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***](#DescriereaModelului)
2. [***Prezentare constrângeri***](#PrezentareConstrângeri)
3. [***Descrierea entităților***](#DescriereaEntităților)
4. [***Descrierea relațiilor***](#DescriereaRelațiilor)
5. [***Descrierea atributelor***](#DescriereaAtributelor)
6. [***Diagrama entitate-relație***](#DiagramaEntitateRelație)
7. [***Diagrama conceptuală***](#DiagramaConceptuală)
8. [***Scheme relaționale***](#SchimeRelaționare)
9. [***Normalizare(FN1-FN3)***](#NormalizareFN1FN3)
   1. *Forma normală 1(FN1)*
   2. *Forma normală 2(FN2)*
   3. *Forma normală 3(FN3)*
10. [***Crearea tabelelor in SQL, inserarea datelor și adăugarea constrângerilor pentru:***](#CreareaTabelelor)
    1. Tabela *Editura*
    2. Tabela *Oras*
    3. Tabela *Biblioteca*
    4. Tabela *Elev*
    5. Tabela *Legitimatie*
    6. Tabela *Autor*
    7. Tabela *Carte*
    8. Tabela *FisaDeLectura*
    9. Tabela *Domeniu*
    10. Tabela *Angajat*
    11. Tabela *Imprumut*
    12. Tabela *Scrisă*
    13. Tabela *Aparține*
11. [***Cereri complexe SQL***](#CererComplexe)
    1. *Prima cerere*
    2. *A doua cerere*
    3. *A treia cerere*
    4. *A prata cerere*
    5. *A cincea cerere*
12. [***Trei cereri de actualizare și suprimare a datelor***](#cererideacutalizare)
13. [***Crearea unei secvențe SQL folosite pentru inserarea înregistrărilor in tabele***](#SecventeSqlPtInserareDate)
14. [***Crearea unei vizualizări compuse***](#Vizulizare)
    1. *Crearea vizualizării*
    2. *Operații LMD permise*
    3. *Operații LMD nepermise*
15. [***Crearea unui index care optimizează cereri de căutare pe 2 criterii***](#Index)
16. [***Cereri SQL care utilizează* outer-join *și* division**](#OuterJoin)
    1. *Outer-join*
    2. *Division*
17. [***Optimizarea unei cereri***](#optimizare)
    1. *Cerere SQL*
    2. *Expresii algebrice*
    3. *Optimizare*
    4. *Arbore algebric*
18. [***Normalizare și denormalizare***](#NormalizareDenormalizare)
    1. *Forma normală Boyce-Codd (BCNF)*
    2. *Forma normală 4 (FN4)*
    3. *Forma normală 5 (FN5)*
    4. *Aplicare denormalizare*
19. [***Bibliografie***](#Bibliografie)

1. Descrierea modelului real și a regulilor de funcționare

Trebuie menţionat faptul că lucrarea de faţă îşi propune înainte de toate să identifice cele mai importante aspecte şi probleme ale biblioteconomiei precum şi 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 inregistrată 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 sa aiba 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 fisă 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 legeitimații 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#***
* **Legitimație** = 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\_lectură#***

4. Descrierea relațiilor

Descrierea relațiilor modelului de date precizând cardinalitatea minimă și maximă.

***Orașul\_are\_Biliotecă*** *= relația care leagă entitățile Oraș și Bibliotecă refelectând legătura dintre acestea.Aceasta are cardinalitatea minimă de 1:1 și maximă de 1:n*

***Biblioteca\_are\_Angajat*** *= relația care leagă entitățile Bibliotecă și Angajat refelectând legătura dintre acestea.Aceasta are cardinalitatea minimă de 1:0 și maximă de 1:n*

***Cartea \_aparține\_Domeniului*** *= relația care leagă entitățile Carte și Domeniu refelectând legătura dintre acestea (o carte aparține unui domeniu).Aceasta are cardinalitatea minimă de 0:1 și maximă de m:n*

***Editura\_publică\_Cartea*** *= relația care leagă entitățile Editură și Carte refelectând 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 refelectând legătura dintre acestea.Aceasta are cardinalitatea minimă de 1:1 și maximă de m:n*

***Cartea\_se\_găseste\_în\_FisaDeLectură*** *= relația care leagă entitățile Carte și Fisă de lectură refelectând 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 refelectând legătura dintre acestea. Aceasta are cardinalitatea minimă de 1:0 și maximă de 1:n*

***Relația de tip 3 dintre entitătile Elev, Bibliotecă și Carte*** *= relația care leagă entitățile Elev,**Bibliotecă și Carte refelectând 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 carcatere de lungime maximă 30, care nu poate fi nulă, reprezentând numele unui domeniu

Entitatea ***Oras***  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 atribute:

* 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 atribute:

* id\_angajat = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui angajat
* id\_bibliotecă = 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 ***Bibliotecă***
* 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 anagajării unui angajat

Entitatea ***Carte***  are ca atribute:

* 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 atribute:

* 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 găseste 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 găseste 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 atribute:

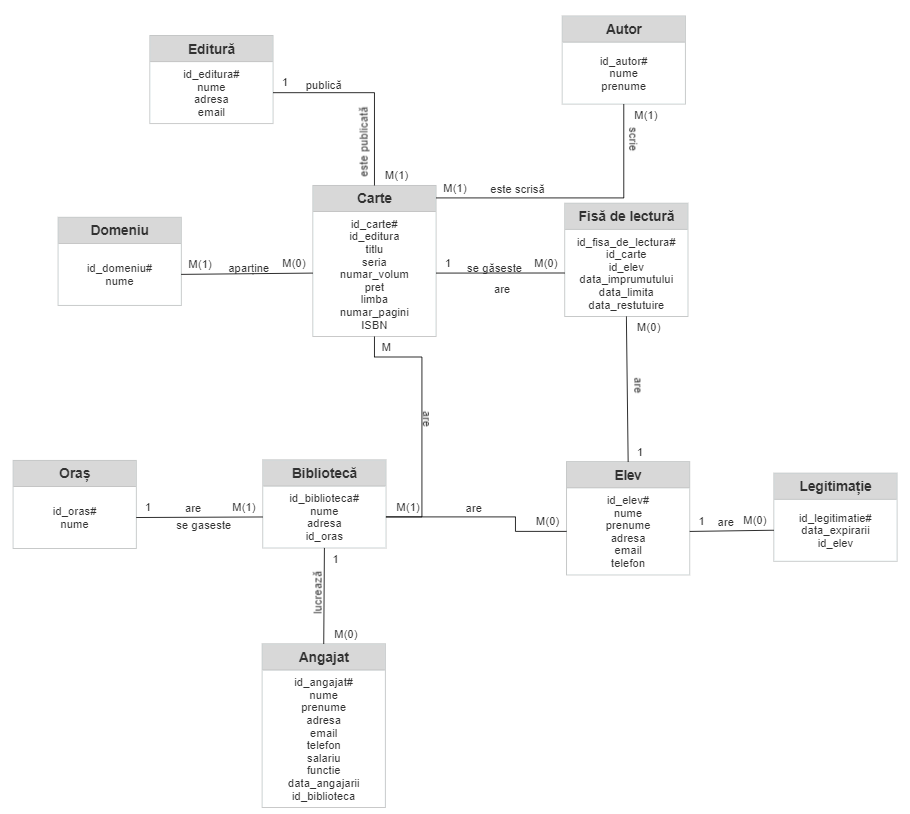
* id\_carte = variabilă de tip întreg care nu poate fi nulă, reprezentând ID-ul unei cărți.Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ***Carte***
* id\_autor = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unui autor. Atributul trebuie să corespundă 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

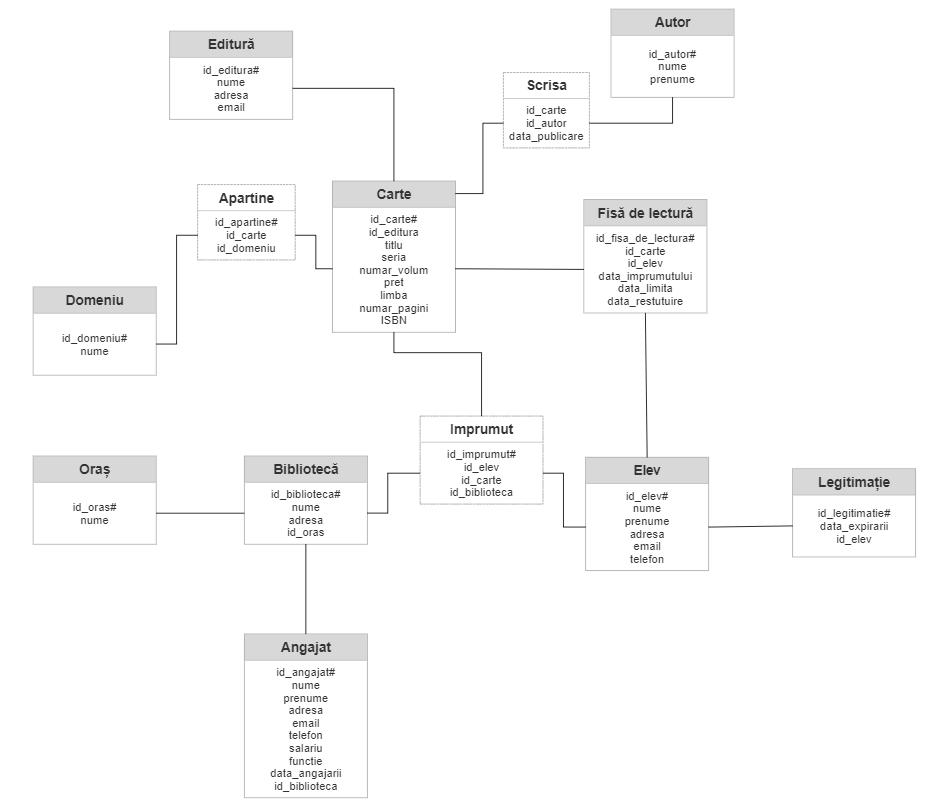
Relația ***Carte\_apartine\_Domeniu*** are ca atribute:

* id\_aparținere = variabilă de tip întreg, care nu poate fi nulă, reprezentând ID-ul unei aparțineri, fiind cheia primară a tabelului asociativ ***Aparține***
* 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 entitătile Elev, Bibliotecă și Carte are ca atribute:***

* 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 biblioteci. 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)

Bibliotecă(id\_biblioteca#, nume, adresa, id\_oras)

Aparține( 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)

Legitimație(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 şi inconsistenţe ale datelor. O bază de date bine proiectată nu permite ca datele să fie redundante, adică aceeaşi informaţie să se găsească în locuri diferite sau să se memoreze în baza de date informaţii care se pot deduce pe baza altor informaţii memorate în aceeaşi bază de date.*

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

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

* 1. ***Forma normală 1(FN1)***

*O entitate se găseşte în prima formă normală dacă şi numai dacă:*

*- nu există atribute cu valori multiple*

*- nu există atribute sau grupuri de atribute care se repetă.*

*Cu alte cuvinte toate atributele trebuie să fie atomice, adică să conţină o singură informaţie.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:*

|  |  |
| --- | --- |
| *Autor* | *Carte* |
| *AUT 1* | *C1,C2,C3* |
| *AUT 2* | *C1,C4* |

*Exemplu non-FN1*

|  |  |
| --- | --- |
| *Autor* | *Carte* |
| *AUT 1* | *C1* |
| *AUT1* | *C2* |
| *AUT1* | *C3* |
| *AUT 2* | *C1* |
| *AUT 2* | *C4* |

*Exemplu FN1*

* 1. ***Forma normală 2(FN2)***

*O entitate se găseşte în a doua formă normală dacă şi numai dacă se găseşte în prima formă normală şi 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:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *id\_autor#* | *nume* | *prenume* | *id\_carte#* | *data\_publicare* |
| *1* | *Armentrout* | *Jennifer* | *C1* | *2018-02-19* |
| *2* | *Armentrout* | *Jennifer* | *C2* | *2011-07-29* |
| *3* | *Morgan* | *Richard* | *C3* | *2002-03-11* |

*Exemplu non-FN2*

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

*{id\_autor#}->{nume,prenume}->determină funcțional autor*

*{id\_autor#,id\_carte#}->{data\_publiare}*

*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*

* 1. ***Forma normală 3(FN3)***

*O entitate se găseşte în a treia formă normală dacă şi numai dacă se găseşte în a doua formă normală şi 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 dependenţe 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:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *id\_carte#* | *id\_editura#* | *nume* | *adresa* | *email* |
| *C1* | *1* | *Leda* | *adr1* | *email1* |
| *C2* | *2* | *Litera* | *adr2* | *email2* |
| *C3* | *3* | *Paladin* | *adr3* | *email3* |

*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:*

|  |  |  |  |
| --- | --- | --- | --- |
| *id\_editura#* | *nume* | *adresa* | *email* |
| *1* | *Leda* | *adr1* | *email1* |
| *2* | *Litera* | *adr2* | *email2* |
| *3* | *Paladin* | *adr3* | *email3* |

|  |  |
| --- | --- |
| *id\_carte#* | *id\_editura#* |
| *C1* | *1* |
| *C2* | *2* |
| *C3* | *3* |

*Exemplu FN3*

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

Secvențele corespunzătoarea creeri și inserări datelor in SQL (constrângerile au fost adăugate o dată cu crearea tabelelor) sunt:

1. ***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.' );

1. ***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

1. ***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);

1. ***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', 'draghicilucian@gmail.com', '0727556710'),

    ('Pop', 'Manuel', 'Calea Mosilor 13,Bucuresti', 'popmanuel@gmail.com', '0773263442'),

    ('Nicolescu', 'Ovidiu', 'Strada Berzei 2,Bucuresti', 'nicolescuovidiu@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');

1. ***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);

1. ***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');

1. ***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);

1. ***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');

1. ***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');

1. ***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', 'codreanudan@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 Vitan 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', 'marinovidiu@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', 'ilieovidiu@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)

1. ***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');

1. ***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

1. ***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 lucreză 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**

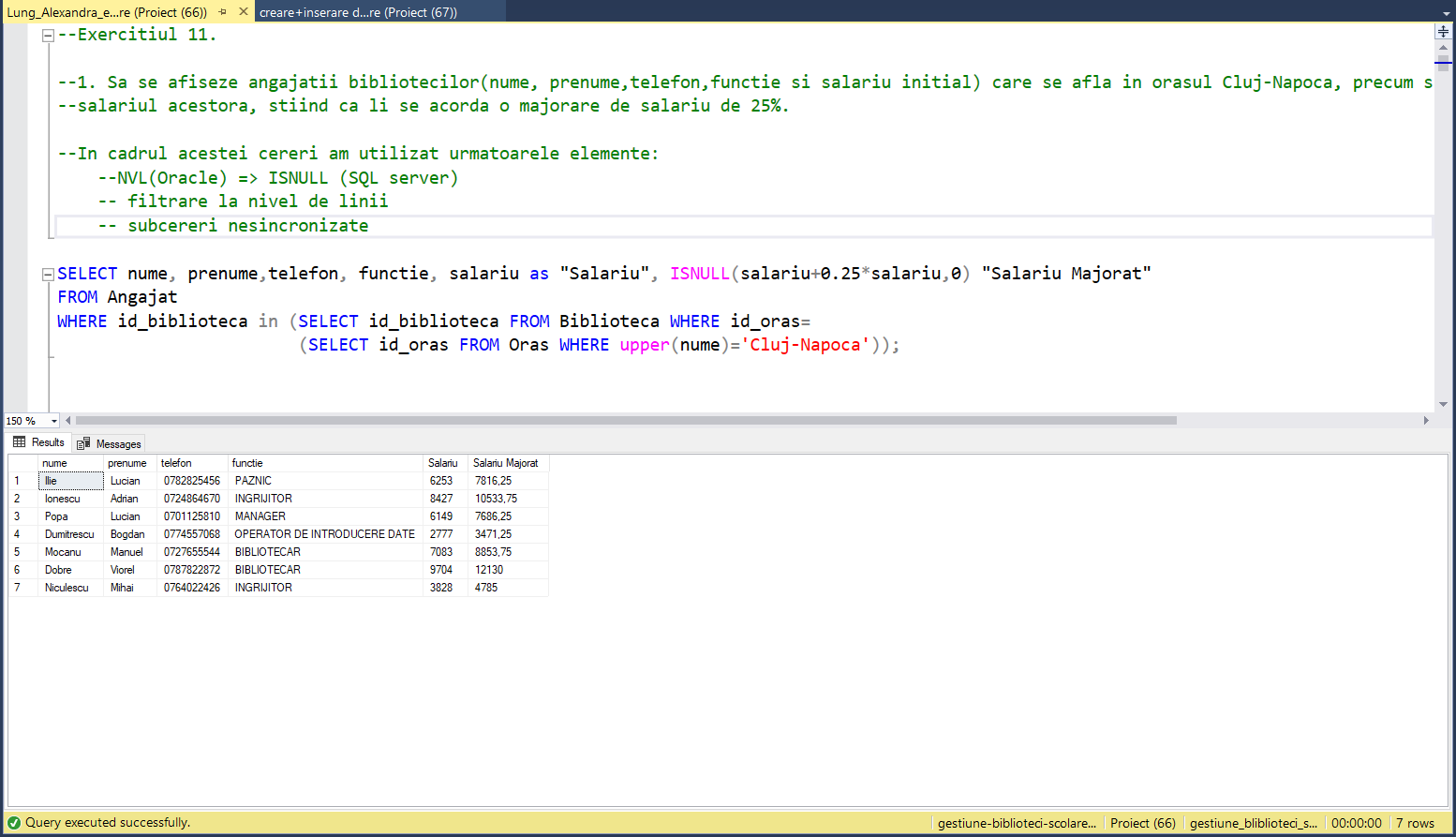
**WHERE id\_biblioteca in (SELECT id\_biblioteca FROM Biblioteca WHERE id\_oras=(SELECT id\_oras FROM Oras WHERE upper(nume)='Cluj-Napoca'));**

*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'));



1. ***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 a șasea 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*
* *funcții pe șiruri de caractere*
* *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.nume ,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 MONTH(f.data\_imprumutului) = 6)

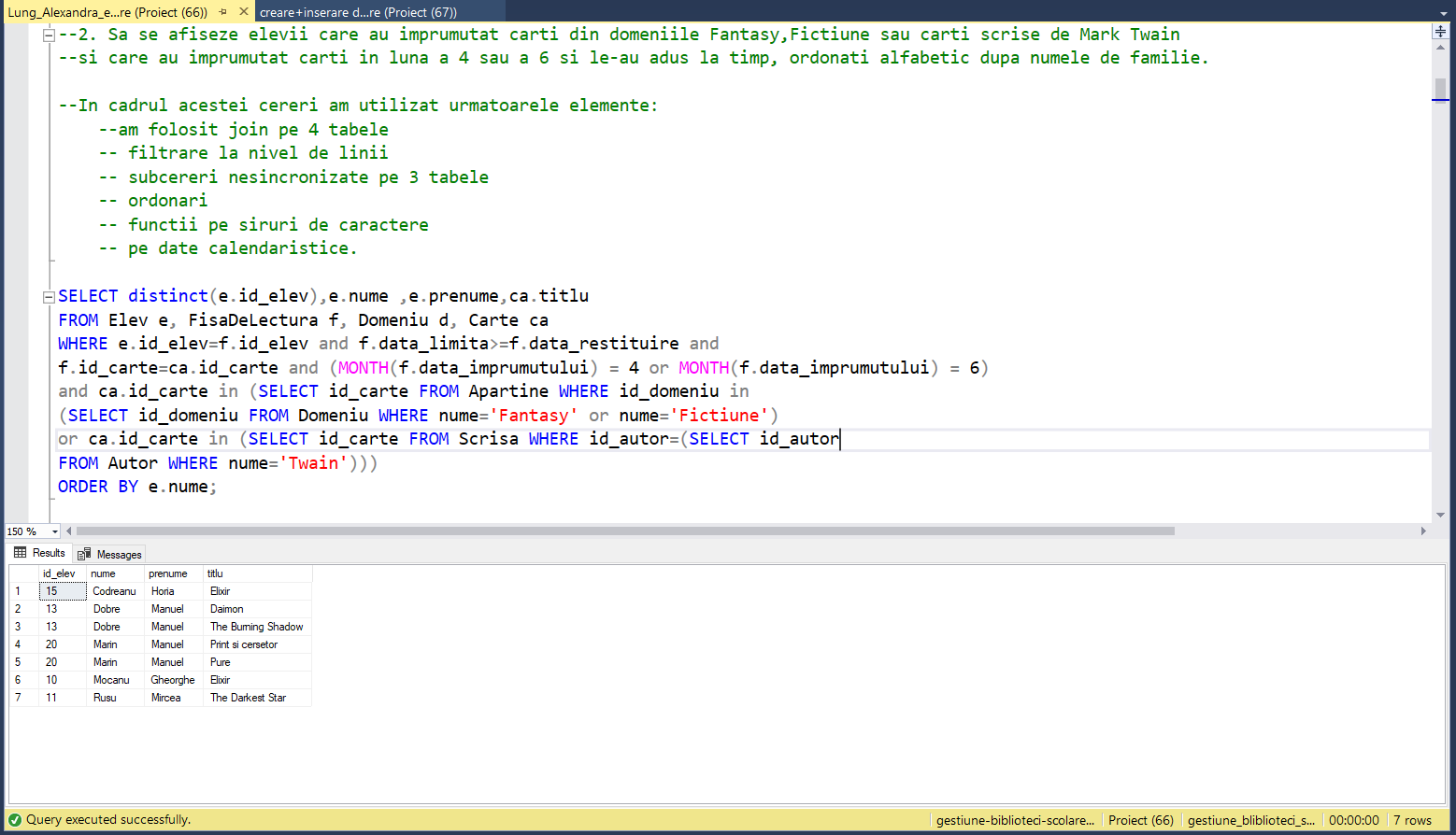
and ca.id\_carte in (SELECT id\_carte FROM Apartine WHERE id\_domeniu in

(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.nume;



1. ***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%.*

*Afisați 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 a.nume, a.prenume, a.salariu, a.data\_angajarii, a.functie, b.nume "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 a.nume,a.prenume,a.salariu,a.data\_angajarii,a.functie,b.nume "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)

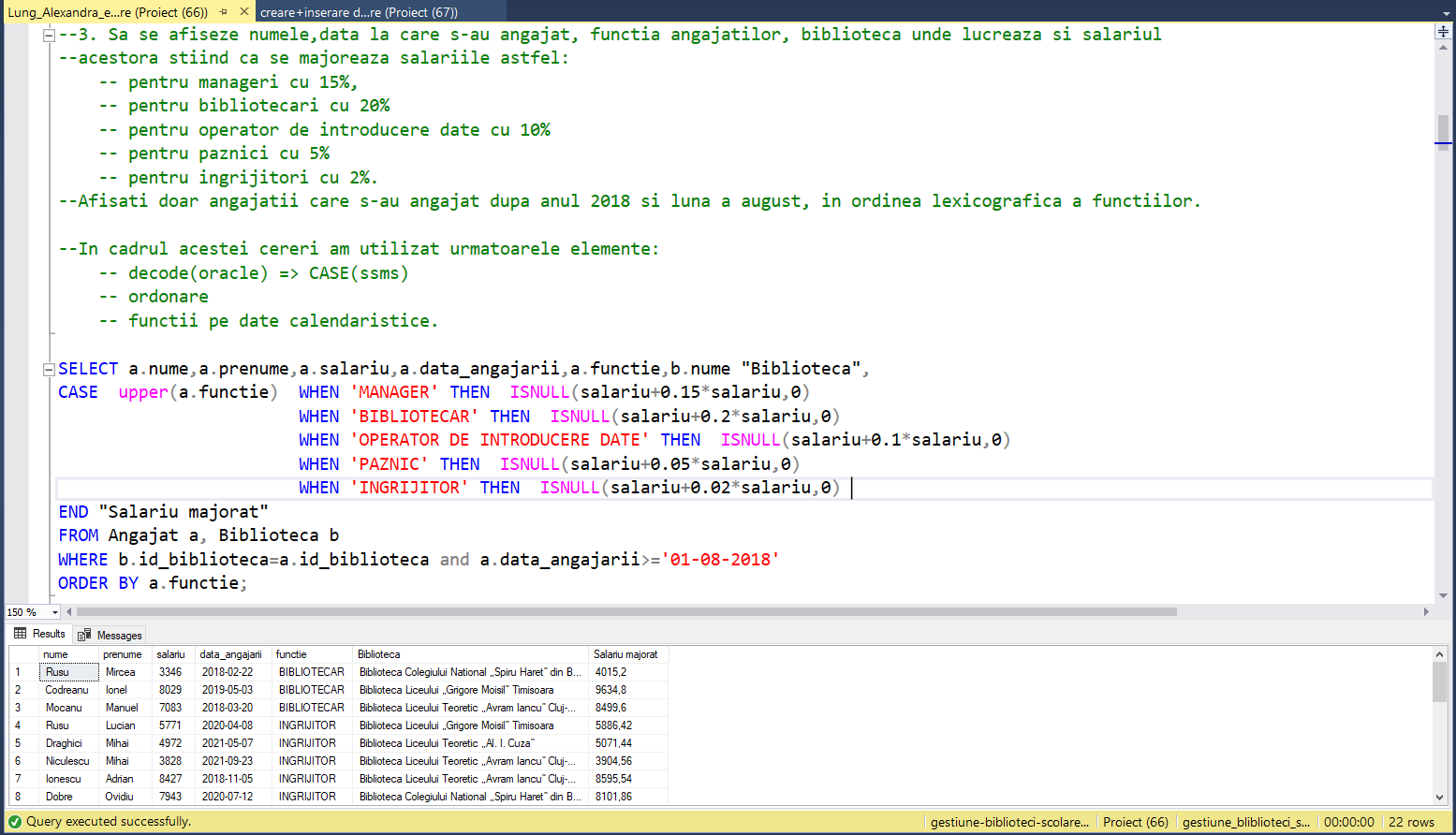
WHEN 'PAZNIC' THEN ISNULL(salariu+0.05\*salariu,0)

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;

1. ***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 si 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"

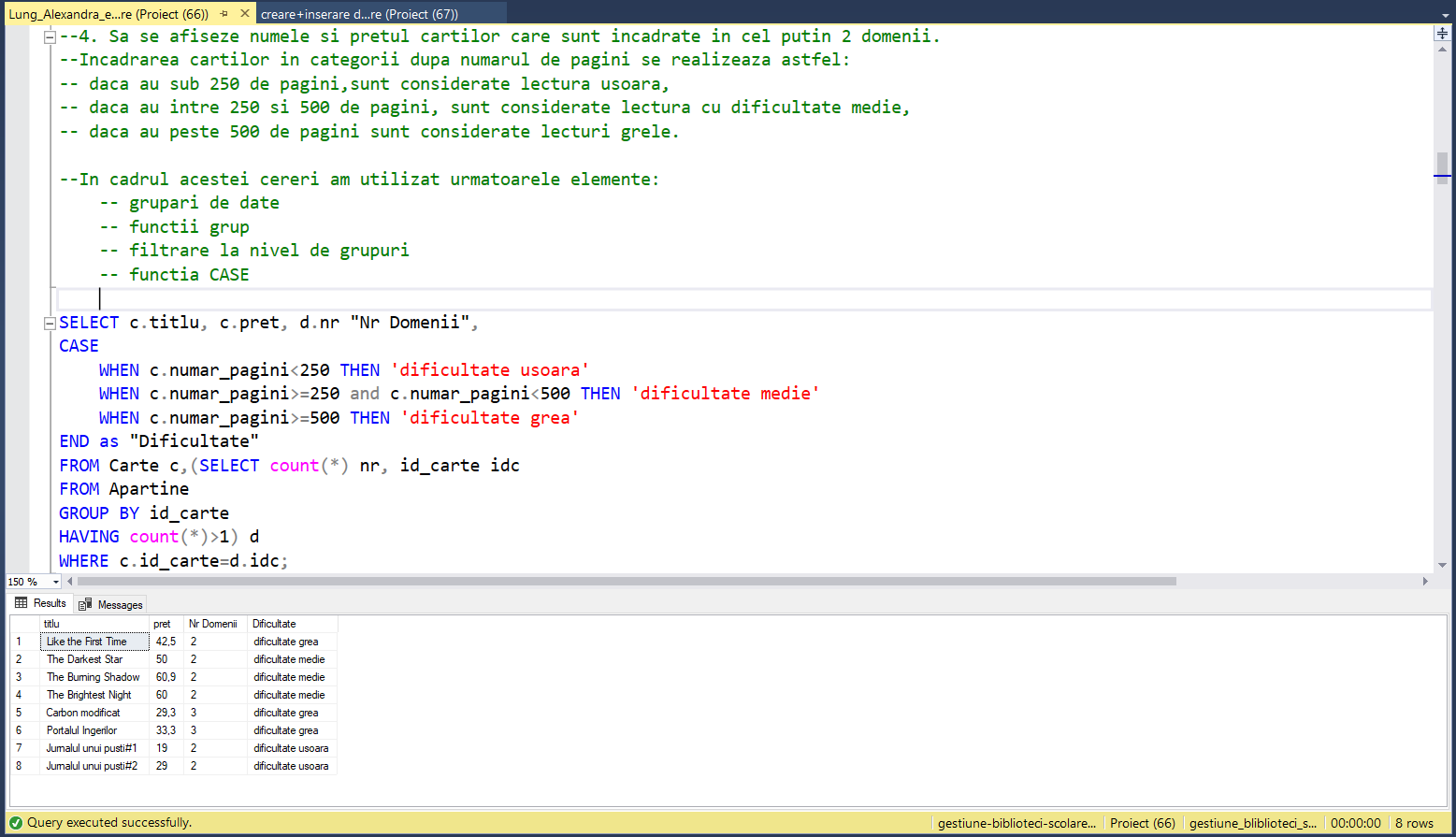
FROM Carte c,(SELECT count(\*) nr, id\_carte idc

FROM Apartine

GROUP BY id\_carte

HAVING count(\*)>1) d

WHERE c.id\_carte=d.idc;



1. ***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

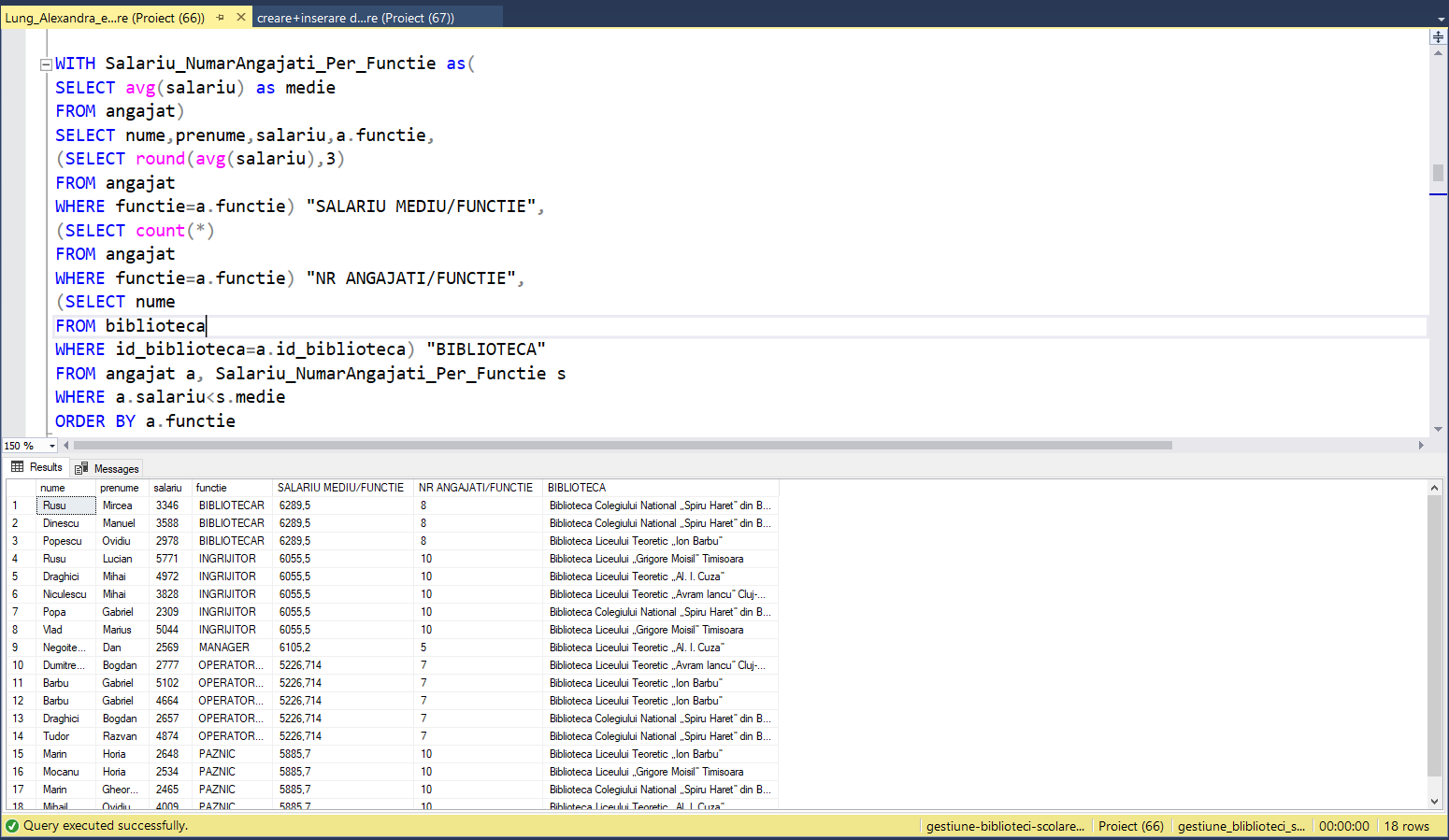
FROM biblioteca

WHERE id\_biblioteca=a.id\_biblioteca) "BIBLIOTECA"

FROM angajat a, Salariu\_NumarAngajati\_Per\_Functie s

WHERE a.salariu<s.medie

ORDER BY a.functie



12. Trei cereri de actualizare și suprimare a datelor

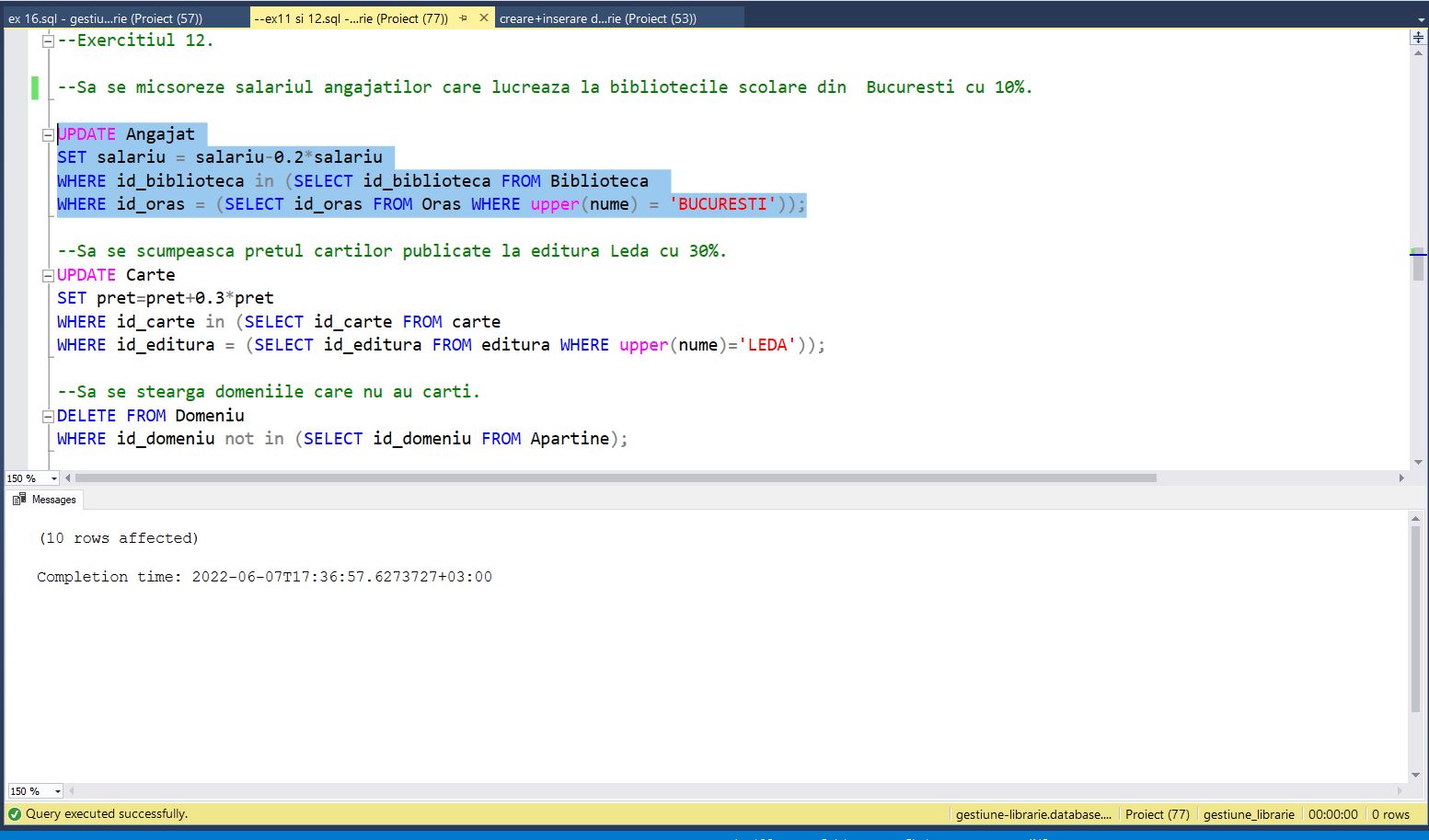
***1.*** *Să se micșoreze salariul angajatilor care lucrează la bibliotecile școlare din București cu 10%.*

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(nume) = 'BUCURESTI'));

******

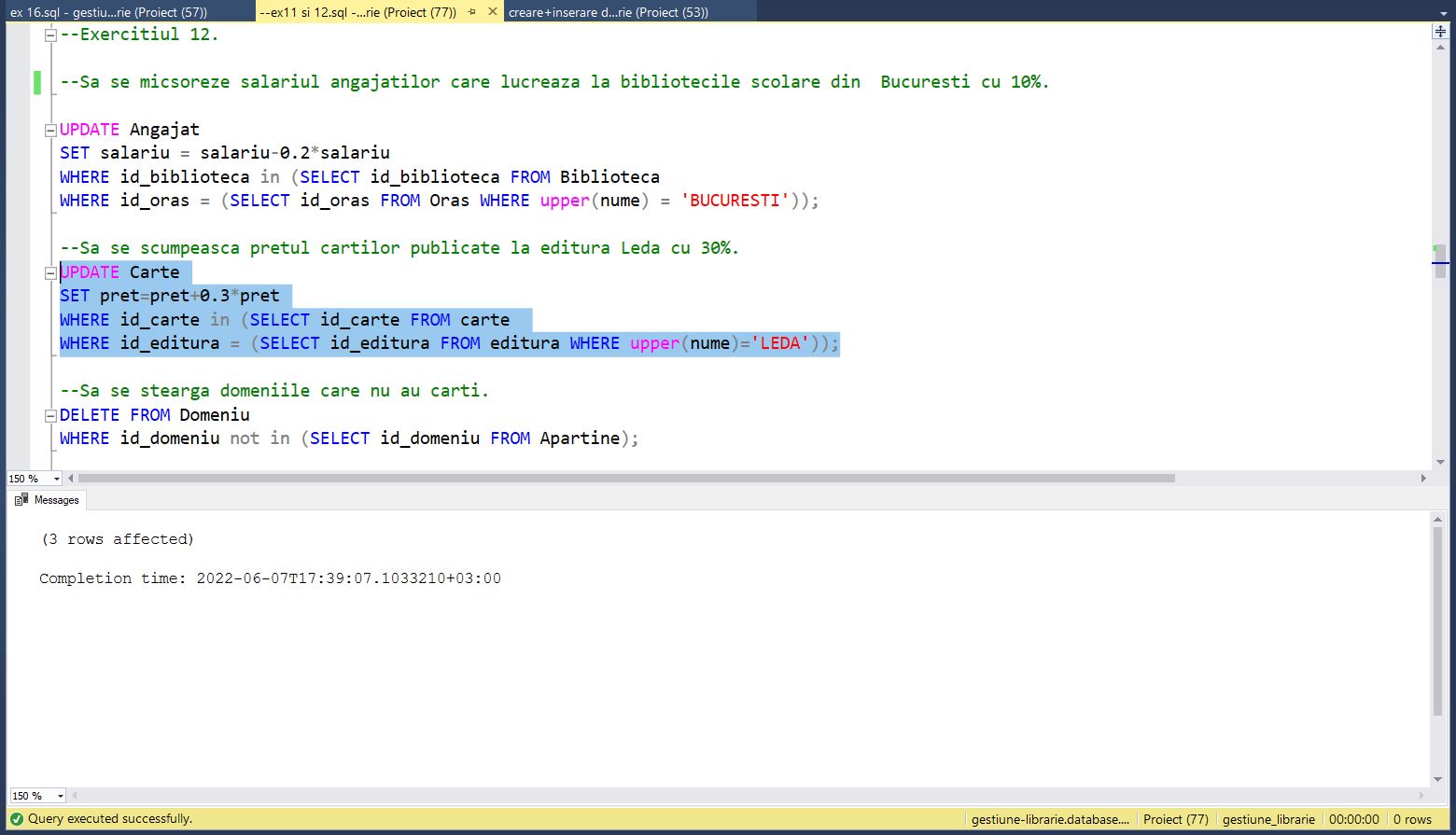
*2.Să se scumpească prețul cărților publicate la editura Leda cu 30%.*

UPDATE Carte

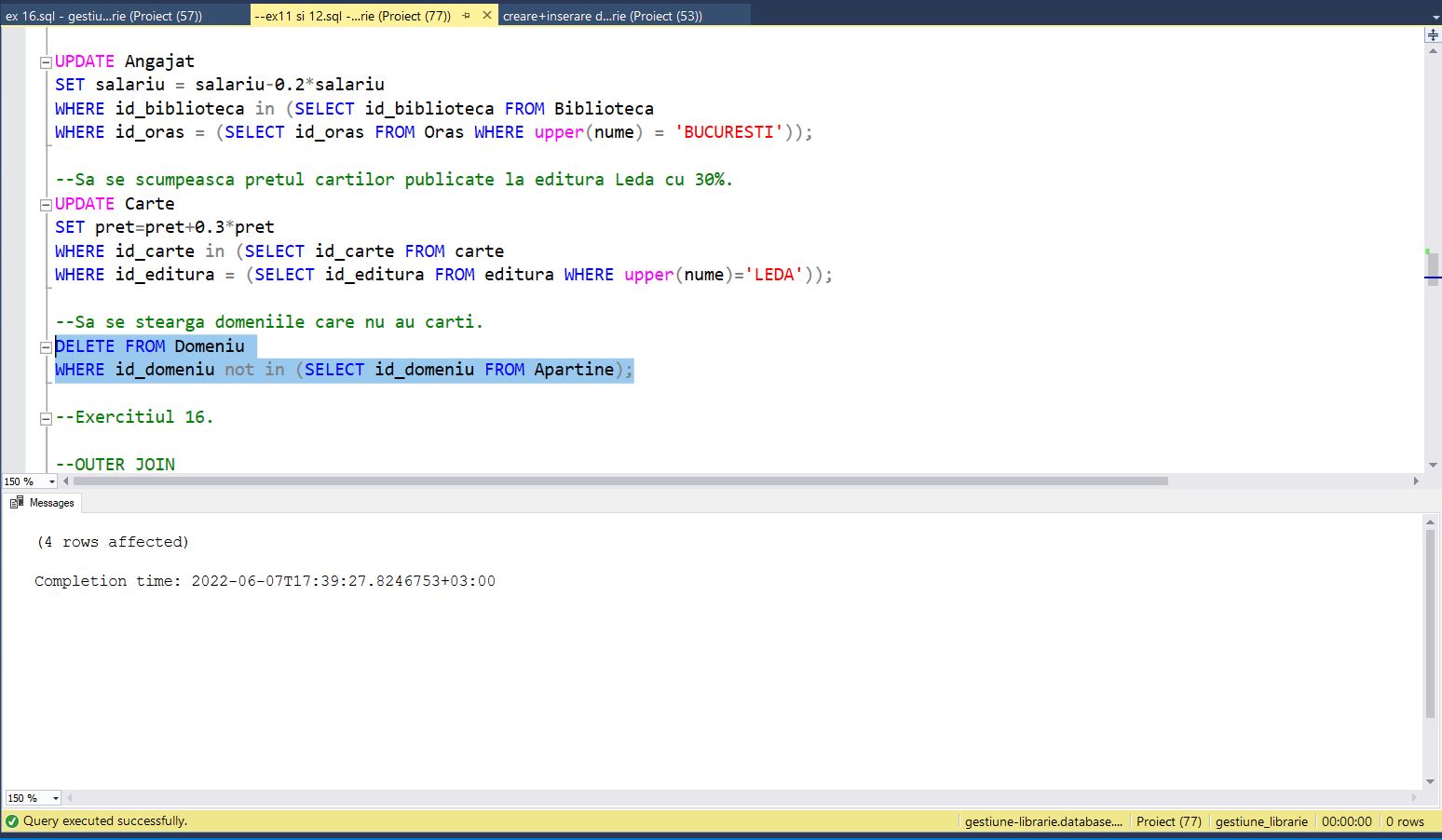
SET pret=pret+0.3\*pret

WHERE id\_carte in (SELECT id\_carte FROM carte

WHERE id\_editura = (SELECT id\_editura FROM editura WHERE upper(nume)='LEDA'));

3. *Să se șteargă domeniile care nu au cărți.*

DELETE FROM Domeniu

WHERE id\_domeniu not in (SELECT id\_domeniu FROM Apartine);

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 şi nu conţine 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 instrucţiuni SELECT în dicţionarul de date.

Vederile se clasifică în două grupe: simple şi complexe. Diferenţa de bază între cele două grupe este legată de operaţiile LMD (inserare, actualizare şi ştergere).

*O vedere simplă este o vedere care:*

* + *furnizează date dintr-un singur tabel;*
  + *nu conţine funcţii sau grupuri de date;*
  + *permite execuţia unor operaţii LMD.*

*O vedere complexă este o vedere care :*

* + *extrage date din mai multe tabele;*
  + *conţine funcţii sau grupuri de date;*
  + *nu permite întotdeauna operaţii LMD.*
  1. ***Crearea vizualizării***

GO

CREATE VIEW sal\_per\_biblioteci(name, minsal, maxsal, avgsal)

AS SELECT d.nume, MIN(e.salariu), MAX(e.salariu), AVG(e.salariu)

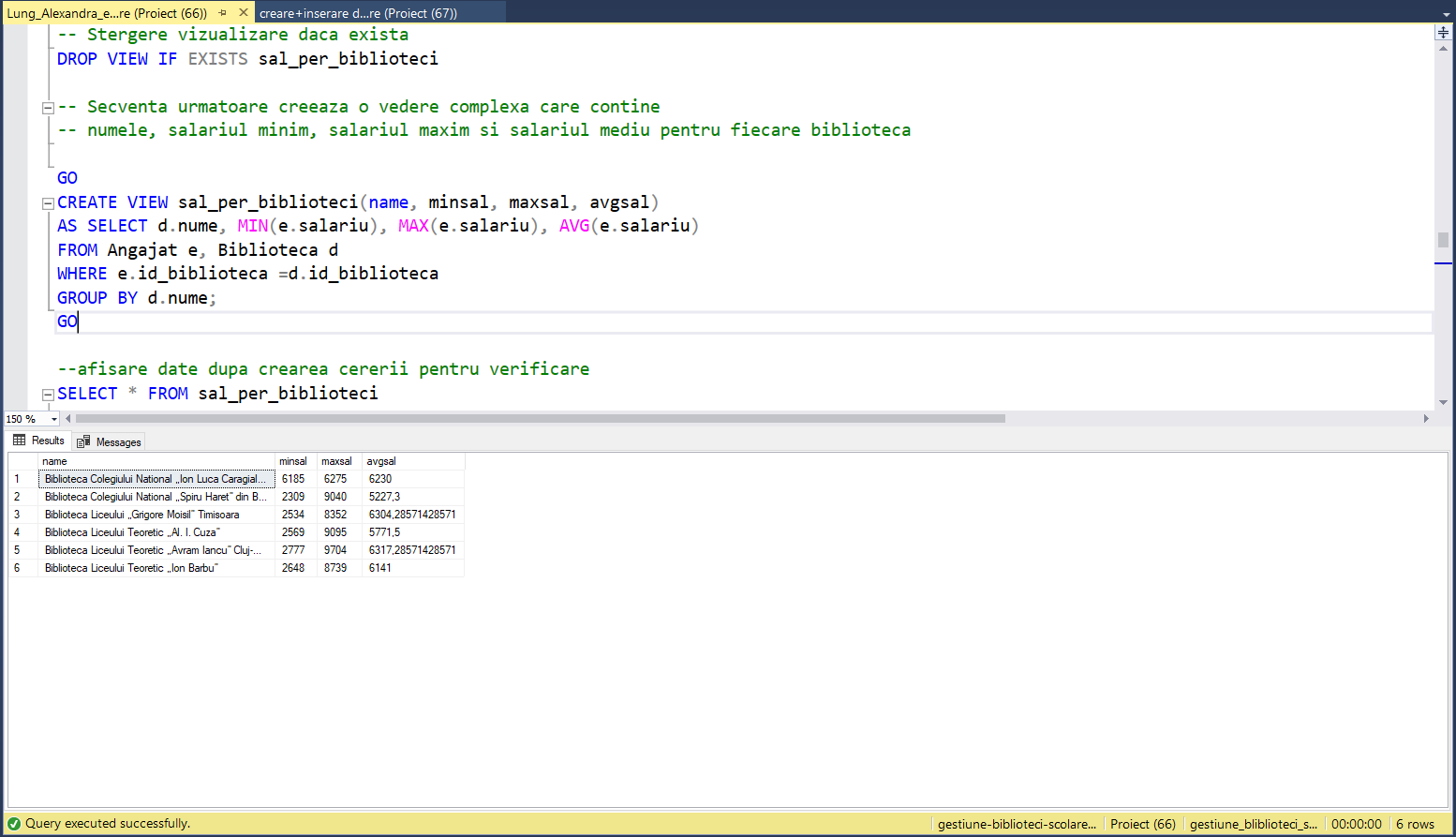
FROM Angajat e, Biblioteca d

WHERE e.id\_biblioteca =d.id\_biblioteca

GROUP BY d.nume;

GO

*Secvența anterioară creează o vedere complexă care conţine numele, salariul minim, salariul maxim şi 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 funcţii sau expresii.*

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

1. Nu se poate şterge o linie aparţinând unei vederi dacă definiţia vederii conţine :

* + funcţii grup;
  + clauză GROUP BY;
  + cuvântul cheie DISTINCT.

2. Nu se pot modifica datele dintr-o vedere dacă vederea conţine:

* + oricare din elementele de la punctul 1;
  + coloane definite prin expresii (de exemplu, SALARY\*12);
  + pseudocoloana ROWNUM.

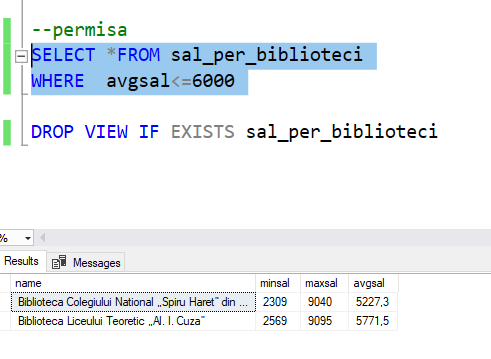
3. Nu pot fi inserate date într-o vedere dacă:

* + vederea conţine oricare din elementele de la punctul 1 şi 2;
  + există coloane NOT NULL fără valoare implicită în tabelul de bază şi care nu au fost selectate în vedere.
  1. ***Operații LMD permise***

SELECT \*FROM sal\_per\_biblioteci

WHERE avgsal<=6000

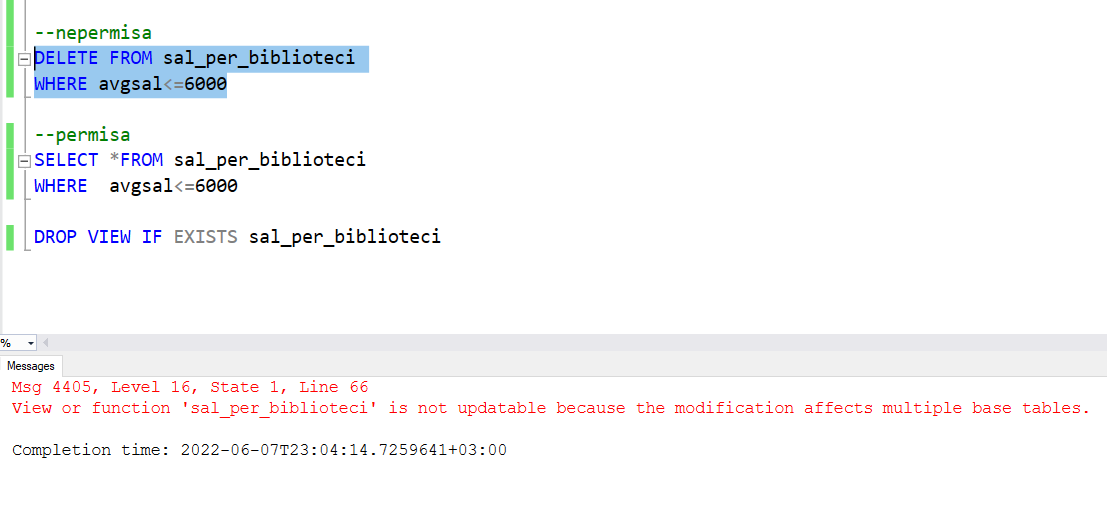
*Secvența anterioară selectează din vederea complexă bibliotecile care au media salariului angajaților mai mică ca 6000 de lei*



* 1. ***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*

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

--neoptimizat

GO

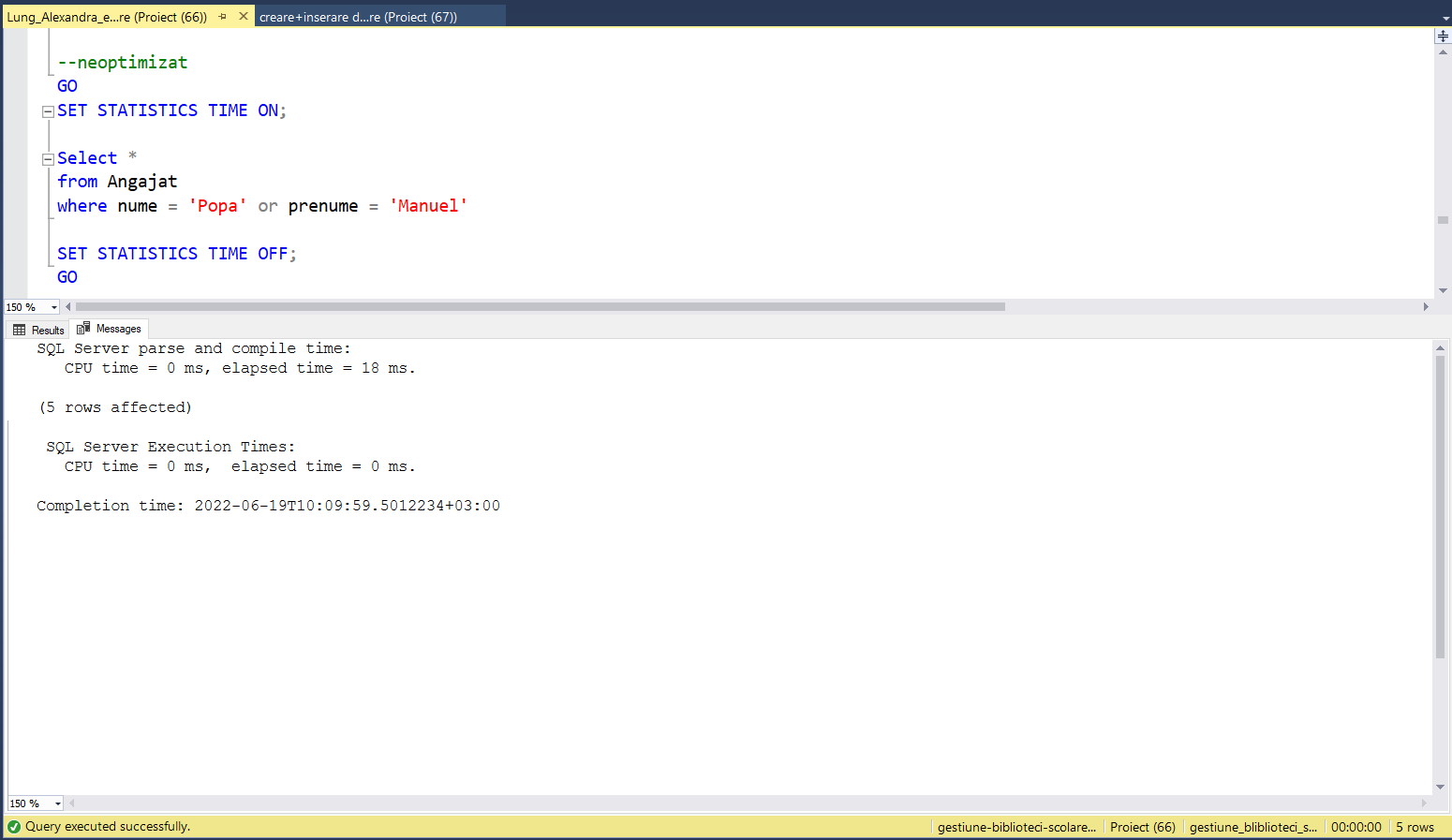
SET STATISTICS TIME ON;

Select \*

from Angajat

where nume = 'Popa' or prenume = 'Manuel'

SET STATISTICS TIME OFF;

GO

--optimizat

CREATE INDEX ang\_num\_pren\_index

ON Angajat (nume, prenume);

GO

SET STATISTICS TIME ON;

Select \*

from Angajat

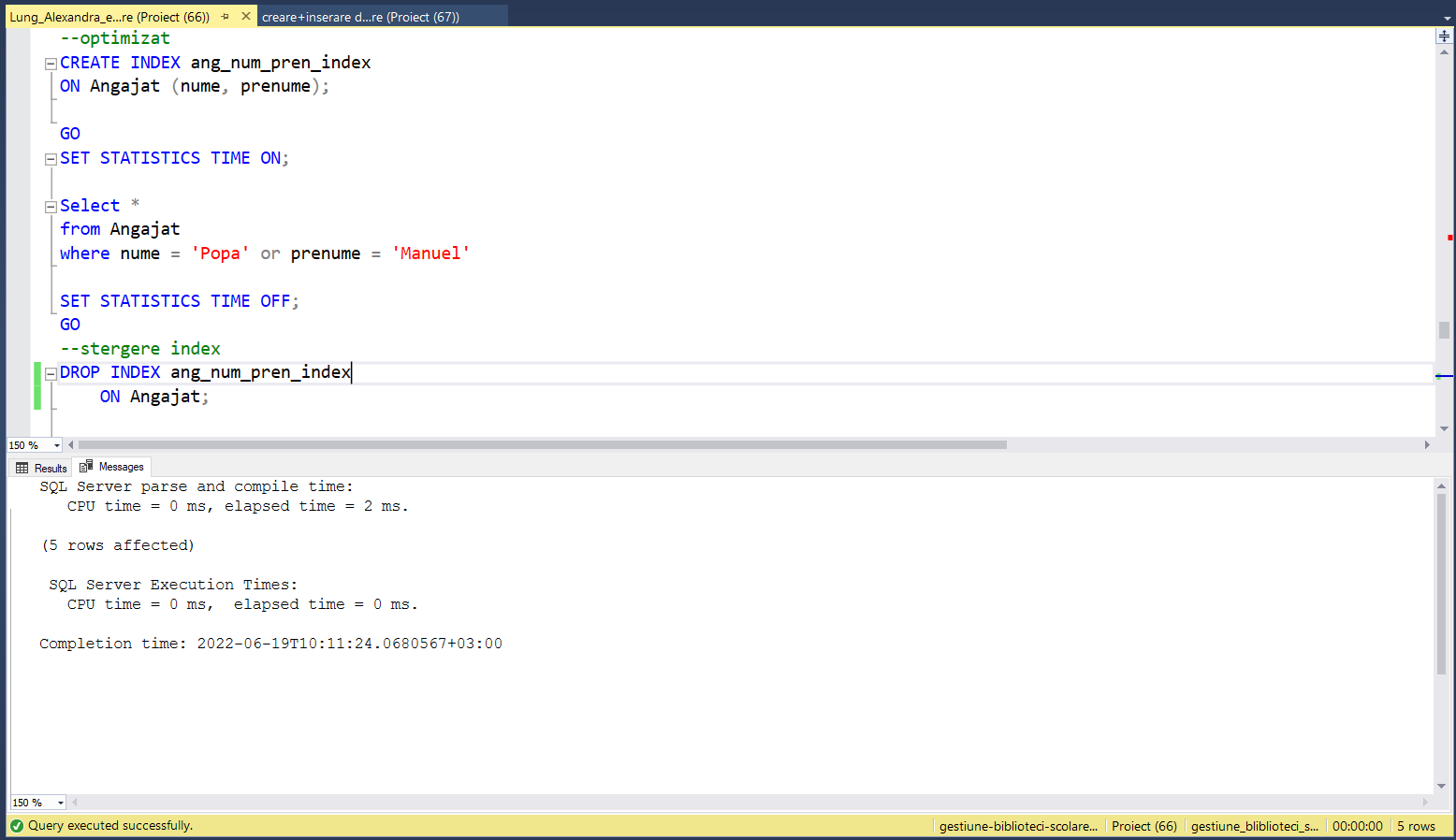
where nume = 'Popa' or prenume = 'Manuel'

SET STATISTICS TIME OFF;

GO

--stergere index

DROP INDEX ang\_num\_pren\_index

 ON Angajat;

16. Cereri SQL care utilizează outer-join și division

1. *Outer-join*

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

SELECT e.nume+' '+e.prenume Elev, c1.titlu Carte, ed.nume EDITURA,a.nume+' '+a.prenume Autor

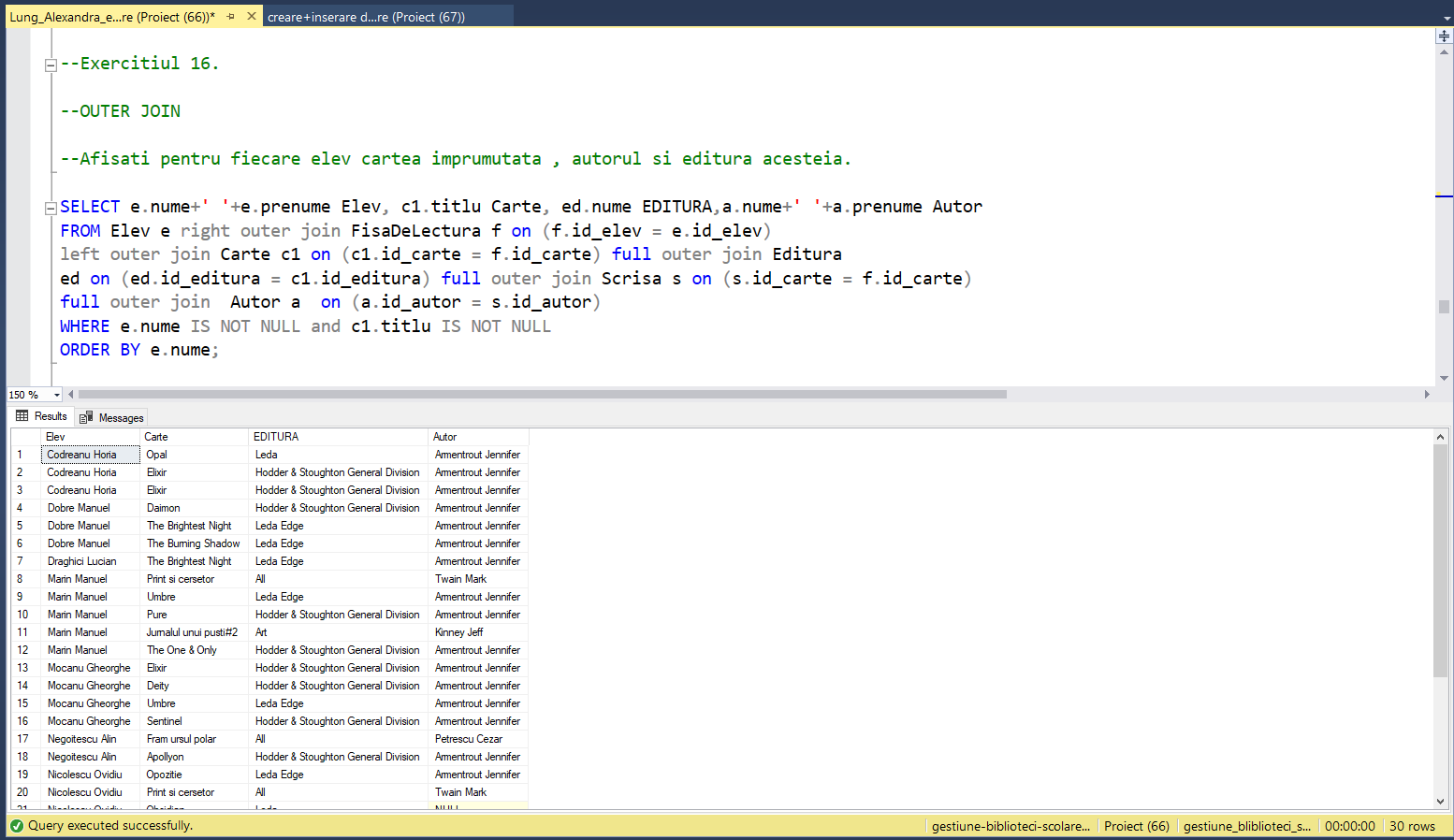
FROM Elev e right outer join FisaDeLectura f on (f.id\_elev = e.id\_elev)

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.nume IS NOT NULL and c1.titlu IS NOT NULL

ORDER BY e.nume;

1. *Division*

*Să se afișeze id-ul, numele și prenumele elevilor,titlul și prețul cărțiilor 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)

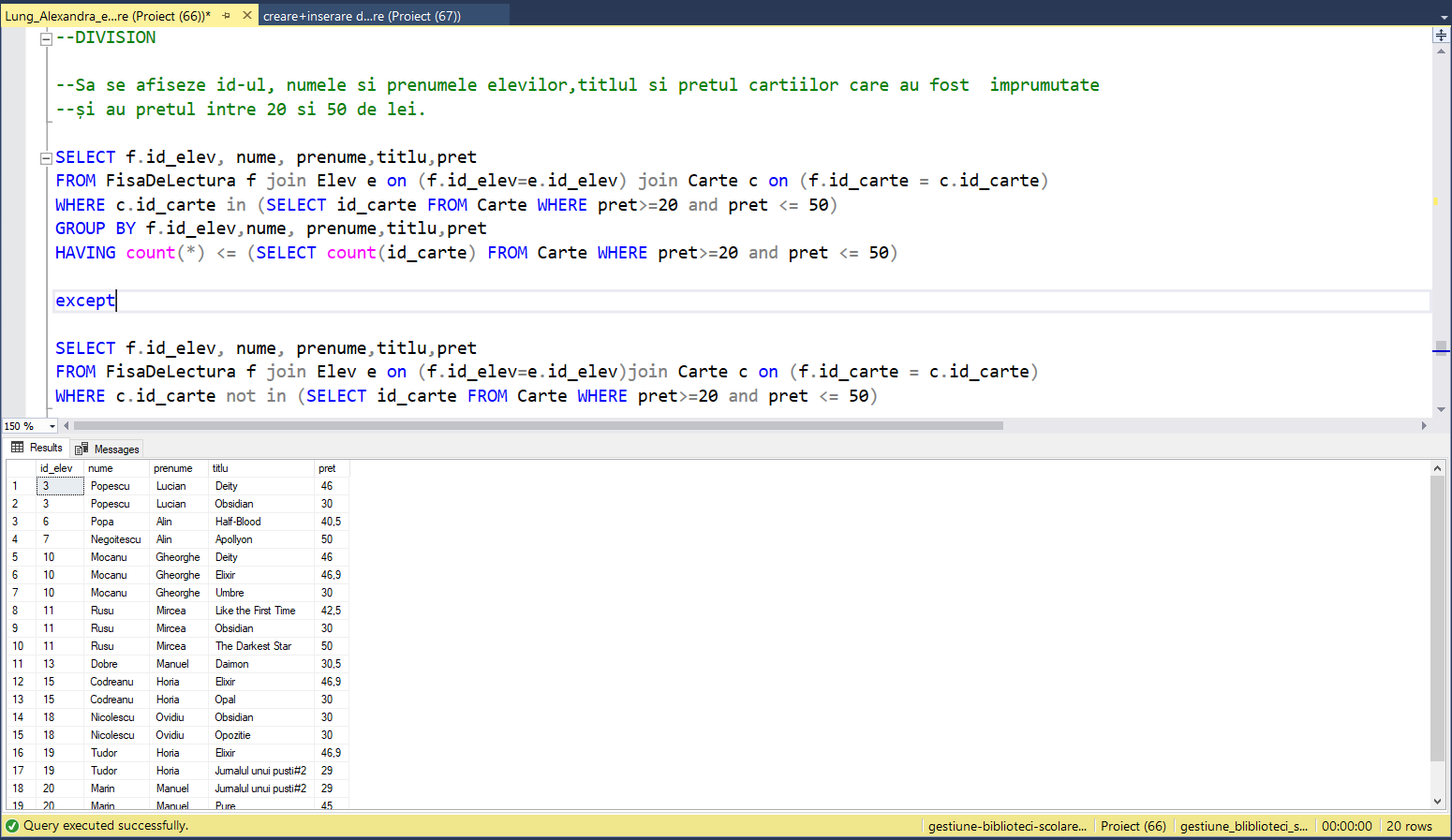
GROUP BY f.id\_elev,nume, prenume,titlu,pret

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)

*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

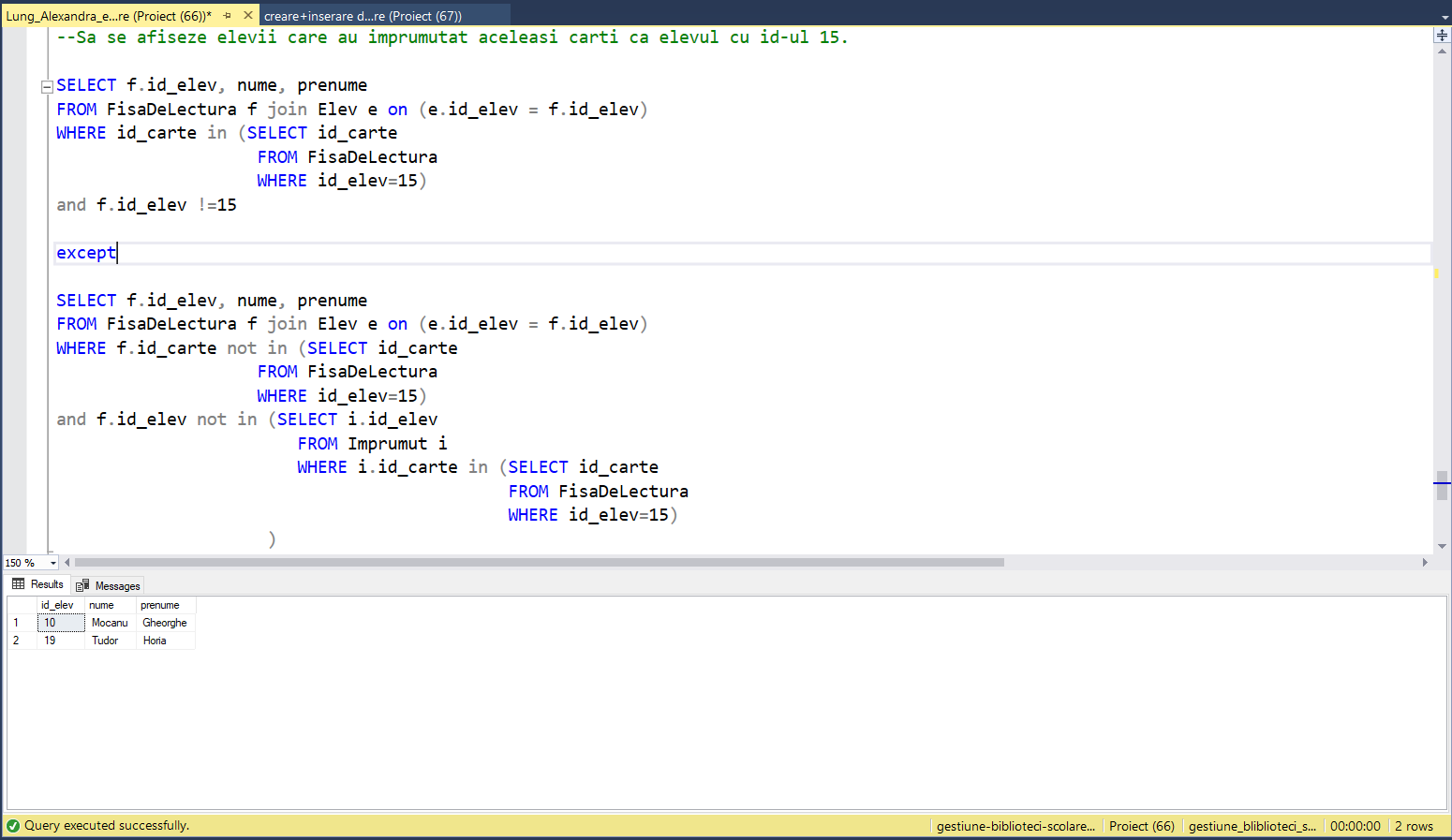
SELECT f.id\_elev, nume, prenume

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))

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.nume)= '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:*

* *Selecţiile se execută cât mai devreme posibil. Motivaţia acestei reguli este că selecţiile reduc substanţial dimensiunea relaţiilor.*
* *Produsurile carteziene se înlocuiesc cu joinuri, ori de câte ori este posibil. Un produs cartezian între două relaţii este de obicei mult mai scump (ca şi cost) decât un join între cele două relaţii, deoarece primul generează concatenarea tuplurilor în mod exhaustiv şi poate genera un rezultat foarte mare.*
* *Un join este mai restrictiv decât altul dacă produce o relaţie mai mică. Se poate determina care join este mai restrictiv pe baza factorului de selectivitate sau cu ajutorul informaţiilor statistice.*
* *Proiecţiile se execută la început pentru a îndepărta atributele nefolositoare. Dacă un atribut al unei relaţii nu este folosit în operaţiile ulterioare atunci trebuie îndepărtat. (am folosit semijoin-uri,eliminând astfel atributele nefolositoare)*

***Arbore algebric***

Rezultat

Titlu

seria

limba

id\_autor

nume = ’Armentrout’

id\_carte

titlu

serie

limba

id\_editura = 2

id\_autor

id\_carte

Scriere Carte

Autor

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 legatură cu acesta.*

1. ***BCNF***

*Determinantul este un atribut sau o mulţime de atribute neredundante, care constituie un identificator unic pentru alt atribut sau altă mulţime de atribute ale unei relaţii date.*

*Intuitiv, o relaţie R este în forma normală Boyce-Codd dacă şi numai dacă fiecare determinant este o cheie candidat.*

*Formal, o relaţie R este în forma normală Boyce-Codd dacă şi numai dacă pentru orice dependenţă funcţională totală X → A, X este o cheie (candidat) a lui R.*

*Regula Casey Delobel pentru R(K1#, K2#, X) presupunând că există dependenţa: X → K2. R1(K1#, X) şi 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* |

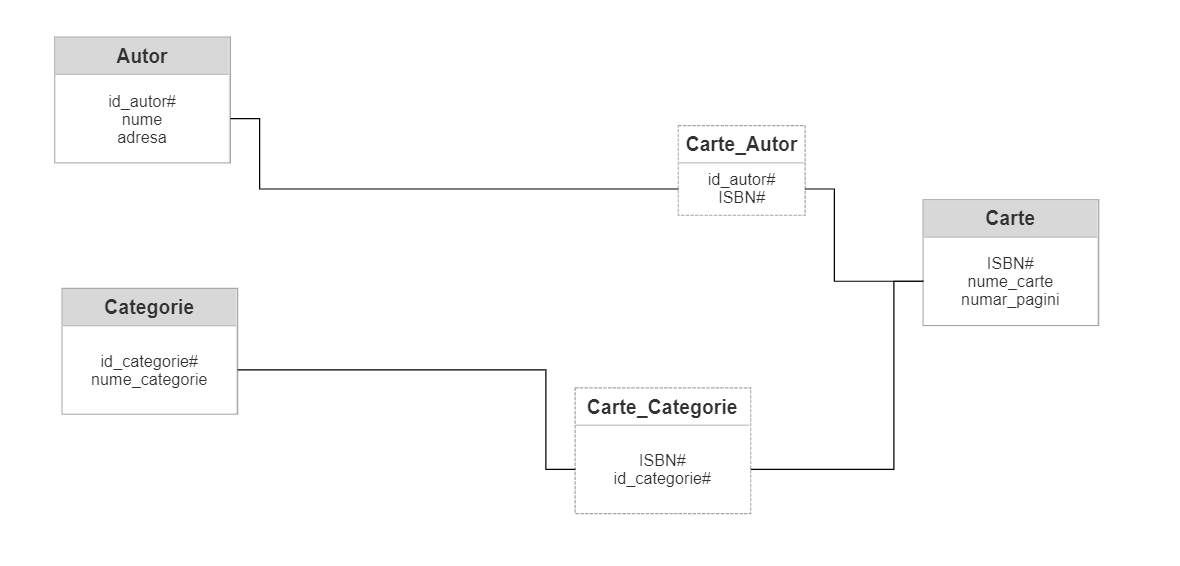
*Exemplu Non-BCNF*

*{nume\_carte,id\_categorie}->{autor} =>(x,y)-> z*

*{autor}->{ nume\_carte } =>z-> 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*

1. ***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 redundanţele datorate relaţiilor m:n, adică datorate dependenţei multiple. Intuitiv, o relaţie R este în a patra formă normală dacă şi numai dacă relaţia este în BCNF şi nu conţine relaţii 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(angajat#,biblioteca#,funcție#)*

*angajat ->biblioteca*

*angajat->funcție*

|  |  |  |
| --- | --- | --- |
| *angajat#* | *biblioteca#* | *funcție#* |
| *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:*

* *R1(angajat#,biblioteca#)*
* *R2(angajat#,funcție#)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *angajat#* | *biblioteca#* |  | *angajat#* | *funcție#* |
| *A1* | *1* |  | *A1* | *Operator de introducere date* |
| *A2* | *2* |  | *A1* | *Bibliotecar* |
| *A2* | *3* |  | *A2* | *Pazinic* |
| *A3* | *2* |  | *A3* | *Manager* |

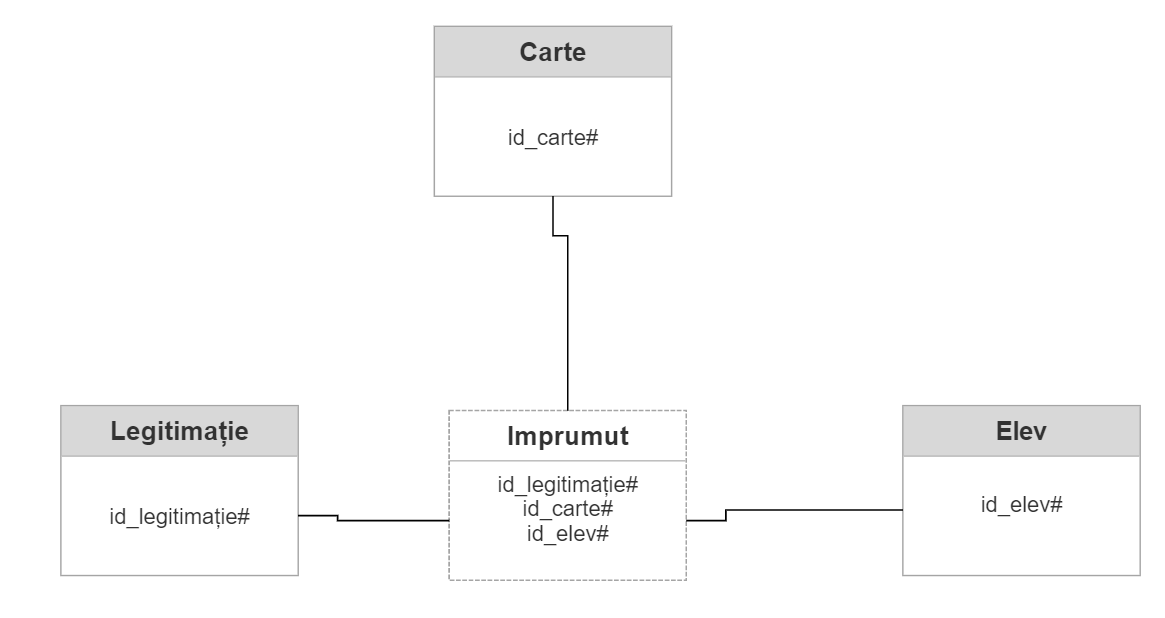
*R1 R2*

1. ***FN5***

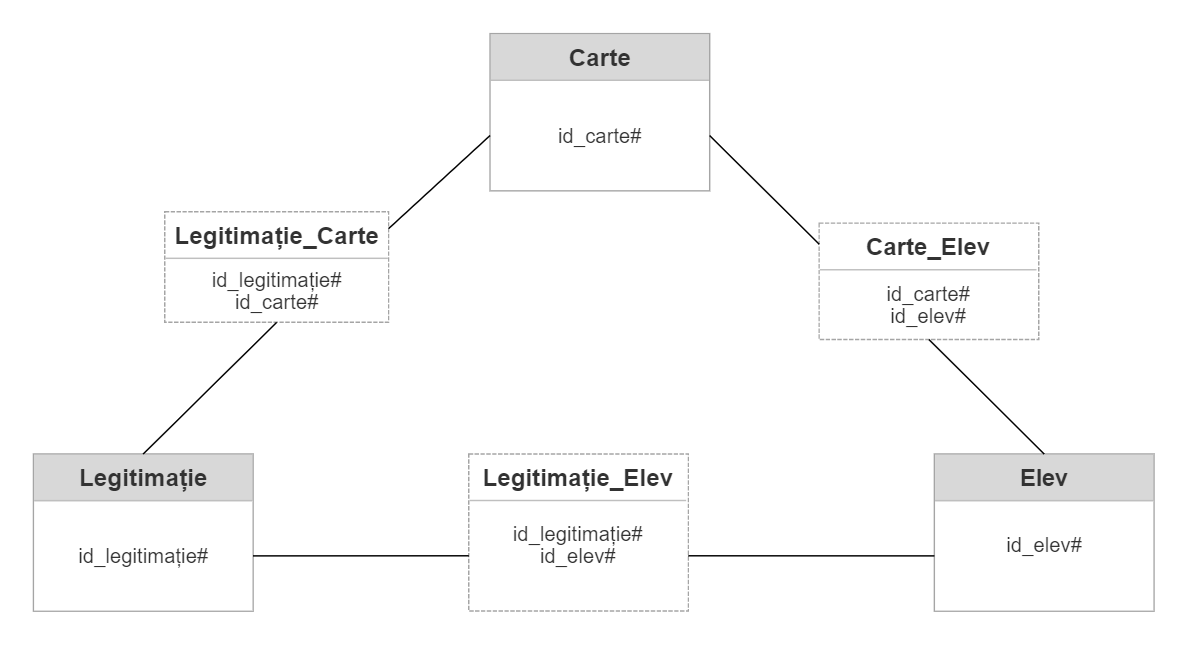
*FN5 îşi propune eliminarea redundanţelor care apar în relaţii m:n dependente. În general, aceste relaţii nu pot fi descompuse. S-a arătat că o relaţie de tip 3 este diferită de trei relaţii de tip 2. Există totuşi o excepţie, şi anume, dacă relaţia este ciclică*

*Intuitiv, o relaţie R este în forma normală 5 dacă şi numai dacă:*

* 1. *relaţia este în FN4;*
  2. *nu conţine dependenţe 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 cilică, atunci când se var 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 in FN5. (Fig 1.)*

1. ***Denormalizare***

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

*Fie R = {R1 , R2 , ..., Rp } o mulţime de relaţii Denormalizarea R înseamnă înlocuirea R cu R’ = JOIN(R1 , R2 , ..., Rp ), astfel încât ∀ 1≤i≤p: proiecţia lui R după atributele lui Ri va produce din nou relaţia R.*

*Obiectivul denormalizarii*

* *mărirea redundanţei (relaţia R’ se află la un nivel de normalizare mai scăzut decât relaţiile R1 , R2 , ..., Rp 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 cosidera 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/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>