Rusu Diana Elena
Grupa 331AB

Proiect Baze de Date
Evidența programărilor într-un salon de cosmetică

Cerință:

-Un client face o programare la salon la un anumit angajat, la o categorie sau la un serviciu, le poate preciza pe toate trei sau doar una, dar cel mai important este serviciul de care vrea să beneficieze. După această precizare, se va completa ora, data la care va avea loc programarea și angajatul care va fi dispobil în acea perioadă de timp și va efectua programarea. Un angajat poate face parte doar dintr-o singură categorie, dar poate presta mai multe servicii. De exemplu: o coafeză poate tunde, vopsi, coafa, împleti, etc. Categoriile sunt împărțite exact ca într-un salon: coafor, cosmetică și unghii. Într-o categorie pot exista mai multe servicii.

- La final clientul face o plată, care se calculează în funcție de serviciile din programarea sa, iar programarea va intra la istoric programări, ce se va lega de client și angajat pentru a putea găsi cu ușurință o programare efectuată in trecut.

-Aplicația la care am lucrat este utilă pentru angații de la recepția salonului, aceștia primesc un cont de logare pentru a accesa aplicația, acesta poate accesa 4 tabele prin butoane, cum ar fi: Clienți, Angajați, Programări și Plăți. La aceste tabele utilizatorul poate adauga informații in tabele, să le actualizeze sau chiar să le șteargă. Pentru fiecare tabelă, am rezolvat câte o interograre simplă sau complexă. Acestea sunt următoarele:

Interogări simple

1) Să se afișeze numele și prenumele clientului împreună cu numărul de programări pe care acesta le are

```
cursor.execute("""
SELECT Clienti.Nume, Clienti.Prenume, COUNT(Programari.ClientID) AS NumarProgramari
FROM Clienti
LEFT JOIN Programari ON Clienti.ClientID = Programari.ClientID
GROUP BY Clienti.ClientID, Clienti.Nume, Clienti.Prenume
""")
```

2) Afișează pentru orice client căutat după nume și prenume ora și data la care are programarea

```
cursor.execute("""
SELECT Programari.Data, Programari.Ora
FROM Programari
JOIN Clienti ON Clienti.ClientID = Programari.ClientID
WHERE Clienti.Nume = ? AND Clienti.Prenume = ?
""", (nume_input, prenume_input))
```

3) Să se afișeze numele și prenumele angajaților din fiecare categorie

```
cursor.execute("""
    SELECT Categorii.Denumire, Angajati.Nume, Angajati.Prenume
    FROM Angajati
    INNER JOIN Categorii ON Angajati.CategorieID = Categorii.CategorieID
    ORDER BY Categorii.CategorieID
""")
```

4) Să se afișeze numele și prenumele angajaților, împreună cu numărul total de ore de la programari, implicit numărul total de ore lucrate

```
cursor.execute("""
    SELECT Angajati.Nume, Angajati.Prenume, SUM(ProgramariAngajati.NrOreProgramare) AS TotalOre
    FROM Angajati
    LEFT JOIN ProgramariAngajati ON Angajati.AngajatID = ProgramariAngajati.AngajatID
    LEFT JOIN Programari ON ProgramariAngajati.ProgramareID = Programari.ProgramareID
    GROUP BY Angajati.AngajatID, Angajati.Nume, Angajati.Prenume
    ORDER BY Angajati.AngajatID
""")
```

5) Să-mi ia ID-ul de la programare cu un anumit ClientID, care e la o anumită dată și oră, acest ID-ul este folosit ulterior la UPDATE-ul plății ca să știm pentru ce programare se face plata.

```
#INTROGARE SIMPLA 5
# Obține ProgramareID pe baza ClientID-ului și datei/orei introduse
cursor.execute(""" SELECT Programari.ProgramareID
                   FROM Programari
                   JOIN Clienti ON Programari.ClientID = Clienti.ClientID
                   WHERE Clienti.ClientID = ? AND Programari.Data = ? AND Programari.Ora = ?""", (client_id, data.get(), ora.get()))
programare_id = cursor.fetchone()[0]
```

6) La fiecare nume și prenume al unui client căutat de utilizator să se afișeze totalul de plată

```
cursor.execute("""
    SELECT c.Nume, c.Prenume, SUM(p.SumaPlata) AS TotalPlati
    FROM Clienti c
    INNER JOIN Programari pr ON c.ClientID = pr.ClientID
    INNER JOIN Plati p ON pr.ProgramareID = p.ProgramareID
    WHERE c.Nume = ? AND c.Prenume = ?
    GROUP BY c.Nume, c.Prenume
""", (nume.get(), prenume.get()))
```

Interogari complexe

1) Angajații cu salariul cel mai mare pe fiecare categorie, afișați în ordine descrescătoare

```
cursor.execute("""
    SELECT Denumire, Nume, Prenume, Salariu
    FROM Categorii c
    INNER JOIN Angajati a ON c.CategorieID = a.CategorieID
    WHERE Salariu IN (
        SELECT MAX(Salariu)
        FROM Angajati
        WHERE CategorieID = c.CategorieID
        GROUP BY CategorieID
    )
    ORDER BY Salariu DESC
""")
```

2) Anul cu cei mai mulți oameni angajați și numărul lor

```
cursor.execute("""
        SELECT TOP 1 Anul, NumarAngajati
        FROM (
            SELECT YEAR(DataAngajarii) AS Anul, COUNT(*) AS NumarAngajati
            FROM Angajati
            GROUP BY YEAR(DataAngajarii)
            ) AS AngajatiPerAn
        ORDER BY NumarAngajati DESC
    """)
```

3) Media prețurilor serviciilor de pe fiecare categorie

```
cursor.execute("""
        SELECT Denumire, CAST(AVG_Pret AS DECIMAL(10, 2)) AS MediePret
        FROM (
            SELECT Categorii.Denumire,
                ROUND(AVG(Servicii.Pret), 2) AS AVG_Pret
            FROM Servicii
            JOIN Categorii ON Servicii.CategorieID = Categorii.CategorieID
            GROUP BY Categorii.Denumire
        ) AS Subcerere
    """)
```

4) Afișează numele și prenumele clienților care au avut mai mult de 3 programări în ultimele 6 luni

```
cursor.execute("""
    SELECT Clienti.Nume, Clienti.Prenume, COUNT(Programari.ProgramareID) AS NumarProgramari
    FROM Clienti
    INNER JOIN Programari ON Clienti.ClientID = Programari.ClientID
    WHERE Clienti.Nume = ? AND Clienti.Prenume = ?
        AND Programari.Data > DATEADD(MONTH, -6, GETDATE())
    GROUP BY Clienti.Nume, Clienti.Prenume
    HAVING COUNT(Programari.ProgramareID) > 3
""", (nume_entry.get(),prenume_entry.get()))
```

INSERT

1) In Clienti

```
cursor.execute("INSERT INTO Clienti(Nume,Prenume,Adresa,NumarTelefon,Email) VALUES (?,?,?,?,?)",
               (
                   nume.get(),
                   prenume.get(),
                   adresa.get(),
                   numartelefon.get(),
                   email.get()
               )
               )
```

2) Angajati

```
cursor.execute("""
    INSERT INTO Angajati(CategorieID, Nume,Prenume,Functie,NumarTelefon,Email,CNP,Adresa,Sex,DataNasterii,DataAngajarii,Salariu)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
""", (
```

3) Programari

```
cursor.execute("SELECT ClientID FROM Clienti WHERE Nume = ? AND Prenume = ?", (nume.get(),prenume.get()))
client_id = cursor.fetchone()[0]

cursor.execute("INSERT INTO Programari(ClientID, Data, Ora) VALUES (?, ?, ?)",
               (client_id, data.get(), ora.get()))
```

4) Plati

```
cursor.execute("SELECT ClientID FROM Clienti WHERE Nume = ? AND Prenume = ?", (nume.get(), prenume.get()))
client_id = cursor.fetchone()[0]

# Obține ProgramareID pe baza ClientID-ului din Programari
cursor.execute("SELECT ProgramareID FROM Programari WHERE ClientID = ? AND Data = ? AND Ora = ?", (client_id,data.get(),ora.get()))
programare_id = cursor.fetchone()[0]

cursor.execute("INSERT INTO Plati(ProgramareID, SumaPlata) VALUES (?, ?)",
               (programare_id, suma.get()))
```

UPDATE

1) Clienti

```
cursor.execute("""
    UPDATE Clienti
    SET Adresa=?, NumarTelefon=?, Email=?
    WHERE Nume=? AND Prenume=?
""", (client_adresa, client_numartelefon, client_email, client_nume, client_prenume))
```

2) Angajati

```
cursor.execute("""
    UPDATE Angajati
    SET CategorieID=?, Functie=?, NumarTelefon=?, Email=?, CNP=?, Adresa=?, Sex=?, DataNasterii=?, DataAngajarii=?, Salariu=?
    WHERE Nume=? AND Prenume=?
""", (
```

3) Programari

```
# Obține ClientID pe baza numelui și prenumelui introduse
cursor.execute("SELECT ClientID FROM Clienti WHERE Nume = ? AND Prenume = ?", (nume.get(), prenume.get()))
client_id = cursor.fetchone()[0]

# Actualizare programare în funcție de ClientID, dată și oră
cursor.execute("UPDATE Programari SET Data = ?, Ora = ? WHERE ClientID = ? AND Data = ? AND Ora = ?",
               (data_noua.get(), ora_noua.get(), client_id, data_veche.get(), ora_veche.get()))
```

4) Plati

```
# Actualizează în tabela Plati
cursor.execute("UPDATE Plati SET SumaPlata = ? WHERE ProgramareID = ?", (suma.get(), programare_id))
```

DELETE

1) Angajati

```
cursor.execute("DELETE FROM Angajati WHERE Nume=? AND Prenume=?", (nume.get(),prenume.get()))
```

2) Programari

```
# Obține ClientID pe baza numelui și prenumelui introduse
cursor.execute("SELECT ClientID FROM Clienti WHERE Nume = ? AND Prenume = ?", (nume.get(), prenume.get()))
client_id = cursor.fetchone()[0]

# Ștergere programare în funcție de ClientID, dată și oră
cursor.execute("DELETE FROM Programari WHERE ClientID = ? AND Data = ? AND Ora = ?",
               (client_id, data.get(), ora.get()))
```

3) Plati

```
# Șterge plata asociată Programarii
cursor.execute("DELETE FROM Plati WHERE ProgramareID = ?", (programare_id,))
```

Descrierea tabelelor și relațiilor dintre tabele:

1:1

- O programare poate avea doar o plată, iar o plată se poate face doar la o programare

1:M

- Un client poate avea mai multe programări, dar o programare are doar un client
- Un client poate avea mai multe plăți, dar o plată poate avea doar un client
- Un angajat poate face parte dintr-o singură categorie, iar o categorie are mai mulți angajați
- O categorie poate avea mai multe servicii, iar un serviciu poate face parte doar dintr-o categorie

M:M

- O programare poate avea mai multi angajati, iar un angajat poate avea mai multe programari
- O programare poate avea mai multe servicii, iar un serviciu poate apărea la mai multe programări
- O programare poate avea mai mulți angajați, iar un angajat poate avea mai multe programări

1:1	1:M	M:M	Tabele in plus pt M:M
Plati:Programari	Clienti:Progamari	Programari:Categorii	ProgramariCategorii
	Client:Plati	Programari:Servicii	ProgramariServicii
	Categorii:Angajat	Programari:Angajat	ProgramariAngajati
	Categorii:Servicii		

1. ****Clienti:****
   - ClientID (cheie primară cu autoincrementare)
   - Nume
   - Prenume
   - Adresa
   - NumarTelefon(length:10)
   - Email

Pot fi NULL: Adresa, Email

2. ****Angajati:****
   - AngajatID (cheie primară cu autoincrementare)
   - CategorieID(cheie externa către tabela Categorii)
   - Nume
   - Prenume
   - Functie
   - NumarTelefon (length:10)
   - Email
   - CNP (Unique, length:13)
   - Adresa
   - Sex (Check Sex(M,F) cu Default Sex(F))
   - DataNasterii
   - DataAngajarii
   - Salariu

Pot fi NULL: Email, Sex, DataNasterii

3. ****Servicii:****
   - ServiciuID (cheie primară cu autoincrementare)
   - CategorieID(cheie externă către tabela Categorii)
   - Denumire
   - Descriere
   - DurataMin
   - Pret

Poate fi NULL: Descriere

4. ****Programari:****
   - ProgramareID (cheie primară cu autoincrementare)
   - ClientID (cheie externă către tabela Clienti)
   - Data
   - Ora

5. ****Plati:****
   - PlataID (cheie primară cu autoincrementare)
   - DataPlata
   - SumaPlata
   - ProgramareID (cheie externă către tabela Programari)
   - ClientID (cheie externă către tabela Clienti)

6. ****Categorie:****
   - CategorieID (cheie primară cu autoincrementare)
   - Denumire
   - Descriere

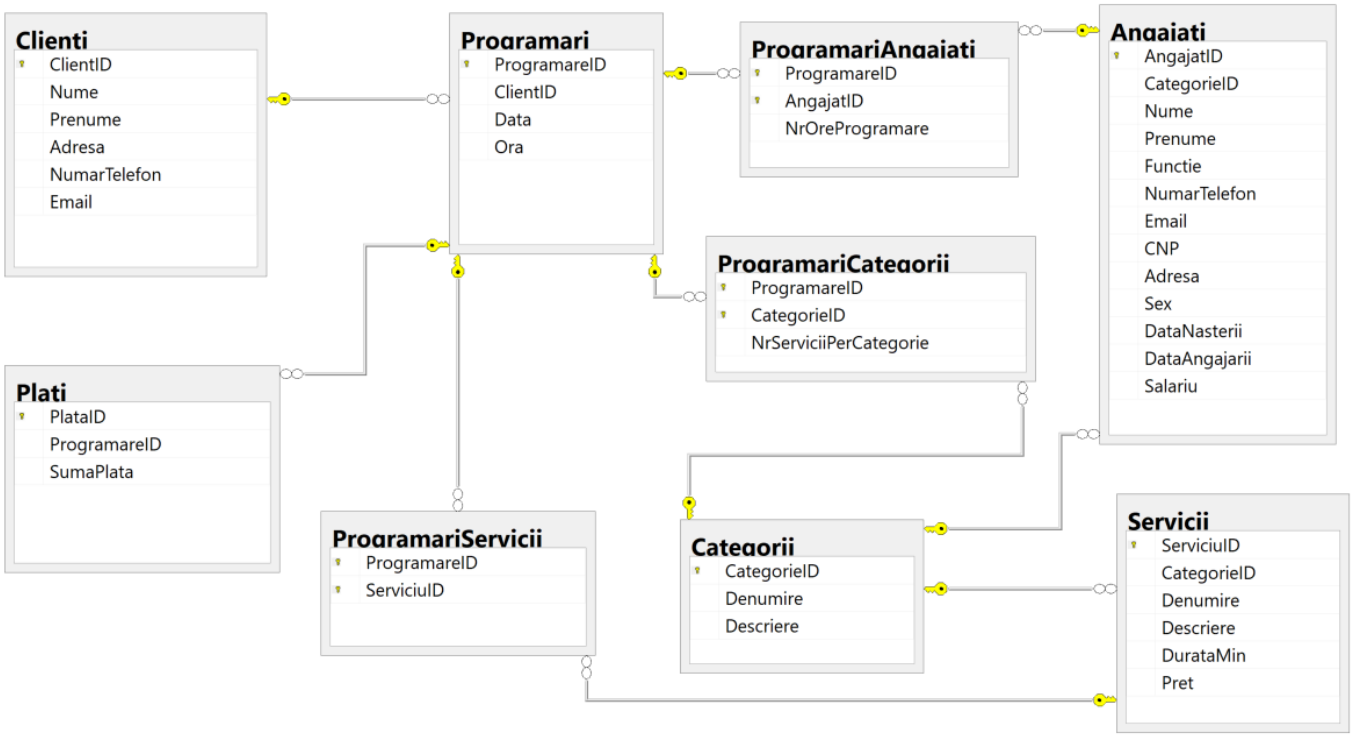
   Poate fi NULL: Descriere

7.** **ProgramariCategorii****
   (cheie primara compusa)
   -ProgramareID
   -CategorieID
   -NrProgramariPerCategorie

8.****ProgramariServicii****
   (cheie primara compusa)
   -ProgramareID
   -CategorieID
   -NrProgramariPerServiciu

9.****ProgramariAngajati****
   (cheie primara compusa)
   -ProgramareID
   -AngajatID
   -NrProgramariPerAngajat

Diagrama:


Clienti
ClientID
Nume
Prenume
Adresa
NumarTelefon
Email
Programari
ProgramareID
ClientID
Data
Ora
ProgramariAngajati
ProgramareID
AngajatID
NrOreProgramare
Angajati
AngajatID
CategorieID
Nume
Prenume
Functie
NumarTelefon
Email
CNP
Adresa
Sex
DataNasterii
DataAngajarii
Salariu
ProgramariCategorii
ProgramareID
CategorieID
NrServiciiPerCategorie
Plati
PlataID
ProgramareID
SumaPlata
ProgramariServicii
ProgramareID
ServiciuID
Categorii
CategorieID
Denumire
Descriere
Servicii
ServiciuID
CategorieID
Denumire
Descriere
DurataMin
Pret