Baza de date a unei biblioteci

Nițu Andreea-Georgiana

Grupa 334AA

Etapa întâi

Cerințe:

Pentru o mai bună organizare și evidență a cărților, biblioteca are nevoie de o bază de date, motiv pentru care am întocmit această listă de cerințe:

- Biblioteca are mai multe săli în care sunt păstrate cărțile, pe diferite categorii
- Într-o sală nu pot fi cărți de categorii diferite
- Pot fi mai multe săli aparținând unei singure categorii (ex: cărțile de beletristică în limba română nu încap într-o singură sală)
- Un student poate face mai multe împrumuturi de cărți
- Un împrumut poate avea mai multe cărți
- O carte poate fi împrumutată de mai multe ori

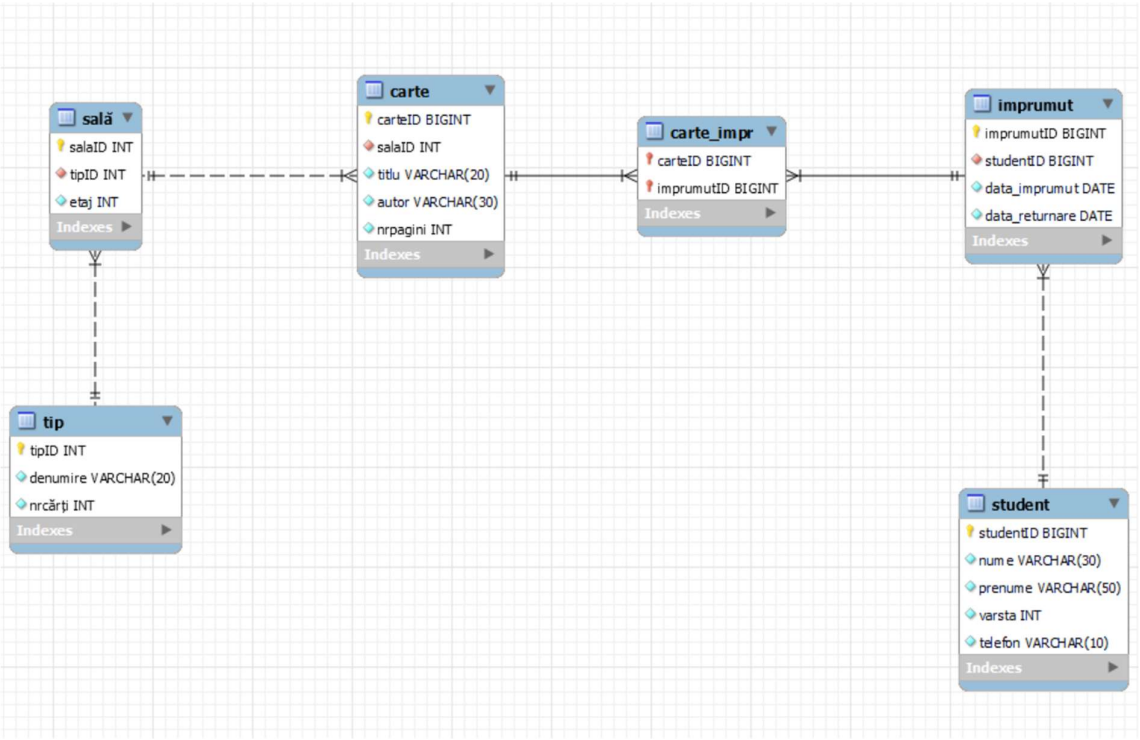
Constrângeri:

- Un student înregistrat în baza de date a bibliotecii trebuie să aibă minim 14 ani
- Biblioteca are 2 etaje (în afară de parter care este considerat etajul 0)

Relații:

Biblioteca prezentată are mai multe săli în care se află numeroase cărți. Fiecare sală poate avea un singur tip de cărți (ex: belestristică, limbi străine, cărți pentru copii), însă pot fi mai multe săli cu același tip de volume. De aceea, relația dintre tabelul **Tip** și dintre tabelul **sală** este de **1:N**. Relația dintre tabelul **sală** și tabelul **Carte** este tot de **1:N**, întrucât într-o sală sunt mai multe cărți, dar o carte nu se poate afla în mai multe săli diferite. Având în vedere că o carte poate apărea în diferite împrumuturi, iar un împrumut poate conține mai multe cărți, relația dintre tabelul **Carte** și tabelul **Imprumut** este de **N:N**, motiv pentru care a fost creat tabelul de legătură numit **Carte impr**. Relația de **1:N** este și între tabelul **Student** și tabelul **Imprumut**, deoarece un student poate face mai multe împrumuturi, dar un singur împrumut nu poate fi făcut de mai mulți studenți.

Arhitectura bazei de date:



Tabelele și tipurile de date:

sală	
salaID	int
tipID	int
etaj	int

tip	
tipID	int
denumire	varchar(20)
nrcărți	int

carte	
carteID	bigint
salaID	int
titlu	varchar(20)
autor	varchar(30)
nrpagini	int

imprumut	
imprumutID	bigint
studentID	bigint
data_imprumut	date
data_returnare	date

carte_impr	
carteID	bigint
imprumutID	bigint

student	
studentID	bigint
nume	varchar(30)
prenume	varchar(50)
varsta	int
telefon	varchar(10)

Etapa a doua

Pentru această etapa am realizat conexiunea bazei de date din MySQL cu interfața. Pentru acest lucru, am folosit limbajul de programare Python, importând bibliotecile necesare. Designul aplicației a fost realizat folosind HTML.

Exemplificarea conectării la baza de date:

C: > Users > maria > OneDrive > Desktop > flask_test > incerc2.py > home

```python
from flask import Flask, redirect, url_for, render_template, request, flash
from flask_mysqldb import MySQL

app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'admin'
app.config['MYSQL_DB'] = 'biblioteca'
app.config['SECRET_KEY'] = 'nu'

mysql = MySQL(app)
```

Etapa a treia

Pentru această etapa am făcut Insert, Update și Delete pentru tabelele carte și student.

- Insert carte/student

```
cur = mysql.connection.cursor()
cur.execute("INSERT INTO carte VALUES(%s,%s,%s,%s,%s)",
            (carteID, salaID, titlu, autor, nrpagini))
mysql.connection.commit()
cur.close()
```

```
cur = mysql.connection.cursor()
cur.execute("INSERT INTO student VALUES(%s,%s,%s,%s,%s)",
            (studentID, nume, prenume, varsta, telefon))
mysql.connection.commit()
cur.close()
```

- Update carte/student

```
cur = mysql.connection.cursor()
if salaID:
    cur.execute("update carte set salaID = %s where carteID = %s",(salaID,carteID))
    mysql.connection.commit()
if nrpagini:
    cur.execute("update carte set nrpagini = %s where carteID = %s",(nrpagini,carteID))
    mysql.connection.commit()


cur.close()
```

```
cur = mysql.connection.cursor()
if varsta:
    cur.execute("update student set varsta = %s where studentID = %s",(varsta,studentID))
    mysql.connection.commit()
if telefon:
    cur.execute("update student set telefon = %s where studentID = %s",(telefon,studentID))
    mysql.connection.commit()

cur.close()
```

- Delete carte/student

```
cur = mysql.connection.cursor()
cur.execute("delete from carte where titlu = %s", (titlu,))
mysql.connection.commit()
cur.close()
```

```
cur = mysql.connection.cursor()
cur.execute("delete from student where studentID = %s", (studentID,))
mysql.connection.commit()
cur.close()
```

Am realizat atât Join-uri simple, cât și Join-uri complexe. Join-urile sunt integrate în interfață atât sub formă de tabele în paginile Evidenta/Statistici, cât și sub forma de căutare după parametri variabili în pagina Carte.

- Join-uri simple

- Afișează sub forma unui tabel ce gen de cărți sunt la fiecare etaj

```
cur.execute("select distinct S.etaj as Etaj, T.denumire as Gen "
"from tip T, sală S "
"where S.tipID=T.tipID "
"order by S.etaj asc")
```

- Afișează sub forma unui tabel ce gen de cărți sunt în fiecare sală

```
cur.execute("select  T.denumire as Gen, S.salaID as SalaID "
"from  sală S, tip T "
"where S.tipID=T.tipID "
"order by S.salaID asc ")
```

- Afișează sub forma unui tabel studenții care au făcut măcar un împrumut

```
cur.execute("select distinct S.nume as Nume, S.prenume as Prenume "
"from student S, imprumut I "
"where S.studentID=I.studentID")
```

- Afișează sub forma unui tabel cărțile care au fost împrumutate măcar o dată

```
cur.execute("select distinct C.titlu as Titlu, C.autor as Autor "
"from carte_impr CI, carte C "
"where CI.carteID=C.carteID")
```

- Afișează ce cărți din bibliotecă aparțin genului literar introdus

```
if val == "1":
    cur = mysql.connection.cursor()
    rez = cur.execute("select T.denumire, C.titlu, C.autor "
    "from tip T, sală S, carte C "
    "where C.salaID=S.salaID  AND S.tipID=T.tipID  and T.denumire=%s", (cheie,))
```

- Afișează ce cărți din bibliotecă au mai multe pagini decât numărul introdus și genul literar căruia aparțin

```
if val == "2":
    cur = mysql.connection.cursor()
    rez = cur.execute("select T.denumire as Gen, C.titlu as Titlu, C.autor as Autor, C.nrpagini as NumarPagini "
    "from tip T, carte C, sală S "
    "where C.salaID=S.salaID  AND S.tipID=T.tipID and C.nrpagini>%s" , (cheie,))
```

- Join-uri complexe

- Afișează sub forma unui tabel cărțile împrumutate în cea mai recentă dată

```
cur.execute("select  I.data_imprumut,C.carteID,CC.titlu,  CC.autor "
            "from imprumut I, carte_impr C, carte CC "
            "where I.data_imprumut=(select max(I.data_imprumut) from imprumut I) "
            " and I.imprumutID=C.imprumutID and C.carteID=CC.carteID")
```

- Afișează sub forma unui tabel studenții, în ordine descrescătoare, în funcție de numărul de împrumuturi efectuate

```
cur.execute("select S.studentID, S.nume, S.prenume, (select COUNT(*) "
            "from imprumut I "
            "where I.studentID=S.studentID) as NumarImprumuturi "
            "from student S "
            "order by NumarImprumuturi DESC")
```

- Afișează sub forma unui tabel, în ordine descrescătoare, numărul de săli pe care îl are fiecare gen literar

```
cur.execute("select T.tipID, T.denumire, (select COUNT(*) from sală S where S.tipID=T.tipID) as nrsali "
            "from tip T "
            "order by nrsali DESC")
```

- Afișează sub forma unui tabel cărțile, în ordine descrescătoare, în funcție de câte ori au fost împrumutate

```
cur.execute("select C.titlu, C.autor, (select COUNT(*) from carte_impr CC where CC.carteID=C.carteID) as NrImprumuturi "
            "from carte C "
            "order by NrImprumuturi DESC")
```

- Afișează ce cărți au fost împrumutate în data introdusă

```
if val == "3":
    cur = mysql.connection.cursor()
    rez = cur.execute("select CC.titlu,  CC.autor "
                      "from imprumut I, carte_impr C, carte CC "
                      "where I.data_imprumut in (select I.data_imprumut from imprumut I where I.data_imprumut=%s)  "
                      "and I.imprumutID=C.imprumutID and C.carteID=CC.carteID" , (cheie,))
```