor
FROM Autor WHERE nume='Twain'))))
ORDER BY e.num;
```


lung_alexandra_ex...re (Proiect (51))

```
--2. Sa se afiseze elevii care au imprumutat carti din domeniile Fantasy,Fictiune sau carti scrise de Mark Twain
--si care au imprumutat carti in luna a 4 sau in ziua a 6 din orice luna si le-au adus la timp, ordonati alfabetic dupa numele de familie

--In cadrul acestei cereri am utilizat urmatoarele elemente:
--am folosit join pe 4 tabele
-- filtrare la nivel de linii
-- subcereri nesincronizate pe 3 tabele
-- ordonari
-- pe date calendaristice.

SELECT distinct(e.id_elev),e.num ,e.prenume,ca.titlu
FROM Elev e, FisaDelectura f, Domeniu d, Carte ca
WHERE e.id_elev=f.id_elev and f.data_limita>=f.data_restituire and
f.id_carte=ca.id_carte and (MONTH(f.data_imprumutului) = 4 or DAY(f.data_imprumutului) = 6)
and ca.id_carte in (SELECT id_carte FROM Apartine WHERE id_domeniu in
(SELECT id_domeniu FROM Domeniu WHERE nume='Fantasy' or nume='Fictiune')
or ca.id_carte in (SELECT id_carte FROM Scrisa WHERE id_autor=(SELECT id_autor
FROM Autor WHERE nume='Twain'))))
ORDER BY e.num;
```

150 %

Results Messages

	id_elev	nume	prenume	titlu
1	15	Codreanu	Horia	Elvir
2	20	Marin	Manuel	Print si cersetor
3	20	Marin	Manuel	Pure
4	6	Popa	Alin	Half-Blood
5	11	Rusu	Mircea	The Darkest Star

Query executed successfully.

gestiune-biblioteci-scolare... Proiect (51) gestiune_biblioteci_s... 00:00:00 5 rows

c. A treia cerere

Să se afișeze numele, data la care s-au angajat, funcția angajaților, biblioteca unde lucrează și salariul acestora știind că se majorează salariile astfel:

- *pentru manageri cu 15%,*
- *pentru bibliotecari cu 20%*
- *pentru operator de introducere date cu 10%*
- *pentru paznici cu 5%*
- *pentru îngrijitori cu 2%.*

Afișați doar angajații care s-au angajat după anul 2018 și luna august, în ordinea lexicografică a funcțiilor.

În cadrul acestei cereri am utilizat următoarele elemente:

- *DECODE(oracle) => CASE(ssms)*
- *ordonare*

Am selectat angajații a căror dată repectă condiția impusă, făcând join cu tabela Biblioteca pentru a afișa numele bibliotecii. De asemenea, am folosit decode(case) pentru a calcula noul salariu.

Comanda în Oracle

```
SELECT concat(a.num, ' ', a.prenume) as NumeAngajat , a.salariu,  
a.data_angajarii, a.functie, b.num "Biblioteca",  
decode(upper(a.functie),  
'MANAGER', NVL(salariu+0.15*salariu,0),  
'BIBLIOTECAR', NVL (salariu+0.2*salariu,0) ,  
'OPERATOR DE INTRODUCERE DATE', NVL (salariu+0.1*salariu,0),  
'PAZNIC', NVL (salariu+0.05*salariu,0),  
'INGRIJITOR', NVL (salariu+0.02*salariu,0)) "Salariu majorat"  
FROM Angajat a, Biblioteca b  
WHERE b.id_biblioteca=a.id_biblioteca and  
a.data_angajarii>=to_date('01/08/2018', 'dd/mm/yyyy')  
ORDER BY a.functie;
```

Comanda în SSMS

```

SELECT concat(a.ume, ' ', a.prenume) as
NumeAngajat, a.salariu, a.data_angajarii, a.functie, b.ume "Biblioteca",
CASE upper(a.functie) WHEN 'MANAGER' THEN ISNULL(salariu+0.15*salariu,0)
                      WHEN 'BIBLIOTECAR' THEN
ISNULL(salariu+0.2*salariu,0)
                      WHEN 'OPERATOR DE INTRODUCERE DATE' THEN
ISNULL(salariu+0.1*salariu,0)
                      WHEN 'PAZNIC' THEN ISNULL(salariu+0.05*salariu,0)
                      WHEN 'INGRIJITOR' THEN
ISNULL(salariu+0.02*salariu,0)
END "Salariu majorat"
FROM Angajat a, Biblioteca b
WHERE b.id_biblioteca=a.id_biblioteca and a.data_angajarii>='01-08-2018'
ORDER BY a.functie;

```

lung_alexandra_ex...re (Proiect 51) ✕

```
--In cadrul acestei cereri am utilizat urmatoarele elemente:
-- decode(oracle) => CASE(ssms)
-- ordonare
-- functii pe date calendaristice
-- functii pe siruri de caractere.
```

```

SELECT concat(a.ume, ' ', a.prenume) as NumeAngajat, a.salariu, a.data_angajarii, a.functie, b.ume "Biblioteca",
CASE upper(a.functie) WHEN 'MANAGER' THEN ISNULL(salariu+0.15*salariu,0)
                      WHEN 'BIBLIOTECAR' THEN ISNULL(salariu+0.2*salariu,0)
                      WHEN 'OPERATOR DE INTRODUCERE DATE' THEN ISNULL(salariu+0.1*salariu,0)
                      WHEN 'PAZNIC' THEN ISNULL(salariu+0.05*salariu,0)
                      WHEN 'INGRIJITOR' THEN ISNULL(salariu+0.02*salariu,0)
END "Salariu majorat"
FROM Angajat a, Biblioteca b
WHERE b.id_biblioteca=a.id_biblioteca and a.data_angajarii>='01-08-2018'
ORDER BY a.functie;

```

150 %

Results Messages

	NumeAngajat	salariu	data_angajarii	functie	Biblioteca	Salariu majorat
1	Rusu Mircea	3345	2018-02-22	BIBLIOTECAR	Biblioteca Colegiului National „Spiru Haret” din B...	4015,2
2	Codreanu Ionel	8029	2019-05-03	BIBLIOTECAR	Biblioteca Liceului „Grigore Moisil” Timisoara	9634,8
3	Mocanu Manuel	7083	2018-03-20	BIBLIOTECAR	Biblioteca Liceului Teoretic „Avram Iancu” Cluj...	8499,6
4	Rusu Lucian	5771	2020-04-08	INGRIJITOR	Biblioteca Liceului „Grigore Moisil” Timisoara	5886,42
5	Draghici Mihai	4972	2021-05-07	INGRIJITOR	Biblioteca Liceului Teoretic „Al. I. Cuza”	5071,44
6	Niculescu Mihai	3828	2021-09-23	INGRIJITOR	Biblioteca Liceului Teoretic „Avram Iancu” Cluj...	3904,56
7	Ionescu Adrian	8427	2018-11-05	INGRIJITOR	Biblioteca Liceului Teoretic „Avram Iancu” Cluj...	8595,54
8	Dobre Ovidiu	7943	2020-07-12	INGRIJITOR	Biblioteca Colegiului National „Spiru Haret” din B...	8101,86
9	Lungu Gabriel	6210	2018-07-07	INGRIJITOR	Biblioteca Liceului Teoretic „Al. I. Cuza”	6334,2
10	Negotescu Dan	2569	2018-06-23	MANAGER	Biblioteca Liceului Teoretic „Al. I. Cuza”	2954,35
11	Vlad Mircea	6185	2018-02-16	MANAGER	Biblioteca Colegiului National „Ion Luca Caragial...	7112,75
12	Tudor Bogdan	7091	2019-04-11	MANAGER	Biblioteca Colegiului National „Spiru Haret” din B...	8154,65
13	Mihail Viorel	8739	2019-04-03	OPERATOR DE INTRODUCERE DATE	Biblioteca Liceului Teoretic „Ion Barbu”	9612,9
14	Niculescu Marius	7774	2020-09-16	OPERATOR DE INTRODUCERE DATE	Biblioteca Liceului Teoretic „Al. I. Cuza”	8551,4
15	Tudor Razvan	4874	2021-12-31	OPERATOR DE INTRODUCERE DATE	Biblioteca Colegiului National „Spiru Haret” din B...	5361,4
16	Barbu Gabriel	5102	2021-04-15	OPERATOR DE INTRODUCERE DATE	Biblioteca Liceului Teoretic „Ion Barbu”	5612,2
17	Ilie Lucian	6253	2020-03-22	PAZNIC	Biblioteca Liceului Teoretic „Avram Iancu” Cluj...	6565,65
18	Niculescu Constantin	8009	2018-06-12	PAZNIC	Biblioteca Colegiului National „Spiru Haret” din B...	8101,86

Query executed successfully.

gestiune-bibliotec-scolare... | Proiect 51 | gestiune_bibliotec_s... | 00:00:00 | 22 rows

d. A patra cerere

Să se afișeze numele și prețul cărților care sunt încadrate în cel puțin 2 domenii. Incadrarea cărților în categorii după numărul de pagini se face astfel:

- dacă au sub 250 de pagini, sunt considerate lectura ușoară*
- dacă au între 250 și 500 de pagini, sunt considerate lectura cu dificultate medie*
- dacă au peste 500 de pagini sunt considerate lecturi grele.*

În cadrul acestei cereri am utilizat următoarele elemente:

- grupări de date*
- funcții grup*
- filtrare la nivel de grupuri*
- funcția CASE*

Am făcut o subcerere în clauza FROM în care am selectat pentru fiecare carte care are numărul de domenii mai mare ca 1, numărul de domenii și id-ul cărții, ulterior am făcut JOIN cu tabela Carte pentru a afișa informațiile cerute

```
SELECT c.titlu, c.pret, d.nr "Nr Domenii",  
CASE  
    WHEN c.numar_pagini<250 THEN 'dificultate usoara'  
    WHEN c.numar_pagini>=250 and c.numar_pagini<500 THEN  
'dificultate medie'  
    WHEN c.numar_pagini>=500 THEN 'dificultate grea'  
END as "Dificultate"  
FROM Carte c, (SELECT count(*) nr, id_carte idc  
FROM Apartine  
GROUP BY id_carte  
HAVING count(*)>1) d  
WHERE c.id_carte=d.idc;
```

Lung_Alexandra_e...re (Proiect (66)) * X creare+inserare d...re (Proiect (67))

--4. Sa se afiseze numele si pretul cartilor care sunt incadrate in cel putin 2 domenii.
--Incadrarea cartilor in categorii dupa numarul de pagini se realizeaza astfel:
-- daca au sub 250 de pagini, sunt considerate lectura usoara,
-- daca au intre 250 si 500 de pagini, sunt considerate lectura cu dificultate medie,
-- daca au peste 500 de pagini sunt considerate lecturi grele.

--In cadrul acestei cereri am utilizat urmatoarele elemente:
-- grupari de date
-- functii grup
-- filtrare la nivel de grupuri
-- functia CASE

```
SELECT c.titlu, c.pret, d.nr "Nr Domenii",  
CASE  
    WHEN c.numar_pagini<250 THEN 'dificultate usoara'  
    WHEN c.numar_pagini>=250 and c.numar_pagini<500 THEN 'dificultate medie'  
    WHEN c.numar_pagini>=500 THEN 'dificultate grea'  
END as "Dificultate"  
FROM Carte c, (SELECT count(*) nr, id_carte idc  
FROM Apartine  
GROUP BY id_carte  
HAVING count(*)>1) d  
WHERE c.id_carte=d.idc;
```

150 %

Results Messages

	titlu	pret	Nr Domenii	Dificultate
1	Like the First Time	42,5	2	dificultate grea
2	The Darkest Star	50	2	dificultate medie
3	The Burning Shadow	60,9	2	dificultate medie
4	The Brightest Night	60	2	dificultate medie
5	Carbon modificat	29,3	3	dificultate grea
6	Portalul Ingerilor	33,3	3	dificultate grea
7	Jumalul unui pusti#1	19	2	dificultate usoara
8	Jumalul unui pusti#2	29	2	dificultate usoara

Query executed successfully.

gestiune-biblioteci-scolare... Proiect (66) gestiune_biblioteci_s... 00:00:00 8 rows

e. A cincea cerere

Să se afișeze pentru fiecare angajat care are salariul mai mic decât media salariului tuturor angajaților, numele, prenumele, salariu, funcția, salariul mediu per funcție, numărul de angajați per funcție și biblioteca școlară unde sunt angajați.

În cadrul acestei cereri am utilizat următoarele elemente:

- *subcereri sincronizate*
- *filtrare la nivel de linii*
- *clauza WITH*

Am selectat clauza WITH ca fiind salariul mediu pentru toți angajații, ulterior folosind subcereri sincronizate am aflat pentru fiecare funcție salariul mediu și numărul de angajați, iar în ultima subcerere sincronizată am selectat numele bibliotecii la care lucrează fiecare angajat.

```
WITH Salariu_NumarAngajati_Per_Functie as(  
SELECT avg(salariu) as medie  
FROM angajat)  
SELECT nume,prenume,salariu,a.functie,  
(SELECT round(avg(salariu),3)  
FROM angajat  
WHERE functie=a.functie) "SALARIU MEDIU/FUNCTIE",  
(SELECT count(*)  
FROM angajat  
WHERE functie=a.functie) "NR ANGAJATI/FUNCTIE",  
(SELECT nume  
FROM biblioteca  
WHERE id_biblioteca=a.id_biblioteca) "BIBLIOTECA"  
FROM angajat a, Salariu_NumarAngajati_Per_Functie s  
WHERE a.salariu<s.medie  
ORDER BY a.functie
```

Lung_Alexandra_e...re (Proiect (66)) × creare+inserare d...re (Proiect (67))

```

WITH Salariu_NumarAngajati_Per_Functie as(
SELECT avg(salariu) as medie
FROM angajat)
SELECT nume,prenume,salariu,a.functie,
(SELECT round(avg(salariu),3)
FROM angajat
WHERE functie=a.functie) "SALARIU MEDIU/FUNCTIE",
(SELECT count(*)
FROM angajat
WHERE functie=a.functie) "NR ANGAJATI/FUNCTIE",
(SELECT nume
FROM biblioteca
WHERE id_biblioteca=a.id_biblioteca) "BIBLIOTECA"
FROM angajat a, Salariu_NumarAngajati_Per_Functie s
WHERE a.salariu<s.medie
ORDER BY a.functie

```

150 %

Results Messages

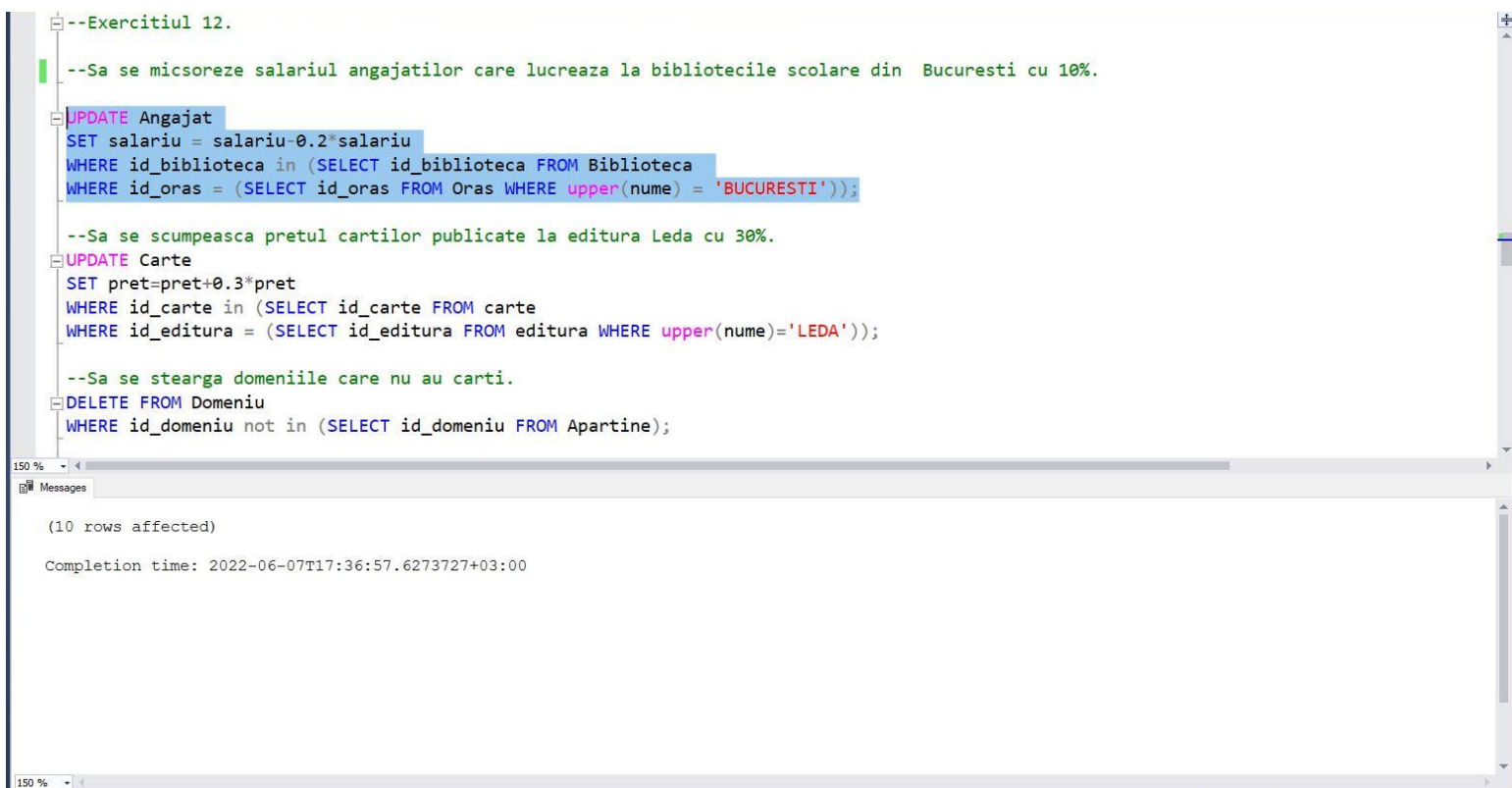
	nume	prenume	salariu	functie	SALARIU MEDIU/FUNCTIE	NR ANGAJATI/FUNCTIE	BIBLIOTECA
1	Rusu	Mircea	3346	BIBLIOTECAR	6289,5	8	Biblioteca Colegiului National „Spiru Haret” din B...
2	Dinescu	Manuel	3588	BIBLIOTECAR	6289,5	8	Biblioteca Colegiului National „Spiru Haret” din B...
3	Popescu	Ovidiu	2978	BIBLIOTECAR	6289,5	8	Biblioteca Liceului Teoretic „Ion Barbu”
4	Rusu	Lucian	5771	INGRIJITOR	6055,5	10	Biblioteca Liceului „Grigore Moisil” Timisoara
5	Draghici	Mihai	4972	INGRIJITOR	6055,5	10	Biblioteca Liceului Teoretic „Al. I. Cuza”
6	Niculescu	Mihai	3828	INGRIJITOR	6055,5	10	Biblioteca Liceului Teoretic „Avram Iancu” Cluj-...
7	Popa	Gabriel	2309	INGRIJITOR	6055,5	10	Biblioteca Colegiului National „Spiru Haret” din B...
8	Viad	Marius	5044	INGRIJITOR	6055,5	10	Biblioteca Liceului „Grigore Moisil” Timisoara
9	Negotie...	Dan	2569	MANAGER	6105,2	5	Biblioteca Liceului Teoretic „Al. I. Cuza”
10	Dumitre...	Bogdan	2777	OPERATOR...	5226,714	7	Biblioteca Liceului Teoretic „Avram Iancu” Cluj-...
11	Barbu	Gabriel	5102	OPERATOR...	5226,714	7	Biblioteca Liceului Teoretic „Ion Barbu”
12	Barbu	Gabriel	4664	OPERATOR...	5226,714	7	Biblioteca Liceului Teoretic „Ion Barbu”
13	Draghici	Bogdan	2657	OPERATOR...	5226,714	7	Biblioteca Colegiului National „Spiru Haret” din B...
14	Tudor	Razvan	4874	OPERATOR...	5226,714	7	Biblioteca Colegiului National „Spiru Haret” din B...
15	Marin	Horia	2648	PAZNIC	5885,7	10	Biblioteca Liceului Teoretic „Ion Barbu”
16	Mocanu	Horia	2534	PAZNIC	5885,7	10	Biblioteca Liceului „Grigore Moisil” Timisoara
17	Marin	Gheor...	2465	PAZNIC	5885,7	10	Biblioteca Colegiului National „Spiru Haret” din B...
18	Mihail	Ovidiu	4009	PAZNIC	5885,7	10	Biblioteca Liceului Teoretic „Al. I. Cuza”

Query executed successfully. gestiune-biblioteci-scolare... Proiect (66) gestiune_biblioteci_s... 00:00:00 18 rows

12. Trei cereri de actualizare și suprimare a datelor

1. Să se micșoreze salariul angajatilor care lucrează la bibliotecile școlare din București cu 10%.

```
UPDATE Angajat
SET salariu = salariu-0.2*salariu
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca
    WHERE id_oras = (SELECT id_oras FROM Oras WHERE upper(ume) =
        'BUCURESTI'));
```



```
--Exercitiul 12.

--Sa se micșoreze salariul angajatilor care lucreaza la bibliotecile școlare din  Bucuresti cu 10%.

UPDATE Angajat
SET salariu = salariu-0.2*salariu
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca
    WHERE id_oras = (SELECT id_oras FROM Oras WHERE upper(ume) = 'BUCURESTI'));

--Sa se scumpeasca pretul cartilor publicate la editura Leda cu 30%.

UPDATE Carte
SET pret=pret+0.3*pret
WHERE id_carte in (SELECT id_carte FROM carte
    WHERE id_editura = (SELECT id_editura FROM editura WHERE upper(ume)='LEDA'));

--Sa se stearga domeniile care nu au carti.

DELETE FROM Domeniu
WHERE id_domeniu not in (SELECT id_domeniu FROM Apartine);

(10 rows affected)

Completion time: 2022-06-07T17:36:57.6273727+03:00
```

2.Să se scumpească prețul cărților publicate la editura Leda cu 30%.

```
UPDATE Carte
SET pret=pret+0.3*pret
WHERE id_carte in (SELECT id_carte FROM carte
    WHERE id_editura = (SELECT id_editura FROM editura WHERE
        upper(ume)='LEDA'));
```



```
--Exercitiul 12.

--Sa se micsoreze salariul angajatilor care lucreaza la bibliotecile școlare din Bucuresti cu 10%.

UPDATE Angajat
SET salariu = salariu-0.2*salariu
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca
WHERE id_oras = (SELECT id_oras FROM Oras WHERE upper(ume) = 'BUCURESTI'));

--Sa se scumpeasca pretul cartilor publicate la editura Leda cu 30%.
UPDATE Carte
SET pret=pret+0.3*pret
WHERE id_carte in (SELECT id_carte FROM carte
WHERE id_editura = (SELECT id_editura FROM editura WHERE upper(ume)='LEDA'));

--Sa se stearga domeniile care nu au carti.
DELETE FROM Domeniu
WHERE id_domeniu not in (SELECT id_domeniu FROM Apartine);

(3 rows affected)
Completion time: 2022-06-07T17:39:07.1033210+03:00
```

3. Să se șteargă domeniile care nu au cărți.

DELETE FROM Domeniu

WHERE id_domeniu not in (SELECT id_domeniu FROM Apartine);

```
UPDATE Angajat
SET salariu = salariu-0.2*salariu
WHERE id_biblioteca in (SELECT id_biblioteca FROM Biblioteca
WHERE id_oras = (SELECT id_oras FROM Oras WHERE upper(ume) = 'BUCURESTI'));

--Sa se scumpeasca pretul cartilor publicate la editura Leda cu 30%.
UPDATE Carte
SET pret=pret+0.3*pret
WHERE id_carte in (SELECT id_carte FROM carte
WHERE id_editura = (SELECT id_editura FROM editura WHERE upper(ume)='LEDA'));

--Sa se stearga domeniile care nu au carti.
DELETE FROM Domeniu
WHERE id_domeniu not in (SELECT id_domeniu FROM Apartine);

--Exercitiul 16.

--OUTER JOIN

(4 rows affected)
Completion time: 2022-06-07T17:39:27.8246753+03:00
```

13. Crearea unei secvențe SQL folosite pentru inserarea înregistrărilor în tabele

```
CREATE SEQUENCE SECV_ORAS
```

```
As int
```

```
INCREMENT by 1
```

```
START WITH 1
```

```
MAXVALUE 500
```

```
NO CYCLE;
```

```
SET IDENTITY_INSERT Oras ON
```

```
INSERT INTO Oras(id_oras,nume) VALUES
```

```
    (NEXT VALUE FOR SECV_ORAS, 'Bucuresti'),
```

```
    (NEXT VALUE FOR SECV_ORAS, 'Iasi'),
```

```
    (NEXT VALUE FOR SECV_ORAS, 'Pitesti'),
```

```
    (NEXT VALUE FOR SECV_ORAS, 'Timisoara'),
```

```
    (NEXT VALUE FOR SECV_ORAS, 'Cluj-Napoca'),
```

```
    (NEXT VALUE FOR SECV_ORAS, 'Ploiesti');
```

```
SET IDENTITY_INSERT Oras OFF
```

14. Crearea unei vizualizări compuse

O vedere/O vizualizare este un tabel logic bazat pe un alt tabel sau pe o altă vedere și nu conține date proprii. Tabelele pe baza cărora sunt create vederile se numesc tabele de bază. La nivelul bazei de date o vedere este stocată sub forma unei instrucțiuni SELECT în dicționarul de date.

Vederile se clasifică în două grupe: simple și complexe. Diferența de bază între cele două grupe este legată de operațiile LMD (inserare, actualizare și ștergere).

O vedere simplă este o vedere care:

- furnizează date dintr-un singur tabel;
- nu conține funcții sau grupuri de date;
- permite execuția unor operații LMD.

O vedere complexă este o vedere care :

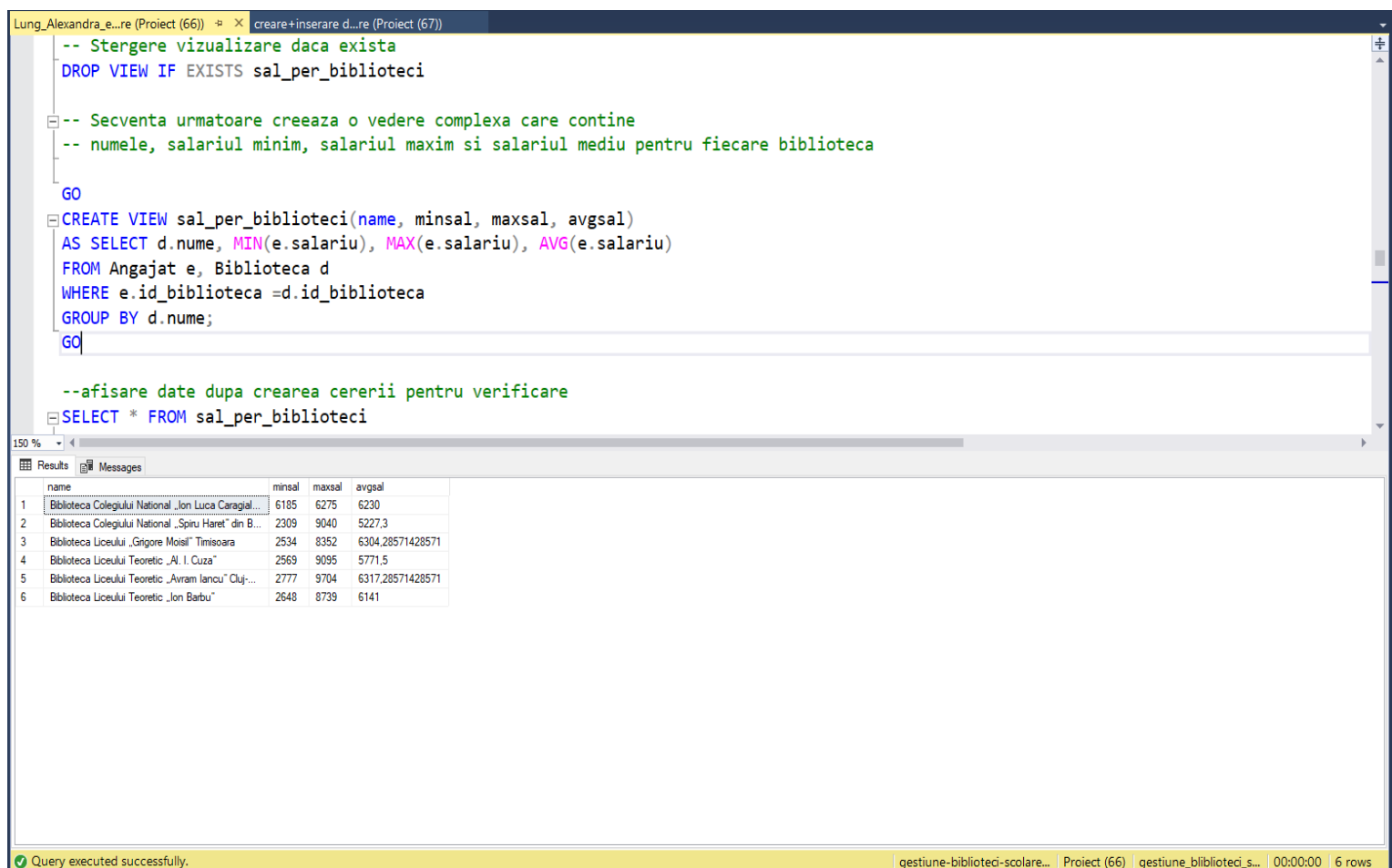
- extrage date din mai multe tabele;
- conține funcții sau grupuri de date;
- nu permite întotdeauna operații LMD.

d. Crearea vizualizării

```
GO
CREATE VIEW sal_per_biblioteci(name, minsal, maxsal,
avgsal)
AS SELECT d.num, MIN(e.salariu), MAX(e.salariu),
AVG(e.salariu)
FROM Angajat e, Biblioteca d
WHERE e.id_biblioteca =d.id_biblioteca
GROUP BY d.num;
GO
```

Secvența anterioară creează o vedere complexă care conține numele, salariul minim, salariul maxim și salariul mediu pentru fiecare bibliotecă. De notat că au fost specificate alias-uri pentru vedere. Acest lucru este necesar dacă una din coloanele vederii rezultă din evaluarea unei funcții sau expresii.

Pentru a putea executa operații LMD asupra datelor din baza de date prin intermediul vederii trebuie avute în vedere o serie de reguli.



```
-- Stergere vizualizare daca exista
DROP VIEW IF EXISTS sal_per_biblioteci

-- Secventa urmatoare creeaza o vedere complexa care contine
-- numele, salariul minim, salariul maxim si salariul mediu pentru fiecare biblioteca
GO

CREATE VIEW sal_per_biblioteci(name, minsal, maxsal, avgsal)
AS SELECT d.num, MIN(e.salariu), MAX(e.salariu), AVG(e.salariu)
FROM Angajat e, Biblioteca d
WHERE e.id_biblioteca =d.id_biblioteca
GROUP BY d.num;
GO

--afisare date dupa crearea cererii pentru verificare
SELECT * FROM sal_per_biblioteci
```

	name	minsal	maxsal	avgsal
1	Biblioteca Colegiului National „Ion Luca Caragial..."	6185	6275	6230
2	Biblioteca Colegiului National „Spiru Haret" din B...	2309	9040	5227,3
3	Biblioteca Liceului „Grigore Moisil" Timisoara	2534	8352	6304,28571428571
4	Biblioteca Liceului Teoretic „Al. I. Cuza"	2569	9095	5771,5
5	Biblioteca Liceului Teoretic „Avram Iancu" Cluj-...	2777	9704	6317,28571428571
6	Biblioteca Liceului Teoretic „Ion Barbu"	2648	8739	6141

Query executed successfully.

1. Nu se poate șterge o linie aparținând unei vederi dacă definiția vederii conține :

- funcții grup;
- clauză GROUP BY;
- cuvântul cheie DISTINCT.

2. Nu se pot modifica datele dintr-o vedere dacă vederea conține:
 - oricare din elementele de la punctul 1;
 - coloane definite prin expresii (de exemplu, SALARY*12);
 - pseudocoloana ROWNUM.
3. Nu pot fi inserate date într-o vedere dacă:
 - vederea conține oricare din elementele de la punctul 1 și 2;
 - există coloane NOT NULL fără valoare implicită în tabelul de bază și care nu au fost selectate în vedere.

e. Operații LMD permise

```
SELECT *FROM sal_per_biblioteci
WHERE avgsal<=6000
```

Secvența anterioară selectează din vederea complexă bibliotecile care au media salariului angajaților mai mică ca 6000 de lei

```
--permisa
SELECT *FROM sal_per_biblioteci
WHERE avgsal<=6000

DROP VIEW IF EXISTS sal_per_biblioteci
```

Results				
Messages				
name	minsal	maxsal	avgsal	
Biblioteca Colegiului National „Spiru Haret” din ...	2309	9040	5227,3	
Biblioteca Liceului Teoretic „Al. I. Cuza”	2569	9095	5771,5	

f. Operații LMD nepermise

```
DELETE FROM sal_per_biblioteci  
WHERE avgsal<=6000
```

Secvența anterioară încearcă să șteargă din vederea complexă bibliotecile care au media salariului angajaților mai mică ca 6000 de lei

```
--nepermisa  
DELETE FROM sal_per_biblioteci  
WHERE avgsal<=6000  
  
--permisa  
SELECT *FROM sal_per_biblioteci  
WHERE avgsal<=6000  
  
DROP VIEW IF EXISTS sal_per_biblioteci
```

Messages

Msg 4405, Level 16, State 1, Line 66

View or function 'sal_per_biblioteci' is not updatable because the modification affects multiple base tables.

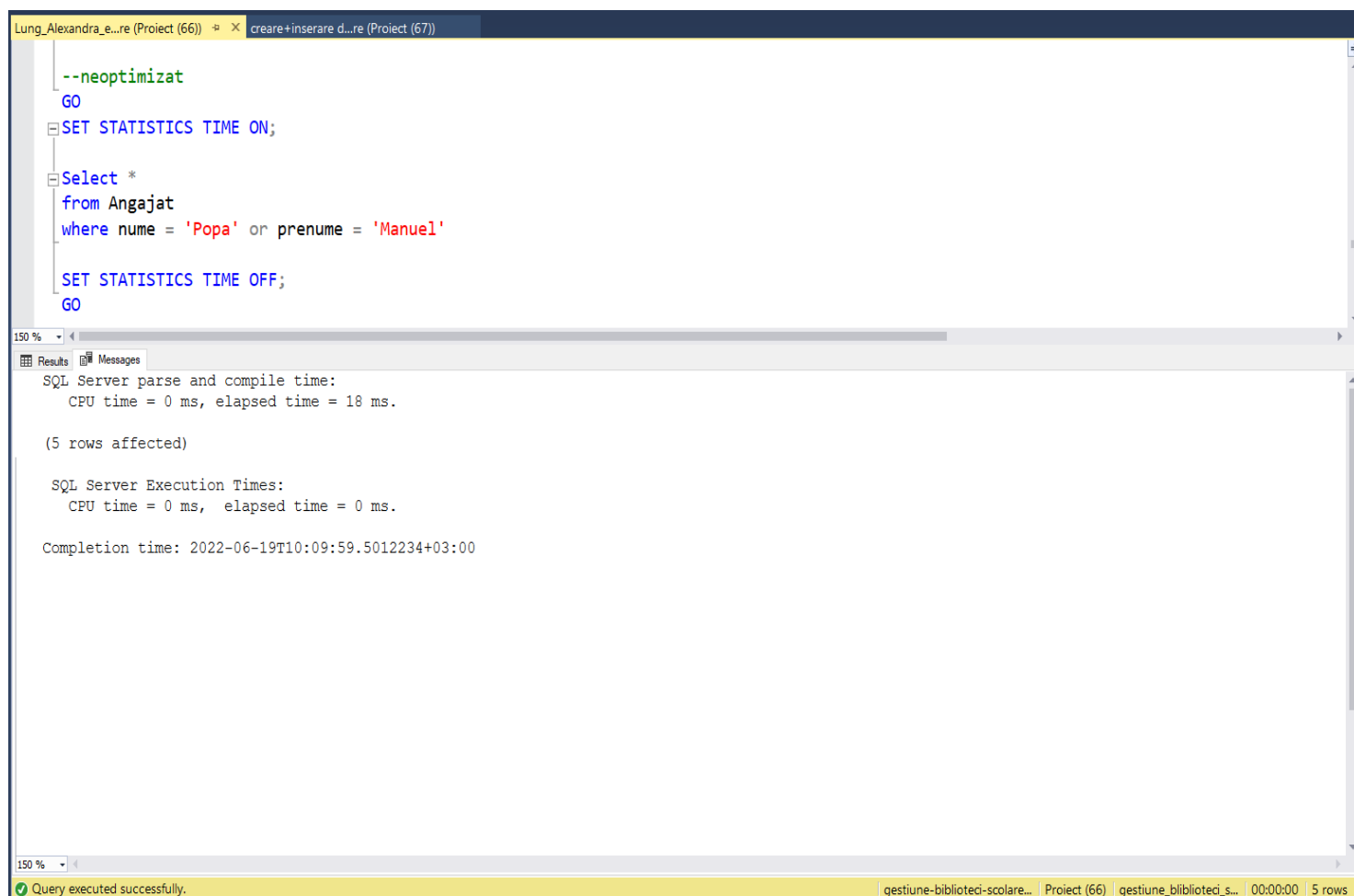
Completion time: 2022-06-07T23:04:14.7259641+03:00

15. Crearea unui index care optimizează cereri de căutare pe 2 criterii

```
--neoptimizat
GO
SET STATISTICS TIME ON;

Select *
from Angajat
where nume = 'Popa' or prenume = 'Manuel'

SET STATISTICS TIME OFF;
GO
```



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window with the following SQL code:

```
--neoptimizat
GO
SET STATISTICS TIME ON;

Select *
from Angajat
where nume = 'Popa' or prenume = 'Manuel'

SET STATISTICS TIME OFF;
GO
```

The bottom pane shows the execution results. The 'Results' tab is active, displaying the following information:

- SQL Server parse and compile time: CPU time = 0 ms, elapsed time = 18 ms.
- (5 rows affected)
- SQL Server Execution Times: CPU time = 0 ms, elapsed time = 0 ms.
- Completion time: 2022-06-19T10:09:59.5012234+03:00

The status bar at the bottom indicates 'Query executed successfully.' and shows the execution time as 00:00:00 and 5 rows affected.

```
--optimizat
CREATE INDEX ang_num_pren_index
ON Angajat (nume, prenume);

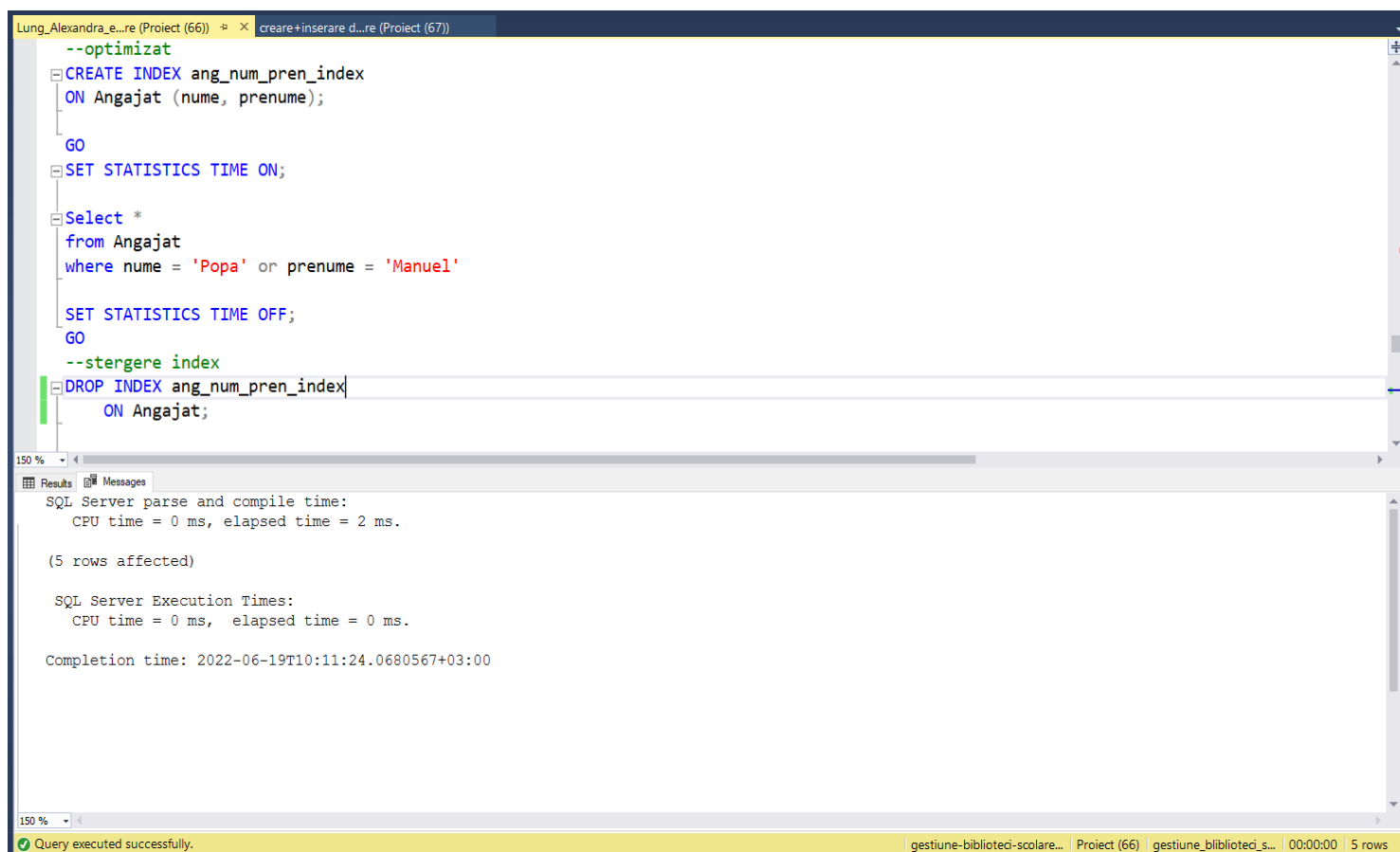
GO

SET STATISTICS TIME ON;

Select *
from Angajat
where nume = 'Popa' or prenume = 'Manuel'

SET STATISTICS TIME OFF;
GO

--stergere index
DROP INDEX ang_num_pren_index
ON Angajat;
```



```
--optimizat
CREATE INDEX ang_num_pren_index
ON Angajat (nume, prenume);

GO

SET STATISTICS TIME ON;

Select *
from Angajat
where nume = 'Popa' or prenume = 'Manuel'

SET STATISTICS TIME OFF;
GO

--stergere index
DROP INDEX ang_num_pren_index
ON Angajat;
```

SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 2 ms.

(5 rows affected)

SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2022-06-19T10:11:24.0680567+03:00

Query executed successfully.

16. Cereri SQL care utilizează outer-join și division

a. Outer-join

Afișați pentru fiecare elev cartea împrumutată, autorul și editura acesteia.

```
SELECT e.ume+ ' '+e.prenume Elev, c1.titlu Carte, ed.ume
EDITURA,a.ume+ ' '+a.prenume Autor
FROM Elev e right outer join FisaDeLectura f on (f.id_elev =
e.id_elev)
left outer join Carte c1 on (c1.id_carte = f.id_carte) full
outer join Editura
ed on (ed.id_editura = c1.id_editura) full outer join Scrisa s
on (s.id_carte = f.id_carte)
full outer join Autor a on (a.id_autor = s.id_autor)
WHERE e.ume IS NOT NULL and c1.titlu IS NOT NULL
ORDER BY e.ume;
```

The screenshot shows a SQL IDE with a query window and a results window. The query is for Exercise 16, which uses outer joins to display book information for each student. The results window shows a table with 20 rows of data.

--Exercitiul 16.

--OUTER JOIN

--Afisati pentru fiecare elev cartea imprumutata , autorul si editura acesteia.

```
SELECT e.ume+ ' '+e.prenume Elev, c1.titlu Carte, ed.ume EDITURA,a.ume+ ' '+a.prenume Autor
FROM Elev e right outer join FisaDeLectura f on (f.id_elev = e.id_elev)
left outer join Carte c1 on (c1.id_carte = f.id_carte) full outer join Editura
ed on (ed.id_editura = c1.id_editura) full outer join Scrisa s on (s.id_carte = f.id_carte)
full outer join Autor a on (a.id_autor = s.id_autor)
WHERE e.ume IS NOT NULL and c1.titlu IS NOT NULL
ORDER BY e.ume;
```

Elev	Carte	EDITURA	Autor
1 Codreanu Horia	Opal	Leda	Amentrout Jennifer
2 Codreanu Horia	Elxir	Hodder & Stoughton General Division	Amentrout Jennifer
3 Codreanu Horia	Elxir	Hodder & Stoughton General Division	Amentrout Jennifer
4 Dobre Manuel	Daimon	Hodder & Stoughton General Division	Amentrout Jennifer
5 Dobre Manuel	The Brightest Night	Leda Edge	Amentrout Jennifer
6 Dobre Manuel	The Burning Shadow	Leda Edge	Amentrout Jennifer
7 Draghici Lucian	The Brightest Night	Leda Edge	Amentrout Jennifer
8 Marin Manuel	Print si cersator	All	Twain Mark
9 Marin Manuel	Umbre	Leda Edge	Amentrout Jennifer
10 Marin Manuel	Pure	Hodder & Stoughton General Division	Amentrout Jennifer
11 Marin Manuel	Jumalul unui pustiR2	Art	Kinney Jeff
12 Marin Manuel	The One & Only	Hodder & Stoughton General Division	Amentrout Jennifer
13 Mocanu Gheorghe	Elxir	Hodder & Stoughton General Division	Amentrout Jennifer
14 Mocanu Gheorghe	Deity	Hodder & Stoughton General Division	Amentrout Jennifer
15 Mocanu Gheorghe	Umbre	Leda Edge	Amentrout Jennifer
16 Mocanu Gheorghe	Sentinel	Hodder & Stoughton General Division	Amentrout Jennifer
17 Negoitescu Alin	Fram ursul polar	All	Petrescu Cezar
18 Negoitescu Alin	Apollyon	Hodder & Stoughton General Division	Amentrout Jennifer
19 Nicolescu Ovidiu	Opozitie	Leda Edge	Amentrout Jennifer
20 Nicolescu Ovidiu	Print si cersator	All	Twain Mark

Query executed successfully.

b. Division

Să se afișeze id-ul, numele și prenumele elevilor, titlul și prețul cărților care au fost împrumutate și au prețul între 20 și 50 de lei.

```
SELECT f.id_elev, nume, prenume, titlu, pret
FROM FisaDeLectura f join Elev e on (f.id_elev=e.id_elev) join
Carte c on (f.id_carte = c.id_carte)
WHERE c.id_carte in (SELECT id_carte FROM Carte WHERE pret>=20
and pret <= 50)
GROUP BY f.id_elev, nume, prenume, titlu, pret
HAVING count(*) <= (SELECT count(id_carte) FROM Carte WHERE
pret>=20 and pret <= 50)
```

except

```
SELECT f.id_elev, nume, prenume, titlu, pret
FROM FisaDeLectura f join Elev e on (f.id_elev=e.id_elev) join
Carte c on (f.id_carte = c.id_carte)
WHERE c.id_carte not in (SELECT id_carte FROM Carte WHERE
pret>=20 and pret <= 50)
```

The screenshot shows a SQL IDE with two tabs: 'Lung_Alexandra_e...re (Proiect (66))' and 'creare+inserare d...re (Proiect (67))'. The active tab contains a SQL query with comments in Romanian. The query is divided into two parts by an 'except' keyword. The first part selects books with prices between 20 and 50 lei, and the second part selects books with prices outside this range. The results pane at the bottom shows a table with 20 rows of data.

```
--DIVISION

--Sa se afiseze id-ul, numele si prenumele elevilor, titlul si pretul cartiilor care au fost imprumutate
--și au pretul între 20 și 50 de lei.

SELECT f.id_elev, nume, prenume, titlu, pret
FROM FisaDeLectura f join Elev e on (f.id_elev=e.id_elev) join Carte c on (f.id_carte = c.id_carte)
WHERE c.id_carte in (SELECT id_carte FROM Carte WHERE pret>=20 and pret <= 50)
GROUP BY f.id_elev, nume, prenume, titlu, pret
HAVING count(*) <= (SELECT count(id_carte) FROM Carte WHERE pret>=20 and pret <= 50)

except

SELECT f.id_elev, nume, prenume, titlu, pret
FROM FisaDeLectura f join Elev e on (f.id_elev=e.id_elev) join Carte c on (f.id_carte = c.id_carte)
WHERE c.id_carte not in (SELECT id_carte FROM Carte WHERE pret>=20 and pret <= 50)
```

	id_elev	nume	prenume	titlu	pret
1	3	Popescu	Lucian	Diety	46
2	3	Popescu	Lucian	Obsidian	30
3	6	Popa	Alin	Half-Blood	40,5
4	7	Negoițescu	Alin	Apollyon	50
5	10	Mocanu	Gheorghe	Diety	46
6	10	Mocanu	Gheorghe	Elxir	46,9
7	10	Mocanu	Gheorghe	Umire	30
8	11	Rusu	Mircea	Like the First Time	42,5
9	11	Rusu	Mircea	Obsidian	30
10	11	Rusu	Mircea	The Darkest Star	50
11	13	Dobre	Manuel	Daimon	30,5
12	15	Codreanu	Horia	Elxir	46,9
13	15	Codreanu	Horia	Opal	30
14	18	Nicolescu	Ovidiu	Obsidian	30
15	18	Nicolescu	Ovidiu	Opotzie	30
16	19	Tudor	Horia	Elxir	46,9
17	19	Tudor	Horia	Jumalul unui pustii#2	29
18	20	Marin	Manuel	Jumalul unui pustii#2	29
19	20	Marin	Manuel	Pune	45

Query executed successfully. | gestiune-bibliotec-scolare... | Proiect (66) | gestiune_bibliotec_s... | 00:00:00 | 20 rows

Să se afișeze elevii care au împrumutat aceleași cărți ca elevul cu id-ul 15.

```
SELECT f.id_elev, nume, prenume
FROM FisaDeLectura f join Elev e on (e.id_elev = f.id_elev)
WHERE id_carte in (SELECT id_carte
                   FROM FisaDeLectura
                   WHERE id_elev=15)
```

```
and f.id_elev !=15
except
```

```
SELECT f.id_elev, nume, prenume
FROM FisaDeLectura f join Elev e on (e.id_elev = f.id_elev)
WHERE f.id_carte not in (SELECT id_carte FROM FisaDeLectura
                        WHERE id_elev=15)
and f.id_elev not in (SELECT i.id_elev FROM Imprumut i WHERE
                     i.id_carte in (SELECT id_carte
                                   FROM FisaDeLectura
                                   WHERE id_elev=15))
```

The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL code:

```
--Sa se afiseze elevii care au imprumutat aceleasi carti ca elevul cu id-ul 15.

SELECT f.id_elev, nume, prenume
FROM FisaDeLectura f join Elev e on (e.id_elev = f.id_elev)
WHERE id_carte in (SELECT id_carte
                  FROM FisaDeLectura
                  WHERE id_elev=15)

and f.id_elev !=15

except

SELECT f.id_elev, nume, prenume
FROM FisaDeLectura f join Elev e on (e.id_elev = f.id_elev)
WHERE f.id_carte not in (SELECT id_carte
                        FROM FisaDeLectura
                        WHERE id_elev=15)
and f.id_elev not in (SELECT i.id_elev
                    FROM Imprumut i
                    WHERE i.id_carte in (SELECT id_carte
                                        FROM FisaDeLectura
                                        WHERE id_elev=15))

)
```

The results pane shows the following data:

id_elev	nume	prenume
10	Mocanu	Gheorghe
19	Tudor	Horia

The status bar at the bottom indicates: Query executed successfully. | gestiune-bibliotec-scolare... | Proiect (66) | gestiune_bibliotec_s... | 00:00:00 | 2 rows

17. Optimizarea unei cereri

Să se afișeze titlul, seria și limba cărților scrise de Armentrout și care au fost publicate de editura cu id-ul egal cu 1.

Cerere SQL

```
SELECT c.titlu,c.seria,c.limba
FROM Carte c join Scrisa s on (c.id_carte = s.id_carte)
join Autor a on (a.id_autor = s.id_autor)
WHERE upper(a.num)= 'ARMENTROUT' and c.id_editura = 2
```

Expresii algebrice

R1 = SELECT(AUTOR,nume='Armentrout') – eliminăm elementele nefolositoare

R2 = PROJECT(R1, id_autor) – îndepărtăm atributele nefolositoare

R3 = PROJECT(SCRIERE, id_autor, id_carte)

R4 = SEMIJOIN(R2, R3, id autor) – SEMIJOIN pentru că avem nevoie doar de id carte

R5 = SELECT(CARTE, id_editura=2)

R6 = PROJECT(R5, id_carte, titlu, serie,limba)

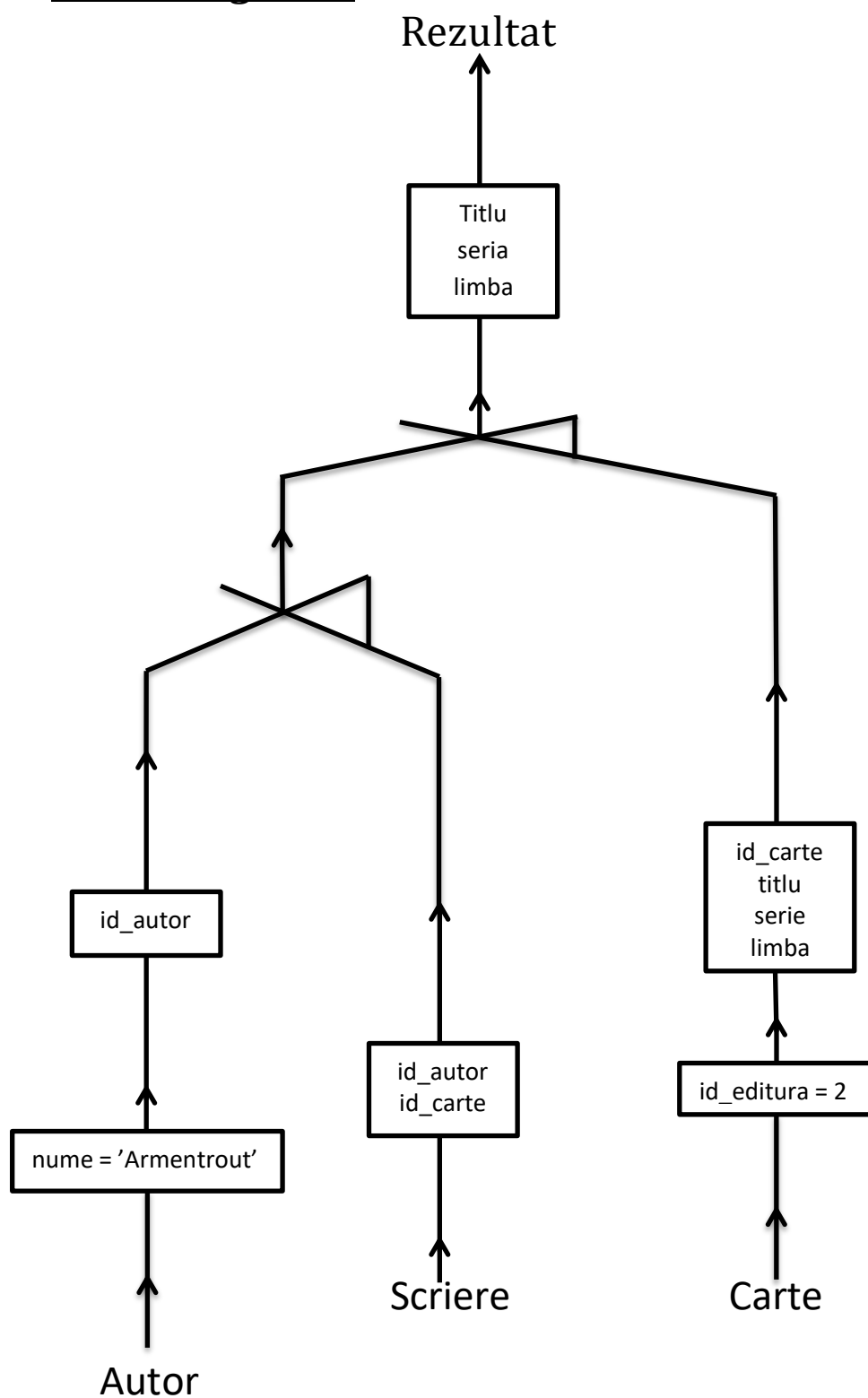
R7 = SEMIJOIN(R4, R6, id_carte) – SEMIJOIN pentru că avem nevoie doar de date dintr-o singura relație

Rezultat = R8 = PROJECT(R7, titlu, serie,limba)

Optimizare

Am proiectat cererea optim de la început, deoarece am ținut cont de regulile de optimizare:

- *Selectiile se execută cât mai devreme posibil. Motivația acestei reguli este că selectiile reduc substanțial dimensiunea relațiilor.*
- *Produsurile carteziane se înlocuiesc cu joinuri, ori de câte ori este posibil. Un produs cartezian între două relații este de obicei mult mai scump (ca și cost) decât un join între cele două relații, deoarece primul generează concatenarea tuplurilor în mod exhaustiv și poate genera un rezultat foarte mare.*
- *Un join este mai restrictiv decât altul dacă produce o relație mai mică. Se poate determina care join este mai restrictiv pe baza factorului de selectivitate sau cu ajutorul informațiilor statistice.*
- *Proiecțiile se execută la început pentru a îndepărta attributele nefolositoare. Dacă un atribut al unei relații nu este folosit în operațiile ulterioare atunci trebuie îndepărtat. (am folosit semijoin-uri, eliminând astfel attributele nefolositoare)*

Arbore algebric

18. Normalizare și denormalizare

Exemplele care urmează pentru a exemplifica normalizarea, se vor considera exemple care nu se regăsesc în modelul implementat, dar care au legătură cu acesta.

a) BCNF

Determinantul este un atribut sau o mulțime de attribute neredundante, care constituie un identificator unic pentru alt atribut sau altă mulțime de attribute ale unei relații date.

Intuitiv, o relație R este în forma normală Boyce-Codd dacă și numai dacă fiecare determinant este o cheie candidat.

Formal, o relație R este în forma normală Boyce-Codd dacă și numai dacă pentru orice dependență funcțională totală $X \rightarrow A$, X este o cheie (candidat) a lui R.

Regula Casey Delobel pentru $R(K1\#, K2\#, X)$ presupunând că există dependența: $X \rightarrow K2$. $R1(K1\#, X)$ și $R2(X\#, K2)$

Nume_carte	ISBN	Id_categorie	Numar_pagini	Autor
C1	ISBN1	Cat1	908	A1
C2	ISBN2	Cat2	487	A2
C3	ISBN3	Cat3	467	A3
C4	ISBN4	Cat4	523	A4
C5	ISBN5	Cat5	376	A5

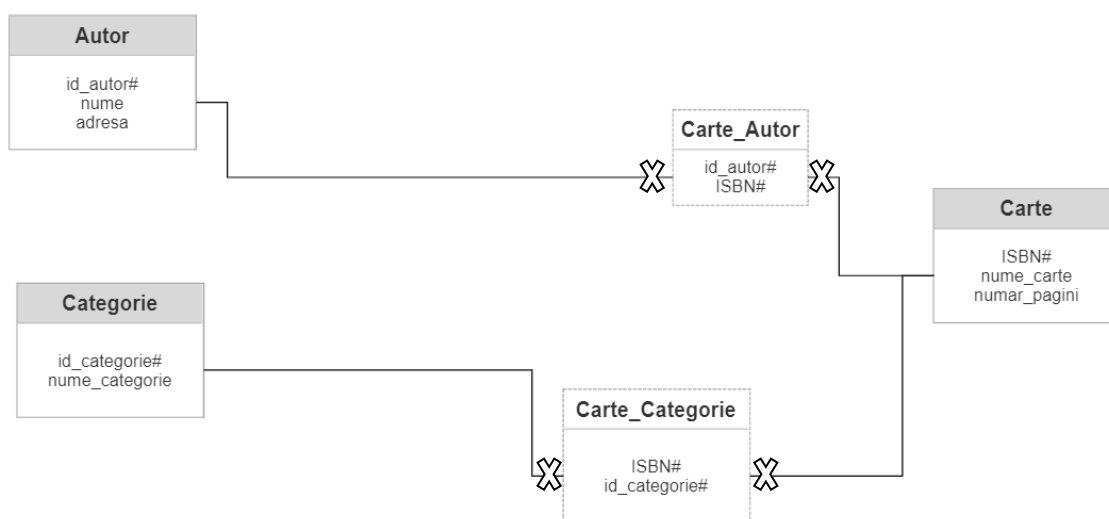
Exemplu Non-BCNF

$\{\text{nume_carte}, \text{id_categorie}\} \rightarrow \{\text{autor}\} \Rightarrow (x, y) \rightarrow z$

$\{\text{autor}\} \rightarrow \{\text{nume_carte}\} \Rightarrow z \rightarrow x$

Prin urmare, avem nevoie de un tabel separat pentru a păstra detaliile AUTORULUI. Dar, la rândul său, autorul va avea id-ul, adresa, telefonul, etc. Prin urmare, un tabel - AUTOR ar trebui să fie creat pentru a păstra detaliile autorului, și un alt tabel - CARTE_AUTOR ar trebui creat pentru a menține maparea între cărți și autor. Similar observăm că se întâmplă și pentru carte și id_categorie.

Prin urmare, tabelele sunt acum mapate după cum urmează:



Exemplu BCNF

b) FN4

Conform celei de-a patra forme normale,

- Ar trebui să îndeplinească toate cerințele 3NF
- Atributul unuia sau mai multor rânduri din tabel nu ar trebui să conducă la mai multe rânduri ale aceluiași tabel care să ducă la dependențe cu mai multe valori

FN4 elimină redundanțele datorate relațiilor m:n, adică datorate dependenței multiple. Intuitiv, o relație R este în a patra formă normală dacă și numai dacă relația este în BCNF și nu conține relații m:n independente.

Vom considera următorul exemplu:

- Un angajat poate lucra la mai multe biblioteci
- Un angajat poate ocupa mai multe funcții în cadrul unei bibliotecii (din lipsă de personal calificat)

Avem următoarea relație:

$R(\text{angajat\#}, \text{biblioteca\#}, \text{funcție\#})$

$\text{angajat} \rightarrow \text{biblioteca}$

$\text{angajat} \rightarrow \text{funcție}$

<i>angajat#</i>	<i>biblioteca#</i>	<i>funcție#</i>
A1	1	Operator de introducere date
A2	2	Paznic
A1	1	Bibliotecar
A3	2	Manager
A2	3	Pazinic

Exemplu Non-FN4

Descompunerea relației se va face după cum urmează:

- $R_1(\text{angajat\#}, \text{biblioteca\#})$
- $R_2(\text{angajat\#}, \text{funcție\#})$

<i>angajat#</i>	<i>biblioteca#</i>
A1	1
A2	2
A2	3
A3	2

R_1

<i>angajat#</i>	<i>funcție#</i>
A1	Operator de introducere date
A1	Bibliotecar
A2	Pazinic
A3	Manager

R_2

c) FN5

FN5 își propune eliminarea redundanțelor care apar în relații $m:n$ dependente. În general, aceste relații nu pot fi descompuse. S-a arătat că o relație de tip 3 este diferită de trei relații de tip 2. Există totuși o excepție, și anume, dacă relația este ciclică

Intuitiv, o relație R este în forma normală 5 dacă și numai dacă:

1. relația este în FN4;
2. nu conține dependențe ciclice.

Vom considera următorul exemplu în care elevul poate împrumuta mai multe cărți pe baza mai multor legitimații:

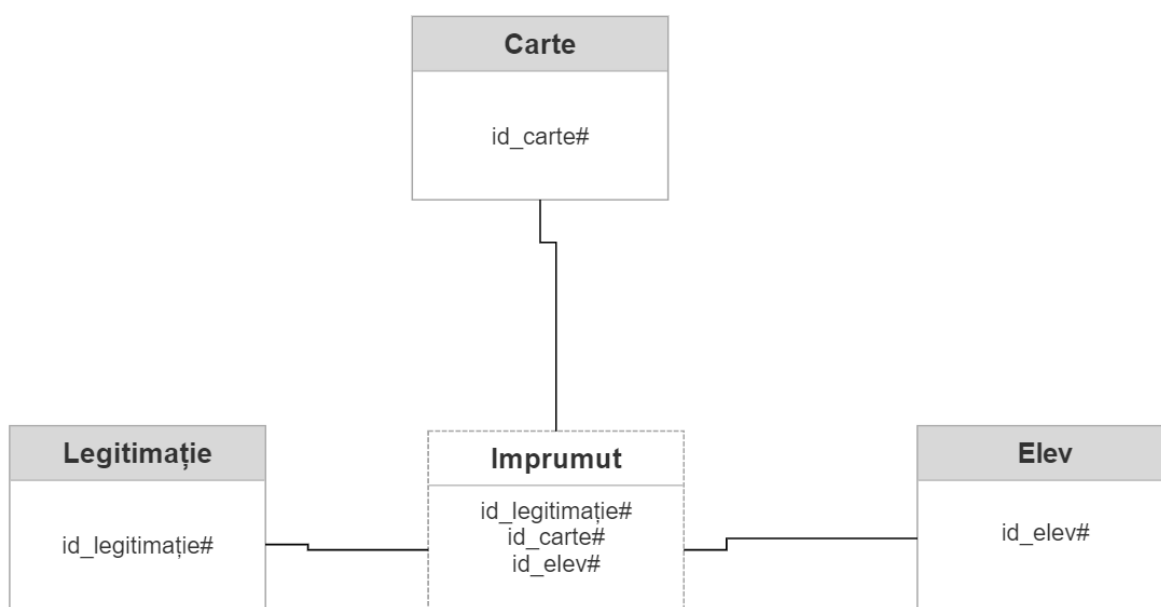
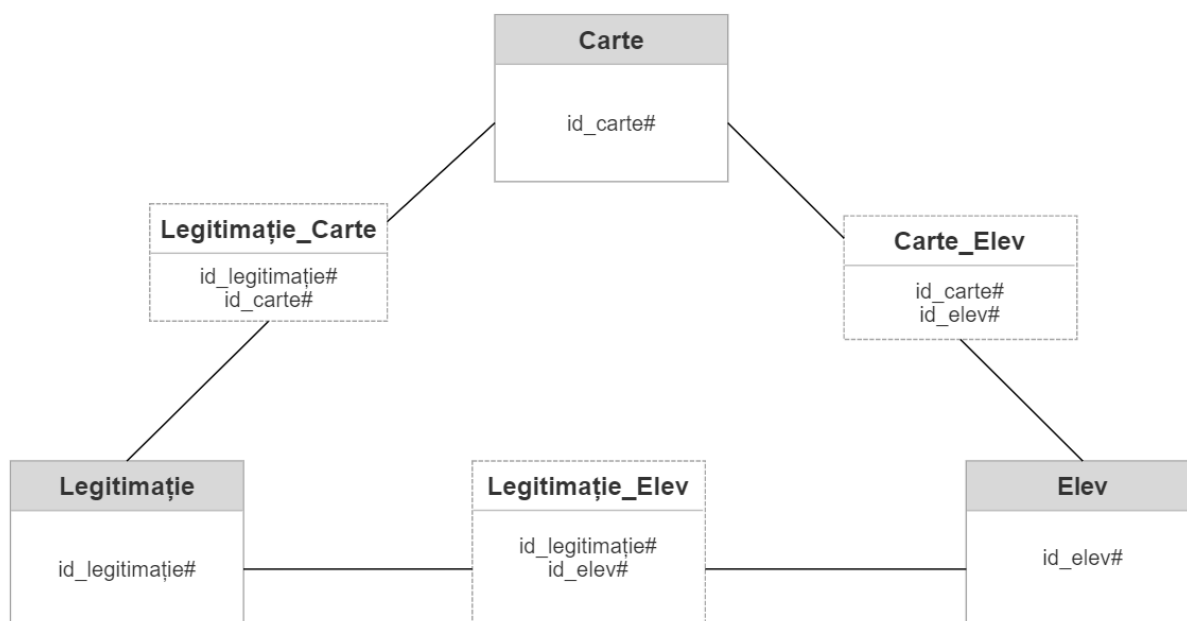


Fig 1.

Această relație din Fig 1. este echivalentă cu 3 relații de tip 3, doar dacă sunt relații ciclice.



Relația fiind ciliară, atunci când se va efectua toate join-urile se va obține un rezultat echivalent cu cel obținut din relația de tip 3. Ca o relație să fie în FN5, trebuie să fie în FN4 și să nu conțină dependențe ciclice. Astfel, se observă, că cele 3 relații de tip 2 compun o diagramă care conține dependențe ciclice, deci relația prezentată anterior nu se află în FN5. De asemenea, relația de tip 3 este în FN5. (Fig 1.)

d) Denormalizare

Denormalizarea este procesul invers al procesului de normalizare. Denormalizarea funcționează adăugând date redundante sau grupând date pentru a optimiza performanța.

Fie $R = \{R_1, R_2, \dots, R_p\}$ o mulțime de relații. Denormalizarea R înseamnă înlocuirea R cu $R' = \text{JOIN}(R_1, R_2, \dots, R_p)$, astfel încât $\forall 1 \leq i \leq p$: proiecția lui R după attributele lui R_i va produce din nou relația R_i .

Obiectivul denormalizării

- *mărirea redundanței (relația R' se află la un nivel de normalizare mai scăzut decât relațiile R_1, R_2, \dots, R_p componente)*
- *reducerea numărului de join-uri care trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans (ca parte din proiectarea bazei de date).*

Vom considera exemplul:

Avem un tabel, numit Carte, în care sunt stocate cărți.

Cărțile au un atribut preț. În această coloană se întâlnesc valori repetitive, deoarece mai multe cărți pot avea același preț.

În acest caz, dacă în baza de date există un tabel separat care reține prețul și id-ul cărții corespunzătoare acestuia, ar fi necesar procesul de denormalizare, întrucât nu este eficient ca atributul preț să se afle într-un tabel separat.

19. Bibliografie(Webografie)

- http://budisteanu.net/Download/DB/curs_7_11_BD.pdf
- <https://ticileananeciu.files.wordpress.com/2019/10/manual-oracle-carmen-popescu.pdf>
- <https://www.yumpu.com/ro/document/read/13467617/s-structuri-si-baze-de-date>
- <https://docs.microsoft.com/en-us/sql/t-sql/statements/statements?view=sql-server-ver16>
- <https://www.scribd.com/doc/74615468/carte>