# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №4**

**«Створення баз даних за допомогою стандартних бібліотек C#»**

**з дисципліни «Програмне забеспечення мережевих технологій»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

ст. викл.Сенів М.М.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Створення баз даних за допомогою стандартних бібліотек C#.

**Мета роботи:** Навчитись елементам програмування мовою C#. Засвоїти основні стандартні класи і методи бібліотек sql та swing, та бібліотек, що дозволяють опрацьовувати дані СУБД.

## 1. Теоретичні відомості

ADO.NET - це набір класів, що надають служби доступу до даних програмісту, що працює на платформі. NET Framework . ADO.NET має багатий набір компонентів для створення розподілених додатків , які разом використовують дані . Це невід'ємна частина платформи. NET Framework , яка надає доступ до реляційних даних, XML - даними і даними додатків. ADO.NET задовольняє різні потреби розробників, включаючи створення клієнтських додатків баз даних , а також бізнес- об'єктів середнього рівня, використовуваних додатками , засобами , мовами і браузерам .

ADO.NET об’єднує можливості ADO та OLE DB в єдиній системі управління послугами доступу до даних. Кожна послуга містить класи для реалізації інтерфейсу до різних джерел даних. Наприклад, ADO Managed Provider (до джерел даних OLE DB), SQL Server Managed Provider (доступ до DBMS Microsoft), Exchange Managed Provider (дані з Microsoft Exchange) тощо. ADO .NET підтримує використання документів в у XML форматі. Це спрощує доступ до даних за протоколом http.

ADO.NET є технологією доступу до баз даних у платформі .NET (Dot Net), яка базується на технології Microsoft ActiveX® Data Objects (ADO). ADO є незалежною від мови програмування технологією об’єктних моделей. ADO.NET є інтегральною частиною засобів .NET Compact Framework і підтримує доступ до реляційних баз даних, до XML документів та даних програм. На її основі можна розробити як клієнтські так і серверні частини проектів широкого спектру задач.

ADO .NET має дві форми доступу до даних:

Доступ без з’єднання: Програма виконує запит і розміщує результати виконання запиту в об’єкті типу DataSet, щоб пізніше їх перенести в задані масиви тощо. В цьому випадку мінімізується час підтримки відкритого з’єднання до бази даних.

Доступ через з’єднання: читання з безадресним курсором ("firehose cursors"). Програма здійснює запит і читає дані з допомогою об’єкту DataReader.

ADO.NET визначає об’єкти типу DataSet та DataTable, які оптимізують перенесення даних по мережах інтранету та Інтернету (в т.ч. крізь firewalls). Технологія надає також традиційні об’єкти Connection і Command.

З допомогою ADO.NET та .NET Framework можна організувати доступ до наступних джерел даних :

• SQL Server (System.Data.SqlClient).

• OLEDB (System.Data.OleDb).

• ODBC (System.Data.Odbc).

• Oracle (System.Data.OracleClient).

Двома основними компонентами в ADO.NET є класи: DataSet, та .NET Framework Data Provider.

В залежності від СУБД, що виступає джерелом даних, до програми залучаються наступні бібліотеки класів (простори назв) :

// СУБД MS Access

using System.Data.OleDb;

//СУБД MS SQL Server

using System.Data.SqlClient;

## 2. Хід роботи

**Завдання:**

1. Створити базу даних, що містить декілька таблиць.

2. Запрограмувати доповнення записів даних в таблицях.,

3. Запрограмувати видалення записів даних.

4. Запрограмувати фільтрацію даних.

5. Написати фрагменти програм доступу до СУБД, формуваннязапитів, отримання таблиць –результатів запитів, перенесення результатів.

В середовищі MS Visual C# 2010 створимо новий проект наберемо текст програми:

using System;

using System.Data.SqlClient;

using System.Data;

namespace Students

{

class Program

{

static void Main(string[] args)

{

//Запит для виводу таблиці студентів

string RequestText = "Select \* From Students";

//Запит для виводу таблиці оцінок студента з кодом 10

string RequestText2 =

"Select " +

"st.ID as ID, " +

"st.StudName as StudName, " +

//"st.GroupNum as GroupNum, " +

"mar.Mark as Mark " +

"FROM Marks mar " +

"Join Students st on st.ID = mar.ID " +

"WHERE st.ID = 10";

//стрічка підключення до бази

string connectionString = "server=TarasHP;uid=sa;pwd=pass45678;database=Test";

SqlConnection conn = new SqlConnection(connectionString);

try

{

//спроба підключитися

conn.Open();

}

catch (SqlException se)

{

Console.WriteLine("Помилка пiдключення:{0}", se.Message);

Console.ReadKey();

return;

}

Console.WriteLine("Успiшно пiдключено до бази");

/\* Створюємо екземпляр класу SqlCommand по імені cmdCreateTable

              і передаємо конструктору цього класу, запит на створення таблиці

              і об'єкт типу SqlConnection

             \*/

//створення таблиць

CreateNewTable(conn, "CREATE TABLE " +

" Students (ID int not null" +

", StudName char(60) not null," +

" GroupNum char(20) not null)");

CreateNewTable(conn, "CREATE TABLE " +

" Marks (ID int not null, Mark int not null)");

Console.WriteLine("Таблицi створенi успiшно");

//додавання нових записiв

Console.WriteLine("Вставляємо записи:");

InsertToTable(conn, 10, "Гринчук Т.", "ПЗС-11");

InsertToTable(conn, 20, "Непомнящий Iгор", "ПЗС-11");

InsertToTable(conn, 30, "Середа Iгор", "ПЗС-11");

//Оцiнки

InsertMarks(conn, 10, 5);

InsertMarks(conn, 10, 4);

InsertMarks(conn, 10, 5);

InsertMarks(conn, 20, 5);

InsertMarks(conn, 20, 3);

InsertMarks(conn, 30, 4);

Console.ReadKey();

//виводимо на екран

ShowFields(conn, RequestText);

//змiна значення рядка

UpdateFromTable(conn);

ShowFields(conn, RequestText);

Console.WriteLine("Оцiнки студента з ID = 10 :");

ShowFields(conn, RequestText2);

Console.ReadKey();

//видалення данних

DeleteFromTable(conn);

ShowFields(conn, RequestText);

//видалення таблиць

DeleteTable(conn, "Students");

DeleteTable(conn, "Marks");

Console.ReadKey();

//закриття пiдключення до бази

conn.Close();

conn.Dispose();

}

private static void DeleteTable(SqlConnection conn, string table)

{

using (SqlCommand cmdDeleteTable = new SqlCommand("DROP TABLE " + table, conn))

{

//вiдправка запиту

try

{

cmdDeleteTable.ExecuteNonQuery();

}

catch

{

Console.WriteLine("Помилка видалення таблицi");

Console.ReadKey();

return;

}

}

Console.WriteLine("Таблиця " + table + " успiшно видалена");

}

private static void DeleteFromTable(SqlConnection conn)

{

using (SqlCommand cmd = new SqlCommand("Delete From Students" +

" where ID = @ID", conn))

{

//Створення параметрiв

SqlParameter param = new SqlParameter();

//задаємо назву параметра

param.ParameterName = "@ID";

//задаємо значення параметра

param.Value = 10;

//задаємо тип параметра

param.SqlDbType = SqlDbType.Int;

//передаємо параметр екземпляру класу SqlCommand

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@StudName";

param.Value = "Иванов Иван";

param.SqlDbType = SqlDbType.Text;

cmd.Parameters.Add(param);

Console.WriteLine("Видалення запису");

try

{

cmd.ExecuteNonQuery();

}

catch

{

Console.WriteLine("Помилка видалення запису");

Console.WriteLine("Можливе запис вже був видалений");

Console.ReadKey();

return;

}

Console.ReadKey();

}

}

private static void UpdateFromTable(SqlConnection conn)

{

using (SqlCommand cmd = new SqlCommand("Update Students" +

" Set StudName = @StudName where ID = @ID", conn))

{

SqlParameter param = new SqlParameter();

param.ParameterName = "@ID";

param.Value = 10;

param.SqlDbType = SqlDbType.Int;

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@StudName";

param.Value = "Гринчук Тарас";

param.SqlDbType = SqlDbType.Text;

cmd.Parameters.Add(param);

Console.WriteLine("Змiнюємо записи");

{

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException se)

{

Console.WriteLine("Помилка, при змiнi запису: {0}",se.Message);

Console.ReadKey();

return;

}

Console.ReadKey();

}

}

}

private static void ShowFields(SqlConnection conn, string text)

{

//Виводимо значення на екран

Console.WriteLine("================================= База даних =================================");

using (SqlCommand cmd = new SqlCommand(text, conn))

{

/\* Метод ExecuteReader() класу SqlCommand повертає

                  об'єкт типу SqlDataReader, з допомогою якого ми можемо

                  прочитати всі рядки, поверненні в результаті виконання запиту

                  CommandBehavior.CloseConnection - закриваємо з'єднання після запиту

                  \*/

using (SqlDataReader dr = cmd.ExecuteReader())

{

//цикл по всіх стовпцях отриманої в результаті запиту таблиці

for (int i = 0; i < dr.FieldCount; i++)

/\* метод GetName() класу SqlDataReader дозволяє отримати ім'я стовпця

                          за номером, який передається як параметр, даному методу

                          і позначає номер стовпчика в таблиці (починається з 0)

                          \*/

Console.Write("{0}\t", dr.GetName(i).ToString().Trim());

/\* читаємо дані з таблиці

                      читання відбувається тільки в прямому напрямку

                      всі прочитання рядка відкидаються \*/

Console.WriteLine();

while (dr.Read())

{

/\* метод GetValue() класу SqlDataReader дозволяє отримати значення стовпця

                                                 за номером, який передається як параметр, даному методу

                                                 і позначає номер стовпчика в таблиці (починається з 0)

                                                 \*/

Console.WriteLine("{0}\t{1}\t{2}", dr.GetValue(0).ToString().Trim(),

dr.GetValue(1).ToString().Trim(),

dr.GetValue(2).ToString().Trim());

}

}

}

Console.WriteLine("==============================================================================");

Console.ReadKey();

}

private static void CreateNewTable(SqlConnection conn, string text)

{

using (SqlCommand cmdCreateTable = new SqlCommand(text, conn))

{

//вiдправка запиту

try

{

cmdCreateTable.ExecuteNonQuery();

}

catch (SqlException se)

{

Console.WriteLine("Помилка створення таблицi: {0}",se.Message);

Console.ReadKey();

return;

}

}

}

private static void InsertToTable(SqlConnection conn, int ID, string StudName, string GroupNum)

{

using (SqlCommand cmd = new SqlCommand("Insert into Students" +

"(ID,StudName,GroupNum) Values (@ID,@StudName,@GroupNum)", conn))

{

SqlParameter param = new SqlParameter();

param.ParameterName = "@ID";

param.Value = ID;

param.SqlDbType = SqlDbType.Int;

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@StudName";

param.Value = StudName;

param.SqlDbType = SqlDbType.Text;

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@GroupNum";

param.Value = GroupNum;

param.SqlDbType = SqlDbType.Text;

cmd.Parameters.Add(param);

try

{

cmd.ExecuteNonQuery();

}

catch

{

Console.WriteLine("Помилка вставки запису");

Console.ReadKey();

return;

}

}

}

private static void InsertMarks(SqlConnection conn, int ID, int Mark)

{

using (SqlCommand cmd = new SqlCommand("Insert into Marks" +

"(ID,Mark) Values (@ID,@Mark)", conn))

{

SqlParameter param = new SqlParameter();

param.ParameterName = "@ID";

param.Value = ID;

param.SqlDbType = SqlDbType.Int;

cmd.Parameters.Add(param);

param = new SqlParameter();

param.ParameterName = "@Mark";

param.Value = Mark;

param.SqlDbType = SqlDbType.Int;

cmd.Parameters.Add(param);

try

{

cmd.ExecuteNonQuery();

}

catch

{

Console.WriteLine("Помилка вставки запису");

Console.ReadKey();

return;

}

}

}

}

}

Запустимо дану програму на виконання (рис. 2.1):

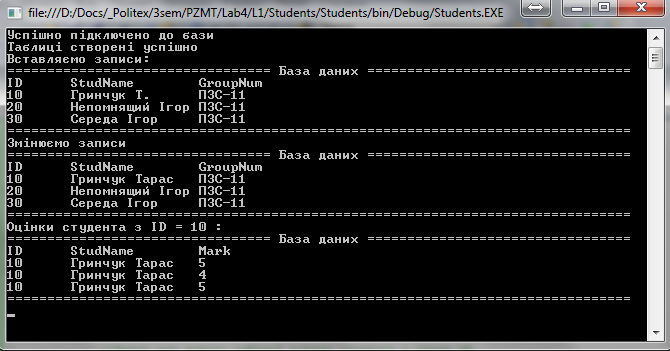


Рис. 2.1. Вікно програми

Як бачимо програма вміє створювати таблиці бази даних в SQL Server. Змінювати їх вміст, додавати стрічки, виводити результати запитів до декількох таблиць бази даних. Що пересвідчитись у правильності виконання програми відкриємо створену БД в SQL Server (рис. 2.2):

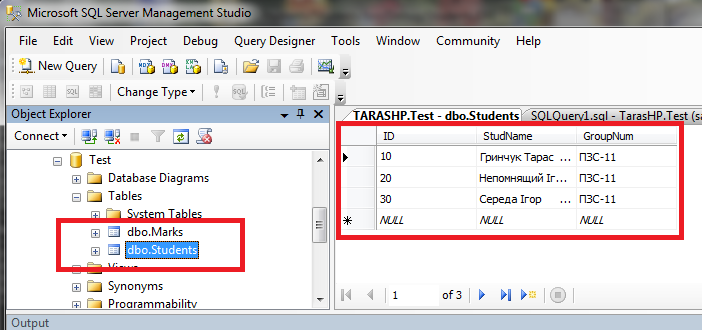


Рис. 2.2. Вигляд БД у SQL Server

Також програмно можна видалити певну стрічку і таблиці взагалі (рис. 2.3 - 2.4):

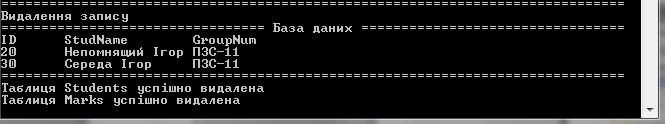


Рис. 2.3. Вікно програми

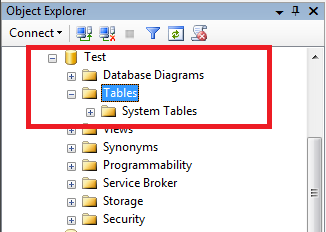


Рис. 2.4. Вигляд пустої бази даних після завершення роботи програми

## ВИСНОВКИ

На даній лабораторній роботі я навчився елементам програмування мовою C#. Засвоїв основні стандартні класи і методи бібліотек sql та swing, та бібліотек, що дозволяють опрацьовувати дані СУБД.