

BAZE DE DATE

GESTIUNEA BIBLIOTECILOR DIN ROMÂNIA

Mai 2021

Toader Vlad - Marian

Grupa 142

Facultatea de Matematică și Informatică

Universitatea din București

Cuprins

| | | |
|-----------|--|-----------|
| 1 | Descrierea modelului real | 4 |
| 2 | Constrângeri | 6 |
| 3 | Entități | 7 |
| 4 | Relații | 8 |
| 5 | Atribute | 9 |
| 6 | Diagrama entitate-relație | 14 |
| 7 | Diagrama conceptuală | 15 |
| 8 | Scheme relaționale | 15 |
| 9 | Realizarea normalizării (FN1-FN3) | 16 |
| 9.1 | Forma normală 1 - FN1 | 16 |
| 9.2 | Forma normală 2 - FN2 | 17 |
| 9.3 | Forma normală 3 - FN3 | 18 |
| 10 | Crearea tabelelor în SQL și inserarea de date | 19 |
| 10.1 | Creare și inserare în tabela <i>AUTOR</i> | 19 |
| 10.2 | Creare și inserare în tabela <i>EDITURA</i> | 20 |
| 10.3 | Creare și inserare în tabela <i>DOMENIU</i> | 21 |
| 10.4 | Creare și inserare în tabela <i>LOCAȚIE</i> | 21 |
| 10.5 | Creare și inserare în tabela <i>CITITOR</i> | 22 |
| 10.6 | Creare și inserare în tabela <i>CARTE</i> | 24 |
| 10.7 | Creare și inserare în tabela <i>BIBLIOTECA</i> | 26 |
| 10.8 | Creare și inserare în tabela <i>APARTINE</i> | 27 |
| 10.9 | Creare și inserare în tabela <i>SCRIERE</i> | 29 |
| 10.10 | Creare și inserare în tabela <i>ANGAJAT</i> | 31 |

| | | |
|-----------|--|-----------|
| 10.11 | Creare și inserare în tabela <i>ABONAMENT</i> | 32 |
| 10.12 | Creare și inserare în tabela <i>IMPRUMUT</i> | 34 |
| 10.13 | Creare și inserare în tabela <i>ISTORIC</i> | 35 |
| 10.14 | Adăguarea constrângerilor <i>not null</i> tabelelor | 39 |
| 11 | Cereri SQL | 39 |
| 11.1 | Cererea nr. 1 | 39 |
| 11.2 | Cererea nr. 2 | 41 |
| 11.3 | Cererea nr. 3 | 42 |
| 11.4 | Cererea nr. 4 | 43 |
| 11.5 | Cererea nr. 5 | 45 |
| 12 | Cereri de actualizare și suprimare a datelor | 46 |
| 13 | Crearea unei secvențe folosită la inserarea datelor în tabele | 49 |
| 14 | Cereri SQL folosind operațiile <i>outer-join</i> și <i>division</i> | 50 |
| 14.1 | Outer-join | 50 |
| 14.2 | Division | 51 |
| 15 | Optimizarea unei cereri | 53 |
| 15.1 | Cerere SQL | 53 |
| 15.2 | Expresii algebrice | 53 |
| 15.3 | Optimizare | 53 |
| 15.4 | Arbore algebric | 54 |
| 16 | Normalizarea BCNF, FN4, FN5. Aplicarea denormalizării | 55 |
| 16.1 | Forma normală Boyce-Codd | 55 |
| 16.2 | Forma normală 4 - FN4 | 56 |
| 16.3 | Forma normală 5 - FN5 | 58 |
| 16.4 | Aplicarea denormalizării | 59 |

1 Descrierea modelului real

Modelul de date va gestiona informații legate de gestionarea bibliotecilor din România. Astfel, pot exista mai multe biblioteci, aflate la diverse locații, atât în cadrul unităților de învățământ, dar pot fi și biblioteci municipale. În cadrul bibliotecilor lucrează angajați, care pot fi manageri, bibliotecari, paznici sau îngrijitori.

De asemenea, în cadrul bazei de date se ține evidența cititorilor fiecărei biblioteci, reținând informații despre împrumuturile efectuate de aceștia. Se rețin și informații despre cărțile disponibile, despre autorii acestora, despre edituri, cât și despre domeniile de care aparțin acestea. Pentru un cititor, se rețin și date despre abonamentele acestuia, dar și despre istoricul cărților.

Utilitatea modelului de date proiectat

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

Reguli de funcționare

Modelul de date respectă anumite restricții de funcționare.

- O bibliotecă poate avea mai mulți angajați sau, în cazul în care aceasta a fost închisă dar încă există în baza de date, niciunul.
- Un angajat lucrează în cadrul unei singuri biblioteci.
- O bibliotecă poate avea mai multe cărți sau, dacă a fost deschisă recent, niciuna.
- O carte este înregistrată la o singură bibliotecă.

- O carte poate aparține mai multor domenii, însă trebuie ca aceasta să aparțină cel puțin unuia.
- Un domeniu poate avea mai multe cărți, sau poate să nu aibă niciuna.
- O carte poate fi publicată de o singură editură.
- O editură poate avea mai multe cărți publicate, însă trebuie să publice cel puțin una.
- Un autor poate publica mai multe cărți, însă, pentru a fi reținut în baza de date, trebuie să fi publicat cel puțin o carte.
- O carte poate fi scrisă de cel puțin un autor.
- O bibliotecă se poate afla într-o singură locație.
- O locație poate găzdui mai multe biblioteci.
- O carte are mai mulți istorici sau poate să nu aibă niciunul, dacă nu a fost împrumutată niciodată.
- Un istoric aparține unei singure cărți.
- Un istoric are un singur cititor.
- Un cititor poate avea mai mulți istorici sau poate să nu aibă niciunul, dacă acesta nu a împrumutat nicio carte.
- Un abonament aparține unui singur cititor.
- Un cititor poate avea mai multe abonamente sau poate să nu aibă niciunul.
- Un cititor poate împrumuta mai multe cărți din biblioteci diferite.

2 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 singuri biblioteci.
- O bibliotecă poate avea mai multe cărți sau niciuna.
- O carte este înregistrată la o singură bibliotecă.
- O carte poate aparține 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 sau cel puțin una.
- Un autor poate publica mai multe cărți sau cel puțin una.
- O carte poate fi scrisă de cel puțin un autor.
- O bibliotecă se poate afla într-o singură locație.
- O locație poate găzdui mai multe biblioteci.
- O carte are mai mulți istorici sau niciunul.
- Un istoric aparține unei singure cărți.
- Un istoric are un singur cititor.
- Un cititor poate avea mai mulți istorici sau niciunul.
- Un abonament aparține unui singur cititor.
- Un cititor poate avea mai multe abonamente sau niciunul.
- Un cititor poate împrumuta mai multe cărți din biblioteci diferite.

3 Entități

Pentru modelul de date proiectat, structurile **LOCAȚIE**, **BIBLIOTECĂ**, **ABONAMENT**, **ANGAJAT**, **CITITOR**, **CARTE**, **AUTOR**, **EDITURĂ**, **DOMENIU**, **ISTORIC** reprezintă entități.

Vom prezenta entitățile modelului de date, dând o descriere completă a fiecăreia. De asemenea, pentru fiecare entitate se va preciza cheia primară.

LOCAȚIE = o locație din România la care se află una sau mai multe biblioteci, și care poate fi și o unitate de învățământ, caz în care biblioteca de la locația respectivă este școlară. Cheia primară a entității este *id_locație#*.

BIBLIOTECĂ = entitate care conține informații despre o bibliotecă. Cheia primară a entității este *id_bibliotecă#*.

ABONAMENT = reține date despre abonamentul unui cititor, înscris la o bibliotecă. Cheia primară a entității este *id_abonament#*.

ANGAJAT = entitate care referă un angajat al unei biblioteci, reținând date relevante despre acesta (nume, prenume, funcția, salariu). Cheia primară a entității este *id_angajat#*.

CITITOR = reține date despre un cititor al unei biblioteci. Cheia primară a entității este *id_cititor#*.

CARTE = entitate care referă o carte aflată într-o bibliotecă. Cheia primară a entității este *id_carte#*.

AUTOR = o entitate care reține date despre o persoană care a publicat cel puțin o carte care se găsește într-una din biblioteci. Cheia primară a entității este *id_autor#*.

EDITURĂ = entitate care referă o editură (o instituție care editează cărți). Cheia primară a entității este *id_editură#*.

DOMENIU = o entitate în care se rețin date despre un anumit domeniu în care se pot încadra cărțile dintr-o bibliotecă. Cheia primară a entității este *id_domeniu#*.

ISTORIC = o entitate în care se rețin date referitoare la istoricul unei cărți: de câte ori a fost împrumutată, ce condiție are cartea. Cheia primară a

entității este *id_istoric#*.

4 Relații

Vom prezenta relațiile modelului de date, precizând pentru fiecare și cardinalitatea minimă și maximă.

BIBLIOTECA_are_ANGAJAT = relație care leagă entitățile BIBLIOTECĂ și ANGAJAT, reflectând legătura dintre acestea (ce angajat lucrează la o bibliotecă). Ea are cardinalitatea minimă 1:0 și cardinalitatea maximă 1:n.

LOCAȚIA_găzduiește_BIBLIOTECA = relație care leagă entitățile LOCAȚIE și BIBLIOTECĂ, reflectând legătura dintre acestea (o locație găzduiește o bibliotecă). Ea are cardinalitatea minimă 1:1 și maximă 1:n.

CARTEA_apartine_DOMENIULUI = relație care leagă entitățile CARTE și DOMENIU, reflectând legătura dintre acestea (o carte aparține unui domeniu). Ea are cardinalitatea minimă 0:1 și maximă m:n.

EDITURA_publică_CARTE = relație care leagă entitățile EDITURĂ și CARTE, reflectând legătura dintre acestea (o editură publică o carte). Ea are cardinalitatea minimă 1:1 și maximă 1:n.

AUTORUL_scrie_CARTE = relație care leagă entitățile AUTOR și CARTE, reflectând legătura dintre acestea (un autor scrie o carte). Ea are cardinalitatea minimă 1:1 și maximă m:n.

CARTEA_are_ISTORIC = relație care leagă entitățile CARTE și ISTORIC, reflectând legătura dintre acestea (o carte are un istoric). Ea are cardinalitatea minimă 1:0 și maximă 1:n.

AUTORUL_scrie_CARTE = relație care leagă entitățile AUTOR și CARTE, reflectând legătura dintre acestea (un autor scrie o carte). Ea are cardinalitatea minimă 1:1 și maximă m:n.

CITITORUL_are_ISTORIC = relație care leagă entitățile CITITOR și ISTORIC, reflectând legătura dintre acestea (un cititor are un istoric). Ea

are cardinalitatea minimă 1:0 și maximă 1:n.

Relația de tip 3 dintre entitățile CITITOR, BIBLIOTECĂ și CARTE - reflectă legătura dintre cele 4 tabele (Un cititor poate împrumuta mai multe cărți din biblioteci diferite).

5 Atribute

Entitatea **EDITURĂ** are ca atribute:

id_editura = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unei edituri.

nume = variabilă de tip șir de caractere, de lungime maximă 25, care reprezintă numele unei edituri, și care nu poate fi nulă,

telefon = variabilă de tip șir de caractere, de lungime 15, care reprezintă numărul de telefon prin care se poate contacta editura.

Entitatea **DOMENIU** are ca atribute:

id_domeniu = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unui domeniu.

nume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă numele unui domeniu, și care nu poate fi nulă.

Entitatea **AUTOR** are ca atribute:

id_autor = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unui autor.

nume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă numele unui autor, și care nu poate fi nulă.

prenume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă prenumele unui autor, și care nu poate fi nulă.

data_nasterii = variabilă de tip dată calendaristică, care reprezintă data nașterii autorului.

Entitatea **CARTE** are ca atribute:

id_carte = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unei cărți.

nume = variabilă de tip șir de caractere, de lungime maximă 40, care reprezintă numele unei cărți, și care nu poate fi nulă.

id_editura = variabilă de tip întreg, care reprezintă id-ul editurii care a publicat cartea, și care nu poate fi nulă. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul EDITURĂ.

numar_pagini = variabilă de tip întreg, care reprezintă numărul de pagini al cărții.

pret = variabilă de tip real, care reprezintă prețul cărții.

Entitatea **CITITOR** are ca atribute:

id_cititor = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unui cititor.

nume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă numele unui cititor, și care nu poate fi nulă.

prenume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă prenumele unui cititor, și care nu poate fi nulă.

telefon = variabilă de tip șir de caractere, de lungime 15, care reprezintă numărul de telefon al cititorului.

Entitatea **ABONAMENT** are ca atribute:

id_abonament = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unui abonament.

id_cititor = variabilă de tip întreg, care reprezintă id-ul cititorului căruia îi corespunde abonamentul, și care nu poate fi nulă. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CITITOR.

pret = variabilă de tip real, care reprezintă prețul abonamentului.

data_expirarii = variabilă de tip dată calendaristică, care reprezintă data expirării abonamentului.

Entitatea **ANGAJAT** are ca atribute:

id_angajat = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unui angajat.

nume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă numele angajatului, și care nu poate fi nulă.

prenume = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă prenumele angajatului, și care nu poate fi nulă.

salariu = variabilă de tip real, care reprezintă salariul angajatului.

functie = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă funcția ocupată de un angajat. Atributul poate lua valorile *manager*, *bibliotecar*, *paznic*, *îngrijitor*.

data_angajarii = variabilă de tip dată calendaristică, care reprezintă data angajării unui angajat.

id_biblioteca = variabilă de tip întreg, care reprezintă id-ul bibliotecii la care lucrează angajatul, și care nu poate fi nulă. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul BIBLIOTECĂ.

Entitatea **LOCATIE** are ca atribute:

id_locatie = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unei locații.

denumire = variabilă de tip șir de caractere, de lungime maximă 20, și care reprezintă numele unei locații, și care nu poate fi nulă.

Entitatea **BIBLIOTECĂ** are ca atribute:

id_biblioteca = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unei biblioteci.

nume = variabilă de tip șir de caractere, de lungime maximă 20, și care

reprezintă numele unei biblioteci, și care nu poate fi nulă.

adresa = variabilă de tip șir de caractere, de lungime maximă 40, și care reprezintă adresa bibliotecii.

tip_biblioteca = variabilă de tip șir de caractere, de lungime maximă 20, care reprezintă tipul bibliotecii, și care nu poate fi nulă. Atributul poate lua valorile *școlară*, *munincipală*.

id_locatie = variabilă de tip întreg, care reprezintă id-ul locației în care se află biblioteca, și care nu poate fi nulă. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCAȚIE.

Entitatea **ISTORIC** are ca atribute:

id_istoric = variabilă de tip întreg, are nu poate fi nulă, și care reprezintă id-ul unui istoric.

id_carte = variabilă de tip întreg, care reprezintă id-ul cărții pentru care se reține istoricul, și care nu poate fi nulă. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CARTE.

id_cititor = variabilă de tip întreg, care reprezintă id-ul cititorului pentru care se reține istoricul, și care nu poate fi nulă. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CITITOR.

data_imprumutului = variabilă de tip dată calendaristică, care reprezintă data la care a fost efectuat împrumutul.

data_limita = variabilă de tip dată calendaristică, care reprezintă data limită până la care se poate restitui cartea împrumutată.

data_restituire = variabilă de tip dată calendaristică, care reprezintă data la care a fost restituită cartea împrumutată.

Relația **AUTOR_scrie_CARTE** are ca atribute:

id_scriere = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unei scrieri.

id_autor = variabilă de tip întreg, care reprezintă id-ul autorului, și care

nu poate fi nulă. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul AUTOR.

id_carte = variabilă de tip întreg, care reprezintă id-ul cărții, și care nu poate fi nulă. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul CARTE.

Relația **CARTE***_aparține_DOMENIU* are ca atribute:

id_apartinere = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unei apartineri, fiind cheia primară a tabelului asociativ APARTINE.

id_carte = variabilă de tip întreg, care reprezintă id-ul cărții, și care nu poate fi nulă. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul DOMENIU.

id_domeniu = variabilă de tip întreg, care reprezintă id-ul domeniului, și care nu poate fi nulă. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul DOMENIU.

Relația de tip 3 dintre entitățile BIBLIOTECĂ, CITITOR, și CARTE are ca atribute:

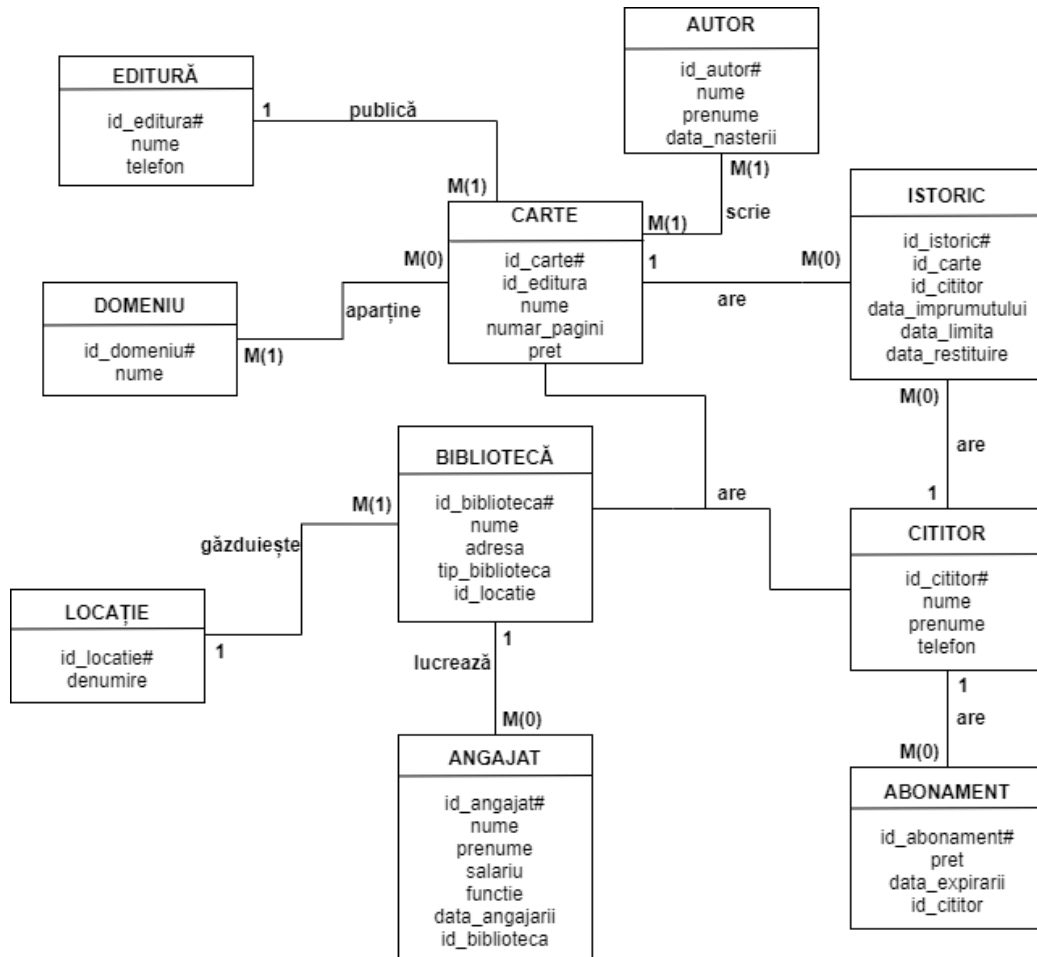
id_imprumut = variabilă de tip întreg, care nu poate fi nulă, și care reprezintă id-ul unui împrumut.

id_carte = variabilă de tip întreg, care reprezintă id-ul cărții. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul CARTE.

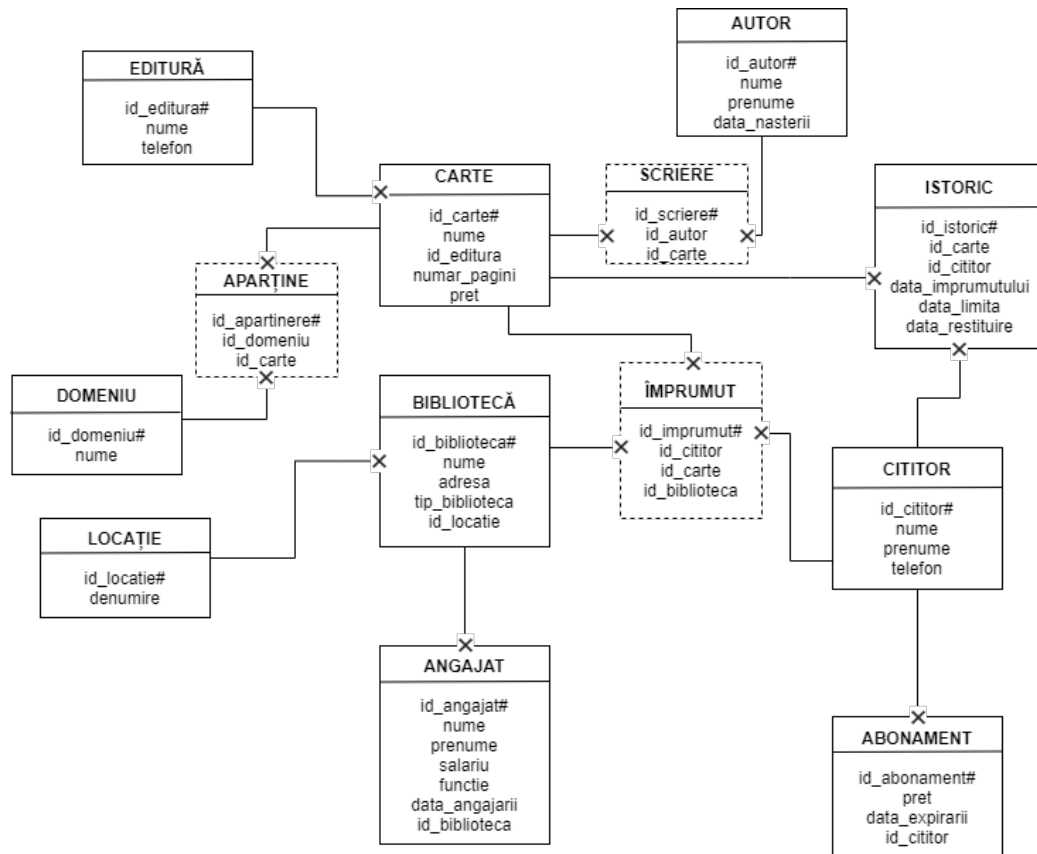
id_biblioteca = variabilă de tip întreg, care reprezintă id-ul bibliotecii. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul BIBLIOTECĂ.

id_cititor = variabilă de tip întreg, care reprezintă id-ul cititorului. Atributul trebuie să coresundă la o valoare a cheii primare din tabelul CITITOR.

6 Diagrama entitate-relație



7 Diagrama conceptuală



8 Scheme relaționale

Schemele relaționale corespunzătoare diagramei conceptuale sunt următoarele:

EDITURĂ (id_editura#, nume, telefon)

CARTE (id_carte#, nume, id_editura, numar_pagini, pret)

AUTOR (id_autor#, nume, prenume, data_nasterii)

SCRIERE (id_scriere#, id_autor, id_carte)

DOMENIU (id_domeniu#, nume)
 APARTINE (id_apartinere#, id_domeniu, id_carte)
 LOCATIE (id_locatie#, denumire)
 ANGAJAT (id_angajat#, nume, prenume, salariu, functie, data_angajarii, id_biblioteca)
 ABONAMENT (id_abonament#, pret, data_expirarii, id_cititor)
 CITITOR (id_cititor#, nume, prenume, telefon)
 ISTORIC (id_istoric#, id_carte, id_cititor, data_imprumutului, data_limita, data_restituirii)
 BIBLIOTECĂ (id_biblioteca#, nume, adresa, tip_biblioteca, id_locatie)
 IMPRUMUT (id_imprumut#, id_cititor, id_carte, id_biblioteca)

9 Realizarea normalizării (FN1-FN3)

9.1 Forma normală 1 - FN1

Prima formă normală exclude posibilitatea existenței grupurilor repetitive, cerând ca fiecare câmp într-o bază de date să cuprindă numai o valoare atomică. 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.

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

Pentru a exemplifica normalizarea, se consideră următorul exemplu:

| CITITOR | CARTE |
|---------|----------|
| CIT1 | C1,C2,C3 |
| CIT2 | C2,C4 |
| CIT3 | C3 |

Exemplu non-FN1

| CITITOR | CARTE |
|---------|-------|
| CIT1 | C1 |
| CIT1 | C2 |
| CIT1 | C3 |
| CIT2 | C2 |
| CIT2 | C4 |
| CIT3 | C3 |

Exemplu FN1

9.2 Forma normală 2 - FN2

O relație R este în a doua formă normală dacă și numai dacă relația R este în FN1, iar fiecare atribut care nu este cheie (nu participă la cheia primară) este dependent de întreaga cheie primară.

Practic, a doua formă normală cere ca toate elementele unei tabele să fie dependente funcțional de totalitatea cheii primare.

Observăm că în modelul implementat se respectă toate cerințele celei de-a doua forme normale: toate relațiile sunt în FN1, iar fiecare atribut din fiecare entitate, care nu este cheie primară, este dependent de aceasta.

Pentru a exemplifica normalizarea, se consideră următorul exemplu:

| COD_DOMENIU# | DOMENIU | COD_CARTE# | PRET |
|--------------|----------|------------|------|
| D1 | Fictiune | C1 | 14.5 |
| D2 | Romance | C1 | 14.5 |
| D1 | Fictiune | C2 | 17 |
| D3 | Aventura | C3 | 15 |

Exemplu non-FN2

Astfel avem următoarele dependențe:

$\{\text{COD_DOMENIU\#}\} \rightarrow \{\text{DOMENIU}\}$ – cod.domeniu determină funcțional domeniu

$\{\text{COD_DOMENIU\#}, \text{COD_CARTE\#}\} \rightarrow \{\text{PRET}\}$

Aplicând regula Casey-Delobel pentru FN2, obținem:

| COD_DOMENIU# | COD_CARTE# | PRET |
|---------------------|-------------------|-------------|
| D1 | C1 | 14.5 |
| D2 | C1 | 14.5 |
| D1 | C2 | 17 |
| D3 | C3 | 15 |

| COD_DOMENIU# | DOMENIU |
|---------------------|----------------|
| D1 | Fictiune |
| D2 | Romance |
| D3 | Aventura |

Exemplu FN2

9.3 Forma normală 3 - FN3

Intuitiv, o relație R este în a treia formă normală dacă și numai dacă relația R este în FN2, iar fiecare atribut care nu este cheie (nu participă la o cheie) depinde direct de cheia primară. Cu alte cuvinte, o relație este în FN3 dacă și numai dacă fiecare atribut (coloană) care nu este cheie, depinde de cheie, de întreaga cheie și numai de cheie.

Observăm că în modelul implementat se respectă toate cerințele celei de-a treia forme normale.

Pentru a exemplifica normalizarea, se consideră următorul exemplu:

| COD_DOMENIU# | COD_CARTE# | PRET |
|---------------------|-------------------|-------------|
| D1 | C1 | 14.5 |
| D2 | C1 | 14.5 |
| D1 | C2 | 17 |
| D3 | C3 | 15 |

Exemplu non-FN3

Pentru a aduce relația R în FN3 se aplică regula Casey-Delobel. Relația se descompune, prin eliminarea dependențelor funcționale tranzitive, în proiecțiile:

R1(COD_CARTE#, PRET)

R2(COD_DOMENIU#, COD_CARTE#)

| COD_CARTE# | PRET |
|------------|------|
| C1 | 14.5 |
| C2 | 17 |
| C3 | 15 |

| COD_DOMENIU# | COD_CARTE# |
|--------------|------------|
| D1 | C1 |
| D2 | C1 |
| D1 | C2 |
| D3 | C3 |

Exemplu FN3

10 Crearea tabelelor în SQL și inserarea de date

10.1 Creare și inserare în tabela *AUTOR*

```
CREATE TABLE autor
(id_autor number(5) constraint pk_autor primary key,
nume varchar2(20),
prenume varchar2(20),
data_nasterii date);
```

- - A se verifica ca prima valoare a secventei sa fie 1, pentru a garanta corectitudinea datelor

```
CREATE SEQUENCE SEQ_AUTOR  
INCREMENT by 1  
START WITH 1  
MAXVALUE 1000  
NOCYCLE;
```

```
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Creanga',  
'Ion', to_date('01/03/1837','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Eminescu',  
'Mihai', to_date('15/01/1850','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Eliade', 'Mircea',  
to_date('13/03/1907','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Sadoveanu',  
'Mihail', to_date('05/11/1880','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Serghi', 'Cella',  
to_date('04/11/1907','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Petrescu',  
'Camil', to_date('22/04/1894','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Slavici', 'Ioan',  
to_date('18/01/1848','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Rowling',  
'Joanne', to_date('31/07/1965','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Silvera', 'Adam',  
to_date('07/06/1990','dd/mm/yyyy'));  
INSERT INTO autor VALUES (SEQ_AUTOR.NEXTVAL, 'Murakami',  
'Haruki', to_date('12/01/1949','dd/mm/yyyy'));  
COMMIT;
```

10.2 Creare și inserare în tabela *EDITURA*

```
CREATE TABLE editura
```

```
(id_editura number(5) constraint pk_editura primary key,  
nume varchar2(20),  
telefon varchar2(15));
```

```
INSERT INTO editura VALUES (1, 'Litera', '0374826635');  
INSERT INTO editura VALUES (2, 'Humanitas', '0214088350');  
INSERT INTO editura VALUES (3, 'Polirom', '0232217440');  
INSERT INTO editura VALUES (4, 'Corint', '0213194820');  
INSERT INTO editura VALUES (5, 'RAO', '0729166965');  
COMMIT;
```

10.3 Creare și inserare în tabela *DOMENIU*

```
CREATE TABLE domeniu  
(id_domeniu number(5) constraint pk_domeniu primary key,  
nume varchar2(20));
```

```
INSERT INTO domeniu VALUES(1,'Fictiune');  
INSERT INTO domeniu VALUES(2,'Literatura clasica');  
INSERT INTO domeniu VALUES(3,'Young adult');  
INSERT INTO domeniu VALUES(4,'Science fiction');  
INSERT INTO domeniu VALUES(5,'Aventura');  
INSERT INTO domeniu VALUES(6,'Literatura romana');  
COMMIT;
```

10.4 Creare și inserare în tabela *LOCATIE*

```
CREATE TABLE locatie  
(id_locatie number(5) constraint pk_locatie primary key,  
nume varchar2(20));
```

```
INSERT INTO locatie VALUES(1,'Bucuresti');
INSERT INTO locatie VALUES(2,'Iasi');
INSERT INTO locatie VALUES(3,'Pitesti');
INSERT INTO locatie VALUES(4,'Timisoara');
INSERT INTO locatie VALUES(5,'Cluj-Napoca');
INSERT INTO locatie VALUES (6, 'Slatina');
INSERT INTO locatie values (7, 'Piatra Neamt');
COMMIT;
```

10.5 Creare și inserare în tabela *CITITOR*

```
CREATE TABLE cititor
(id.cititor number(5) constraint pk_cititor primary key,
nume varchar2(20),
prenume varchar2(20),
telefon varchar2(15));
```

```
CREATE SEQUENCE SEQ_CITITOR
INCREMENT by 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
```

```
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Toader',
'Vlad',
'0751968771');
INSERT INTO cititor VALUES(SEQ_CITITOR.NEXTVAL, 'Popescu',
'Ilinca',
'0775200135');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Fierbin-
teanu' , 'David',
```

```
'0762111320');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Florea' ,
'Alexandra',
'0734231224');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Ilinca',
'Flavius',
'0745874914');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Sandu',
'Maria',
'0756204355');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Morar',
'Ramona',
'0721234222');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Maria',
'Viorica',
'0786235333');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Ilie' , 'Ovidiu',
'0763223453');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Toma',
'Carmen',
'0792331234');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Nicolae',
'Florina',
'0745375312');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Marin',
'Dorin',
'0742543212');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Grigore',
'Denisa',
'0756320488');
```

```
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Bratu', 'Sonia',
'0785698210');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Toderoiu',
'Gabriel',
'0794212333');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Boboc',
'Georgiana',
'0764122353');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Vladau',
'Adrian',
'0764122331');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Bican', 'Serban',
'0754231567');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Alexe', 'Stefana',
'0752122353');
INSERT INTO cititor VALUES (SEQ_CITITOR.NEXTVAL, 'Szabo',
'Richard',
'0753211354');
COMMIT;
```

10.6 Creare și inserare în tabela *CARTE*

```
CREATE TABLE carte
(id_carte number(5) constraint pk_carte primary key,
nume varchar2(20),
numar_pagini number(4),
pret number(3),
id_editura number(5),
```



```
constraint fk_carte foreign key (id_editura) references EDITURA ( id_editura
));
```

```
CREATE SEQUENCE SEQ_CARTE
INCREMENT by 1
START WITH 2
MAXVALUE 1000
NOCYCLE;
```

```
ALTER TABLE carte MODIFY nume varchar2(40);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Amintiri din
copilarie', 269, 9.5, 1);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Poezii', 256, 8,
2);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL, 'Maitreyi',
200,18,3);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL, 'La tiganci',
188, 17.34,3);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL, 'Romanul ado-
lescentului miop', 368, 35, 4);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL, 'Nunta in cer',
160,22,2);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Dumbrava mi-
nunata', 48,15,1);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Baltagul', 210,
22, 4);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Hanu Ancutei',
152, 19.5, 5);
```

```
INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Cartea Mironei',
```

496, 19.9, 4);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Panza de paianjen', 480, 36.6, 3);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Patul lui Procust', 336, 27, 1);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Jocul Ielelor', 320, 32, 2);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Mara', 319,12,4);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Moara cu noroc', 192, 13, 3);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Harry Potter si piatra filosofala', 256,69,4);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Harry Potter si prizonierul din Azkaban', 462,49,4);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Harry Potter si Talismanele Mortii', 784,65,4);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Padurea norvegiana', 360,27,3);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'1Q84', 448,28,3);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Trecutul e tot ce mi-ai lasat', 376,33,5);

INSERT INTO carte VALUES (SEQ_CARTE.NEXTVAL,'Amandoi mor la sfarsit', 336,45,5);

COMMIT;

10.7 Creare și inserare în tabela *BIBLIOTECA*

```
CREATE TABLE biblioteca
```

```
(id_biblioteca number(5) constraint pk_biblioteca primary key,
```

```
nume varchar2(20),
```

```
adresa varchar2(40),
```

```
tip_biblioteca varchar2(20),
id_locatie number(5),
constraint fk_biblioteca foreign key (id_locatie) references LOCATIE (
id_locatie),
constraint check_tip_biblioteca check(upper(tip_biblioteca) in ('MUNINCI-
PALA','SCOLARA')) );
```

```
INSERT INTO biblioteca VALUES(1,'Mihai Eminescu','Bd. Unirii, nr. 5,
sector 1','Municipala',1);
```

```
INSERT INTO biblioteca VALUES (2,'Dinicu Golescu', 'Piata Juraman-
tului, nr. 12','Municipala',3);
```

```
INSERT INTO biblioteca VALUES (3,'Tudor Musatescu','Strada Negru
Voda, nr. 66','Scolara',2);
```

```
INSERT INTO biblioteca VALUES (4,'Municipala nr.5','Bd. Eroilor, nr.
336, corp A','Municipala',4);
```

```
INSERT INTO biblioteca VALUES (5,'Dan Barbilian','Strada Dezrobirii,
nr. 37','Scolara',5);
```

```
COMMIT;
```

10.8 Creare și inserare în tabela *APARTINE*

```
CREATE TABLE apartine
(id_apartinere number(5) constraint pk_apartinere primary key,
id_carte number(5),
id_domeniu number(5),
constraint fk_apartine_domeniu foreign key (id_domeniu) references DOME-
NIU(id_domeniu),
constraint fk_apartine_carte foreign key (id_carte) references CARTE
(id_carte));
```

```
CREATE SEQUENCE SEQ_APARTINE
```

```
INCREMENT by 1  
START WITH 1  
MAXVALUE 1000  
NOCYCLE;
```

```
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,3,2);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,3,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,3,5);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,4,2);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,4,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,5,2);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,5,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,5,5);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,6,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,7,2);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,7,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,8,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,9,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,8,1);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,9,1);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,10,1);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,10,2);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,10,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,10,5);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,11,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,11,2);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,12,1);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,12,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,13,6);  
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,14,1);
```

```
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,14,6);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,15,6);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,16,6);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,16,2);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,16,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,17,2);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,17,6);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,18,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,18,3);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,18,5);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,19,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,19,3);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,19,5);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,20,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,20,3);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,20,5);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,21,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,22,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,23,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,23,3);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,24,1);
INSERT INTO apartine VALUES(SEQ_APARTINE.NEXTVAL,24,3);
COMMIT;
```

10.9 Creare și inserare în tabela *SCRIERE*

```
CREATE TABLE scriere
(id_scriere number(5) constraint pk_scriere primary key,
id_autor number(5),
id_carte number(5),
constraint fk_scriere_autor foreign key (id_autor) references AUTOR (id_autor),
```

constraint fk_scriere_carte foreign key (id_carte) references CARTE (id_carte));

```
INSERT INTO scriere VALUES (1,1,3);
INSERT INTO scriere VALUES (2,2,4);
INSERT INTO scriere VALUES (3,3,5);
INSERT INTO scriere VALUES (4,3,6);
INSERT INTO scriere VALUES (5,3,7);
INSERT INTO scriere VALUES (6,3,8);
INSERT INTO scriere VALUES (7,4,9);
INSERT INTO scriere VALUES (8,4,10);
INSERT INTO scriere VALUES (9,4,11);
INSERT INTO scriere VALUES (10,5,12);
INSERT INTO scriere VALUES (11,5,13);
INSERT INTO scriere VALUES (12,6,13);
INSERT INTO scriere VALUES (13,6,14);
INSERT INTO scriere VALUES (14,6,15);
INSERT INTO scriere VALUES (15,3,15);
INSERT INTO scriere VALUES (16,7,16);
INSERT INTO scriere VALUES (17,7,17);
INSERT INTO scriere VALUES (18,8,18);
INSERT INTO scriere VALUES (26,8,19);
INSERT INTO scriere VALUES (19,8,20);
INSERT INTO scriere VALUES (20,10,21);
INSERT INTO scriere VALUES (21,10,22);
INSERT INTO scriere VALUES (22,9,23);
INSERT INTO scriere VALUES (23,9,24);
INSERT INTO scriere VALUES (24,8,23);
INSERT INTO scriere VALUES (25,10,23);
COMMIT;
```

10.10 Creare și inserare în tabela *ANGAJAT*

```
CREATE TABLE angajat
(id_angajat number(5) constraint pk_angajat primary key,
nume varchar2(20),
prenume varchar2(20),
salariu number(5),
functie varchar2(20),
data_angajarii date,
id_biblioteca number(5),
constraint fk_angajat foreign key (id_biblioteca) references BIBLIOTECA
(id_biblioteca),
constraint check_functie_angajat check (upper(functie) in ('MANAGER',
'BIBLIOTECAR', 'PAZNIC', 'INGRIJITOR')) );
```

```
CREATE SEQUENCE SEQ_ANGAJAT
INCREMENT by 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
```

```
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL, 'Antonescu',
'Ovidiu', 3500, 'Manager', to_date('30/06/2015', 'dd/mm/yyyy'), 2);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL, 'Cordun',
'Diana', 2900, 'Bibliotecar', to_date('12/11/2017', 'dd/mm/yyyy'), 2);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL, 'Scheianu',
'Anisia', 2800, 'Bibliotecar', to_date('08/05/2019', 'dd/mm/yyyy'), 2);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL, 'Avram',
'Catalin', 2300, 'Paznic', to_date('23/01/2020', 'dd/mm/yyyy'), 2);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL, 'Catana',
'Ionela', 2000, 'Ingrijitor', to_date('15/08/2014', 'dd/mm/yyyy'), 2);
```

```
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Cuza',
'Catalina', 3600, 'Manager', to_date('11/11/2011','dd/mm/yyyy'),3);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Satarbasa',
'Carmen',3000,'Bibliotecar',to_date('01/06/2012','dd/mm/yyyy'),3);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Parghel',
'Petre',3700, 'Manager', to_date('01/09/2008','dd/mm/yyyy'),4);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Negus',
'Laura', 3500, 'Bibliotecar', to_date('14/05/2011','dd/mm/yyyy'),4);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Oproiu',
'Laurentiu', 3600, 'Bibliotecar', to_date('26/02/2012','dd/mm/yyyy'),4);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Dorcioman',
'Dana', 3000, 'Paznic', to_date('23/06/2007','dd/mm/yyyy'),4);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Bratu',
'Ionut', 2400, 'Bibliotecar', to_date('24/05/2015','dd/mm/yyyy'),5);
INSERT INTO angajat VALUES (SEQ_ANGAJAT.NEXTVAL,'Broscoteanu',
'David', 3250, 'Bibliotecar', to_date('11/04/2021','dd/mm/yyyy'),2);
COMMIT;
```

10.11 Creare și inserare în tabela *ABONAMENT*

```
CREATE TABLE abonament
(id.abonament number(5) constraint pk_abonament primary key,
pret number(4),
data_expirarii date,
id_cititor number(5),
constraint fk_abonament foreign key(id_cititor) references CITITOR (id_cititor)
);
```

```
CREATE SEQUENCE SEQ_ABONAMENT
INCREMENT by 1
START WITH 1
```


MAXVALUE 1000

NOCYCLE;

```
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,20,
to_date('30/06/2021','dd/mm/yyyy'),2);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,25,
to_date('01/01/2022','dd/mm/yyyy'),3);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,20,
to_date('04/02/2021','dd/mm/yyyy'),4);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,22.5,
to_date('13/06/2021','dd/mm/yyyy'),5);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,24,
to_date('30/07/2021','dd/mm/yyyy'),7);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,25,
to_date('01/03/2022','dd/mm/yyyy'),2);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,30,
to_date('30/06/2022','dd/mm/yyyy'),8);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,15,
to_date('12/02/2021','dd/mm/yyyy'),9);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,17.5,
to_date('22/03/2021','dd/mm/yyyy'),11);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,23,
to_date('17/09/2021','dd/mm/yyyy'),12);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,14,
to_date('01/01/2021','dd/mm/yyyy'),14);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,20,
to_date('19/10/2021','dd/mm/yyyy'),14);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,18,
to_date('28/04/2021','dd/mm/yyyy'),15);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,35,
```

```
to_date('31/12/2022','dd/mm/yyyy'),16);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,22.5,
to_date('22/06/2021','dd/mm/yyyy'),18);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,25,
to_date('31/07/2021','dd/mm/yyyy'),19);
INSERT INTO abonament VALUES (SEQ_ABONAMENT.NEXTVAL,31.5,
to_date('22/04/2022','dd/mm/yyyy'),20);
COMMIT;
```

10.12 Creare și inserare în tabela *IMPRUMUT*

```
CREATE TABLE imprumut
(id_imprumut number(5) constraint pk_imprumut primary key,
id_cititor number(5),
id_carte number(5),
id_biblioteca number(5),
constraint fk_imprumut_cititor foreign key(id_cititor) references CITITOR
(id_cititor),
constraint fk_imprumut_carte foreign key(id_carte) references CARTE
(id_carte),
constraint fk_imprumut_biblioteca foreign key(id_biblioteca) references
BIBLIOTECA(id_biblioteca) );
```

```
CREATE SEQUENCE SEQ_IMPRUMUT
INCREMENT BY 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
```

```
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,2,23,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,3,4,3);
```

```
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,4,15,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,5,8,1);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,6,12,4);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,7,7,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,8,15,5);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,9,3,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,10,7,5);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,11,9,4);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,12,19,3);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,13,22,1);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,14,24,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,15,16,3);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,16,9,4);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,17,3,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,18,4,4);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,19,13,5);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,20,3,1);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,21,18,3);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,3,6,3);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,4,8,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,2,22,2);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,7,15,5);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,13,5,3);
INSERT INTO imprumut VALUES (SEQ_IMPRUMUT.NEXTVAL,18,3,5);
COMMIT;
```

10.13 Creare și inserare în tabela *ISTORIC*

```
CREATE TABLE istoric
(id.istoric number(5) constraint pk_istoric primary key,
id_carte number(5),
```

```
id_cititor number(5),
data_imprumutului date,
data_limita date,
data_restituire date,
constraint fk_istoric_carte foreign key(id_carte) references CARTE (id_carte),
constraint fk_istoric_cititor foreign key(id_cititor) references CITITOR
(id_cititor),
constraint check_data check(data_limita < data_imprumutului)
);
```

```
CREATE SEQUENCE SEQ_ISTORIC
INCREMENT by 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
```

```
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 23, 2, to_date
('23/01/2021', 'dd/mm/yyyy'), to_date('31/01/2021', 'dd/mm/yyyy'), to_date
('30/01/2021', 'dd/mm/yyyy'));
```

```
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 4, 3, to_date
('13/03/2021', 'dd/mm/yyyy'),to_date ('27/03/2021', 'dd/mm/yyyy'), to_date
('27/03/2021', 'dd/mm/yyyy'));
```

```
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 15, 4, to_date
('27/12/2020', 'dd/mm/yyyy'), to_date('11/01/2021', 'dd/mm/yyyy'), to_date
('09/01/2021', 'dd/mm/yyyy'));
```

```
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 8, 5, to_date
('06/02/2021', 'dd/mm/yyyy'), to_date ('20/02/2021', 'dd/mm/yyyy'), to_date
('26/02/2021', 'dd/mm/yyyy'));
```

```
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 12, 6, to_date
('11/04/2021', 'dd/mm/yyyy'), to_date ('25/04/2021', 'dd/mm/yyyy'), to_date
```

```
( '21/04/2021', 'dd/mm/yyyy');
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 15, 8, to_date
( '13/06/2021', 'dd/mm/yyyy'), to_date ('27/06/2021', 'dd/mm/yyyy'), to_date
( '14/06/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 3, 9, to_date
( '09/12/2020', 'dd/mm/yyyy'), to_date ('23/12/2020', 'dd/mm/yyyy'), to_date
( '20/12/2020', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 7, 10, to_date
( '01/03/2021', 'dd/mm/yyyy'), to_date ('15/03/2021', 'dd/mm/yyyy'), to_date
( '06/04/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 9, 11, to_date
( '18/04/2021', 'dd/mm/yyyy'), to_date ('25/04/2021', 'dd/mm/yyyy'), to_date
( '23/04/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 19, 12,
to_date ( '28/01/2021', 'dd/mm/yyyy'), to_date ('07/02/2021', 'dd/mm/yyyy'),
to_date ( '07/02/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 22, 13,
to_date ( '09/03/2021', 'dd/mm/yyyy'), to_date ('16/03/2021', 'dd/mm/yyyy'),
to_date ( '17/03/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 24 ,14,
to_date ( '17/02/2021', 'dd/mm/yyyy'), to_date ('24/02/2021', 'dd/mm/yyyy'),
to_date ( '23/02/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 16, 15,
to_date ( '18/03/2021', 'dd/mm/yyyy'), to_date ('01/04/2021', 'dd/mm/yyyy'),
to_date ( '03/04/2021', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES( SEQ_ISTORIC.NEXTVAL, 9, 16, to_date
( '01/11/2020', 'dd/mm/yyyy'), to_date ('15/11/2020', 'dd/mm/yyyy'), to_date
('16/11/2020', 'dd/mm/yyyy'));
INSERT INTO istoric VALUES (SEQ_ISTORIC.NEXTVAL, 3, 17, to_date
( '19/05/2021', 'dd/mm/yyyy'), to_date ('13/06/2021', 'dd/mm/yyyy'), to_date
```

```
( '06/06/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 4,18, to_date  
( '30/05/2021', 'dd/mm/yyyy'), to_date( '15/06/2021','dd/mm/yyyy'),NULL);  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 13, 19,  
to_date ( '06/06/2021', 'dd/mm/yyyy'), to_date ( '30/06/2021','dd/mm/yyyy'),  
NULL);  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 3, 20,  
to_date ( '25/02/2021', 'dd/mm/yyyy'), to_date ('05/03/2021', 'dd/mm/yyyy'),  
to_date ( '04/03/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 18, 21,  
to_date ( '18/01/2021', 'dd/mm/yyyy'), to_date ('01/02/2021', 'dd/mm/yyyy'),  
NULL);  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 6, 3, to_date  
( '19/03/2021', 'dd/mm/yyyy'), to_date ('03/04/2021', 'dd/mm/yyyy'), to_date  
( '24/03/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 8, 4, to_date  
( '01/03/2021', 'dd/mm/yyyy'), to_date ('01/04/2021', 'dd/mm/yyyy'), to_date  
( '27/03/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 22, 2,  
to_date ( '19/05/2021', 'dd/mm/yyyy'), to_date ('03/06/2021', 'dd/mm/yyyy'),  
to_date ( '20/05/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 15, 7,  
to_date ( '04/02/2021', 'dd/mm/yyyy'), to_date ('18/02/2021', 'dd/mm/yyyy'),  
to_date ( '27/02/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 5, 13,  
to_date ( '15/01/2021', 'dd/mm/yyyy'), to_date ('29/01/2021', 'dd/mm/yyyy'),  
to_date ( '31/01/2021', 'dd/mm/yyyy'));  
INSERT INTO istoric VALUES ( SEQ_ISTORIC.NEXTVAL, 3, 18,  
to_date ( '07/03/2021', 'dd/mm/yyyy'), to_date ('03/04/2021', 'dd/mm/yyyy'),  
to_date ( '19/03/2021', 'dd/mm/yyyy'));
```

COMMIT;

10.14 Adăguarea constrângerilor *not null* tabelelor

```
ALTER TABLE editura MODIFY nume not null;
ALTER TABLE domeniu MODIFY nume not null;
ALTER TABLE autor MODIFY nume not null;
ALTER TABLE autor MODIFY prenume not null;
ALTER TABLE carte MODIFY nume not null;
ALTER TABLE carte MODIFY id_editura not null;
ALTER TABLE abonament MODIFY id_cititor not null;
ALTER TABLE cititor MODIFY nume not null;
ALTER TABLE cititor MODIFY prenume not null;
ALTER TABLE angajat MODIFY nume not null;
ALTER TABLE angajat MODIFY prenume not null;
ALTER TABLE angajat MODIFY id_biblioteca not null;
ALTER TABLE locatie MODIFY nume not null;
ALTER TABLE biblioteca MODIFY nume not null;
ALTER TABLE biblioteca MODIFY tip_biblioteca not null;
ALTER TABLE biblioteca MODIFY id_locatie not null;
ALTER TABLE istoric MODIFY id_cititor not null;
ALTER TABLE istoric MODIFY id_carte not null;
ALTER TABLE scriere MODIFY id_autor not null;
ALTER TABLE scriere MODIFY id_carte not null;
```

11 Cereri SQL

11.1 Cererea nr. 1

Să se afișeze angajații bibliotecilor care se află în Pitești, precum și salariul acestora, știind că li se acordă o majorare de salariu de 15%.

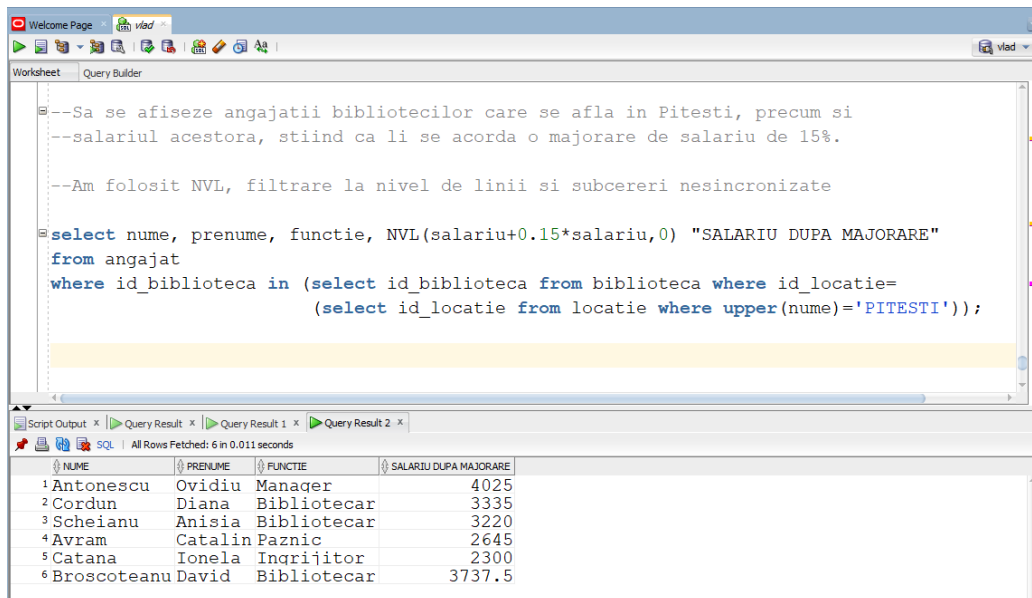
În cadrul acestei cereri, am folosit NVL, filtrare la nivel de linii și subcereri nesincronizate.

Mai întâi am selectat id-ul locației cu numele Pitești, apoi am selectat bibliotecile al căror cod de locație este egal cu id-ul găsit anterior, iar apoi am selectat datele referitoare la angajații care lucrează în aceste biblioteci.

```

select nume, prenume, functie, NVL(salariu+0.15*salariu,0)
"SALARIU DUPA MAJORARE"
from angajat
where id_biblioteca in
(select id_biblioteca from biblioteca where id_locatie=
(select id_locatie from locatie where upper(nume)='PITESTI'));

```



The screenshot shows a SQL query editor window with a query that filters employees based on their library location (Pitești) and calculates a 15% salary increase. The query is executed, and the results are displayed in a table below.

| | NUME | PRENUME | FUNCTIE | SALARIU DUPA MAJORARE |
|---|-------------|---------|-------------|-----------------------|
| 1 | Antonescu | Ovidiu | Manager | 4025 |
| 2 | Cordun | Diana | Bibliotecar | 3335 |
| 3 | Scheianu | Anisia | Bibliotecar | 3220 |
| 4 | Avram | Catalin | Paznic | 2645 |
| 5 | Catana | Ionela | Ingritor | 2300 |
| 6 | Broscoteanu | David | Bibliotecar | 3737.5 |

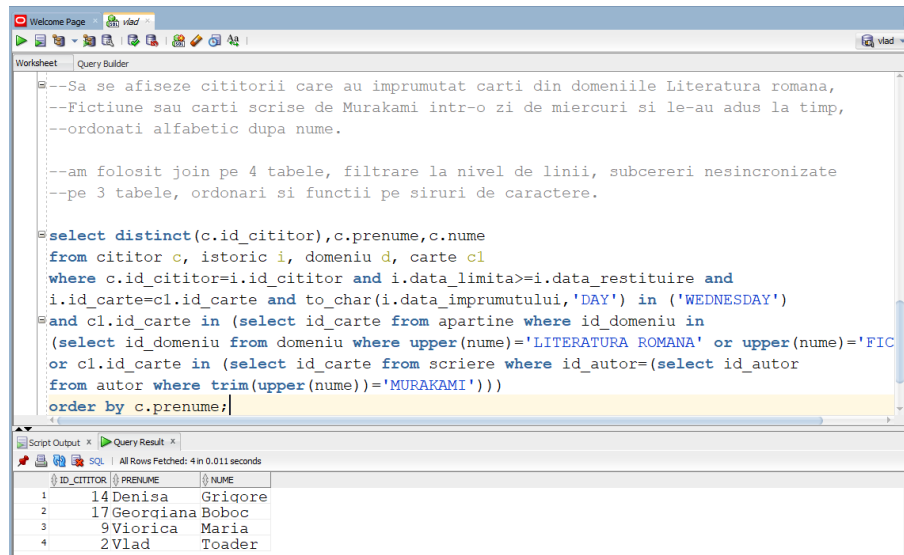
11.2 Cererea nr. 2

Să se afișeze cititorii care au împrumutat cărți din domeniile "Literatură română", "Ficțiune" sau carti scrise de Murakami într-o zi de miercuri și le-au adus la timp, ordonați alfabetic după nume.

În cadrul acestei cereri, am folosit operația join pe 4 tabele, filtrare la nivel de linii, subcereri nesincronizate pe 3 tabele, ordonare, funcții pe șiruri de caractere și pe date calendaristice.

Am selectat întâi cărțile care aparțin domeniilor cerute sau cele scrise de Murakami, iar mai apoi am selectat din cititorii care au împrumutat aceste cărți într-o zi de miercuri, făcând join-uri pentru a le afla numele și prenumele.

```
select distinct(c.id_cititor), c.prename, c.numename
from cititor c, istoric i, domeniu d, carte c1
where c.id_cititor=i.id_cititor and i.data_limita=i.data_restituire
and
i.id_carte=c1.id_carte and to_char(i.data_imprumutului,'DAY')
in ('WEDNESDAY')
and c1.id_carte in
(select id_carte from apartine where id_domeniu in
(select id_domeniu from domeniu where upper(numename)= 'LITER-
ATURA ROMANA' or upper(numename)='FICTIUNE')
or c1.id_carte in
(select id_carte from scriere where id_autor=
(select id_autor from autor where upper(numename)='MURAKAMI'))))
order by c.prename;
```



11.3 Cererea nr. 3

Să se afișeze numele, funcția angajaților, biblioteca unde lucrează și salariul știind că se majorează salariile astfel: pentru manageri cu 20%, pentru bibliotecari cu 15%, pentru paznici cu 10% și pentru îngrijitori cu 5%. Afișați doar angajații care s-au angajat după anul 2015, în ordinea lexicografică a funcțiilor.

În cadrul acestei cereri, am folosit DECODE, NVL, ordonare și o funcție pentru a prelucra date calendaristice.

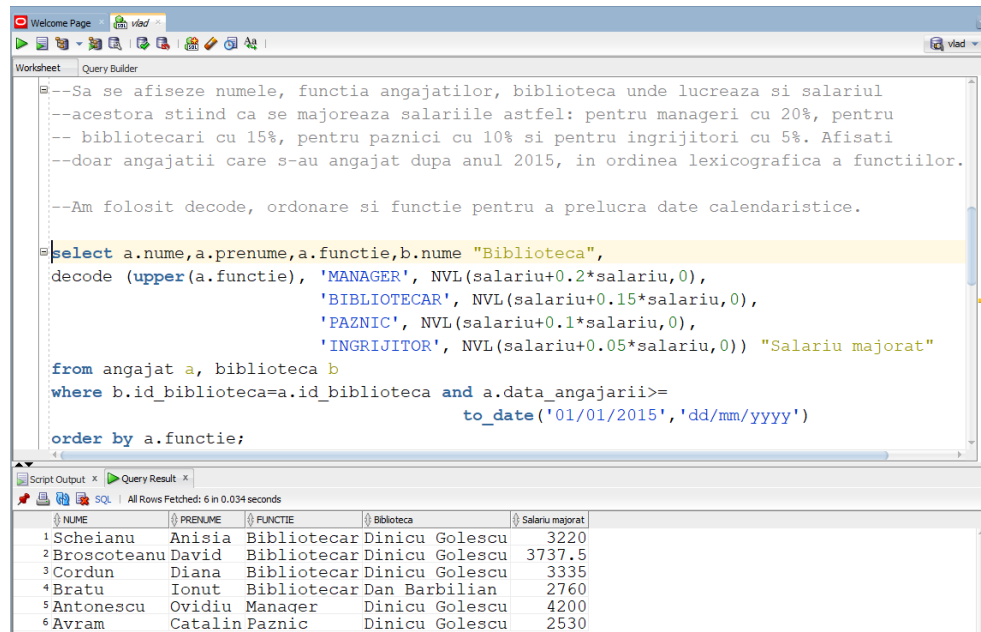
Am selectat angajații a căror dată respectă condiția dată, făcând join cu tabela BIBLIOTECĂ pentru a afișa numele bibliotecii, folosind și decode pentru a calcula salariul.

```
select a.nume, a.preume, a.functie, b.nume "Biblioteca",
decode (upper(a.functie),
'MANAGER', NVL(salariu+0.2*salariu,0),
'BIBLIOTECAR', NVL(salariu+0.15*salariu,0),
```

```

'PAZNIC', NVL(salariu+0.1*salariu,0),
'INGRIJITOR', NVL(salariu+0.05*salariu,0)) "Salariu majorat"
from angajat a, biblioteca b
where b.id_biblioteca=a.id_biblioteca and a.data_angajarii >=
to_date ('01/01/2015','dd/mm/yyyy')
order by a.functie;

```



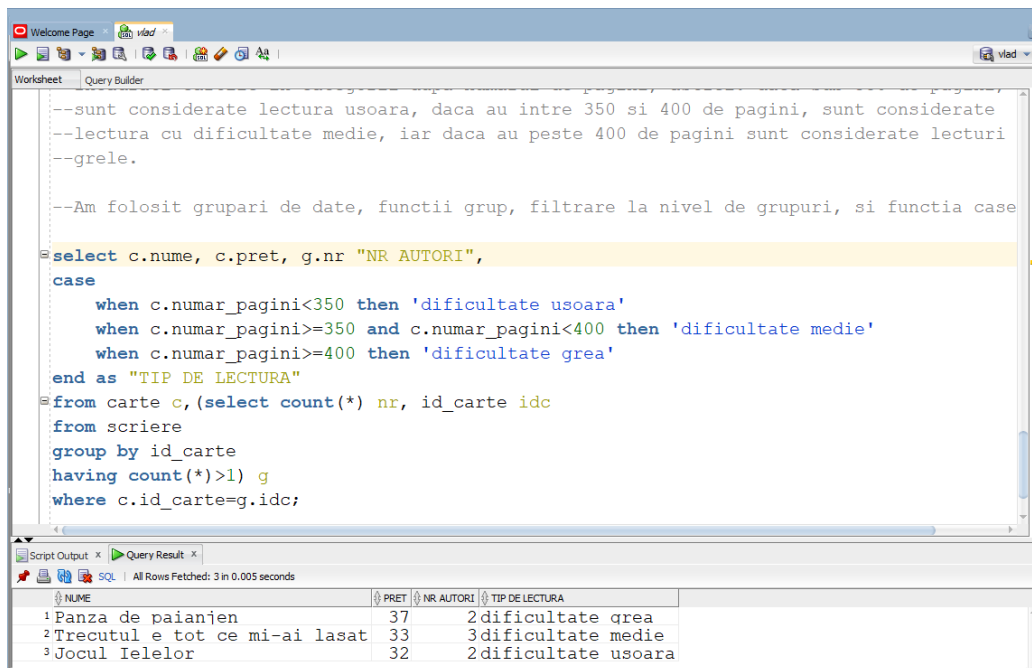
11.4 Cererea nr. 4

Să se afișeze numele și prețul cărților care sunt scrise de cel puțin 2 autori. Încadrați cărțile în categorii în funcție de numărul de pagini, astfel: dacă au sub 350 de pagini, sunt considerate lectură ușoară; dacă au între 350 și 400 de pagini, sunt considerate lectură cu dificultate medie, iar dacă au peste 400 de pagini sunt considerate lecturi grele.

În cadrul acestei cereri, am folosit grupări de date, funcții grup, filtrare la nivel de grupuri și funcția CASE.

Am realizat o subcerere în clauza FROM, în care am selectat pentru fiecare carte care are numărul de autori mai mare decât 1 numărul respectiv și id-ul cărții, iar apoi am făcut join cu tabela CARTE, pentru a afișa informațiile complete.

```
select c.nume, c.pret, g.nr "NR AUTORI",
case when c.numar_pagini<350 then 'dificultate usoara'
when c.numar_pagini>=350 and c.numar_pagini<400 then 'dificultate medie'
when c.numar_pagini>=400 then 'dificultate grea'
end as "TIP DE LECTURA"
from carte c, (select count(*) nr, id_carte idc from scriere group
by id_carte having count(*)>1) g
where c.id_carte=g.idc;
```



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

```
--sunt considerate lectura usoara, daca au intre 350 si 400 de pagini, sunt considerate
--lectura cu dificultate medie, iar daca au peste 400 de pagini sunt considerate lecturi
--grele.

--Am folosit grupari de date, functii grup, filtrare la nivel de grupuri, si functia case

select c.nume, c.pret, g.nr "NR AUTORI",
case
when c.numar_pagini<350 then 'dificultate usoara'
when c.numar_pagini>=350 and c.numar_pagini<400 then 'dificultate medie'
when c.numar_pagini>=400 then 'dificultate grea'
end as "TIP DE LECTURA"
from carte c, (select count(*) nr, id_carte idc
from scriere
group by id_carte
having count(*)>1) g
where c.id_carte=g.idc;
```

The results window shows the following data:

| NUME | PRET | NR AUTORI | TIP DE LECTURA |
|---------------------------------|------|-----------|--------------------|
| 1 Panza de paianjen | 37 | 2 | dificultate grea |
| 2 Trecutul e tot ce mi-ai lasat | 33 | 3 | dificultate medie |
| 3 Jocul Ielelor | 32 | 2 | dificultate usoara |

11.5 Cererea nr. 5

Să se afișeze pentru fiecare angajat care are salariul mai mare decât media salariilor tuturor angajaților numele, prenumele, salariul, funcția, salariul mediu per funcție, numărul de angajați per funcție și biblioteca unde lucrează.

În cadrul acestei cereri, am folosit subcereri sincronizate, filtrare la nivel de linii și clauza WITH.

Am selectat în clauza WITH salariul mediu pentru toți angajații; apoi, 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. Am afișat angajații care au salariul mai mare decât salariul mediu al tuturor angajaților (calculat în clauza WITH).

```
WITH tabtemp as (select avg(salariu) as medie from angajat)
select nume, prenume, salariu, e.functie,
(select round(avg(salariu)) from angajat where functie=e.functie)
" SALARIU MEDIU/FUNCTIE",
(select count(*) from angajat where functie=e.functie) "NR AN-
GAJATI/FUNCTIE",
(select nume from biblioteca where id_biblioteca=e.id_biblioteca)
" BIBLIOTECA"
from angajat e, tabtemp t
where e.salariu>t.medie
order by e.functie
```

Welcome Page

vlad

Worksheet Query Builder

```
select nume,prenume,salariu,e.functie,
(select round(avg(salariu))
from angajat
where functie=e.functie) "SALARIU MEDIU/FUNCTIE",
(select count(*)
from angajat
where functie=e.functie) "NR ANGAJATI/FUNCTIE",
(select nume
from biblioteca
where id_biblioteca=e.id_biblioteca) "BIBLIOTECA"
from angajat e
order by e.functie;
```

Script Output x Query Result x

SQL All Rows Fetched: 12 in 0.003 seconds

| ID_ANGAJAT | NUME | PRENUME | SALARIU | FUNCTIE | DATA_ANGAJARII | ID_BIBLIOTECA | |
|------------|------|-------------|-----------|---------|----------------|---------------|---|
| 1 | 2 | Antonescu | Ovidiu | 3500 | Manager | 30-JUN-15 | 2 |
| 2 | 3 | Cordun | Diana | 2900 | Bibliotecar | 12-NOV-17 | 2 |
| 3 | 4 | Scheianu | Anisia | 2800 | Bibliotecar | 08-MAY-19 | 2 |
| 4 | 5 | Avram | Catalin | 2300 | Paznic | 23-JAN-20 | 2 |
| 5 | 6 | Catana | Ionela | 2000 | Ingrijitor | 15-AUG-14 | 2 |
| 6 | 7 | Cuza | Catalina | 3600 | Manager | 11-NOV-11 | 3 |
| 7 | 8 | Satarbasa | Carmen | 3000 | Bibliotecar | 01-JUN-12 | 3 |
| 8 | 9 | Pardhel | Petre | 3700 | Manager | 01-SEP-08 | 4 |
| 9 | 10 | Nequs | Laura | 3500 | Bibliotecar | 14-MAY-11 | 4 |
| 10 | 11 | Oproiu | Laurentiu | 3600 | Bibliotecar | 26-FEB-12 | 4 |
| 11 | 12 | Broscoteanu | David | 3250 | Bibliotecar | 11-APR-21 | 2 |
| 12 | 13 | Bratu | Ionut | 2400 | Bibliotecar | 24-MAY-15 | 5 |

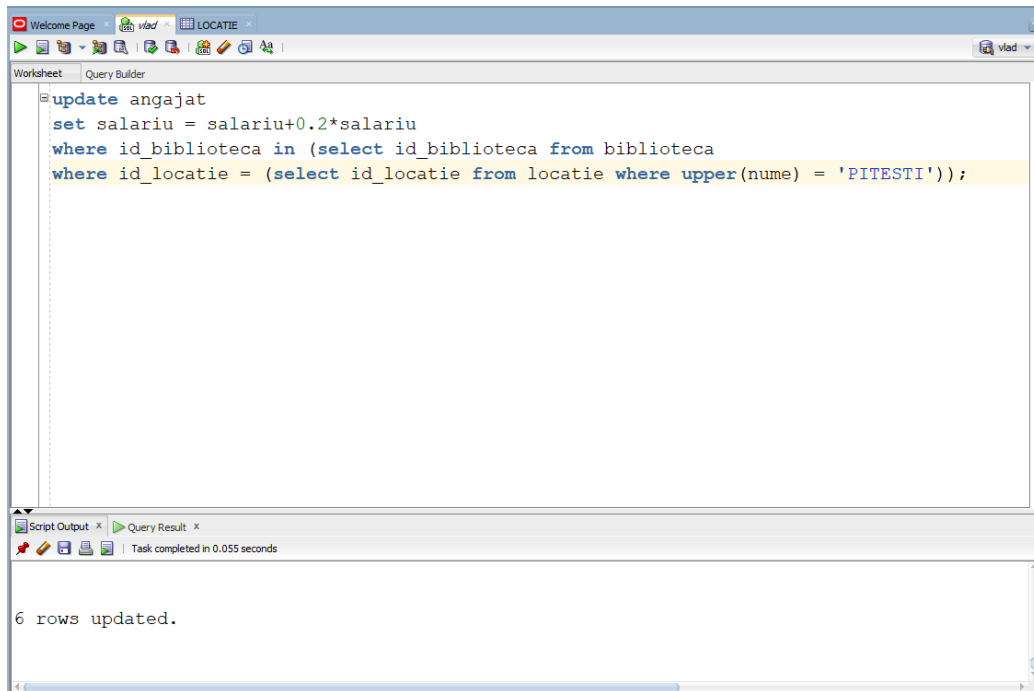
12 Cereri de actualizare și suprimare a datelor

1. Să se mărească salariul angajaților care lucrează la biblioteci aflate în Pitești cu 20

```

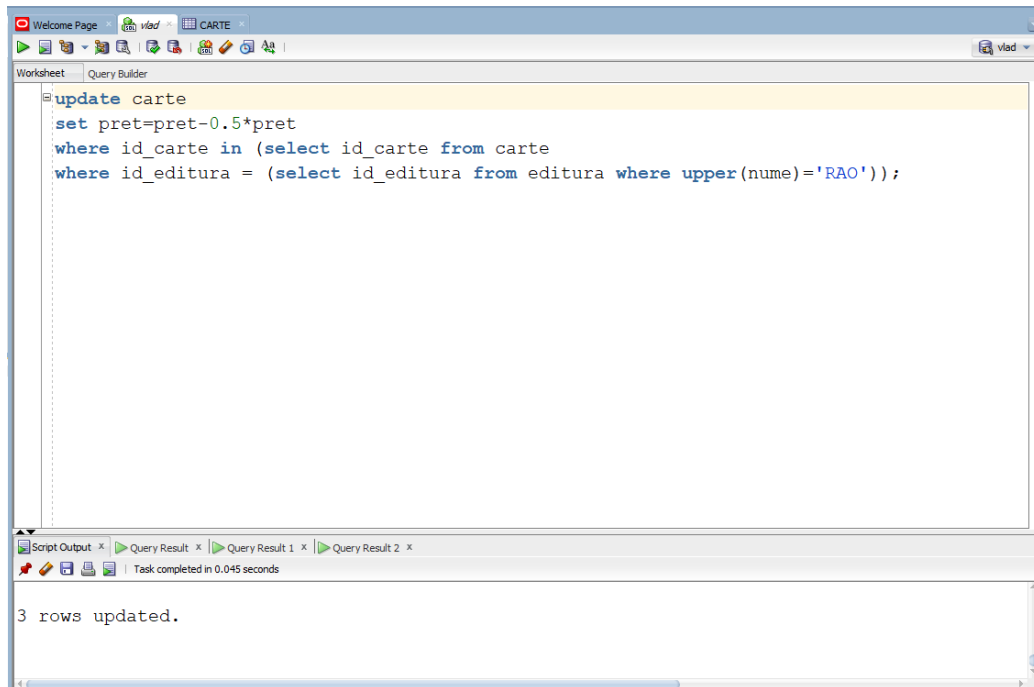
update angajat
set salariu = salariu+0.2*salariu
where id_biblioteca in (select id_biblioteca from biblioteca
where id_locatie = (select id_locatie from locatie where upper(nume)
= 'PITESTI'));

```



2. Să se reducă prețul cărților publicate la editura RAO cu 50%.

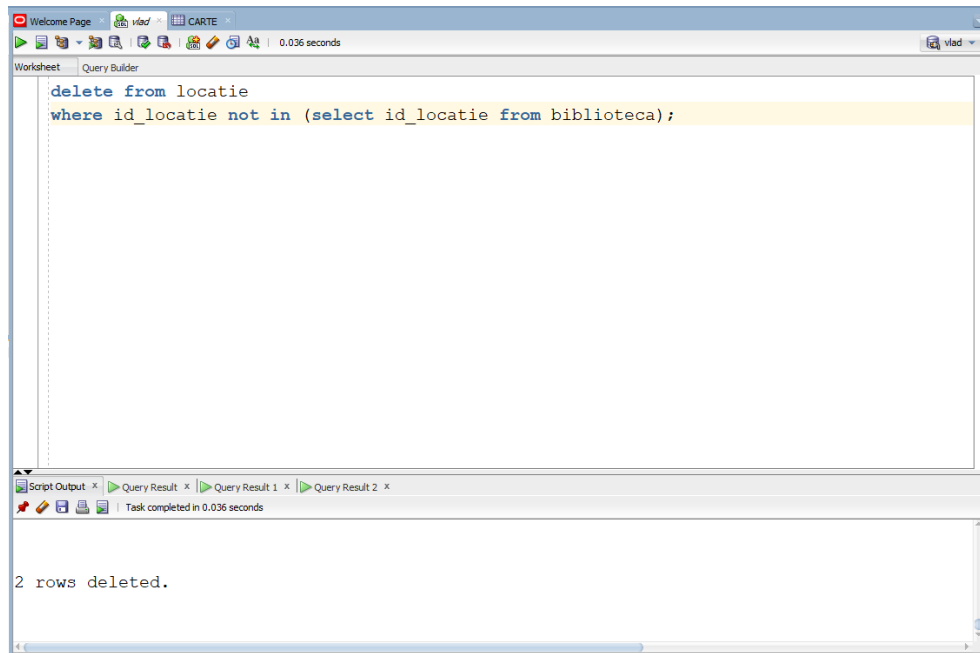
```
update carte  
set pret=pret-0.5*pret  
where id_carte in (select id_carte from carte  
where id_editura = (select id_editura from editura where up-  
per(ume)='RAO'));
```



3. Să se șteargă din baza de date locațiile în care nu se află nicio bibliotecă.

delete from locatie

where id_locatie not in (select id_locatie from biblioteca);



13 Crearea unei secvențe folosită la inserarea datelor în tabele

```
CREATE SEQUENCE SEQ_CITITOR
INCREMENT by 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
```

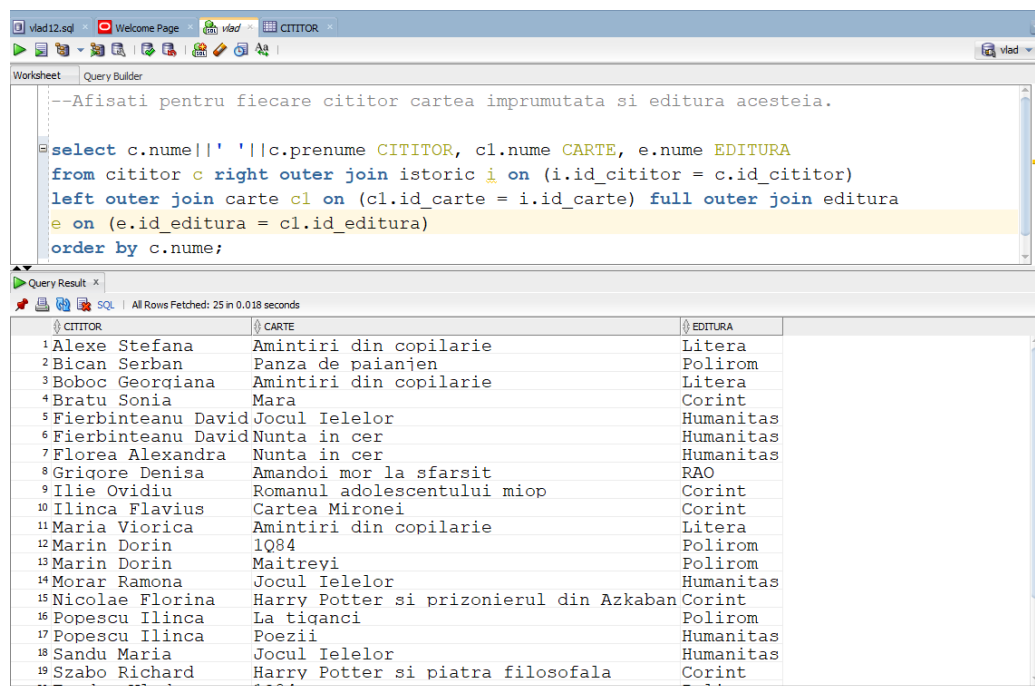
```
CREATE SEQUENCE SEQ_IMPRUMUT
INCREMENT BY 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
```

14 Cereri SQL folosind operațiile *outer-join* și *division*

14.1 Outer-join

Afișați, pentru fiecare cititor, cărțile împrumutate și editurile corespunzătoare fiecărei cărți.

```
select c.id_cititor, concat(concat(c.numa,' '),c.prenume) CITI-
TOR, c1.numa CARTE, e.numa EDITURA
from cititor c right outer join istoric i on (i.id_cititor = c.id_cititor)
left outer join carte c1 on (c1.id_carte = i.id_carte)
full outer join editura e on (e.id_editura = c1.id_editura)
order by c.numa;
```



--Afișati pentru fiecare cititor cartea împrumutată și editura acesteia.

```
select c.numa||' '||c.prenume CITITOR, c1.numa CARTE, e.numa EDITURA
from cititor c right outer join istoric i on (i.id_cititor = c.id_cititor)
left outer join carte c1 on (c1.id_carte = i.id_carte) full outer join editura
e on (e.id_editura = c1.id_editura)
order by c.numa;
```

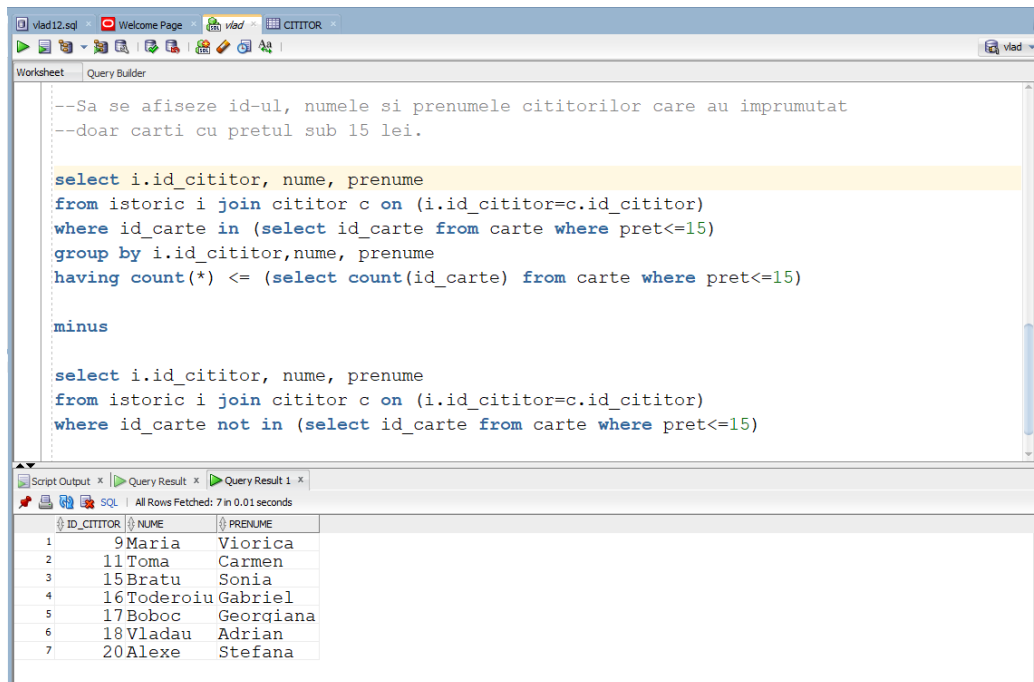
Query Result: All Rows Fetched: 25 in 0.018 seconds

| CITITOR | CARTE | EDITURA |
|----------------------|---|-----------|
| 1 Alexe Stefana | Amintiri din copilărie | Litera |
| 2 Bican Serban | Panza de paianjen | Polirom |
| 3 Boboc Georgiana | Amintiri din copilărie | Litera |
| 4 Bratu Sonia | Mara | Corint |
| 5 Fierbinteanu David | Jocul Ielelor | Humanitas |
| 6 Fierbinteanu David | Nunta in cer | Humanitas |
| 7 Florea Alexandra | Nunta in cer | Humanitas |
| 8 Griore Denisa | Amandoi mor la sfarsit | RAO |
| 9 Ilie Ovidiu | Romanul adolescentului miop | Corint |
| 10 Ilinca Flavius | Cartea Mironei | Corint |
| 11 Maria Viorica | Amintiri din copilărie | Litera |
| 12 Marin Dorin | IQ84 | Polirom |
| 13 Marin Dorin | Maitreyi | Polirom |
| 14 Morar Ramona | Jocul Ielelor | Humanitas |
| 15 Nicolae Florina | Harry Potter si prizonierul din Azkaban | Corint |
| 16 Popescu Ilinca | La tiganci | Polirom |
| 17 Popescu Ilinca | Poezii | Humanitas |
| 18 Sandu Maria | Jocul Ielelor | Humanitas |
| 19 Szabo Richard | Harry Potter si piatra filosofala | Corint |

14.2 Division

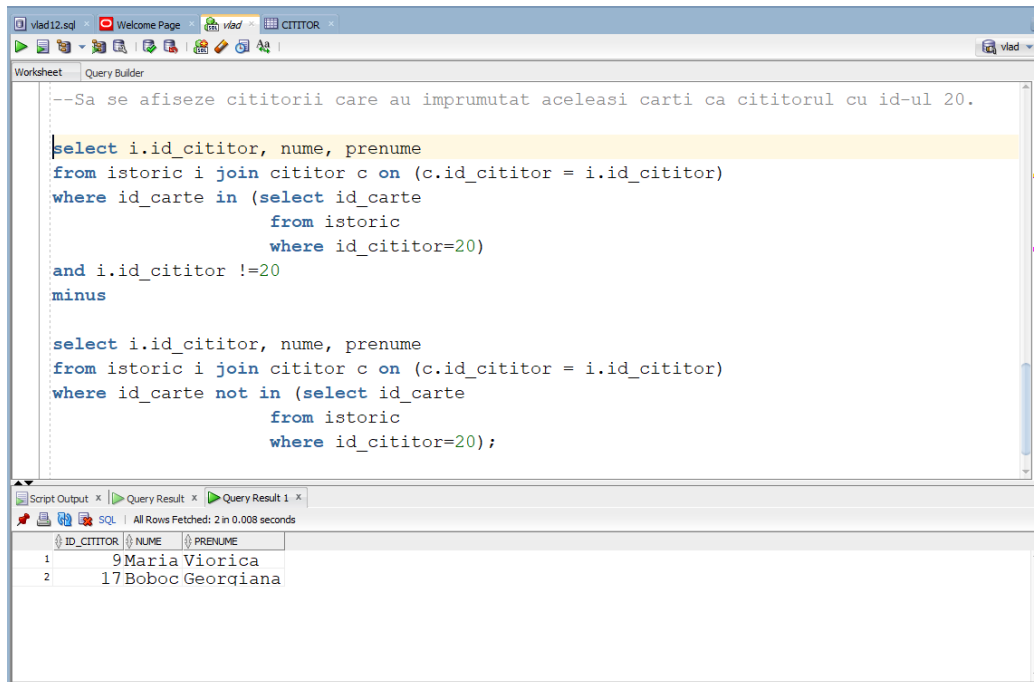
1. Să se afișeze id-ul, numele și prenumele cititorilor care au împrumutat doar cărți cu prețul sub 15 lei.

```
select i.id_cititor, nume, prenume
from istoric i join cititor c on (i.id_cititor=c.id_cititor)
where id_carte in (select id_carte from carte where pret<=15)
group by i.id_cititor,nume, prenume
having count(*) != (select count(id_carte) from carte where
pret<=15)
minus
select i.id_cititor, nume, prenume
from istoric i join cititor c on (i.id_cititor=c.id_cititor)
where id_carte not in (select id_carte from carte where pret<=15);
```



2. Să se afișeze cititorii care au împrumutat aceleași cărți ca cititorul cu id-ul 20.

```
select i.id_cititor, nume, prenume
from istoric i join cititor c on (c.id_cititor = i.id_cititor)
where id_carte in (select id_carte from istoric where id_cititor=20)
and i.id_cititor !=20
minus
select i.id_cititor, nume, prenume
from istoric i join cititor c on (c.id_cititor = i.id_cititor)
where id_carte not in (select id_carte from istoric where id_cititor=20);
```



15 Optimizarea unei cereri

Să se afișeze numele și prețul cărților scrise de Eliade și care au fost publicate de editura cu id-ul egal cu 3.

15.1 Cerere SQL

```
select c.num, c.pret
from carte c join scriere s on (c.id_carte=s.id_carte) join autor
a on (a.id_autor=s.id_autor)
where upper(a.num)='ELIADE' and c.id_editura=3;
```

15.2 Expresii algebrice

$R1 = \text{SELECT}(\text{AUTOR}, \text{nume}='Eliade')$ – eliminăm elementele nefolositoare

$R2 = \text{PROJECT}(R1, \text{id_autor})$ – îndepărtăm attributele nefolositoare

$R3 = \text{PROJECT}(\text{SCRIERE}, \text{id_autor}, \text{id_carte})$

$R4 = \text{SEMIJOIN}(R2, R3, \text{id_autor})$ – SEMIJOIN pentru că avem nevoie doar de id_carte

$R5 = \text{SELECT}(\text{CARTE}, \text{id_editura}=3)$

$R6 = \text{PROJECT}(R5, \text{id_carte}, \text{nume}, \text{pret})$

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

$\text{Rezultat} = R8 = \text{PROJECT}(R7, \text{nume}, \text{pret})$

15.3 Optimizare

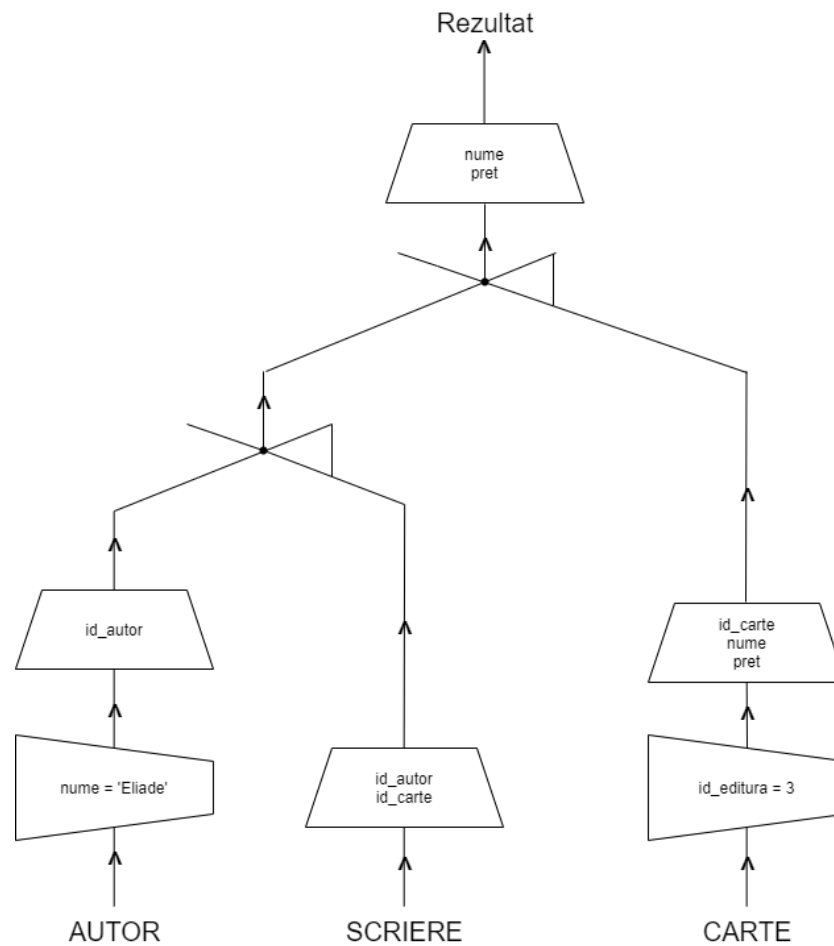
Cererea a fost proiectată optim de la început, deoarece s-a ținut cont de regulile de optimizare, și anume:

1. Selecțiile se execută cât mai devreme posibil, pentru a elimina elementele nefolositoare.

2. Produsele carteziane au fost înlocuite cu join-uri.
3. Proiecțiile se execută la început pentru a îndepărta atributele nefolositoare.

În plus, am înlocuit join-urile clasice cu semijoin-uri, deoarece aveam nevoie de date dintr-o singură relație, nu din ambele, astfel eliminând atributele nefolositoare.

15.4 Arbore algebric



16 Normalizarea BCNF, FN4, FN5. Aplicarea denormalizării

Pentru a exemplifica normalizarea în cadrul acestei secțiuni, se vor considera exemple care nu se regăsesc în modelul implementat, dar care au legătură cu acesta.

16.1 Forma normală Boyce-Codd

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.

Fie următoarea relație: R(cod cititor#, cod carte#, cod abonament, data imprumut, data restituire) - un cititor poate împrumuta mai multe cărți pe baza unui abonament.

$\{\text{cod cititor\#}, \text{cod carte\#}\} \rightarrow \{\text{cod abonament}, \text{data imprumut}, \text{data restituire}\}$

$\{\text{cod abonament}\} \rightarrow \{\text{cod cititor}\}$ (atributul cod abonament este o cheie candidat deoarece un abonament aparține unui singur cititor)

| COD CITI-TOR# | COD CARTE# | COD ABONA-MENT | DATA IM-PRUMUT | DATA RESTI-TUIRE |
|----------------------|-------------------|-----------------------|-----------------------|-------------------------|
| CIT1 | C1 | A1 | 14.05.2021 | 22.05.2021 |
| CIT1 | C2 | A1 | 23.05.2021 | 30.05.2021 |
| CIT2 | C3 | A2 | 28.04.2021 | 16.05.2021 |
| CIT3 | C2 | A3 | 14.05.2021 | 22.05.2021 |

Exemplu non-BCNF

Aplicând regula Casey-Delobel, se obțin relațiile:

R1(cod_cititor#, cod_carte#, data_imprumut, data_restituire)

R2(cod_cititor#, cod_abonament#)

| COD CITI-TOR# | COD CARTE# | DATA IMPRU-MUT | DATA RESTI-TUIRE |
|----------------------|-------------------|-----------------------|-------------------------|
| CIT1 | C1 | 14.05.2021 | 22.05.2021 |
| CIT1 | C2 | 23.05.2021 | 30.05.2021 |
| CIT2 | C3 | 28.04.2021 | 16.05.2021 |
| CIT3 | C2 | 14.05.2021 | 22.05.2021 |

R1

| COD CITITOR# | COD ABONAMENT# |
|---------------------|-----------------------|
| CIT1 | A1 |
| CIT2 | A2 |
| CIT3 | A3 |

R2

16.2 Forma normală 4 - FN4

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 (adica fiecare determinant trebuie să fie o cheie candidat) și nu conține relații $m:n$ independente.

Fie următorul exemplu: un angajat poate lucra la mai multe biblioteci, ocupând mai multe funcții (full-time/part-time). Astfel, dacă este necesară inserarea unui nou angajat se poate realiza această operație doar dacă se inserează și o bibliotecă, dar și o funcție.

Există următoarea relație: $R(\text{ANGAJAT}\#, \text{BIBLIOTECĂ}\#, \text{FUNCȚIE}\#)$.
În acest exemplu avem următoarele dependențe:

angajat- \rightarrow -bibliotecă

angajat- \rightarrow -funcție

| ANGAJAT# | BIBLIOTECĂ# | FUNCȚIE# |
|-----------------|--------------------|-----------------|
| Popescu | Nr. 1 | Manager |
| Popescu | Nr. 1 | Bibliotecar |
| Ionescu | Nr. 2 | Bibliotecar |
| Ionescu | Nr. 3 | Bibliotecar |
| Marinescu | Nr. 2 | Manager |

Exemplu non-FN4 (R)

Relația R se va descompune astfel:

$R_1(\text{ANGAJAT}\#, \text{BIBLIOTECĂ}\#)$

$R_2(\text{ANGAJAT}\#, \text{FUNCȚIE}\#)$

| ANGAJAT# | BIBLIOTECĂ# |
|-----------------|--------------------|
| Popescu | Nr. 1 |
| Ionescu | Nr. 2 |
| Ionescu | Nr. 3 |
| Marinescu | Nr. 2 |

R_1

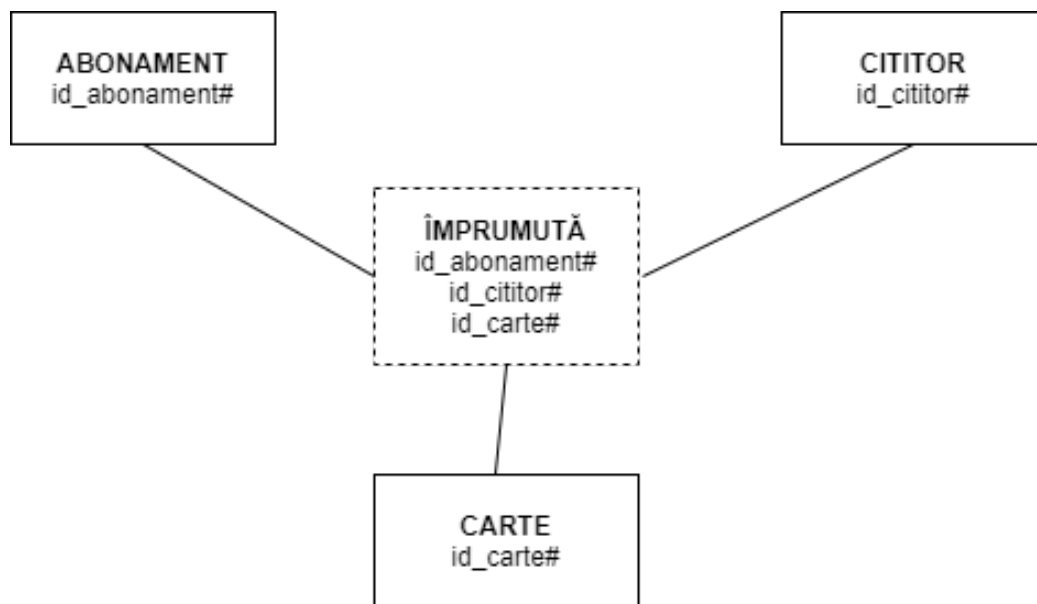
| ANGAJAT# | FUNCTIE# |
|-----------|-------------|
| Popescu | Manager |
| Popescu | Bibliotecar |
| Ionescu | Bibliotecar |
| Marinescu | Manager |

R2

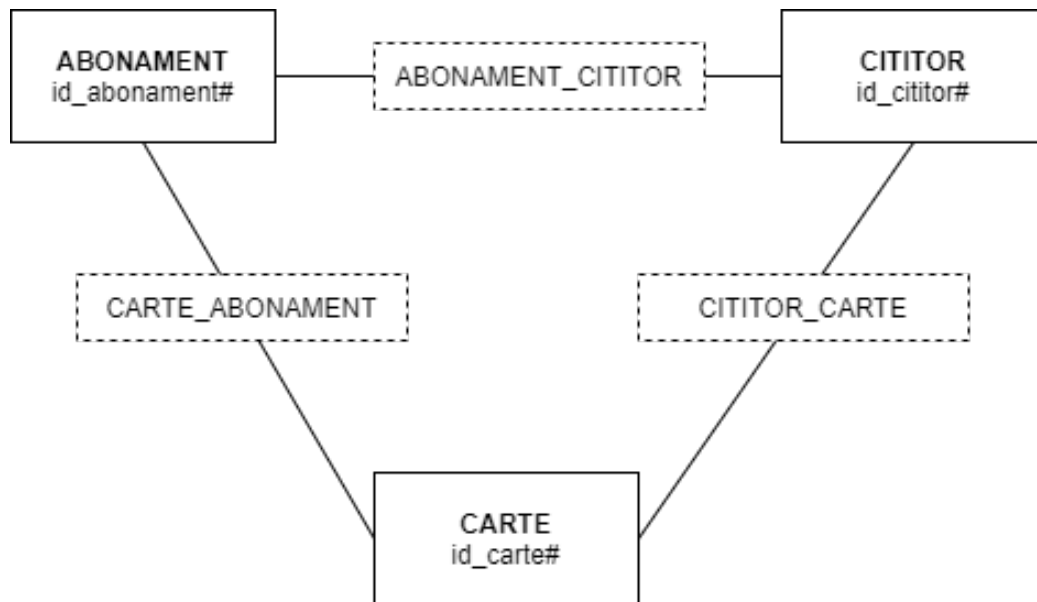
16.3 Forma normală 5 - FN5

FN5 își propune eliminarea redundanțelor care apar în relații m:n dependente. Intuitiv, o relație R este în forma normală 5 dacă și numai dacă este în FN4, iar aceasta nu conține dependențe ciclice.

Fie următorul exemplu: un cititor poate împrumuta mai multe cărți pe baza mai multor abonamente.



Relația de tip 3 precedentă poate fi echivalentă cu 3 relații de tip 3, doar dacă aceste relații sunt ciclice.



Relația fiind ciclă, atunci când vor fi efectuate toate join-urile, se va obține un rezultat echivalent cu cel obținut din relația de tip 3. O relație, pentru a fi în FN5, trebuie să fie în FN4 și să nu conțină dependențe ciclice.

Se observă astfel că cele trei relații de tip 2 compun o diagramă care conține dependențe ciclice, deci relația de mai sus nu se află în FN5. Pe de altă parte, relația de tip 3 este în FN5.

16.4 Aplicarea denormalizării

Obiectivul denormalizării constă în reducerea numărului de join-uri care trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans, ca făcând parte din proiectarea bazei de date.

Ca o regulă empirică, se poate afirma că, dacă performanțele nu sunt satisfăcătoare și relația are o rată de reactualizare scăzută, dar o rată a interogărilor foarte ridicată, denormalizarea poate constitui o opțiune viabilă.

Se consideră următorul exemplu: avem un tabel în care sunt stocate cărți, numit CARTE. Cărțile au și un atribut *preț*. Această coloană conține valori repetitive, același preț definind mai multe cărți.

În acest caz, dacă în baza de date există un tabel separat în care se află prețul împreună cu id-ul cărții căreia îi corespunde, este necesar procesul de denormalizare, în urma căruia atributul *greutate* se va plasa în tabelul CARTE, deoarece nu este eficient ca acest atribut să se afle într-un tabel separat.