**Aplicație cu baze de date relaționale având interfață virtuală**

**Evidența programărilor la un salon de cosmetică**

Neagu Alexandra Carola

Grupa 331 AA

**1.Tematica aplicației**

28. Aplicație pentru evidența programărilor la un salon de cosmetică.

În cadrul aplicației vom lua în considerare următoarele lucruri:

* Un client al salonului poate avea mai multe programări
  + Pentru fiecare client se rețin: nume, prenume, CNP, număr de telefon și email
* O programare conție data și ora la care se desfășoară serviciile incluse
* Serviciile se împart în diferite categorii, iar fiecare serviciu aparține unei singure categorii
* Un angajat al salonului poate presta o singură categorie de servicii
  + Pentru fiecare angajat se rețin următoarele informații: nume, prenume, CNP, adresă, număr de telefon, data nașterii, data angajării și categoria de servicii pe care o prestează
* În cadrul fiecărei categorii de servicii pot fi angajate mai multe persoane
* Fiecare serviciu oferit are prețul său
* Salonul este împărțit în mai multe săli (fiecare sală cu denumirea ei). Într-o sala se poate presta o singură categorie de servicii (ex: doar masaj sau doar unghii sau doar păr, etc), dar fiecare categorie de servicii are mai multe săli la dispoziție
* În sală nu pot intra mai mulți clienți în același timp. Sala trebuie eliberată pentru ca următorul client să poată aibă acces la ea (astfel nu e nevoie sa luam în considerare capacitatea sălii)
* O programare poate conține mai multe servicii

**2.Identificarea tabelelor**

Din cele relatate identificăm următoarele entități cu care lucrează salonul: clienți, angajați, săli, servicii și programări. Pentru fiecare dintre aceste entități vom avea câte un tabel în baza noastră de date. Astfel vom avea următoarele 5 tabele: Clienți, Angajați, Săli, Servicii, Categorii și Programări.

**3.Identificarea câmpurilor**

|  |
| --- |
| Clienți |
| Nume |
| Prenume |
| CNP |
| Telefon |
| Email |

|  |
| --- |
| Săli |
| Denumire |

|  |
| --- |
| Programare |
| Data |
| Ora |

|  |
| --- |
| Categorii |
| Denumire |
| Nr angajați |

|  |
| --- |
| Servicii |
| Denumire |
| Preț |

Din motive de confidențialitate adresa clienților nu este necesară pentru a efectua o programare, astfel, aceasta nu se va afla în baza de date.

|  |
| --- |
| Angajați |
| Nume |
| Prenume |
| CNP |
| Data nașterii |
| Data angajării |
| Telefon |
| Oraș |
| Județ |
| Stradă |
| Număr |

Pentru respectarea caracteristicilor tabelului și câmpului ideal, adresa angajaților am spart-o în stradă, număr, oraș, județ/sector. În acest fel, am evitat stocarea unui câmp cu mai multe părți (multiplu).

**4.Stabilirea cheilor primare**

|  |
| --- |
| Angajați |
| ID Angajat *CP* |
| Nume |
| Prenume |
| CNP |
| Data nașterii |
| Data angajării |
| Telefon |
| Oraș |
| Județ |
| Stradă |
| Număr |

Pentru identificarea în mod unic a fiecărei înregistrări vom folosi o cheie primara artificială.

|  |
| --- |
| Clienți |
| ID Client *CP* |
| Nume |
| Prenume |
| CNP |
| Telefon |
| Email |

|  |
| --- |
| Servicii |
| ID Serviciu *CP* |
| Denumire |
| Preț |

|  |
| --- |
| Săli |
| ID Sală *CP* |
| Denumire |

|  |
| --- |
| Categorii |
| ID Categorie  *CP* |
| Denumire |
| Nr angajați |

|  |
| --- |
| Programare |
| ID Programare *CP* |
| Data |
| Ora |

Câmpurile ID Client, ID Angajat, ID Serviciu, ID Sală, ID Categorie, ID Programare sunt cheile primare ale tabelelor din baza de date a salonului.

**5.Identificarea relațiilor dintre tabele**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Clienți | Angajați | Săli | Servicii | Categorii | Programări |
| Clienți | - | 1:n | 1:n | 1:n | - | 1:n |
| Angajați | 1:n | - | 1:n | 1:n | 1:1 | 1:n |
| Săli | 1:n | 1:n | - | 1:n | 1:1 | 1:n |
| Servicii | 1:n | 1:n | 1:n | - | 1:1 | 1:n |
| Categorii | - | 1:n | 1:n | 1:n | - | 1:n |
| Programări | 1:1 | 1:n | 1:n | 1:n | 1:n | - |

Nu avem relație angajat-angajat deoarece am decis că șeful este proprietarul iar acesta nu prestează niciun serviciu. Astefl acesta nu este propriuzis angajat ci doar proprietar.

Din moment ce între relațiile serviciu-sală, angajat-sală, client-angajat și programare-sală sunt relații de n:n am ales sa folosesc ca tabel de legătură tabelul Detalii Programare.

**6.Structura bazei de date**

**Clienți (**ID Client, Nume, Prenume, CNP, Telefon, Email**)**

**Angajați (**ID Angajat**,** Nume, Prenume, CNP, Telefon, Data angajării, Data nașterii, Oraș, Județ, Stradă, Nr, ID Categorie**)**

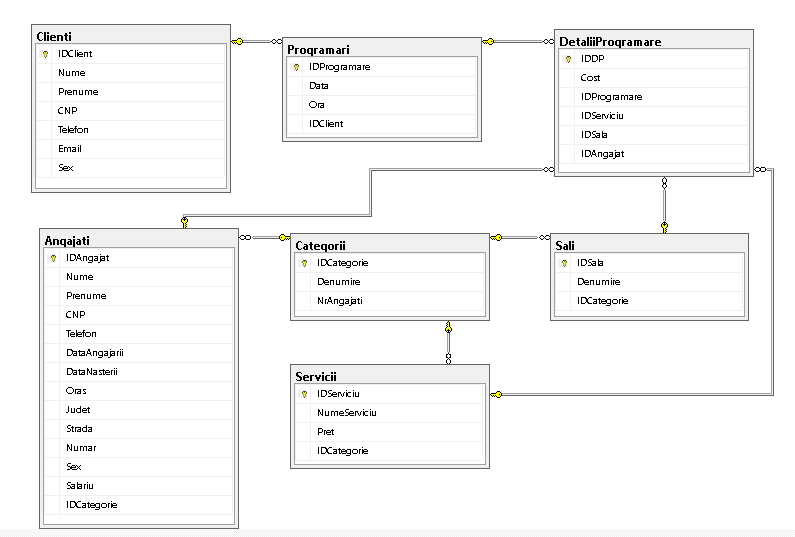
**Săli (**ID Sală, Denumire, ID Categorie**)**

**Categorii (**ID Categorie, Denumire, Nr angajați**)**

**Servicii (**ID Serviciu, Nume serviciu, Preț, ID Categorie**)**

**Programări (**ID Programare, Dată, Oră, ID Client**)**

**Detalii Programare (**ID DP**,** ID Programare, ID Serviciu, ID Sală, Cost, ID Angajat**)**

ID-urile subliniate cu 2 linii și evidențiate cu albastru sunt cheile externe.

**7.Creare tabele și constrângeri**

CREATE TABLE Clienti (

IDClient SERIAL PRIMARY KEY,

Nume VARCHAR(50) NOT NULL,

Prenume VARCHAR(50) NOT NULL,

CNP VARCHAR(13) NOT NULL,

Telefon VARCHAR(10),

Email VARCHAR(10),

Sex CHAR(1) DEFAULT 'M' CHECK (sex IN ('M', 'F'))

);

CREATE TABLE Categorii (

IDCategorie SERIAL PRIMARY KEY,

Denumire NVARCHAR(50) NOT NULL,

NrAngajati INT NOT NULL

);

CREARE TABLE Angajati (

IDAngajat SERIAL PRIMARY KEY,

Nume VARCHAR(50) NOT NULL,

Prenume VARCHAR(50) NOT NULL,

CNP VARCHAR(13) NOT NULL,

Telefon VARCHAR(10),

DataAngajarii DATETIME NULL,

DataNasterii DATETIME NULL,

Oras NVARCHAR(50) NOT NULL,

Judet NVARCHAR(50) NOT NULL,

Strada NVARCHAR(50) NOT NULL,

Numar VARCHAR(10) NOT NULL,

Sex CHAR(1) DEFAULT 'M' CHECK (sex IN ('M', 'F')),

Salariu NUMERIC(8,2) NOT NULL,

IDCategorie SERIAL REFERENCES Categorii(IDCategorie)

);

CREATE TABLE Sali (

IDSala SERIAL PRIMARY KEY,

Denumire NVARCHAR(50) NOT NULL,

IDCategorie SERIAL REFERENCES Categorii(IDCategorie)

);

CREATE TABLE Servicii (

IDServiciu SERIAL PRIMARY KEY,

NumeServiciu VARCHAR(50) NULL,

Pret INT NULL,

IDCategorie SERIAL REFERENCES Categorii(IDCategorie)

);

CREATE TABLE Programari (

IDProgramare SERIAL PRIMARY KEY,

Data DATETIME NULL,

Ora TIME(7) NULL

);

CREATE TABLE DetaliiProgramare (

IDDP SERIAL PRIMARY KEY,

Cost INT NULL,

ID Programare SERIAL REFERENCES Programari(IDProgramare),

IDServiciu SERIAL REFERENCES Servicii(IDServiciu),

IDSala SERIAL REFERENCES Sali(IDSala),

IDAngajat SERIAL REFERENCES Angajati(IDAngajat)

);

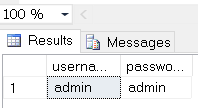
**8.Proiectarea interfeței vizuale**

Aplicația a fost proectată folosint C#, backend-ul se folosește de WindowsFormsAplication, iar baza de date este construită cu ajutorul SQL Server.

**9.Corelarea interfeței vizuale cu baza de date**

Un administrator se poate conecta la baza de date cu nume si parola.

Am creat o pagina de LOG IN prin care am făcut conectarea la baza de date și fără de care nu există posibilitatea de a face nicio operație asupra bazei de date.

Operația de LOG IN s-a efectuat prin intermediul tabelei suplimentar introduse AUTH, care nu este legată practic de restul tabelelor prin nicio cheie. E folosită strict pentru logare.

Singura persoană care se poate conecta la baza de date este administratorul, care poate fi proprietarul salonului sau angajatul care răspunde la telefon în momentul când un client sună să facă o programare.

**10.Insert, Update, Delete**

În fiecare caz mai întâi se verifică dacă exista clientul/categoria cu ID-ul respectiv.

Dacă există, în cazul pentru Insert nu mai inserează iar pentru Update și Delete înseamnă că poate face operația. Dacă nu există, in cazul pentru Insert poate face inserarea, dar pentru Update și Delete înseamnă că nu poate efectua operația.

com.CommandText = "select count(\*) from Clienti where IDClient = @IDClient";

com.Parameters.AddWithValue("@IDClient", textBox1.Text);

com.CommandText = "select count(\*) from Categorii where IDCategorie = @IDCategorie";

com.Parameters.AddWithValue("@IDCategorie", textBox1.Text);

**Insert**

Pentru tabela Clienți:

com.CommandText = "insert into Clienti(Nume,Prenume,CNP,Telefon,Email,Sex) values('" + textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text + "','" + textBox5.Text + "','" + textBox6.Text + "','" + textBox7.Text + "')";

Pentru tabela Categorii:

com.CommandText = "insert into Categorii(Denumire,NrAngajati) values('" + textBox2.Text + "','" + textBox3.Text + "')";

**Update**

Pentru tabela Clienți:

StringBuilder updateQuery = new StringBuilder("update Clienti set ");

if (!string.IsNullOrEmpty(textBox2.Text))

{

updateQuery.Append("Nume = @Nume, ");

com.Parameters.AddWithValue("@Nume", textBox2.Text);

nr++;

}

if(!string.IsNullOrEmpty(textBox3.Text))

{

updateQuery.Append("Prenume = @Prenume, ");

com.Parameters.AddWithValue("@Prenume", textBox3.Text);

nr++;

}

if(!string.IsNullOrEmpty(textBox4.Text))

{

updateQuery.Append("CNP = @CNP, ");

com.Parameters.AddWithValue("@CNP", textBox4.Text);

nr++;

}

if(!string.IsNullOrEmpty(textBox5.Text))

{

updateQuery.Append("Telefon = @Telefon, ");

com.Parameters.AddWithValue("@Telefon", textBox5.Text);

nr++;

}

if(!string.IsNullOrEmpty(textBox6.Text))

{

updateQuery.Append("Email = @Email, ");

com.Parameters.AddWithValue("@Email", textBox6.Text);

nr++;

}

if(!string.IsNullOrEmpty(textBox7.Text))

{

updateQuery.Append("Sex = @Sex, ");

com.Parameters.AddWithValue("@Sex", textBox7.Text);

nr++;

}

updateQuery.Length -= 2;

updateQuery.Append(" where IDClient = @IDClient");

Pentru tabela Categorii:

StringBuilder updateQuery = new StringBuilder("update Categorii set ");

if (!string.IsNullOrEmpty(textBox2.Text))

{

updateQuery.Append("Denumire = @Denumire, ");

com.Parameters.AddWithValue("@Denumire", textBox2.Text);

nr2++;

}

if (!string.IsNullOrEmpty(textBox3.Text))

{

updateQuery.Append("NrAngajati = @NrAngajati, ");

com.Parameters.AddWithValue("@NrAngajati", textBox3.Text);

nr2++;

}

updateQuery.Length -= 2;

updateQuery.Append(" where IDCategorie = @IDCategorie");

**Delete**

Pentru tabela Clienți:

com.CommandText = "delete from Clienti where IDCLient= '"+ textBox1.Text+"'";

Pentru tabela Categorii:

com.CommandText = "delete from Categorii where IDCategoriet= '" + textBox1.Text + "'";

**11.Interogări simple (cu JOIN)**

QS1: Cele mai mari 2 salarii ale angajatilor ce lucreaza pentru categoria pe care o doriti.

SELECT TOP 2 A.Salariu

FROM Angajati A INNER JOIN Categorii C ON A.IDCategorie = C.IDCategorie

WHERE C.Denumire = '" +textBox2.Text+ "'

ORDER BY A.Salariu DESC

QS2: Clienții care au programări pentru servicii cu un cost mai mare decât 50.

SELECT C.IDClient, C.Nume, C.Prenume, S.NumeServiciu, DP.Cost

FROM Clienti C JOIN Programari P ON C.IDClient = P.IDClient JOIN DetaliiProgramare DP ON P.IDProgramare = DP.IDProgramare JOIN Servicii S ON DP.IDServiciu = S.IDServiciu

WHERE DP.Cost > 50

QS3: Nume si prenumele angajatilor care lucreaza in categoria Machiaj.

SELECT A.Nume, A.Prenume

FROM Angajati A INNER JOIN Categorii C ON A.IDCategorie = C.IDCategorie

WHERE C.Denumire = 'Machiaj'

QS4: Numarul de servicii pe care le coordoneaza fiecare categorie.

SELECT C.Denumire, COUNT(S.IDServiciu) AS NrServicii

FROM Categorii C LEFT JOIN Servicii S ON C.IDCategorie= S.IDCategorie

GROUP BY C.Denumire

QS5: Numele si prenumele clientilor care au cel putin 2 servicii programate.

SELECT Cl.Nume, Cl.Prenume

FROM Clienti Cl JOIN Programari P ON Cl.IDClient=P.IDClient JOIN DetaliiProgramare Dp ON P.IDProgramare=Dp.IDProgramare

GROUP BY Cl.Nume, Cl.Prenume

HAVING COUNT(DISTINCT Dp.IDServiciu) >= 2

QS6: Numele si prenumele clientilor care au cel putin o programare in sala cu ID-ul 5.

SELECT DISTINCT C.IDClient, C.Nume, C.Prenume

FROM Clienti C JOIN Programari P ON C.IDClient = P.IDClient JOIN DetaliiProgramare Dp ON P.IDProgramare = Dp.IDProgramare JOIN Sali S ON Dp.IDSala = S.IDSala

WHERE S.IDSala = 5

**12.Interogări complexe (SELECT in SELECT)**

QC1: Afisati numele categoriilor pentru care media salariilor angajatilor este mai mare decat media salariului pe intreaga companie.

SELECT C.Denumire

FROM Categorii C JOIN Angajati A ON C.IDCategorie = A.IDCategorie

GROUP BY C.Denumire

HAVING AVG(A.Salariu) > (SELECT AVG(Salariu) FROM Angajati)

QC2: Afisati angajatii care nu presteaza niciun serviciu.

SELECT IDAngajat, Nume, Prenume

FROM Angajati WHERE IDAngajat NOT IN (SELECT DISTINCT IDAngajat FROM DetaliiProgramare)

QC3: Afisati angajatii care castiga cel mai mare salariu pentru fiecare categorie. Sortati in ordinea descrescatoare a salariului.

SELECT A.Nume, A.Prenume, A.Salariu

FROM Angajati A, (SELECT MAX(Salariu) AS SAL, IDCategorie FROM Angajati GROUP BY IDCategorie) AS B

WHERE A.Salariu = B.SAL AND A.IDCategorie = B.IDCategorie

ORDER BY A.Salariu DESC

QC4: Clienții care au programări pentru servicii oferite de un anumit angajat cu un salariu mai mare decât media salariilor angajaților și aceste programări au avut loc într-o anumită lună. (o sa mearga numai pentru luna 12 pentru ca numai atunci am programri)

SELECT C.IDClient, C.Nume, C.Prenume

FROM Clienti C JOIN Programari P ON C.IDClient = P.IDClient JOIN DetaliiProgramare DP ON P.IDProgramare = DP.IDProgramare JOIN Angajati A ON DP.IDAngajat = A.IDAngajat

WHERE A.Salariu > (SELECT AVG(Salariu) FROM Angajati) AND MONTH(P.Data) = '"+textBox2.Text+"'