Lucrare de laborator nr. 2
Conexiunea la baze de date și operații specifice

2.1 Scopul lucrării

În cadrul acestei lucrări se urmărește consolidarea operațiilor de creare a unei bazei de date, de realizare a conexiunii la baza de date, de creare a tabelelor și de interogare a tabelelor bazei de date, precum și de scriere a testelor unitate pentru fiecare operație specifică bazei de date.

2.2 Exerciții rezolvate

Exercițiu 2.1. Se dorește dezvoltarea unei aplicații software care să fie utilizată la un concurs de matematică pentru elevi de liceu. În acest scop să se creeze o bază de date relațională care să conțină tabelele prezentate în diagrama entitate-relație din figura 2.1. Apoi să se implementeze metode pentru operațiile de adăugare, actualizare, ștergere și căutare, precum și teste unitate corespunzătoare fiecărei operații implementate.

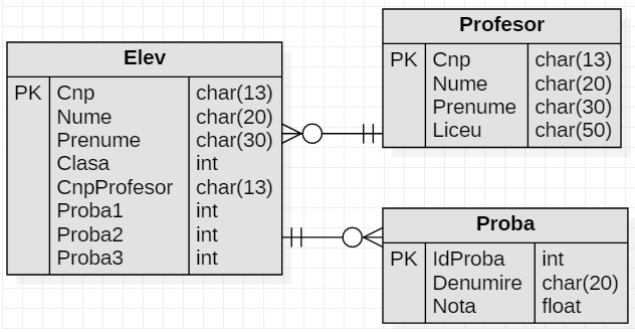

Figura 2.1. Diagramă entitate-relație

Soluție:

În soluția prezentată se utilizează sistemul de gestionare de baze de date relaționale ***SQL Server***, iar pentru implementarea testelor unitate se utilizează framework-ul ***NUnit***.

Pachetul ***Model*** cuprinde 3 clase având structura prezentată în figura 2.2. Pachetul ***Persistenta*** (figura 2.3) cuprinde 3 clase:

❖ Clasa ***CreareBazaDateTabele*** care cuprinde metode ce permit crearea bazei de date și a tabelului ***Profesor***.

❖ Clasa ***Persistenta*** cuprinde metode pentru deschiderea/închiderea conexiunii la baza de date, dar și pentru interogări.

❖ Clasa ***ProfesorPersistenta*** cuprinde metode pentru adăugarea, actualizarea, ștergerea, căutarea unui profesor, dar și pentru obținerea listei de profesori.

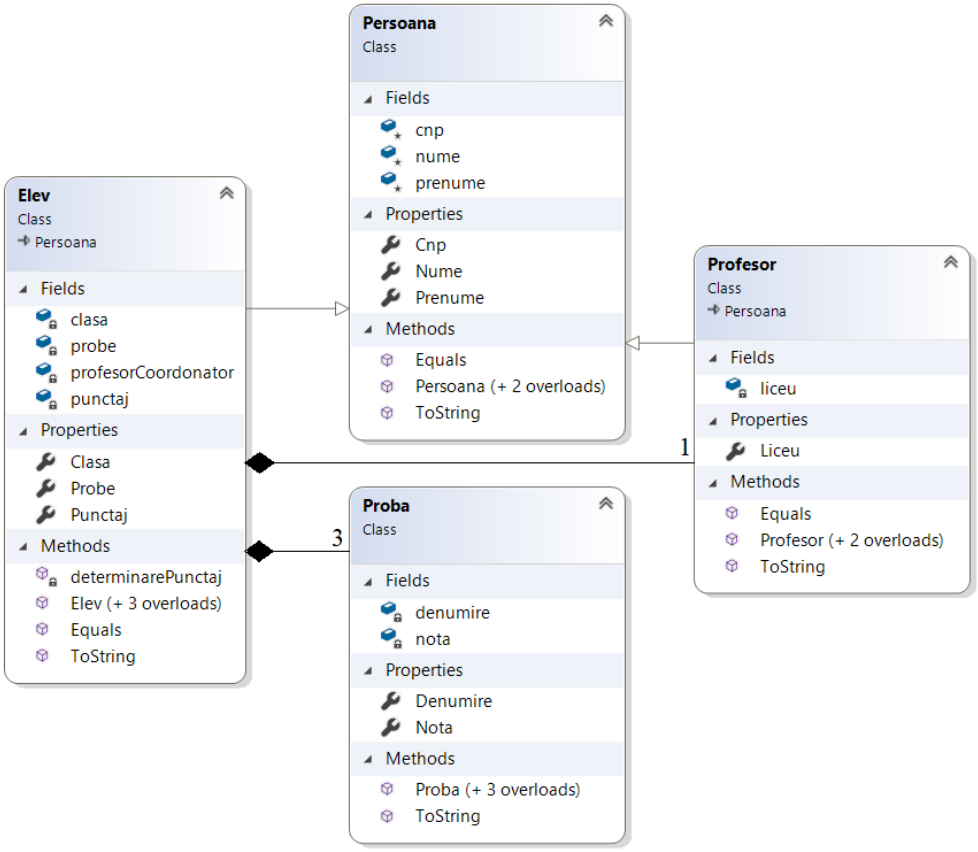

Figura 2.2. Clasele pachetului "Model"

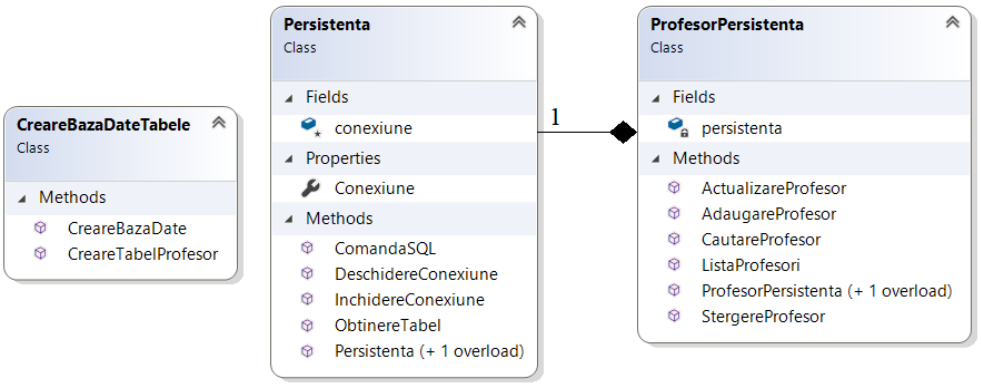

Figura 2.3. Clasele pachetului "Persistenta"

Pentru testarea unitară a metodelor celor 3 clase din pachetul ***Persistenta*** se implementează un proiect pentru testare ce cuprinde 3 clase (figura 2.4).

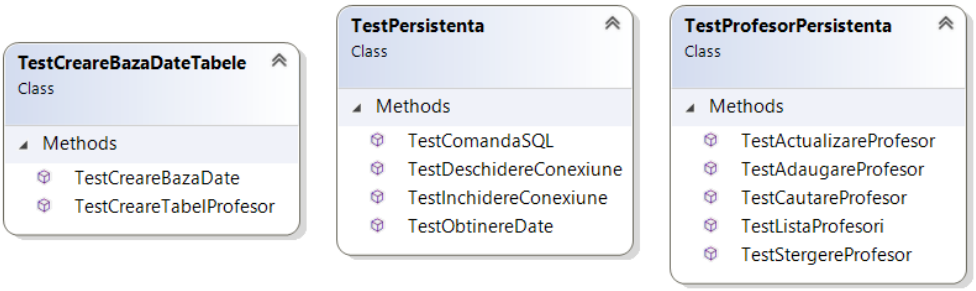

Figura 2.4. Clasele pentru testarea unitară

Pentru crearea bazei de date ***ConcursMatematica*** se implementează metoda ***CreareBazaDate()*** aparținând clasei ***CreareBazaDateTabele***:

```
public void CreareBazaDate()
{
    SqlConnection conexiune = new SqlConnection("Server=localhost\\
sqlexpress;Integrated  security=SSPI;database=master");
    String str = "CREATE DATABASE ConcursMatematica ON " +
     "PRIMARY (NAME = ConcursMatematica_Data, " +
     "FILENAME = 'D:\\ConcursMatematica.mdf', " +
     "SIZE = 5MB, MAXSIZE = 10MB, FILEGROWTH = 10%)" +
     "LOG ON (NAME = ConcursMatematica_Log, " +
     "FILENAME = 'D:\\ConcursMatematicaLog.ldf', " +
     "SIZE = 1MB, MAXSIZE = 5MB, FILEGROWTH = 10%)";
    SqlCommand comandaSQL = new SqlCommand(str, conexiune);
    try
    {
        conexiune.Open();
        comandaSQL.ExecuteNonQuery();
        Console.WriteLine("Baza de date a fost creata cu succes!");
    }
    catch (System.Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    finally
    {
        if (conexiune.State != ConnectionState.Closed)
            conexiune.Close();
    }
}
```

Metoda ***TestCreareBazaDate()*** a clasei ***TestCreareBazaDateTabele*** verifică dacă baza de date a fost creată.

```
[Test]
public void TestCreareBazaDate()
{
    // Arrange
    CreareBazaDateTabele bd = new CreareBazaDateTabele();
    // Assert
    FileAssert.DoesNotExist("D:\\ConcursMatematica.mdf");
    FileAssert.DoesNotExist("D:\\ConcursMatematicaLog.ldf");
    // Act
    bd.CreareBazaDate();
    // Assert
    FileAssert.Exists("D:\\ConcursMatematica.mdf");
    FileAssert.Exists("D:\\ConcursMatematicaLog.ldf");
}
```

Pentru crearea tabelului ***Profesor*** având structura prenzentată în diagrama din figura 2.1 se implementează metoda ***CreareTabelProfesor()*** aparținând clasei ***CreareBazaDateTabele***:

```
public bool CreareTabelProfesor()
{
    string connectionString = "Integrated Security=SSPI; Initial
Catalog=ConcursMatematica;Data Source=localhost \\sqlexpress;";
    Persistenta p = new Persistenta(connectionString);
    string sql = "CREATE TABLE Profesor (Cnp CHAR(13) CONSTRAINT
PKeyCnpP PRIMARY KEY, Nume CHAR(20), Prenume CHAR(30), Liceu
CHAR(50))";
    bool rezultat = true;
    try
    {
        p.DeschidereConexiune();
        SqlCommand comanda = new SqlCommand(sql, p.Conexiune);
        if (comanda.ExecuteNonQuery() == 0)
            rezultat = false;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        rezultat = false;
    }
    finally
    {
        p.InchidereConexiune();
    }
    return rezultat;
}
```

Metoda ***TestCreareTabelProfesor()*** a clasei ***TestCreareBazaDateTabele*** verifică dacă tabelul ***Profesor*** a fost creat și are structura dorită.

```
[Test]
public void TestCreareTabelProfesor()
{
    // Arrange
    CreareBazaDateTabele bd = new CreareBazaDateTabele();
    string s = "Data Source=localhost\\sqlexpress;Initial
Catalog=ConcursMatematica;Integrated Security=True;";
    // Act
    bd.CreareTabelProfesor();
    // Arrange
    SqlConnection conexiune = new SqlConnection(s);
    conexiune.Open();
    DataTable dt = conexiune.GetSchema("Tables");
    List<string> tabele = new List<string>();
    foreach (DataRow row in dt.Rows)
        tabele.Add(row[2].ToString());
    bool rezultat1 = tabele.Contains("Profesor");
    bool rezultat2 = tabele.Contains("Profesori");
    string sqlComanda = "SELECT COUNT(*) FROM
INFORMATION_SCHEMA.COLUMNS WHERE table_name = 'Profesor'";
    SqlCommand comanda = new SqlCommand(sqlComanda, conexiune);
    int numarColoane = (int)comanda.ExecuteScalar();
    string sqlComanda1 = "SELECT * FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'Profesor'";
    comanda = new SqlCommand(sqlComanda1, conexiune);
    SqlDataReader coloane = comanda.ExecuteReader();
    List<string> listaColoane = new List<string>();
    while (coloane.Read())
    listaColoane.Add(string.Format("{0} ", coloane[3]).Trim(' '));
    coloane.Close();
    // Assert
    Assert.IsTrue(rezultat1);
    Assert.IsFalse(rezultat2);
    Assert.AreEqual(4, numarColoane);
    Assert.AreEqual("Cnp", listaColoane[0]);
    Assert.AreEqual("Nume", listaColoane[1]);
    Assert.AreEqual("Prenume", listaColoane[2]);
    Assert.AreEqual("Liceu", listaColoane[3]);
}
```

Metodele pentru deschiderea/închiderea conexiunii la baza de date sunt definite în clasa ***Persistenta***:

```
public void DeschidereConexiune()
{
    if (this.conexiune.State != ConnectionState.Open)
        this.conexiune.Open();
}
public void InchidereConexiune()
{
    if (this.conexiune.State != ConnectionState.Closed)
        this.conexiune.Close();
}
```

Metoda ***TestDeschidereConexiune()*** a clasei ***TestPersistenta*** verifică dacă a fost deschisă conexiunea la baza de date specificată.

```
[Test]
public void TestDeschidereConexiune()
{
    // Arrange
    Persistenta p = new Persistenta();
    // Act
    p.DeschidereConexiune();
    // Assert
    Assert.IsTrue(p.Conexiune.State == ConnectionState.Open);
}
```

Metoda ***TestInchidereConexiune()*** a clasei ***TestPersistenta*** verifică dacă a fost înschisă conexiunea la baza de date specificată.

```
[Test]
public void TestInchidereConexiune()
{
    // Arrange
    Persistenta p = new Persistenta();
    // Act
    p.InchidereConexiune();
    // Assert
    Assert.IsTrue(p.Conexiune.State == ConnectionState.Closed);
}
```

Metoda ***ComandaSQL()*** a clasei ***Persistenta*** permite executarea unei comezi SQL transmisă prin intermediul parametrului.

```
public bool ComandaSQL(string comandaSQL)
{
    bool rezultat = true;
    try
    {
        this.DeschidereConexiune();
        SqlCommand comanda = new SqlCommand(comandaSQL,
this.conexiune);
        if (comanda.ExecuteNonQuery() == 0)
            rezultat = false;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        rezultat = false;
    }
    finally
    {
```

```
        this.InchidereConexiune();
    }
    return rezultat;
}
```

Metoda ***TestComandaSQL()*** a clasei ***TestPersistenta*** verifică dacă comanda SQL a fost executată cu succes.

```
[Test]
public void TestComandaSQL()
{
    // Arrange
    Persistenta p = new Persistenta();
    string comandaSQL1 = "insert into Profesor
values('1981011234565','Pop','Ioan','LTNB')";
    string comandaSQL2 = "insert into Profesor
values('Pop','Ioan','LTNB')";
    string comandaSQL3 = "delete from Profesor where Cnp =
'1921011234565'";
    string comandaSQL4 = "delete from Profesor where Cnp =
'1981011234565'";
    // Act
    bool rezultat1 = p.ComandaSQL(comandaSQL1);
    bool rezultat2 = p.ComandaSQL(comandaSQL2);
    bool rezultat3 = p.ComandaSQL(comandaSQL3);
    bool rezultat4 = p.ComandaSQL(comandaSQL4);
    // Assert
    Assert.IsTrue(rezultat1);
    Assert.IsFalse(rezultat2);
    Assert.IsFalse(rezultat3);
    Assert.IsTrue(rezultat4);
}
```

Metoda ***ObtineTabel()*** a clasei ***Persistenta*** permite obținerea datelor dintr-un tabel în baza comenzii SQL transmise ca parametru.

```
public DataTable ObtinereTabel(string comandaSQL)
{
    DataTable rezultat = null;
    try
    {
        this.DeschidereConexiune();
        SqlCommand comanda = new SqlCommand(comandaSQL,
this.conexiune);
        SqlDataAdapter dateCitite = new SqlDataAdapter(comanda);
        DataTable dateTabel = new DataTable();
        dateCitite.Fill(dateTabel);
        rezultat = dateTabel;
    }
    catch (Exception ex)
```

```
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        this.InchidereConexiune();
    }
    return rezultat;
}
```

Metoda ***TestObtinereDate()*** a clasei ***TestPersistenta*** verifică dacă au fost obținute date corecte în urma executării comenzii SQL.

```
[Test]
public void TestObtinereDate()
{
    // Arrange
    Persistenta p = new Persistenta();
    string vizualizareSQL = "Select * from Profesor order by Nume";
    // Act
    DataTable rezultat = p.ObtinereTabel(vizualizareSQL);
    // Assert
    Assert.IsNotNull(rezultat);
    Assert.AreEqual(1, rezultat.Rows.Count);
    Assert.AreEqual(4, rezultat.Columns.Count);
}
```

Metoda ***AdaugareProfesor()*** a clasei ***ProfesorPersistenta*** permite adăugarea informațiilor despre un profesor în baza de date.

```
public bool AdaugareProfesor(Profesor prof)
{
    string comandaSQL = "insert into Profesor values('" + prof.Cnp +
"','" + prof.Nume + "','" +prof.Prenume + "','" + prof.Liceu + "')";
    return this.persistenta.ComandaSQL(comandaSQL);
}
```

Metoda ***TestAdaugareProfesor()*** a clasei ***TestProfesorPersistenta*** verifică dacă adăugarea unui obiect de tip ***Profesor*** se realizează cu succes.

```
[Test]
public void TestAdaugareProfesor()
{
    // Arrange
    ProfesorPersistenta pp = new ProfesorPersistenta();
    Profesor prof = new Profesor("Pop", "Vlad", "1981011234565",
"LTNB");
    // Act
    bool rezultat1 = pp.AdaugareProfesor(prof);
```

```
    bool rezultat2 = pp.AdaugareProfesor(prof);
    // Assert
    Assert.IsTrue(rezultat1);
    Assert.IsFalse(rezultat2);
}
```

Metoda ***ActualizareProfesor()*** a clasei ***ProfesorPersistenta*** permite actualizarea informațiilor despre un profesor în baza de date.

```
public bool ActualizareProfesor(string cnp, Profesor prof)
{
    string comandaSQL = "update Profesor set Nume = '" + prof.Nume +
"', Prenume = '" + prof.Prenume + "', Liceu = '" + prof.Liceu + "'
where Cnp = '" + cnp + "'";
    return this.persistenta.ComandaSQL(comandaSQL);
}
```

Metoda ***TestActualizareProfesor()*** a clasei ***TestProfesorPersistenta*** verifică dacă actualizarea unui obiect de tip ***Profesor*** se realizează cu succes.

```
[Test]
public void TestActualizareProfesor()
{
    // Arrange
    ProfesorPersistenta pp = new ProfesorPersistenta();
    Profesor prof = new Profesor("Popescu", "Ionel",
"1981011234565", "LTNB");
    // Act
    bool rezultat1 = pp.ActualizareProfesor("1981011234565",prof);
    bool rezultat2 = pp.ActualizareProfesor("1988011234565", prof);
    // Assert
    Assert.IsTrue(rezultat1);
    Assert.IsFalse(rezultat2);
}
```

Metoda ***StergereProfesor()*** a clasei ***ProfesorPersistenta*** permite ștergerea informațiilor despre un profesor din baza de date.

```
public bool StergereProfesor(string cnp)
{
    string comandaSQL="delete from Profesor where Cnp = '"+cnp+ "'";
    return this.persistenta.ComandaSQL(comandaSQL);
}
```

Metoda ***TestStergereProfesor()*** a clasei ***TestProfesorPersistenta*** verifică dacă ștergerea unui obiect de tip ***Profesor*** se realizează cu succes.

```
[Test]
public void TestStergereProfesor()
{
    // Arrange
    ProfesorPersistenta pp = new ProfesorPersistenta();
    // Act
    bool rezultat1 = pp.StergereProfesor("2981331234565");
    bool rezultat2 = pp.StergereProfesor("1981011234565");
    // Assert
    Assert.IsFalse(rezultat1);
    Assert.IsTrue(rezultat2);
}
```

Metoda ***CautareProfesor()*** a clasei ***ProfesorPersistenta*** permite căutarea unui profesor în baza de date.

```
public Profesor CautareProfesor(string cnpProfesor)
{
    string cautareSQL = "Select * from Profesor where Cnp = '" +
cnpProfesor + "'";
DataTable tabelProfesori=this.persistenta.ObtinereTabel(cautareSQL);
    if (tabelProfesori == null || tabelProfesori.Rows.Count == 0)
        return null;
    DataRow dr = tabelProfesori.Rows[0];
    return new Profesor(dr["Nume"].ToString(), dr["Prenume"].
ToString(), dr["Cnp"].ToString(), dr["Liceu"].ToString());
}
```

Metoda ***TestCautareProfesor()*** a clasei ***TestProfesorPersistenta*** verifică dacă căutarea unui obiect de tip ***Profesor*** se realizează cu succes.

```
[Test]
public void TestCautareProfesor()
{
    // Arrange
    ProfesorPersistenta pp = new ProfesorPersistenta();
    Profesor prof1 = new Profesor("Ionescu", "Lucia",
"2961021234565", "CND");
    Profesor prof2 = new Profesor("Pop", "Vlad", "1981011234565",
"LTNB");
    // Act
    Profesor rezultat1 = pp.CautareProfesor("2961021234565");
    Profesor rezultat2 = pp.CautareProfesor("1981011234577");
    // Assert
    Assert.IsNotNull(rezultat1);
    Assert.IsNull(rezultat2);
}
```

Metoda ***ListaProfesori()*** a clasei ***ProfesorPersistenta*** permite obținerea tuturor înregistrărilor din tabelul Profesor.

```
public List<Profesor> ListaProfesori()
{
    string vizualizareSQL = "Select * from Profesor order by Nume";
    DataTable tabelProfesori =
this.persistenta.ObtinereTabel(vizualizareSQL);
    if (tabelProfesori == null || tabelProfesori.Rows.Count == 0)
        return null;
    List<Profesor> lista = new List<Profesor>();
    foreach (DataRow dr in tabelProfesori.Rows)
    {
        Profesor prof = new Profesor((string)dr["Nume"],
(string)dr["Prenume"], (string)dr["Cnp"], (string)dr["Liceu"]);
        lista.Add(prof);
    }
    return lista;
}
```

Metoda ***TestListaProfesori()*** a clasei ***TestProfesorPersistenta*** verifică dacă obținerea tuturor înregistrărilor din tabelul ***Profesor*** se realizează cu succes.

```
[Test]
public void TestListaProfesori()
{
    // Arrange
    ProfesorPersistenta pp = new ProfesorPersistenta();
    Profesor prof1 = new Profesor("Ionescu", "Lucia",
"2961021234565", "CND");
    Profesor prof2 = new Profesor("Pop", "Vlad", "1981011234565",
"LTNB");
    // Act
    List<Profesor> rezultat = pp.ListaProfesori();
    // Assert
    Assert.IsNotNull(rezultat);
    Assert.AreEqual(2, rezultat.Count);
    Assert.AreEqual(prof1, rezultat[0]);
    Assert.AreEqual(prof2, rezultat[1]);
}
```

2.3 Exerciții propuse spre rezolvare

Exercițiu 2.2. Să se modifice proiectul anterior astfel încât:

❖ În clasa ***CreareBazaDateTabele*** să se adauge 2 metode pentru crearea tabelor ***Elev*** și ***Proba*** conform diagramei ER (figura 2.1), iar în clasa ***TestCreareBazaDateTabele*** să se adauge metode pentru testarea noilor metode.

❖ Să se implementeze 2 clase pentru realizarea persistenței obiectelor de tip ***Elev***, respectiv ***Proba***, iar în proiectul de testare să se implementeze 2 clase pentru testare.