

**ДРОГОБИЦЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА**

Ірина Шаклеїна, Надія Ших, Роман Білий

Опрацювання баз даних засобами MS Visual Studio

**Методичні рекомендації до виконання
лабораторних робіт**

**Дрогобич
2018**

УДК 004.65(075.8)

Ш 17

Рекомендовано до друку вченою радою Дрогобицького державного педагогічного університету імені Івана Франка (протокол № 7 від 17.05.2018 р.)

Рецензенти:

Григорович Андрій Геннадійович, кандидат технічних наук, доцент кафедри фізики навчально-дослідного ІФМЕІТ Дрогобицького державного педагогічного університету імені Івана Франка;

Нищак Іван Дмитрович, доктор педагогічних наук, доцент кафедри технологічної та професійної освіти навчально-дослідного ІФМЕІТ Дрогобицького державного педагогічного університету імені Івана Франка.

Шаклеїна І., Ших Н., Білий Р.

Ш 17 Опрацювання баз даних засобами MS Visual Studio: методичні рекомендації до виконання лабораторних робіт / Ірина Шаклеїна, Надія Ших, Роман Білий. – Дрогобич : Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка, 2018. – 95 с.

Посібник укладено відповідно до програми навчальної дисципліни «Системи баз даних та знань» для підготовки фахівців першого (бакалаврського) рівня вищої освіти галузі знань: 12 «Інформаційні технології» напряму підготовки 122 «Комп'ютерні науки», затвердженої вченою радою Дрогобицького державного педагогічного університету імені Івана Франка (протокол № 9 від 11.11.2017).

УДК 004.65(075.8)

Зміст

Передмова	4
Роз'єднаний режим роботи з базою даних у MS Visual Studio	6
Опрацювання баз даних засобами Visual Studio у з'єднаному режимі ..	14
Робота з компонентами SqlConnection, SqlDataAdapter та Dataset.....	19
Опрацювання зв'язків між таблицями бази даних засобами MS Visual Studio.....	28
Робота з BLOB-даними.....	39
Розробка багатовіконного додатку для роботи з базою даних.....	45
Реєстрація та авторизація користувачів у клієнтській програмі, розподіл ролей	50
Опрацювання збережуваних процедур засобами Visual Studio	56
Опрацювання баз даних SQLite. Обробка та пошук даних.....	69
Опрацювання баз даних MongoDB. Робота з даними в клієнтській програмі.....	79
Варіанти завдань для розробки власної бази даних	89
Література	94

Передмова

Практично будь-яка сучасна галузь пов'язана з накопиченням та обробкою даних. З огляду на це, одним з важливих завдань для фахівця з інформаційних технологій є знання технологій роботи з базами даних.

Центром будь-якої бази даних є її прикладна частина, що складається із сервера баз даних, джерел даних і мережевого програмного забезпечення для підключення клієнта до мережі. Для зручності роботи з даними зазвичай розробляється спеціальне програмне забезпечення, що використовується на робочому місці користувача бази даних, – інтерфейсна частина.

Методичні рекомендації до виконання лабораторних робіт з курсу «Системи баз даних та знань», що мають на меті опрацювання баз даних різного типу засобами середовища MS Visual Studio, укладені на базі семестрового курсу, який читається для студентів напряму підготовки 122 «Комп'ютерні науки». До посібника увійшли рекомендації до виконання лабораторних робіт щодо опрацювання в середовищі MS Visual Studio даних, які містяться у базах даних. Розглядаються реляційні бази даних, розроблені засобами SQL Server й SQLite, та документоорієнтовані бази даних MongoDB. Наведено особливості основних режимів роботи з даними, засоби опрацювання зв'язків між таблицями баз даних та збережуваних процедур у проектах з графічним інтерфейсом.

Усі лабораторні роботи мають однакову структуру: тема, мета,

теоретичні відомості, завдання для самостійного виконання та питання для самоконтролю за темою роботи. Теоретичні відомості охоплюють основний матеріал, який потрібно знати студенту для виконання завдань лабораторної роботи.

Під час підготовки до заняття студент повинен опрацювати теоретичні відомості, спробувати самостійно виконати завдання, передбачені в лабораторній роботі, та вміти дати відповіді на контрольні запитання. У деяких випадках при підготовці до роботи потрібно скористатися додатковою літературою, поданою наприкінці посібника.

Роз'єднаний режим роботи з базою даних у MS Visual Studio

META: ознайомлення з методами підключення бази даних SQL Server до проекту MS Visual Studio; засобами й особливостями опрацювання даних.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

У MS Visual Studio можна виділити два режими роботи з даними, що містяться у базі даних (БД):

- з'єднаний режим – режим обробки даних, що передбачає безпосереднє отримання даних з джерела (БД);
- роз'єднаний режим – автономна робота з даними, що реалізується за допомогою об'єктів класу DataSet.

За зв'язок додатку з джерелом даних і маніпуляції даними в ADO.NET відповідає провайдер даних. Розглянемо основні класи, спільні для всіх провайдерів даних (рис. 1):

Connection – застосовується для створення з'єднання із джерелом даних і керування транзакціями. Якщо використовується SQL Data Provider, використовуються об'єкти класу SqlConnection, якщо ж OLE DB Data Provider, об'єкти класу OleDbConnection;

Command – призначається для виконання команд джерелом даних (SqlCommand й OleDbCommand відповідно до провайдера);

DataReader – використовується для читання даних з їхньою однобічною вибіркою. Якщо додаток клієнта не модифікує дані і не потрібна довільна вибірка даних, а досить їх одноразового перегляду, то використання DataReader замість DataSet уможливить збереження

ресурсів RAM і CPU, а також підняття швидкодії програми;

DataAdapter – вживається для читання даних й зберігання змін, що зроблені в них. DataAdapter служить сполучною ланкою між DataSet і джерелом даних. Він використовує Command для виконання команд SQL як для заповнення DataSet даними, так і для зворотної передачі змінених клієнтом даних до джерела. Для виконання цих функцій об'єкт має чотири методи: SelectCommand, InsertCommand, UpdateCommand і DeleteCommand (рис. 1).

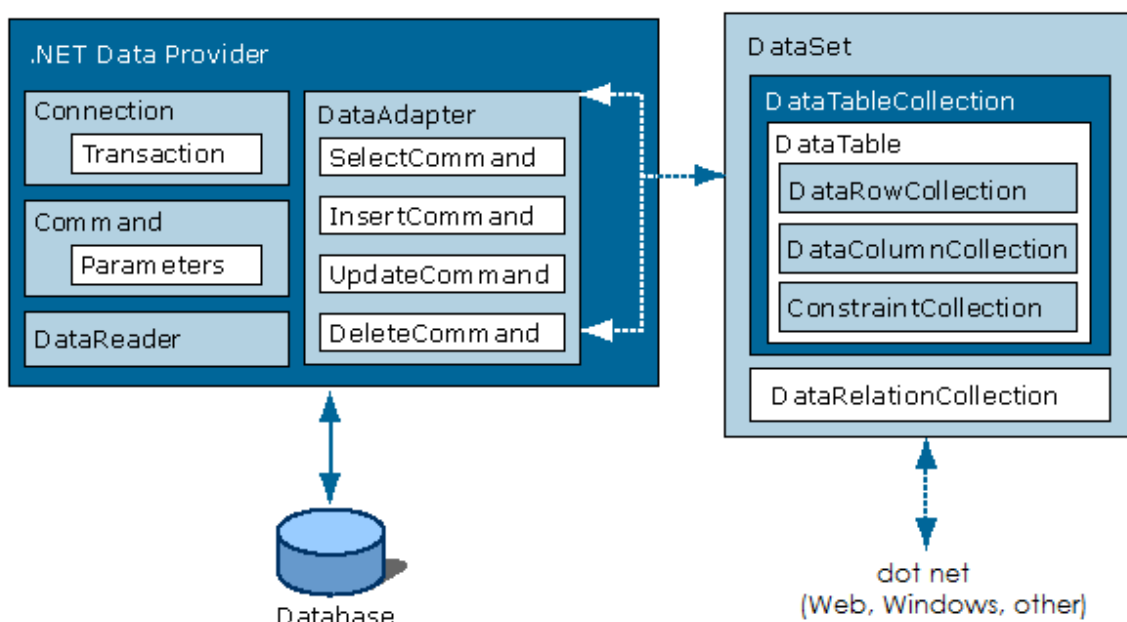


Рис. 1. Схема роботи з джерелом даних

Для роботи з даними в автономному режимі використовують об'єкти класу **DataSet**, розробленого як автономне сховище даних. Він складається з набору таблиць, кожна з яких містить множину рядків і стовпчиків даних. У класі DataSet можна визначити зв'язки між таблицями. DataSet – це набір об'єктів DataTable. Кожен такий об'єкт містить багато об'єктів DataColumn і DataRow.

Доступ до даних

Для підключення БД MS SQL Server до проекту MS Visual Studio можна скористатись командою Add New Data Source пункту меню Data. В

діалоговому вікні, що відкриється (рис 1.), оберіть об'єкт DataBase та натисніть на кнопку Далі. У наступному діалоговому вікні потрібно обрати пункт New Connection і як джерело даних слід обрати Microsoft SQL Server або Файл бази даних Microsoft SQL Server (для роботи з вбудованим у MS Visual Studio сервером баз даних) (рис.1).

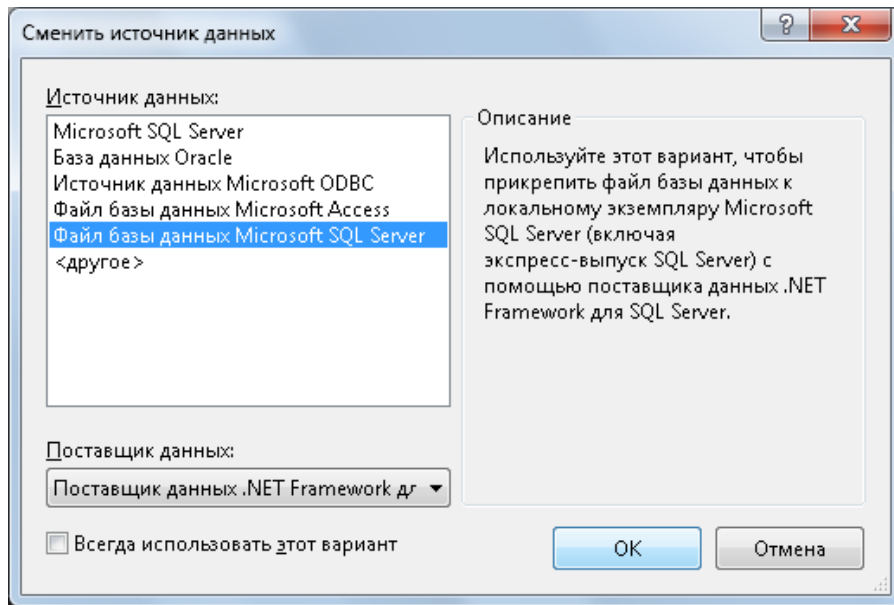


Рис. 1. Підключення до бази даних

На наступному кроці слід підключитись до сервера та вказати назву потрібної бази даних (рис. 2).

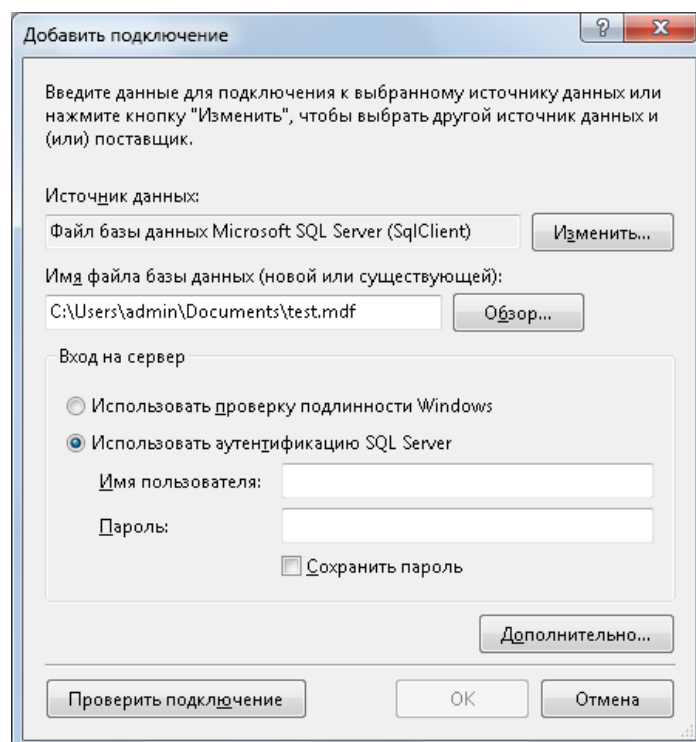


Рис. 2. Вибір бази даних

Далі потрібно відмітити пункт Tables (рис.3) та натиснути на кнопку готово. Створена БД MS SQL Server буде підключена до проекту.

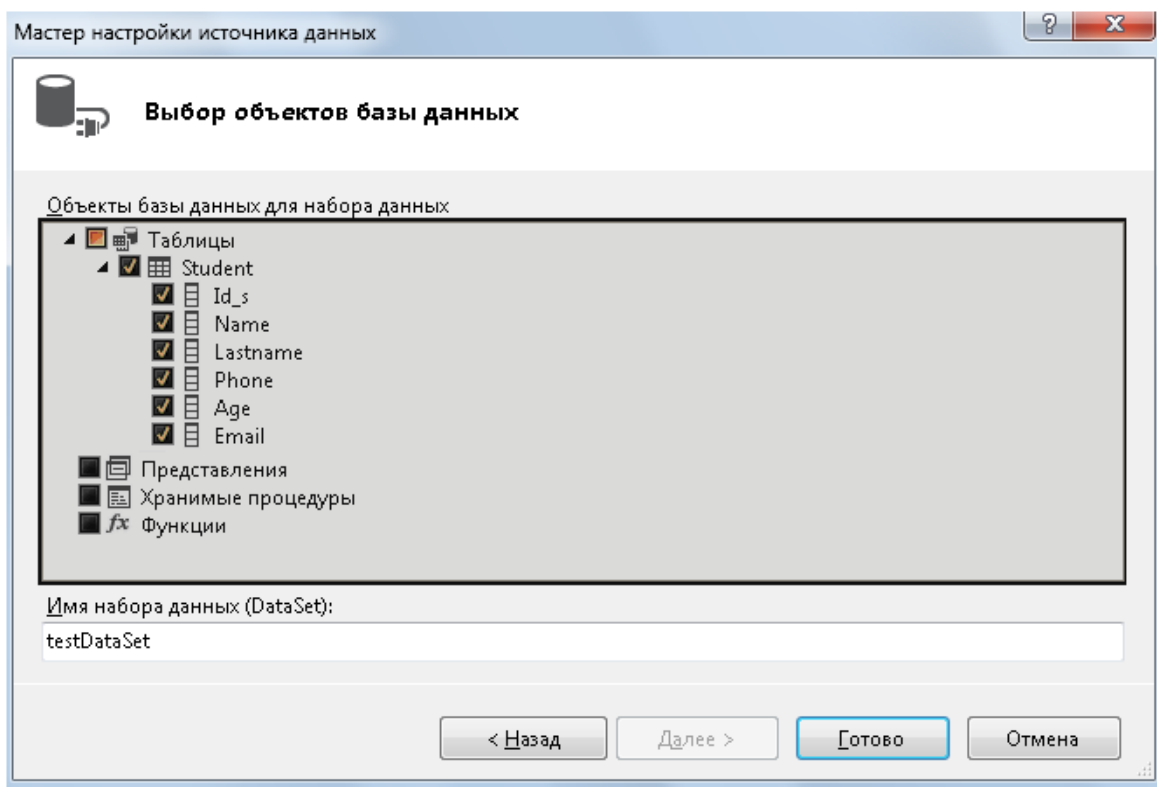


Рис. 3. Вибір вмісту бази даних, що підключається до проекту

Якщо скористатись командою Show Data Sources з пункту Data, можна переглянути вміст створеного автоматично об'єкта Data Set. DataSet – це спеціалізований об'єкт, що містить образ бази даних. Цей компонент не є візуальним, тому поле форми проекту залишається порожнім.

Відображення та опрацювання даних

Для відображення даних, що містяться в підключеній БД, потрібно обрати відповідні візуальні компоненти (DataGridView, ComboBox, TextBox, ListBox та інші).

Зручним засобом для відображення даних і роботи з ними є компонент DataGridView. Після того, як обрана база даних MS SQL Server підключена до проекту, потрібно помістити на форму компонент DataGridView та викликати для нього вікно DataGridView Tasks,

зв'язатись з потрібною базою даних (рис. 4) або налаштувати властивості DataSource та DataMember. Після побудови та запуску проекту в компоненті DataGridView буде виведено потрібну інформацію, що містилась у вказаній таблиці БД MS SQL Server.

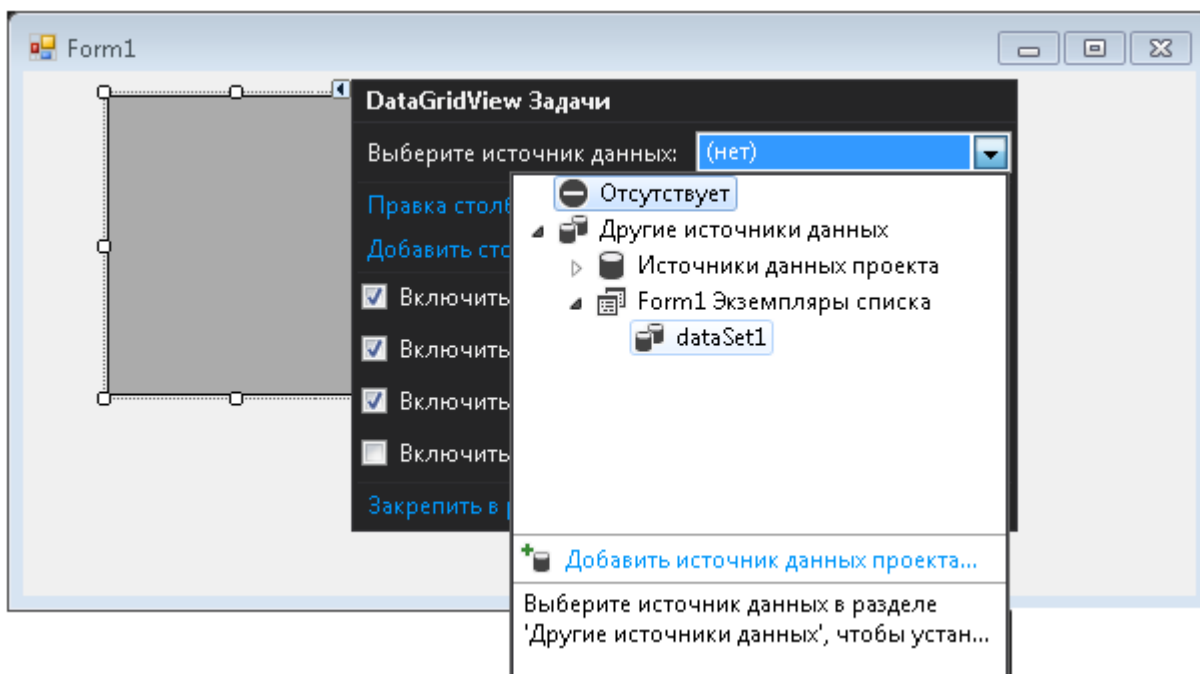


Рис. 4. Зв'язок таблиці БД з компонентом DataGridView

Зверніть увагу, що окрім компонента DataSet, що містить образ бази даних, до проекту автоматично додано компоненти BindingSource (забезпечує зв'язок з таблицею БД) та TableAdapter (відповідає за виконання запитів по кожній таблиці).

Для роботи з записами зручно скористатись компонентом BindingNavigator. Після того, як компонент додано на форму, його потрібно зв'язати з відповідною таблицею за допомогою властивості BindingSource, в ролі значення якої слід обрати назву таблиці. У режимі форми цей компонент дає змогу гортати записи, редагувати вміст полів та видаляти записи з БД.

Робота з даними

Для пошуку даних зручно скористатись командою **Додати запит** вікна DataGridView Tasks. Для вибору певної інформації з бази даних у

діалоговому вікні, що відкриється, потрібно написати відповідний запит, використовуючи основні команди мови SQL. Якщо запит формується під час роботи програми, то слід використовувати параметри. Параметр в SQL-команді задається знаком "@". Наприклад, команда з параметром, що забезпечує вибірку з таблиці student записів по вмісту поля name, може мати вигляд (рис. 5):

```
SELECT * FROM student WHERE name = @name
```

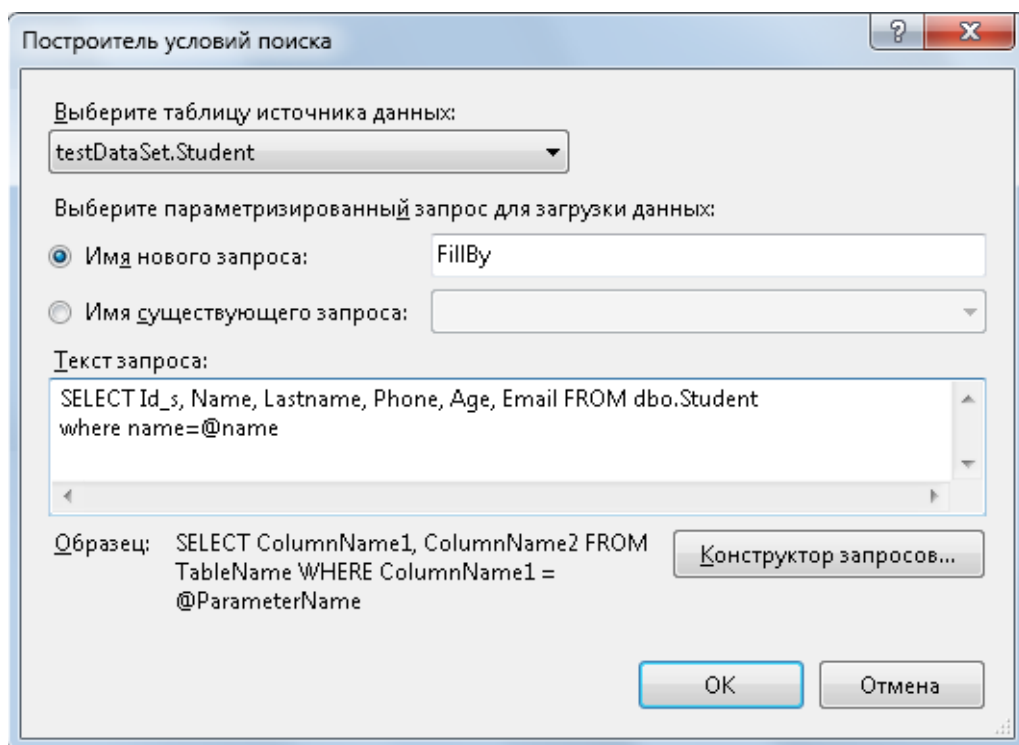


Рис.5. Формування запиту за допомогою пункту Додати запит

Для перевірки коректності побудованого запиту можна скористатись командою Конструктор запитів.

У результаті виконання цієї команди на формі буде розташований новий компонент ToolStrip, який дає змогу вибрати записи за введеним значенням вказаного поля.

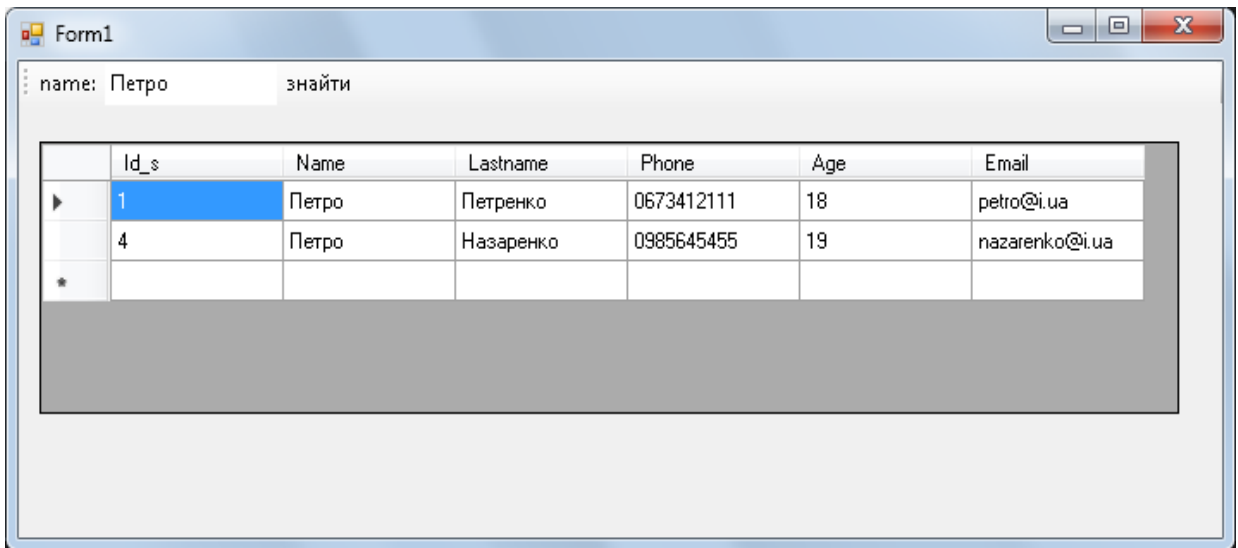


Рис. 6. Результат пошуку за введеним значенням поля "Name"

Пошук даних з декількох таблиць

Зазвичай БД містить декілька зв'язаних між собою таблиць. Під'єднується до проекту така база даних автоматично, причому у вікні Data Sources одразу після під'єднання буде в зручному вигляді відображена вся структура (всі таблиці) даної БД.

Для опрацювання даних, установлення та редагування наявних зв'язків зручно скористатись пунктом Edit DataSet with Designer вкладки Data Sources. Для встановлення зв'язків між таблицями безпосередньо в проекті потрібно мишкою «зв'язати» відповідні поля, після чого відкриється вікно Relation, яке дає змогу обрати та встановити потрібні зв'язки (рис. 7).

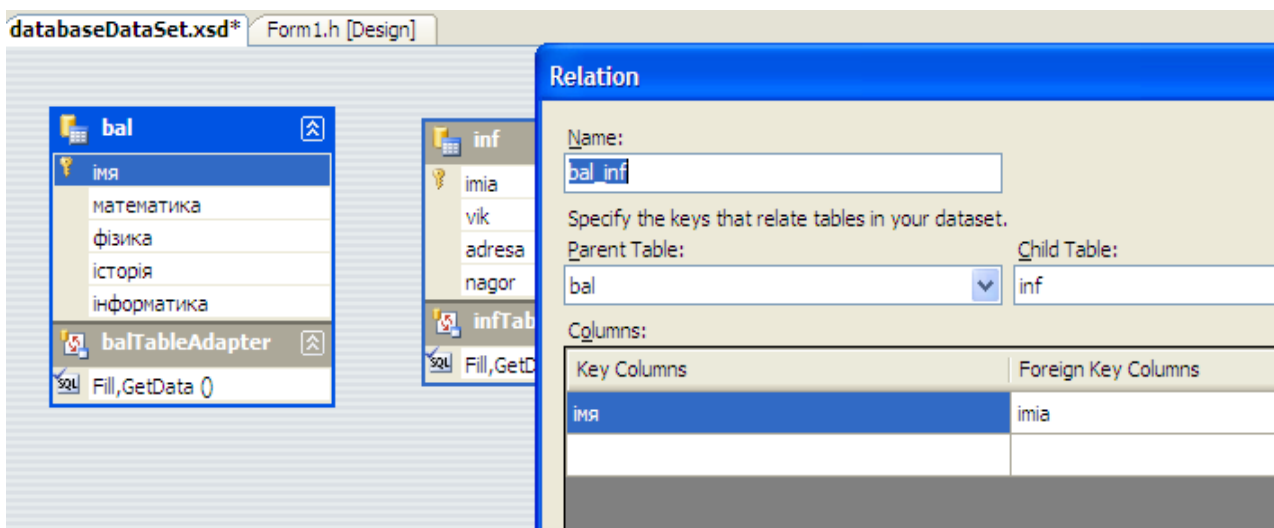


Рис. 7. Зв'язування таблиць

Слід зауважити, що всі зв'язки, встановлені між таблицями таким чином, існують не для реальних таблиць в БД, а лише для їхнього відображення в додатку.

Зауважимо також, що якщо таблиці не зв'язані в БД, не можливо забезпечити їхню спільну роботу в одній формі по введенню даних, оскільки середовище цього не дозволяє. Усі елементи для керування записами будуються лише для однієї таблиці.

Для вибору певної інформації з бази даних, що міститься в різних таблицях, у діалоговому вікні, що відкриється, потрібно написати відповідний багатотабличний запит, використовуючи основні команди мови SQL.

Завдання для самостійного виконання

1. Створити реляційну БД в середовищі MS SQL Server, що складається з двох таблиці, згідно з Вашим варіантом.
2. Розробити графічний інтерфейс користувача засобами MS Visual Studio
3. Установити з'єднання з базою даних (за можливості розглянути різні способи підключення БД до проекту).
4. Переглянути вміст створеного автоматично об'єкта DataSet, скориставшись командою Show Data Sources. Розглянути можливі компоненти для відображення даних.
5. Забезпечити коректне відображення даних (компоненти DataGridView, TextBox, ComboBox) та їх опрацювання за допомогою компонента BindingNavigator.
6. Реалізувати пошук даних згідно з деяким критерієм.

Контрольні запитання

1. Які режими роботи з даними з БД передбачаються у середовищі MS Visual Studio?
2. Яке призначення об'єктів класу SqlConnection?
3. Для чого використовуються об'єкти класу DataSet?
4. Об'єкти якого класу використовуються для заповнення DataSet даними та для зворотної передачі змінених клієнтом даних до джерела?
5. Назвіть основні методи об'єкта DataAdapter?
6. Яким чином можна підключити БД MS SQL Server до проекту MS Visual Studio?
7. Як переглянути вміст об'єкта Data Set (вміст БД, підключеної до проекту)?
8. З якою метою використовують пункт Edit DataSet with Designer вкладки Data Sources?
9. Яке призначення невізуального компонента BindingSource?
10. Який метод використовується для завантаження даних у DataSet?

Опрацювання баз даних засобами VisualStudio у з'єднаному режимі

МЕТА: вивчення особливостей опрацювання даних засобами MS Visual Studio у з'єднаному режимі роботи з БД; робота з об'єктами класів Connection, Command та DataReader.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Для роботи з даними у з'єднаному режимі, який передбачає

безпосереднє отримання даних з бази даних, використовують таку схему:

1. Створення з'єднання із джерелом за допомогою об'єктів класу **Connection**.

2. Налаштування об'єкта **Command** – призначається для виконання команд джерелом даних (SqlCommand й OleDbCommand відповідно до провайдера). Для виконання цих функцій об'єкт має чотири методи: SelectCommand, InsertCommand, UpdateCommand і Deletecommand.

3. Читання даних з їхньою однобічною вибіркою або запис даних за допомогою об'єктів класу **DataReader**.

Робота з об'єктом Connection

Для роботи з MS SQL Server потрібно додати наступні бібліотеки

```
using System.Data;  
using System.Data.SqlClient;
```

У збірці System.Data.dll є класи ADO.NET. Типи даних з System.Data відтворюють не тільки відображення рядків і стовпців, а також відносини між таблицями, первинні ключі тощо. Для БД типу Access слід використовувати бібліотеку System.Data.OleDb.

Об'єкт підключення до бази даних можна одержати, створюючи нові екземпляри відповідних класів підключення. Наприклад, для класу SqlConnection:

```
SqlConnection Conn = new SqlConnection(connectionString);
```

До викладеного додатково потрібно вказати об'єкт підключення (до якої бази даних треба під'єднатися), які права доступу використати тощо. Ці параметри можна задати за допомогою властивості ConnectionString. Наприклад:

```
string connectionString = "Data Source=  
(LocalDB)\MSSQLLocalDB; AttachDbFilename=test.mdf;
```

Integrated Security=True"; – для з'єднання з БД test.mdf, розробленою засобами вбудованого у MS Visual Studio 2015 серверу Microsoft SQL Server.

Наступним кроком є виклик методу Open():

```
Conn.Open();
```

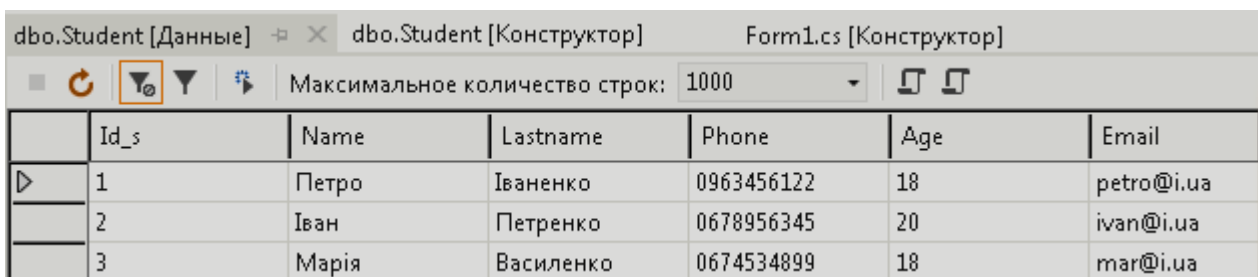
Виконання операцій у з'єднаному режимі роботи з БД

Розглянемо роботу з об'єктами Command та DataReader на прикладі розроблення віконного додатку, який забезпечує з'єднання з БД MS SQL Server та опрацювання даних (оновлення, додавання та пошук згідно критерію) у з'єднаному режимі. Для повернення даних до клієнта у вигляді рядків використовують команду ExecuteReader класу Command; для зміни даних в БД можна скористатись командою ExecuteNonQuery.

Нехай БД test.mdf містить таблицю Student, що має таку структуру:

	Имя	Тип данных	Допустимы значения NULL
Id_s		int	<input type="checkbox"/>
Name		nchar(10)	<input type="checkbox"/>
Lastname		nchar(15)	<input type="checkbox"/>
Phone		nchar(10)	<input checked="" type="checkbox"/>
Age		int	<input checked="" type="checkbox"/>
Email		nchar(15)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

і такі дані:



The screenshot shows a SQL Server Enterprise Manager window displaying the data for the 'Student' table in the 'dbo' schema. The table has columns: Id_s, Name, Lastname, Phone, Age, and Email. The data is as follows:

Id_s	Name	Lastname	Phone	Age	Email
1	Петро	Іваненко	0963456122	18	petro@i.ua
2	Іван	Петренко	0678956345	20	ivan@i.ua
3	Марія	Василенко	0674534899	18	mar@i.ua

Використаємо її для організації опрацювання даних у з'єднаному режимі. Для виведення даних на форму можна використати компонент

ListBox.

Після того, як встановлено з'єднання з БД (створено та налаштовано об'єкт `SqlConnection`), потрібно створити та налаштувати об'єкти `SqlCommand` та `SqlDataReader`:

```
//оголошуємо змінні відповідних класів

static SqlCommand dbCmd;

static SqlDataReader dbReader;

dbCmd = new SqlCommand();

//реалізуємо вибірку даних з таблиці Student

dbCmd.CommandText = "SELECT Id_s, Name, Lastname, Age
FROM Student";

dbReader =
dbCmd.ExecuteReader(CommandBehavior.SingleResult);

while (dbReader.Read())

// виводимо результат запиту у компонент listBox

    listBox1.Items.Add(dbReader.GetValue(0) + "\t" +
dbReader.GetValue(1) + "\t" + dbReader.GetValue(2) + "\t" +
dbReader.GetValue(3) + "\t" + dbReader.GetValue(4) + "\t" +
dbReader.GetValue(5));

dbReader.Close();
```

Після запуску проекту на виконання, форма має такий вигляд:

1	Петро	Іваненко	0963456122	18	petro@i.ua
2	Іван	Петренко	0678956345	20	ivan@i.ua
3	Марія	Василенко	0674534899	18	mar@i.ua

Рис. 1. Виведення даних з таблиці Student

Для **видалення** даних з таблиці можна використовувати запит:

```
dbCmd.CommandText = "Delete from Student where Id_s=1";
// видаляємо запис з ID=1.

dbCmd.ExecuteNonQuery(); // змінюємо дані в БД
```

Для **додавання** даних можна використовувати запит

```
dbCmd.CommandText = "Insert into Student (Id_s, Name,
Lastname, Age) Values (6, 'Андрій', 'Антоненко', 19)";
dbCmd.ExecuteNonQuery(); // змінюємо дані в БД
```

Для **оновлення** значення CommandText слід вказати відповідний Sql-запит на оновлення даних.

Завдання для самостійного виконання

1. Створити реляційну БД в середовищі SQL Server або MS Access, що складається з однієї таблиці, згідно Вашого варіанту.
2. Створити новий проект Windows Forms в середовищі MS Visual Studio та підключити розроблену БД до проекту.
3. Розглянути особливості роботи з даними у з'єднаному режимі: відображення всіх даних з таблиці та даних згідно з деяким

критерієм, видалення та оновлення даних (значення поля задається з клавіатури під час виконання проекту).

4. Реалізувати додавання даних до таблиці БД та пошук даних згідно з деяким критерієм.

Контрольні запитання

1. Яка особливість опрацювання даних у з'єднаному режимі роботи з джерелом даних?
2. Наведіть приклади, коли рекомендовано працювати з джерелом даних у з'єднаному режимі.
3. Назвіть призначення та основні методи об'єкта SqlCommand.
4. Для чого використовуються об'єкти класу DataReader?
5. Як задати значення властивості ConnectionString об'єкта SqlConnection?
6. Для чого використовують команду ExecuteReader класу Command?
7. Як організувати видалення даних з таблиці у з'єднаному режимі роботи з джерелом даних?
8. Як реалізувати запис даних до таблиці у з'єднаному режимі роботи з джерелом даних?
9. Яким способом можна змінити дані в БД?
10. Яке призначення методів ExecuteReader та ExecuteNonQuery()?

Робота з компонентами SqlConnection, SqlDataAdapter та DataSet

МЕТА: ознайомлення з особливостями роботи з компонентами SqlConnection, SqlDataAdapter і DataSet; розроблення клієнтської програми для роботи з даними, що містяться в БД SQL Server, використовуючи відповідні властивості та методи компонентів.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Доступ до даних, що містяться в БД MS SQL Server, та їх опрацювання можна реалізувати за допомогою невізуальних компонентів SqlConnection, SqlDataAdapter і DataSet. Як зазначалось раніше, компонент SqlConnection забезпечує з'єднання з базою даних (джерелом даних), компонент SqlDataAdapter – взаємодію з базою даних, а DataSet – зберігання даних, отриманих від джерела даних в рядок виконання SQL-запиту. Механізм взаємодії компонентів доступу до даних і їхнє відображення показані на рис. 1.

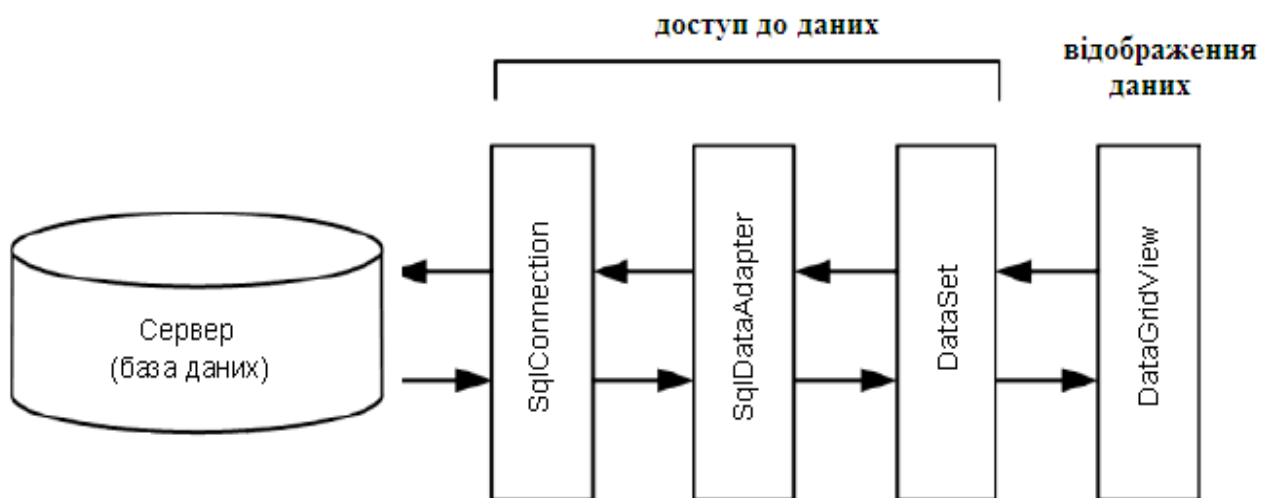


Рис. 1. Взаємодія компонентів, що забезпечують доступ та відображення даних

Для реалізації віконного додатка опрацювання даних з БД, використаємо БД Test із таблицею Student, що має наступну структуру:

	Имя	Тип данных	Допустимы значения NULL
PK	Id_s	int	<input type="checkbox"/>
	Name	nchar(10)	<input type="checkbox"/>
	Lastname	nchar(15)	<input type="checkbox"/>
	Phone	nchar(10)	<input checked="" type="checkbox"/>
	Age	int	<input type="checkbox"/>
	Email	nchar(15)	<input checked="" type="checkbox"/>

Розглянемо детальніше підключення та опрацювання даних. Спочатку на форму треба помістити компонент SqlConnection, потім – SqlDataAdapter, DataSet і DataGridView (для відображення даних з таблиці БД).

Компонент SqlConnection налаштовується таким чином. Спочатку потрібно клікнути на значку списку, який є в рядку властивості ConnectionString (рядок з'єднання) і вибрати New Connection (Нове з'єднання). У діалоговому вікні, що відкриється, потрібно обрати кнопку Change і у вікні Change Data Source вибрати тип джерела даних (у нашому випадку – "файл бази даних Microsoft Sql Sever"). Далі у вікні Add Connection треба клікнути на кнопці Browse і вказати файл бази даних.

Тоді потрібно налаштувати компонент SqlDataAdapter, що забезпечує взаємодію з базою даних. Виконати налаштування компонента SqlDataAdapter можна за допомогою майстра налаштування або вручну. Перше вікно майстра (рис. 2) з'являється при додаванні на форму компонента SqlDataAdapter.

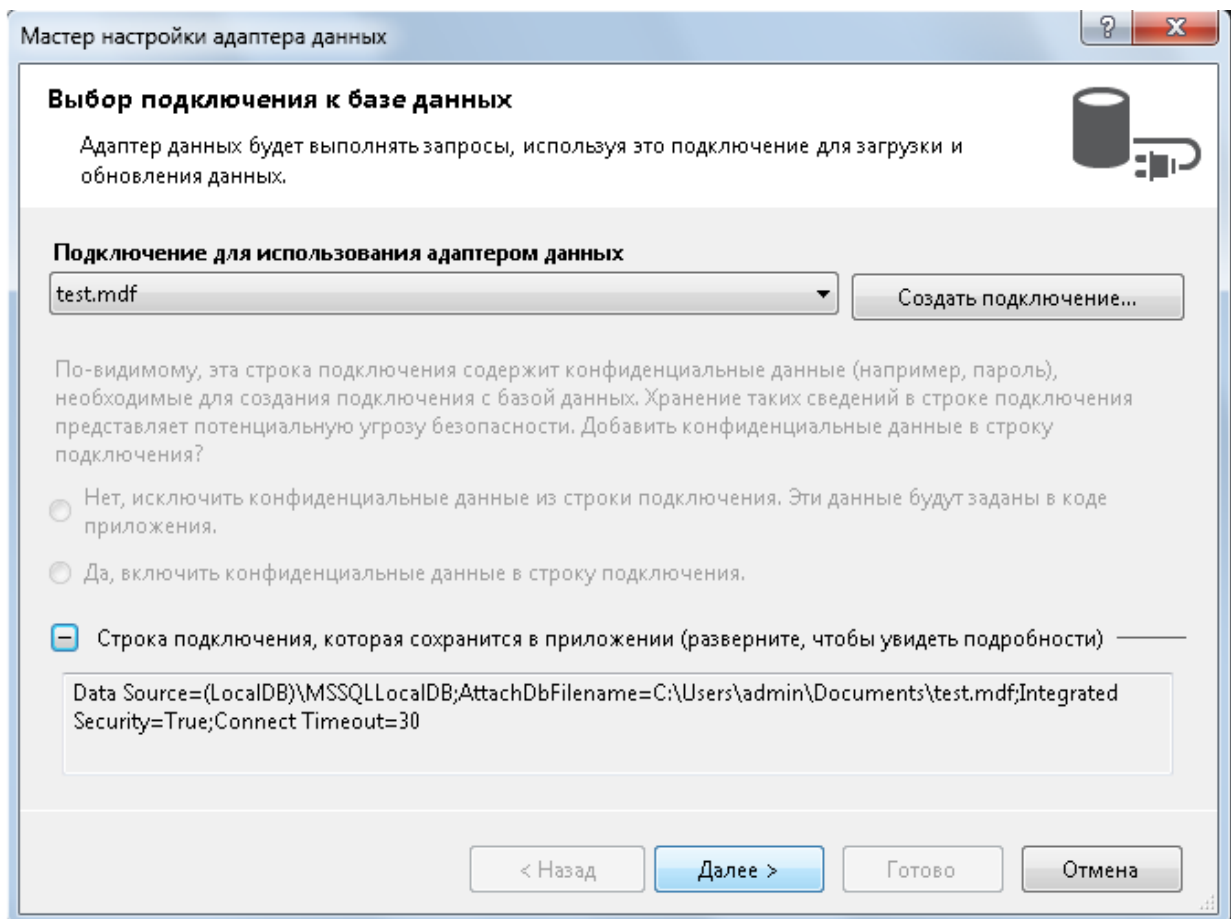


Рис. 2. Підключення до бази даних

У цьому вікні потрібно вибрати з'єднання (connection) і натиснути на кнопку **Далі**. У наступному вікні також треба натиснути на кнопку **Далі**. У вікні, яке з'являється на третьому кроці, треба натиснути на кнопку **Конструктор запитів**, а потім у вікні **Додати таблицю** (рис. 3) вибрати таблицю даних і натиснути на кнопку **Додати**.

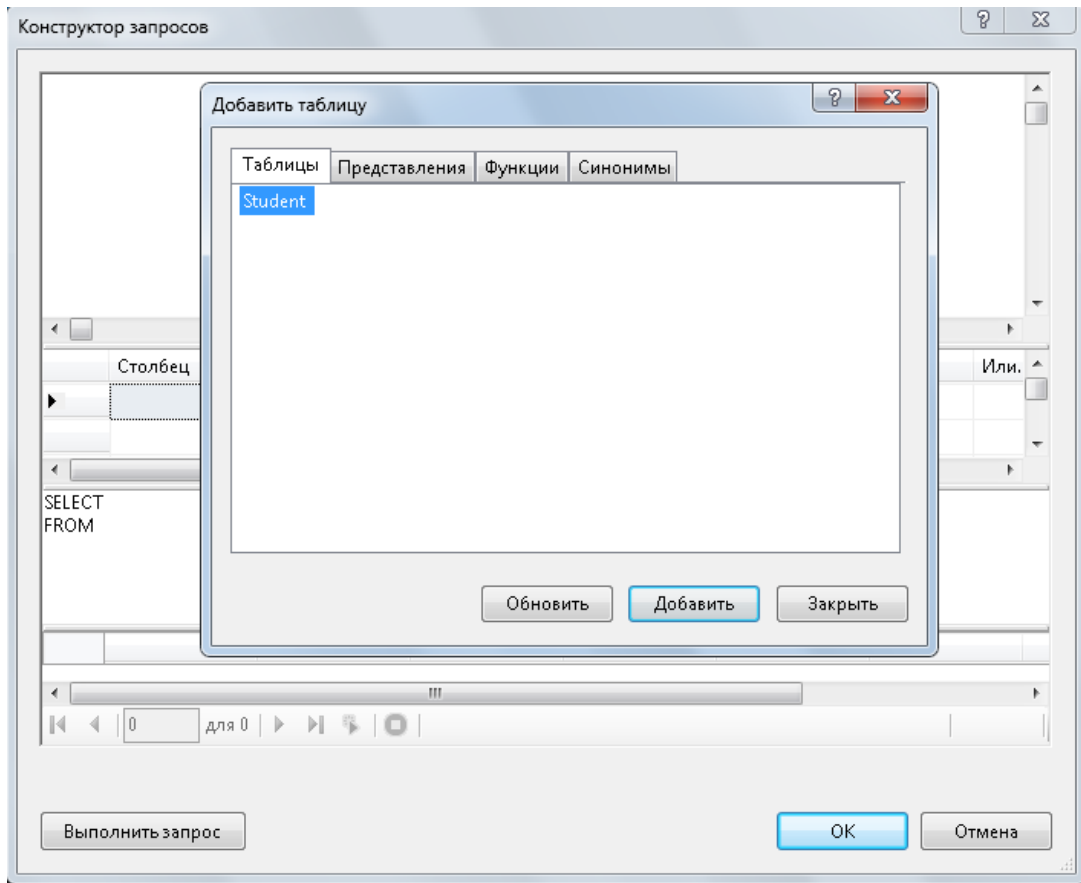


Рис. 3. Вибір таблиці з даними

Далі у вікні **Конструктор запитів** слід вказати поля таблиці, з яких треба вибрати дані (якщо необхідно отримати дані з усіх полів, слід обрати All Columns). У результаті виконання описаних дій компонент SqlDataAdapter буде налаштований. У властивостях SelectCommand, DeleteCommand, InsertCommand і UpdateCommand будуть записані команди, що забезпечують взаємодію з базою даних.

Компонент DataSet (набір даних) зберігає дані, отримані з бази даних. Налаштування компонента DataSet виконується так. Спочатку у вікні **Додавання набору даних** (рис. 4), яке з'являється на екрані в момент додавання компонента на форму, треба вибрати перемикач **Нетипізований набір даних** і натиснути на кнопку OK.

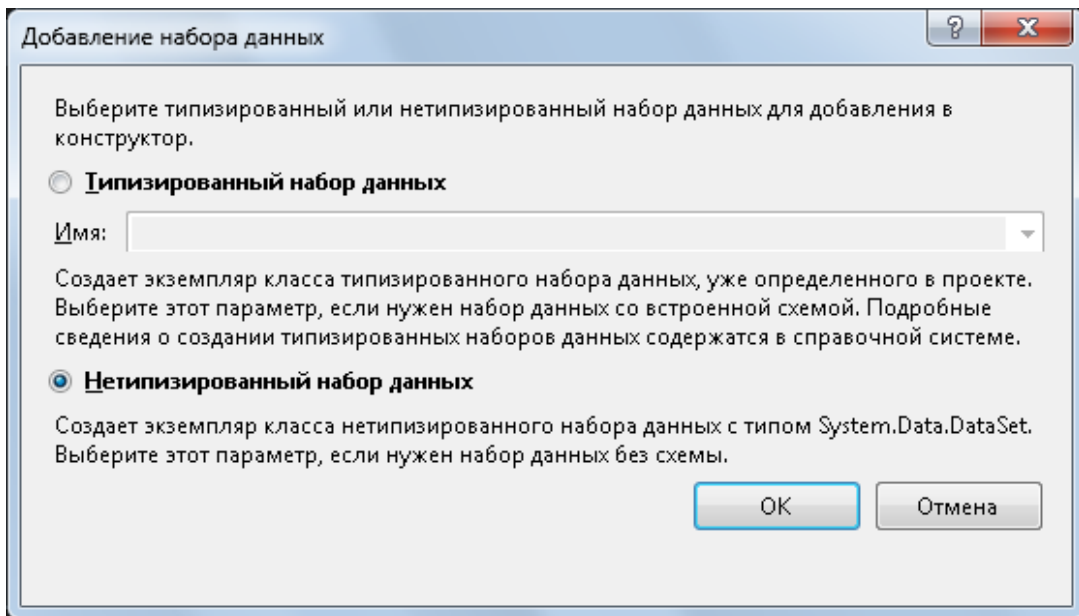


Рис. 4. Наладштування компонента Dataset

У колекцію Tables треба додати таблицю (рис. 5), а в колекцію Columns – стовпці і у кожного елемента колекції Columns встановити значення властивості ColumnName.

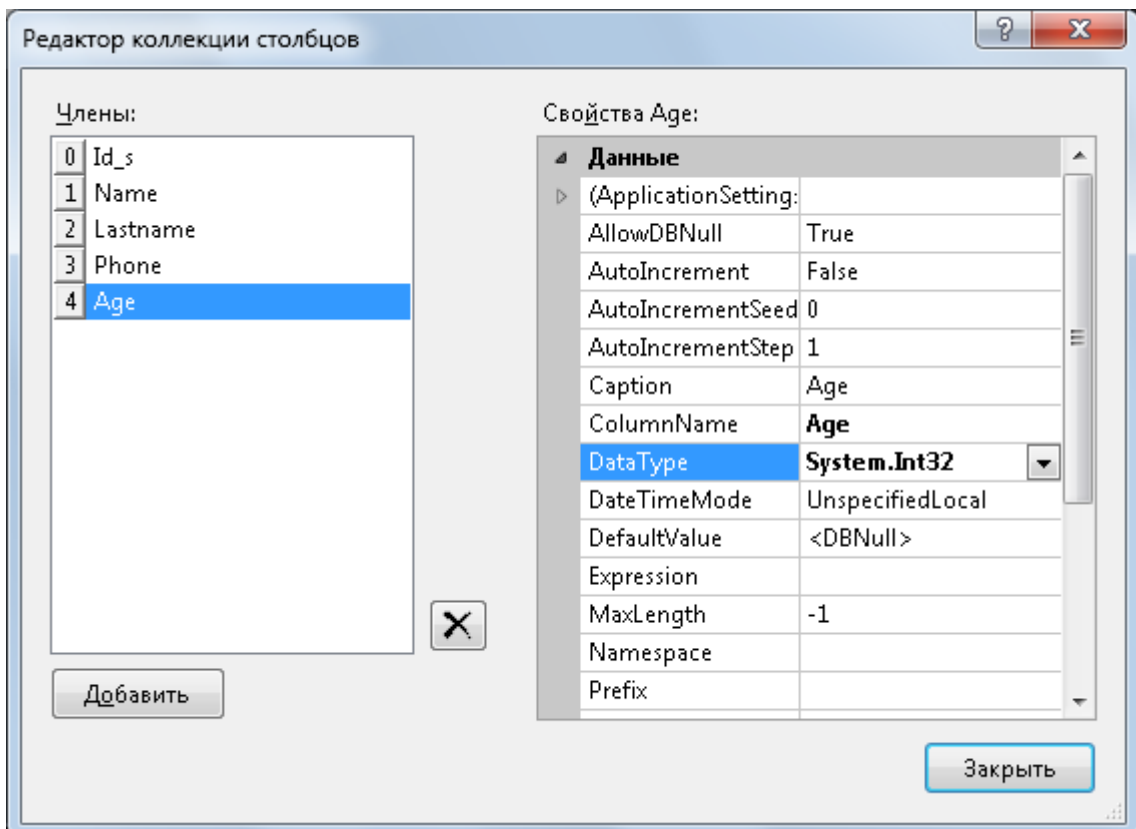


Рис. 5. Створення структури таблиці

Відображення даних

Відображення даних у формі таблиці можна реалізувати за допомогою компонента DataGridView. Для цього потрібно у вікні властивостей компонента у властивості DataSource обрати як джерело даних налаштований перед цим DataSet, а у властивості DataMember вказати відповідну таблицю БД (рис. 6).

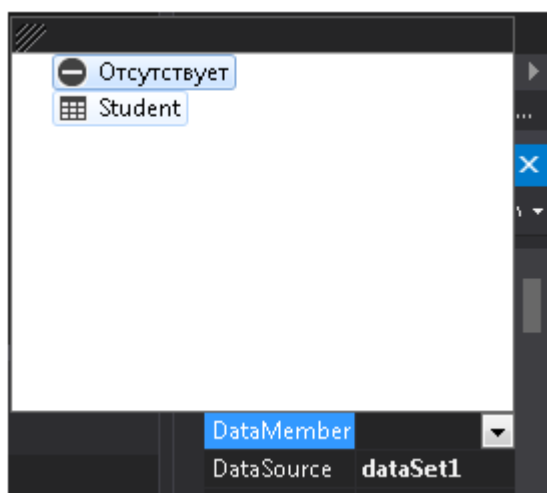


Рис. 6. Приклад налаштування компонента DataGridView

Наступне, що треба зробити, – створити функції обробки подій FormLoad і FormClosing форми. Функція обробки події FormLoad повинна завантажити дані, події FormClosing – зберегти зміни, зроблені користувачем. Завантаження даних виконує метод Fill (в результаті серверу направляється команда SELECT) компонента SqlDataAdapter, якому як параметр передається таблиця об'єкта DataSet, що заповнюється в результаті виконання команди:

```
// початок роботи
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e)
{
// завантажити дані
SqlDataAdapter1.Fill(dataTable1);
}
```

```
// оновлення даних в момент завершення роботи програми
private: System::Void Form1_FormClosing(System::Object^
sender,
System::Windows::Forms::FormClosingEventArgs^ e)
{
SqlDataAdapter1.Update(dataSet1.Tables["Student"]);
}
```

Вибір інформації з бази даних

При роботі з базою даних користувача, як правило, цікавить не весь її вміст, а деяка конкретна інформація. Вибрати потрібну інформацію з бази даних можна, направивши серверу SQL-команду SELECT.

Для реалізації вибору необхідної інформації на форму потрібно додати поле редагування та кнопку для здійснення пошуку:

```
// клік на кнопці Знайти
private: System::Void button1_Click(System::Object^
sender,
System::EventArgs^ e)
{
dataSet1.Clear(); // видаляємо старі дані
sqlDataAdapter1.SelectCommand.CommandText = " select *
from student where name =' " + textBox1.Text + " ' ";
//або через параметри
//sqlDataAdapter1.SelectCommand.Parameters["name"].Value
= "%" + textBox1.Text + "%";
// виконуємо команду
sqlDataAdapter1.Fill(dataTable1);
}
```

Завдання для самостійного виконання

1. Створити реляційну БД у середовищі Microsoft Sql Server згідно з Вашим варіантом.
2. Створити новий проект Windows Form у MS Visual Studio
3. Установити з'єднання з базою даних, використовуючи компоненти SqlConnection, SqlDataAdapter та DataSet.
4. Забезпечити коректне відображення даних за допомогою DataGridView.
5. Реалізувати можливість видалення та додавання даних до таблиці БД.
6. Реалізувати пошук даних згідно з варіантом. Розробити зручний графічний інтерфейс користувача.

Контрольні запитання

1. Поясніть схему взаємодії користувача з БД у роз'єднаному режимі роботи з даними.
2. Яке значення може мати властивість ConnectionString SqlConnection?
3. Яке призначення компонента SqlDataAdapter ?
4. Який метод компонента SqlDataAdapter відповідає за завантаження даних в DataSet?
5. Яким чином реалізувати оновлення даних в БД?
6. Який метод компонента DataSet дає змогу видалити дані?
7. Як реалізувати пошук даних у таблиці БД згідно з деяким критерієм?
8. Як реалізувати в клієнтській програмі видалення запису з таблиці БД?

9. Яка властивість компонента DataSet відповідає за зв'язок з джерелом даних?
10. Які компоненти MS Visual Studio можна використати для відображення даних з БД?

Опрацювання зв'язків між таблицями бази даних засобами MS Visual Studio

МЕТА: вивчення особливостей реалізації зв'язків виду «один до багатьох» та «багато до багатьох» між таблицями бази даних MS SQL Server засобами MS Visual Studio.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Зазвичай, розрізняють такі типи зв'язків між таблицями БД: "один до одного", "один до багатьох" та "багато до багатьох". Для реалізації між таблицями БД зв'язку виду «багато до багатьох», використовують, як правило, додаткову стикувальну таблицю.

Для вивчення особливостей опрацювання зв'язків між таблицями БД в клієнтській програмі потрібно реалізувати відповідну базу даних. Розглянемо БД MS SQL Server з такою структурою (рис.1):

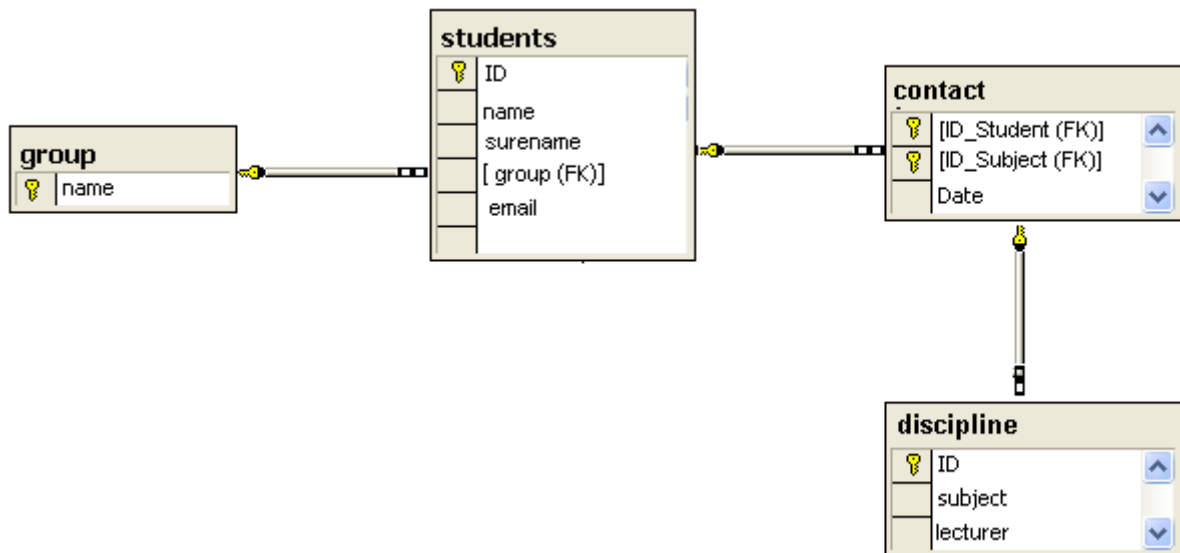


Рис. 1. Перегляд структури бази даних в MS SQL Server

У середовищі MS Visual Studio створимо новий проект. Скориставшись функціями доступу до даних, додамо нове джерело даних.

При виборі об'єктів, які слід включити в нове джерело даних, потрібно обрати всі створені таблиці (рис. 2).

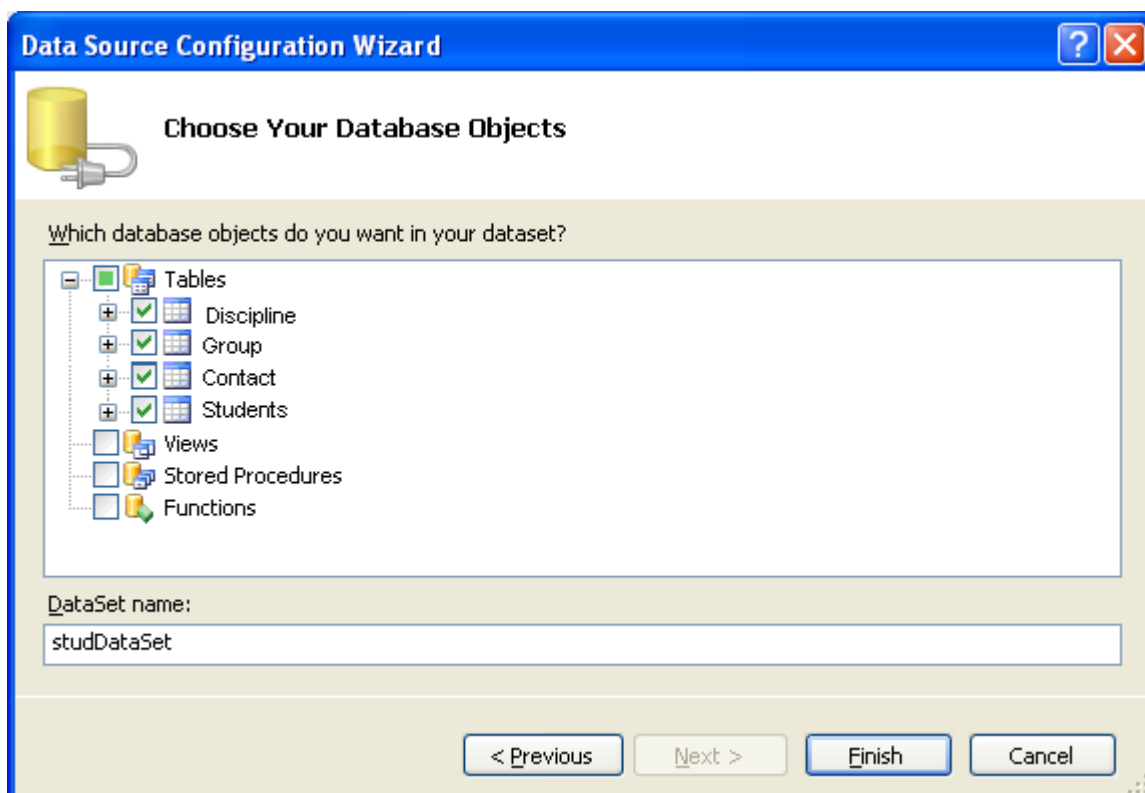


Рис. 2. Вибір об'єктів, які слід включити в нове джерело даних

У результаті дій майстра буде згенерована XSD-схема зі строго типізованим об'єктом DataSet. Для відображення структури можна скористатись вікном Solution Explorer. Можна переглянути властивості кожного встановленого зв'язку між таблицями. Для цього слід вибрати цей зв'язок, натиснути на праву клавішу миші та в контекстному меню вибрати пункт Edit Relation.

Об'єкт DataRelation має два важливих параметри:

– **Relation** встановлює механізм для відображення зв'язаних даних у двох таблицях;

– **ForeignKeyConstraint** накладає обмеження зовнішнього ключа на батьківську таблицю.

За замовчуванням для об'єктів DataRelation, що автоматично згенеровані середовищем Visual Studio, є встановлений лише параметр Relation. Тому рекомендують встановлювати режим Both Relation and Foreign Constraint та значення Cascade для правил Update Rule та Delete Rule, а для Accept/Reject Rule значення None.

Для уникнення помилок зв'язування для цього прикладу слід встановити зв'язки на XSD-схемі у режим Relation Only

Відображення на формі зв'язку «один до багатьох»

Нехай зв'язок «один до багатьох» має на меті реалізувати підтримку цілісності значень. Тобто значення, які заносяться до комірок таблиці, повинні відповідати наперед заданій множині значень, наприклад, вибиратися зі списку. Розглянемо детальніше реалізацію цього випадку у клієнтській програмі.

У вікні Data Sources встановимо залежний стовпець Group таблиці **Students** у режим графічного компонента **ComboBox**. Для таблиці **Students** встановимо режим Details та перетягнемо її на форму (рис. 3).

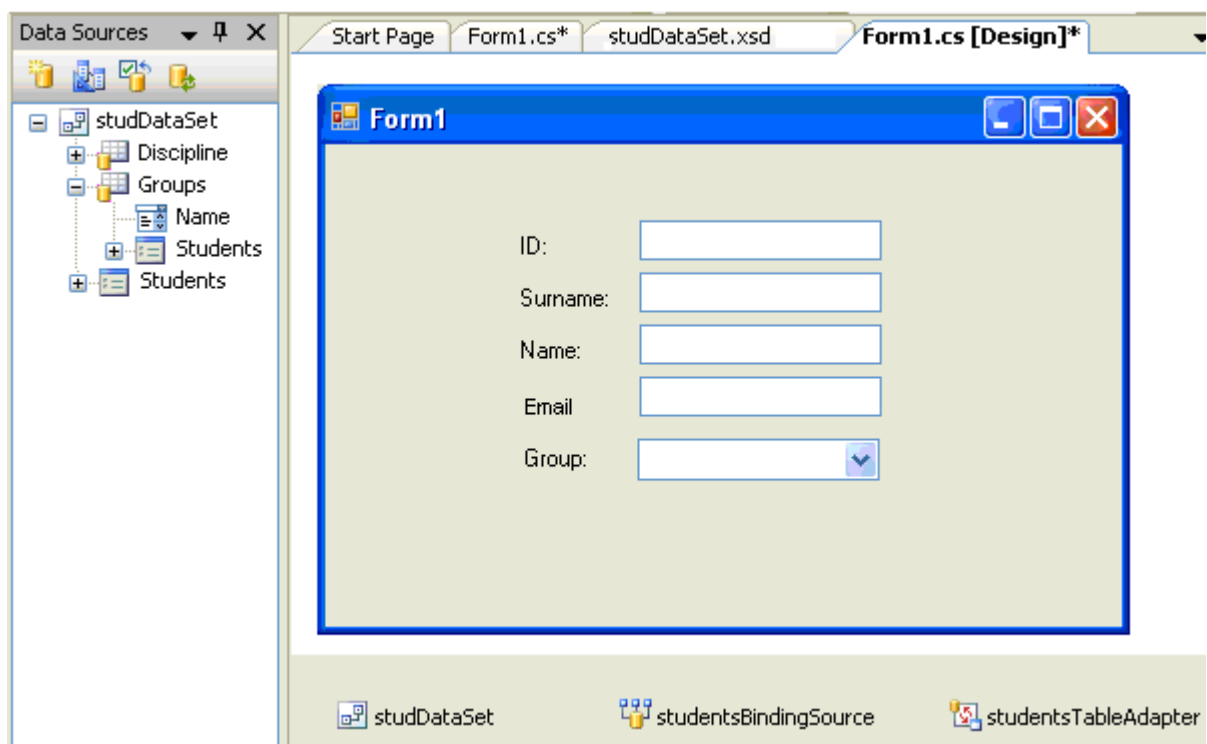


Рис. 3 . Реалізація зв'язку «один до багатьох» для забезпечення цілісності значень

Далі перетягуємо лівою клавішею миші таблицю **Group** на відповідний графічний компонент ComboBox з міткою Group на формі. У результаті цих дій майстер додасть у розділі Components Tray компоненти GroupBindingSource та GroupTableAdapter. Після цього можемо запустити клієнтську програму на виконання (рис. 4).

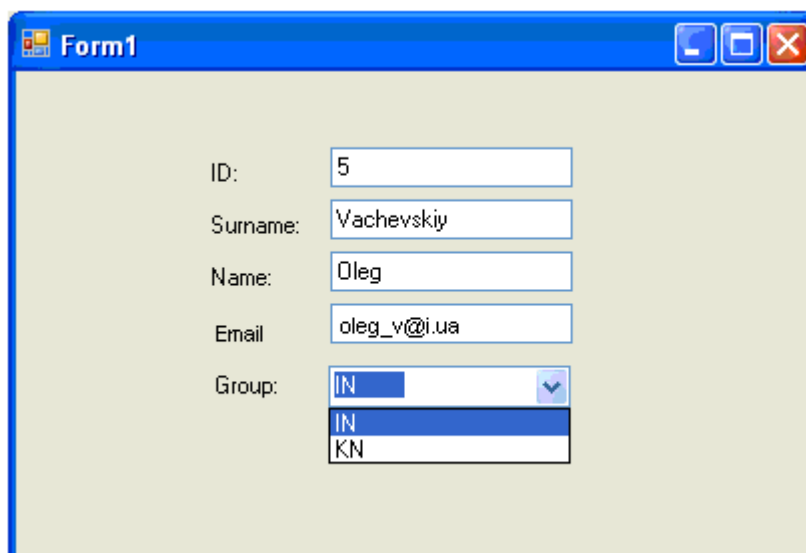


Рис. 4. Програма-клієнт з елементами ComboBox

Подібному налаштуванню з вибором значень зі списку піддається і компонент **DataGridView**. Для цього необхідно з вікна DataSources перетягнути таблицю Students на форму. Потім натиснути правою клавішею миші на компонент DataGridView, що є на формі, та у контекстному меню вибрати пункт Edit Columns.

У діалоговому вікні Edit Columns для стовпців, що мають мати випадні списки, необхідно встановити параметр ColumnType у значення DataGridViewComboBoxColumn. Далі у групі параметрів Data потрібно вибрати джерело даних для параметра DataSource. Якщо для таблиці, з якою необхідно зв'язатися, ще немає у розділі Components Tray форми відповідної компоненти BindingSource, тоді слід розкрити гілки дерев Other Data Sources, Project Data Sources та в строго типізованому об'єкті DataSet нашого проекту вибрати необхідну для зв'язування таблицю.

Для двох інших параметрів ValueMember та DisplayMember необхідно вибрати з таблиці, з якою зв'язуємося, такі стовпці: перший, що виступає у ролі ключа для зв'язування, другий – що призначений для відображення.

Після запуску на виконання клієнтська програма може виглядати як на рис. 5.

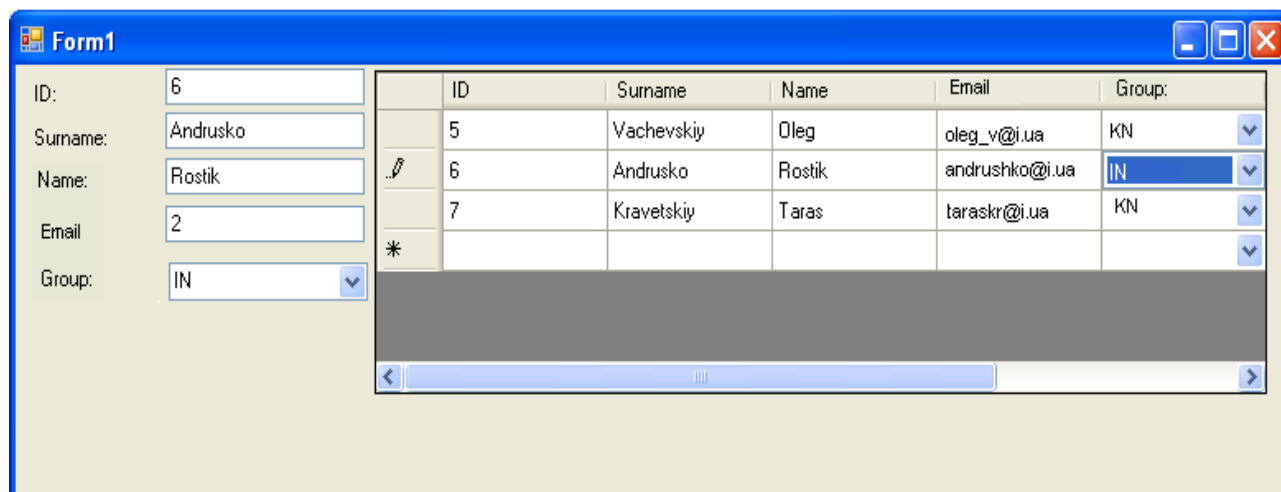


Рис. 5. Програма-клієнт з DataGridView із випадними списками

Відображення на формі зв'язку «багато до багатьох»

Такий зв'язок, що у базі даних реалізується за допомогою стикувальної таблиці та двох зв'язків «один до багатьох», на формі програми відображається у вигляді двох таблиць: однією з учасників зв'язку та додаткової стикувальної таблиці. Між головною та стикувальною таблицею встановлюється зв'язок за реляційною схемою «головний – підлеглий». Між стикувальною таблицею та другим учасником зв'язку «багато до багатьох» налаштовується зв'язок у вигляді підтримки цілісності значень.

Перетягнемо з вікна Data Sources на форму таблицю **Students**, підлеглу таблицю **Contact** та її стовпець Date.

Таблицю **Students** можемо розмістити на формі у вигляді сітки або ж у деталізованому вигляді. Для стикувальної таблиці **Contact** з метою полегшення введення значення дати додатково розмістимо на формі стовпець Date у вигляді компонента DateTimePicker.

Таблиця **Contact** повинна відображати зв'язані дані з таблиці **Discipline** для поточної стрічки таблиці **Students**. З таблиці **Discipline** необхідно відображати значення двох стовпців Subject та Lecture. Тому перед підстановкою їх у таблицю **Contact** попередньо необхідно їх об'єднати за допомогою формули.

Зовнішнє об'єднання стовпців

Перейдемо у графічний режим XSD-схеми, виберемо мишею метод Fill, GetData() таблиці **Discipline** та у вікні Properties відкорегуємо SQL-запит з бази даних, натиснувши на кнопку параметра CommandText.

У вікні редактора SQL-запитів Query Builder (рис. 6) додамо у конструкцію SELECT обчислювальний стовпець, який є результатом конкатенації двох стовпців з усіченими пробілами на кінці:

```
RTRIM(Subject) + ' ' + RTRIM(Lecturer) AS SubjLect
```

Після закриття вікна редактора система видасть повідомлення з пропозицією перебудувати команду передачі оновлення у базу даних.

Цю пропозицію слід відхилити та залишити команду оновлення без змін

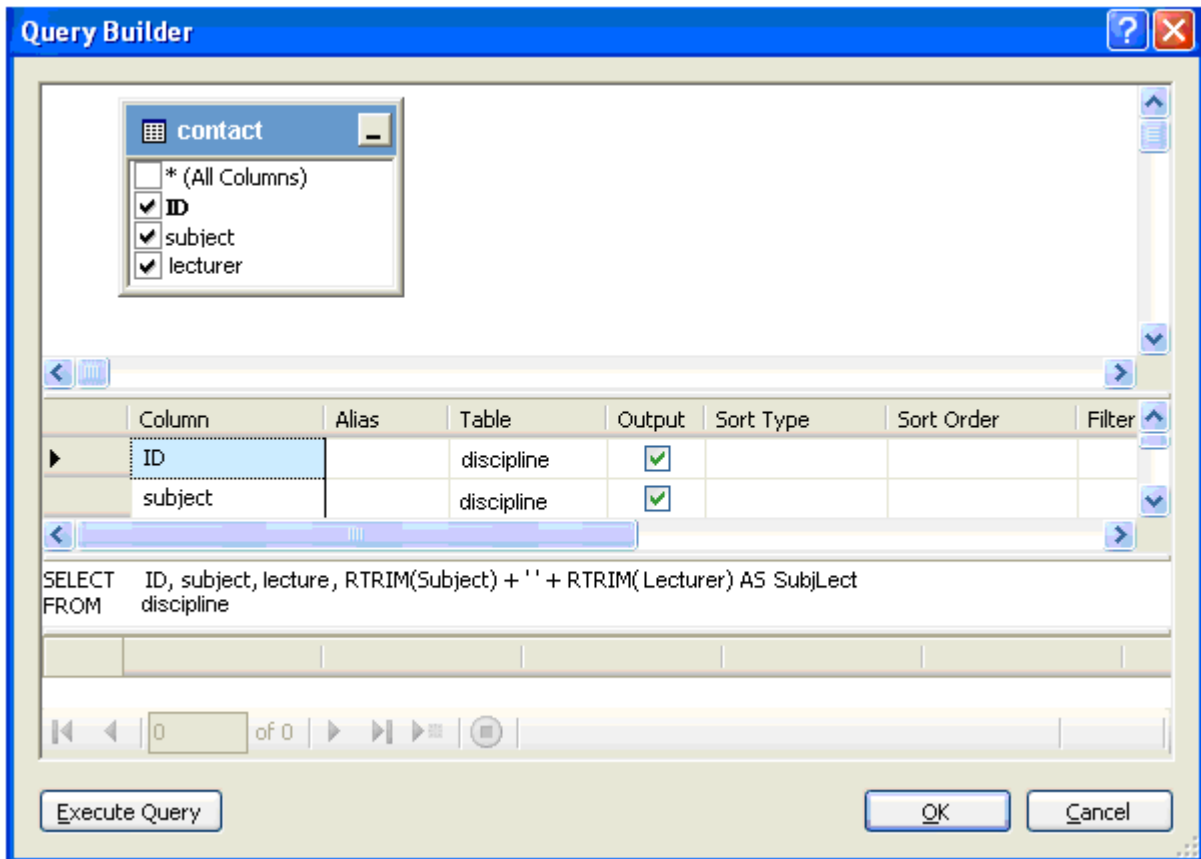


Рис. 6. Редактор Query Builder для побудови SQL-запитів

Після коригування XSD-схему необхідно зберегти. Після цього переходимо знову у режим дизайну форми програми.

Далі налаштовуємо таблицю **Contact** у вигляді підтримки цілісності значень, тобто відображення значень таблиці **Discipline** у вигляді стовпців з випадними списками. Для цього натискаємо на праву кнопку миші на компоненті DataGridView таблиці **Contact** та у контекстному меню вибираємо пункт Edit Columns.

Стовпець ID_Student стикувальної таблиці використовується лише для забезпечення зв'язку з головною таблицею, і тому може бути вилучений з перегляду (рис. 7).

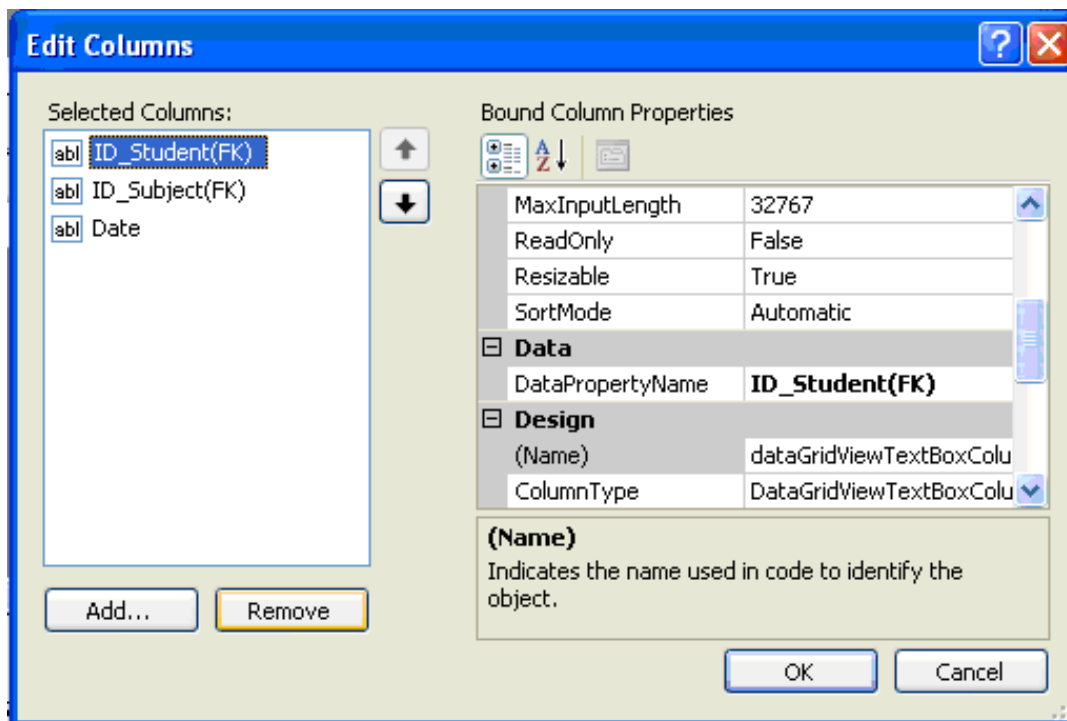
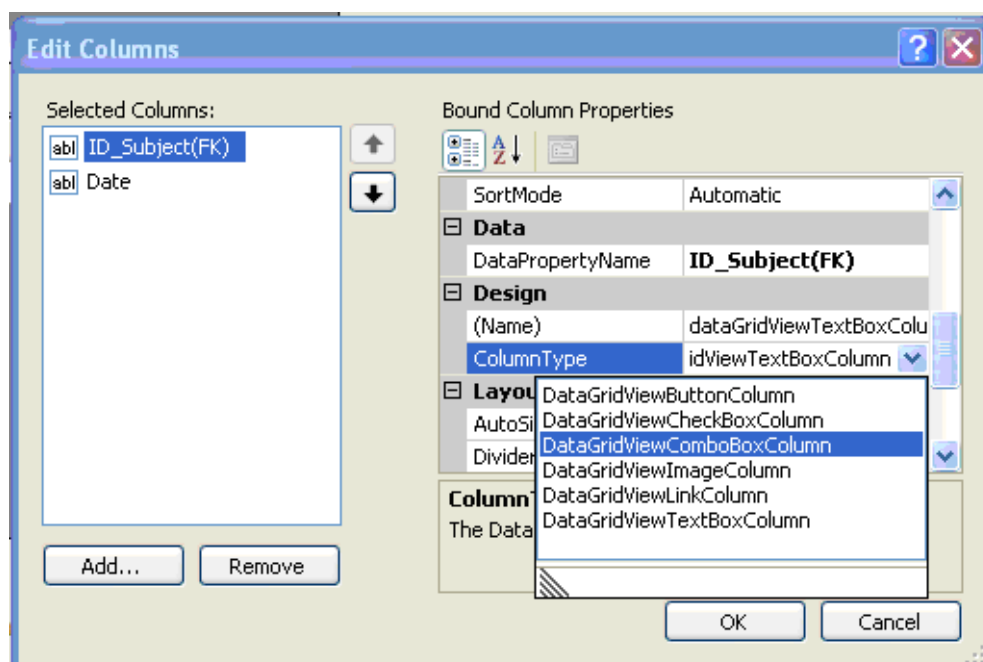
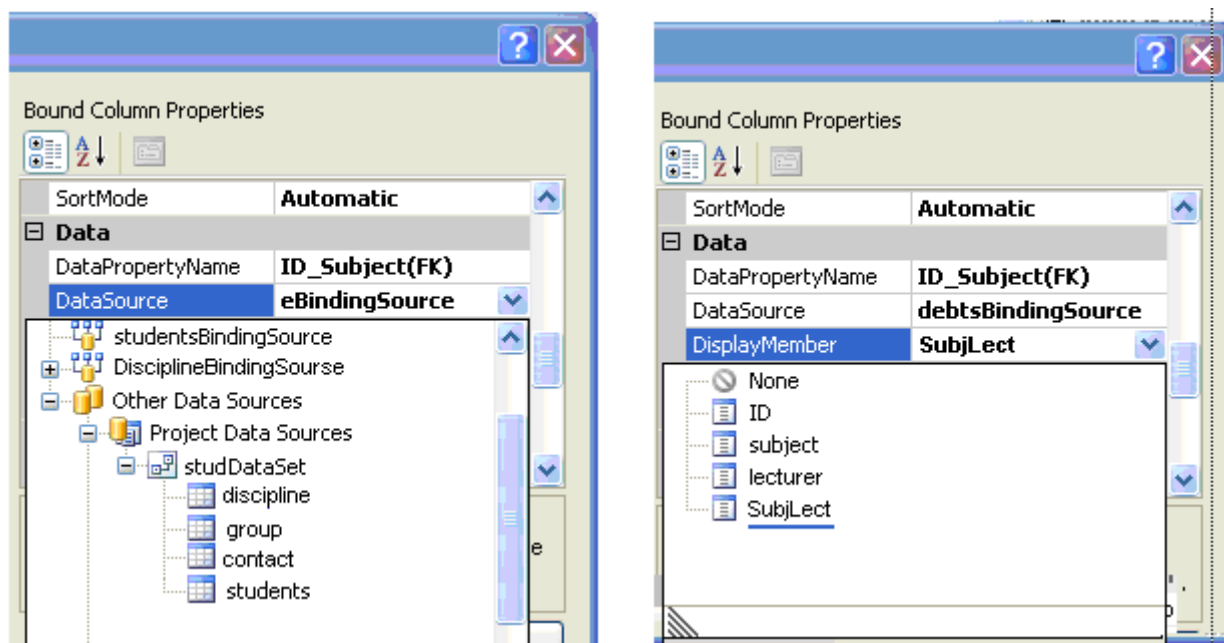


Рис. 7. Вилучення стовпця з компоненти DataGridView

Тепер необхідно налаштувати стовпець ID_Subject на відображення значення за зовнішнім ключем з таблиці **Discipline** (рис. 9).



a)



б)

в)

Рис. 8. Налаштування стовпця на відображення значення за зовнішнім ключем

Для параметра `ColumnType` обираємо значення `DataGridViewComboBoxColumn` та обираємо джерело даних для параметра `DataSource`. Оскільки дані з таблиці **Discipline** ще не були відображені на формі, то для цього необхідно розкрити гілки дерев `Other Data Sources`, `Project Data Sources` та в об'єкті `DataSet` вибрати таблицю **Discipline**.

Для параметра `ValueMember` потрібно вибрати стовець таблиці **Discipline**, що виступає у ролі ключа для зв'язування. Для параметра `DisplayMember` виберіть стовець таблиці **Discipline**, значення якого має відображатися.

У параметрі `HeaderText` потрібно задати надпис для стовпця `ID_Subject`, що відображається на компоненті `DataGridView`. Далі слід налаштувати компоненту `DataGridView` для таблиці **Students**. Стовпець `Group` цієї таблиці має давати можливість вибирати значення зі списку, тобто мати реляційну залежність, відповідно, з таблицею **Group**.

Для уникнення помилки часу виконання, що пов'язана з

обмеженнями зовнішнього ключа, таблиці повинні завантажуватися в об'єкт DataSet у строго визначеній послідовності. У нашому випадку таблиці **Students** та **Contact** повинні завантажуватися останніми.

```
private: System::Void Form1_Load
(System::Object^ sender, System::EventArgs^ e)
{
    this.GroupTableAdapter.Fill(this.StudDataSet.Group);
    this.DisciplineTableAdapter.Fill(this.StudDataSet.Discipline);
    this.StudentsTableAdapter.Fill(this.StudDataSet.Students);
    this.ContactTableAdapter.Fill(this.StudDataSet.Contact);
}
```

Також необхідно налаштувати кнопку з дискетою, що розміщена на панелі навігатора таблиці **Students**, для збереження у базі даних інформації зі стикувальної таблиці **Contact**.

```
private: System::Void StudentsBindingNavigatorSaveItem_Click
(System::Object^ sender, System::EventArgs^ e)
{
    this.Validate();
    this.StudentsBindingSource.EndEdit();
    this.StudentsTableAdapter.Update(this.StudDataSet.Students);
    this.ContactBindingSource.EndEdit();
    this.ContactTableAdapter.Update(this.StudDataSet.Contact);
}
```

Завдання для самостійного виконання

1. Продумати та створити реляційну БД в середовищі MS SQL Server згідно з Вашою предметною областю з 3 – 4 таблиць, що обов'язково містить зв'язки "один до багатьох" та "багато до багатьох".

Установити всі відповідні зв'язки. Заповнити таблиці деякими тестовими даними.

2. Створити проект Windows Forms у середовищі MS Visual Studio. Скориставшись функціями доступу до даних, установити з'єднання з базою даних.

3. Згідно з описаним алгоритмом розробити клієнтську програму, що дає змогу реалізувати зв'язок типу "один до багатьох".

4. Згідно з описаним алгоритмом розробити клієнтську програму, що дає змогу реалізувати зв'язок «багато до багатьох», використовуючи з'єднання стовпців.

5. Оформити інтерфейс програми якнайкраще.

Контрольні запитання

1. Які типи зв'язків можна встановити між таблицями бази даних?
2. Яким чином реалізувати зв'язок типу "багато до багатьох"?
3. Наведіть приклади масивів даних, між якими є зв'язок типу "один до багатьох" або "багато до одного"?
4. Який пункт меню середовища MS Visual Studio дає змогу переглянути зв'язки між таблицями підключеної БД?
5. Як у MS Visual Studio переглянути властивості кожного встановленого зв'язку між таблицями підключеної до проекту БД?
6. Для чого призначений параметр Relation об'єкта DataRelation?
7. Які компоненти MS Visual Studio можна використати для забезпечення опрацювання зв'язку типу "один до багатьох"?
8. Чи має значення послідовність завантажень таблиць БД в об'єкт DataSet? В яких випадках?
9. Як реалізувати об'єднання двох стовпців таблиць БД?

10. Яким чином вилучити сповпець таблиці DataGridView з перегляду?

Робота з BLOB-даними

МЕТА: ознайомлення з особливостями запису до поля таблиці БД зображень та масивів тексту; розроблення клієнтської програми, що дає змогу записувати та зчитувати BLOB-дані до БД.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

BLOB (Binary Large Object) – це загальний термін для типів даних text, ntext, varchar(max), nvarchar(max), image та varbinary(max) із максимальним об'ємом даних до 2 Гбайт. BLOB-дані можна записувати в базу даних у вигляді двійкових чи символьних даних, залежно від типу поля джерела даних.

Для збереження великих об'ємів тексту, як звичайного, так і форматowanego (RTF), рекомендується використовувати тип nvarchar(max), для збереження об'єктів, наприклад, фотографій, використовувати тип даних varbinary(max).

Роботу з даними такого типу (BLOB-полями) розглянемо на такому прикладі: створимо базу даних MS SQL Server з однієї таблиці teacher (рис. 1).

	Имя	Тип данных	Допустимы значения NULL
PK	Id	int	<input type="checkbox"/>
	surname	nchar(10)	<input checked="" type="checkbox"/>
	post	nchar(10)	<input checked="" type="checkbox"/>
	photo	varbinary(MAX)	<input checked="" type="checkbox"/>
	notes	nvarchar(MAX)	<input checked="" type="checkbox"/>

Рис. 1. Структура таблиці з бази BLOB-полями

Стовпець photo зарезервуємо для зберігання в базі даних фотографій, а стовпець notes – для текстових блоків, у тому числі й у RTF-форматі.

Для розробки програми-клієнта опрацювання даних зазначених типів створимо новий Windows Forms проект у середовищі MS Visual Studio та налаштуємо за допомогою майстра строго типізований об'єкт DataSet. У вікні Data Sources (рис. 2) стовпець Photo зв'яжемо з компонентом PictureBox, що призначений для відображення графічних зображень.

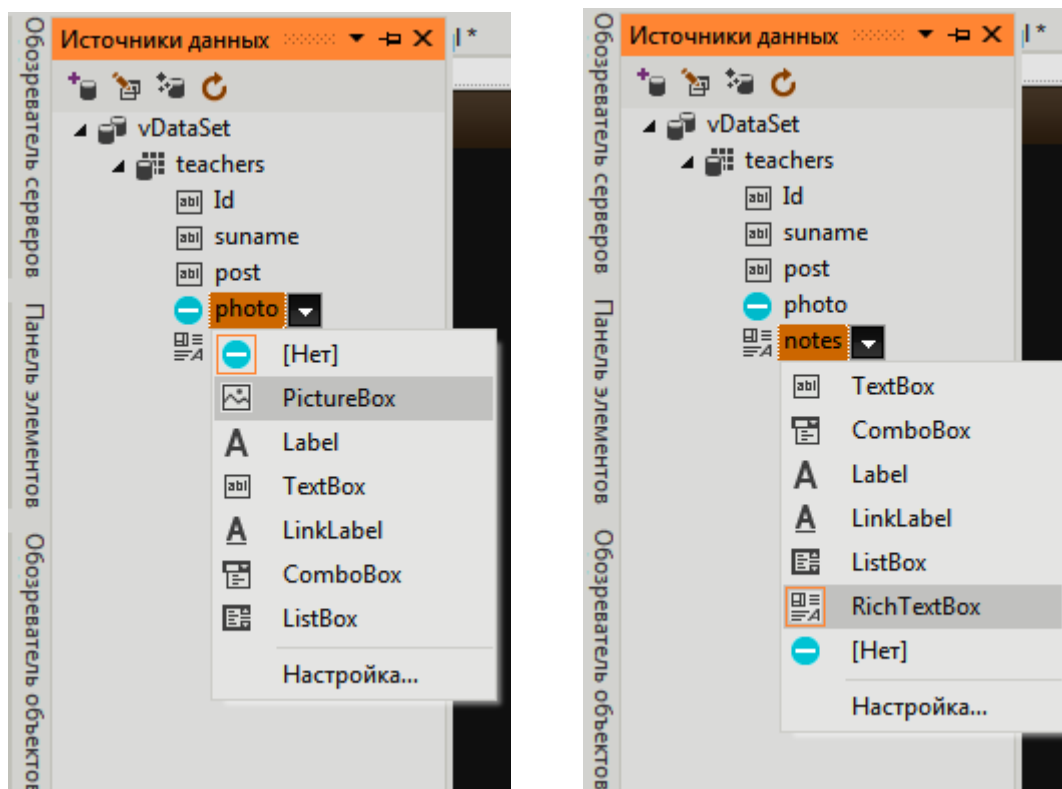


Рис. 2. Налаштування режиму відображення графічних зображень

Компонент PictureBox може відображати такі формати зображень: растрові (файли BMP, GIF, JPEG, TIFF та PNG), векторні (файли WMF або EMF), піктограми (файл ICO).

Тепер налаштуємо стовпець notes на відображення тексту. За замовчуванням з ним зв'язаний компонент TextBox. Після перетягування його на форму програми він буде відображати звичайний текст стовпця. Для відображення блоку тексту необхідно встановити для параметра

Multiline значення True. Для відображення форматowanego тексту необхідно стовпець зв'язати з компонентом RichTextBox. Для цього необхідно додати його до переліку зв'язаних компонентів, шляхом вибору відповідного пункту у вікні Options.

Після проведення відповідних налаштувань стовпців потрібно налаштувати форму вікна програми (рис. 3).

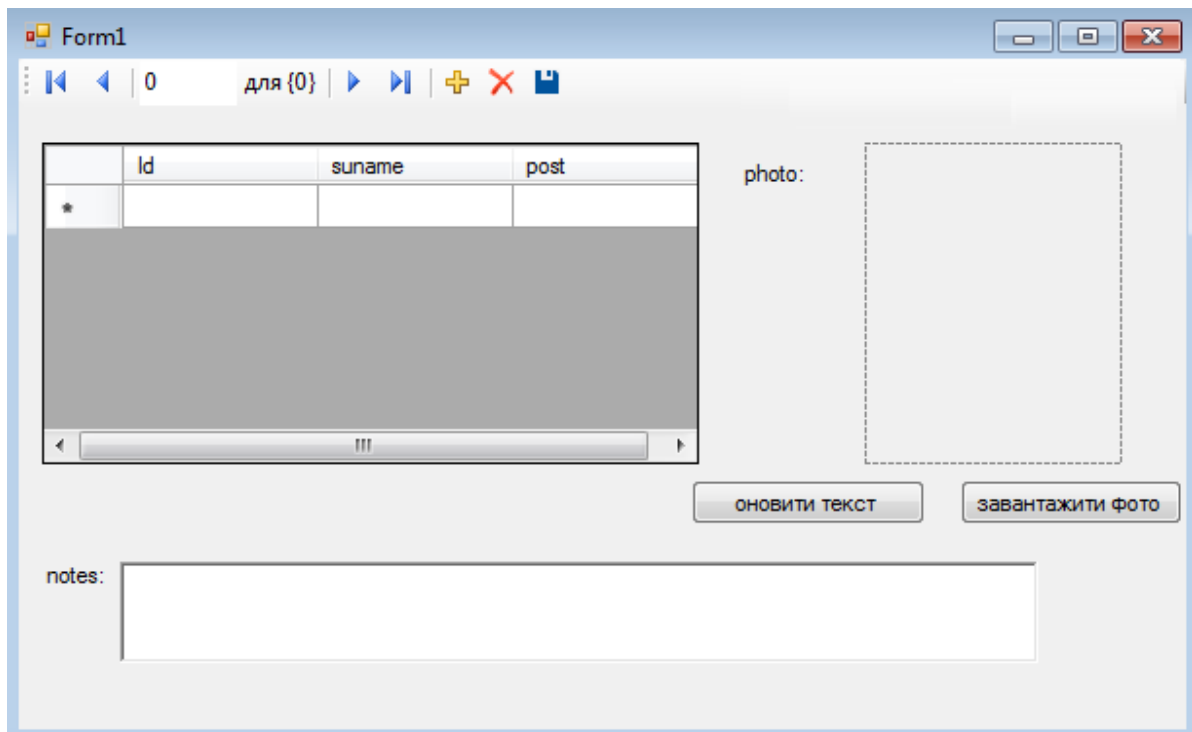


Рис. 3. Форматування форми програми

Перетягнемо з вікна Data Sources на форму програми таблицю teacher, стовпець з фотографіями photo та стовпець для форматowanego блоку тексту notes. Компонент DataGridView має можливість відображати також і графічні об'єкти. Однак доцільніше рисунки відображати в окремих вікнах. Тому вилучимо стовпці Photo та Notes з переліку відображуваних стовпців компонента DataGridView. Для завантаження у базу даних фотографій з файлів слід перетягнути на форму програми також діалоговий компонент OpenFileDialog.

Налаштуємо відповідним чином обрані компоненти. Для того, щоб рисунки зменшувалися до розміру компонента PictureBox, встановимо

для параметра SizeMode значення Zoom. Для завантаження фотографій додамо на форму програми кнопку (Button) та запрограмуємо функцію оброблення події натиснення кнопки таким чином:

```
// завантаження фотографії у базу даних
private: System::Void button1_Click
(System::Object^ sender, System::EventArgs^ e)
{
    if (open_dialog.ShowDialog() ==
        System.Windows.Forms.DialogResult.OK)
    { try
        { byte[] photo = GetPhoto(open_dialog.FileName);
            using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=test.mdf;Integ
rated Security=True;Connect Timeout=30"))
            {
                con.Open();
                using (SqlCommand command = new SqlCommand("UPDATE
dbo.teachers SET photo = @Photo WHERE Id = " +
this.bindingNavigatorPositionItem.Text, con))
                {command.Parameters.Add("@Photo", SqlDbType.Image,
photo.Length);
                    command.Parameters["@Photo"].Value = photo;
                    command.ExecuteNonQuery();
                }
                con.Close();}
            }
        catch(Exception n)
            { DialogResult result = MessageBox.Show(n.Message,
n.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        this.teachersTableAdapter.Fill(this.dBDataSet.teachers); }

//Функція GetPhoto:
public static byte[] GetPhoto(string filePath)
```

```

        {
            FileStream fs = new FileStream(filePath,
            FileMode.Open, FileAccess.Read);
            BinaryReader br = new BinaryReader(fs);
            byte[] photo = br.ReadBytes((int)fs.Length);
            br.Close();
            fs.Close();
            return photo;
        }

```

Налаштуємо компонент RichTextBox, зв'язаний зі стовпцем notes.

Для оновлення даних в полі notes додамо кнопку "оновити":

```

// оновлення тексту-примітки з TextRichTextBox
this.TextRichTextBox.DataBindings.Add((gcnew
System::Windows::
Forms::Binding(L"Rtf", this.PeopleBindingSource, L"Text",
true)));

```

Остаточний вигляд вікна програми наведений на рис. 4.

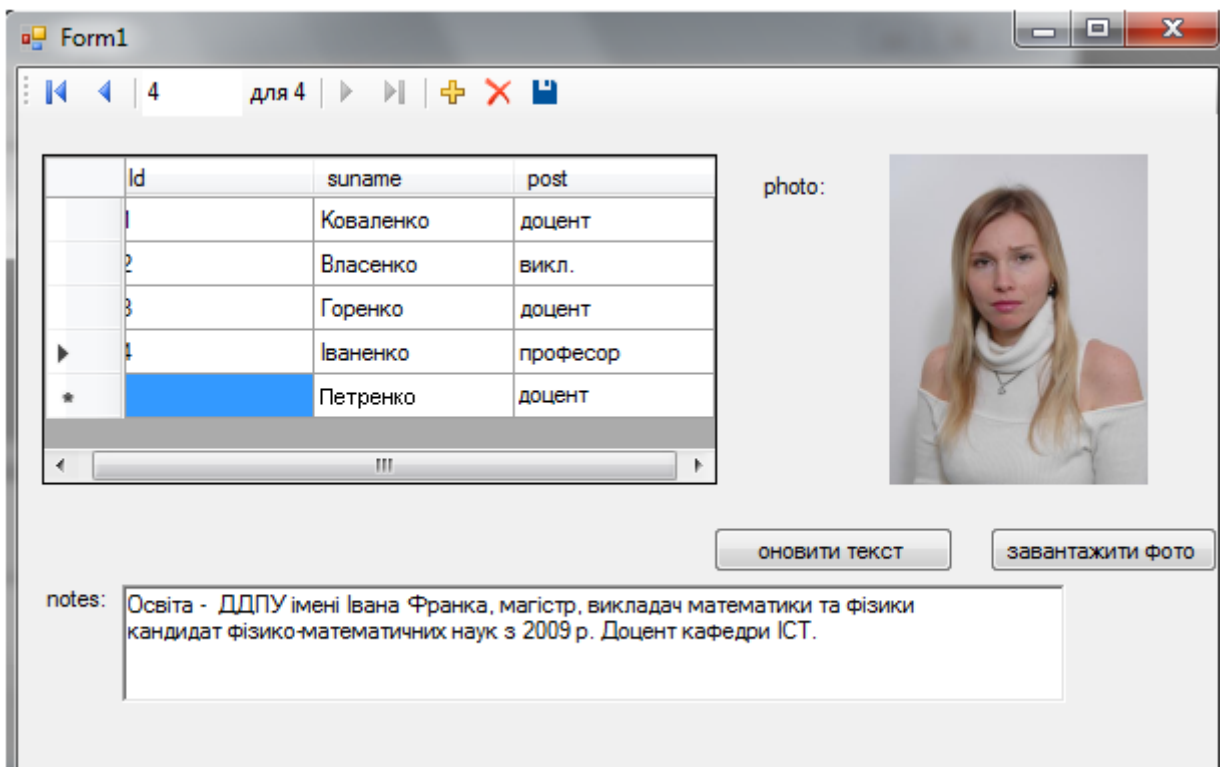


Рис. 4. Вигляд вікна програми опрацювання BLOB-даних

Завдання для самостійного виконання

1. Відповідно до Вашого варіанта, створіть базу даних у середовищі MS SQL Server з однієї таблиці, що має поля типу nvarchar(max) (для збереження текстового блоку) та image або varbinary(max) (для зображень).

2. Для створеної БД, згідно з описаним в теоретичних відомостях алгоритмом, розробіть клієнтську програму засобами C# (проект Windows Forms у середовищі MS Visual Studio), що забезпечує зручну роботу з даними різних типів.

3. Передбачте можливість для користувача завантажувати зображення та текстові масиви в компоненти PictureBox та RichTextBox під час роботи програми.

4. Для форматування тексту додайте до форми програми панель з кнопками, як у редакторі Microsoft Word, та запрограмуйте кожну з них на окрему функцію форматування тексту.

Контрольні запитання

1. Чи можна зберігати у БД зображення?
2. Що означає термін «BLOB-дані» ?
3. У якому форматі BLOB-дані можна записувати в базу даних?
4. Який тип даних можна використовувати для збереження зображення у БД, розробленій у MS SQL Server?
5. Як обрати для відображення даних поля БД компонент PictureBox?
6. Який тип даних можна використовувати для збереження у полі таблиці БД великих об'ємів тексту?
7. Які компоненти можна використати для відображення полів типу nvarchar(max), varchar(max) та text?

8. Який компонент дає змогу організувати вибір зображення користувачем під час роботи програми?
9. Як перетворити завантажене зображення у масив байтів?
10. Яке призначення функції FileStream?

Розробка багатовіконного додатка для роботи з базою даних

МЕТА: розроблення засобами MS Visual Studio додатку, що дає змогу користувачу редагувати довідкові таблиці БД в окремому вікні.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Зазвичай, бази даних складаються з декількох таблиць, що зв'язані між собою. Нехай маємо БД **library**, структура якої наведена на рис.1.

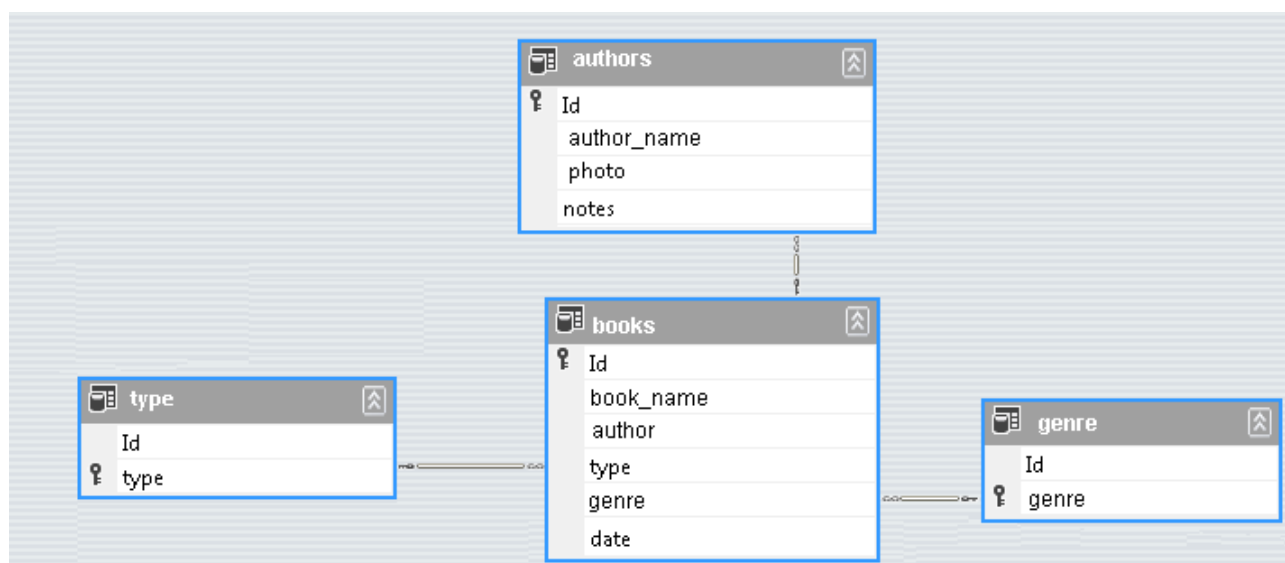


Рис. 1. Структура бази даних **library**

Розглянемо як приклад розроблення додатка для роботи з даними,

що містяться в таблицях БД. Нехай для таблиць **authors** та **genre** необхідно передбачити можливість редагування даних в окремому вікні. Для цього створимо меню для програми-клієнта, яка уможливить вибір відповідних таблиць та роботу з даними.

З метою організації меню додамо на основну форму програми компонент MenuStrip та заповнимо його назвами відповідних таблиць (рис. 2).

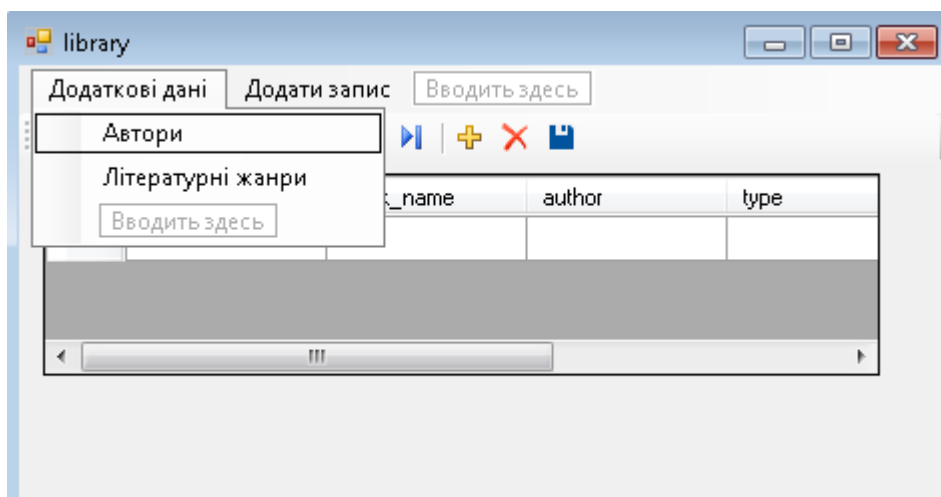


Рис. 2. Налаштування компонента MenuStrip

Далі слід додати до проекту нові Windows Forms вікна та розмістити у них зазначені таблиці.

Для створення нового Windows Forms вікна необхідно додати до проекту новий елемент. Для цього слід натиснути правою кнопкою миші на назві проекту у вікні Solution Explorer та в контекстному підменю обрати пункт New Item. Далі у діалоговому вікні Add New Item слід обрати елемент Windows Form, вказати для нього назву *Authors* та додати його до проекту.

У новостворене вікно слід перетягнути з Data Sources таблицю **authors** (рис. 3) та зберегти проект (в цьому випадку відображення даних у стовпці, що має містити фото, доцільно реалізувати за допомогою

компонента PictureBox.

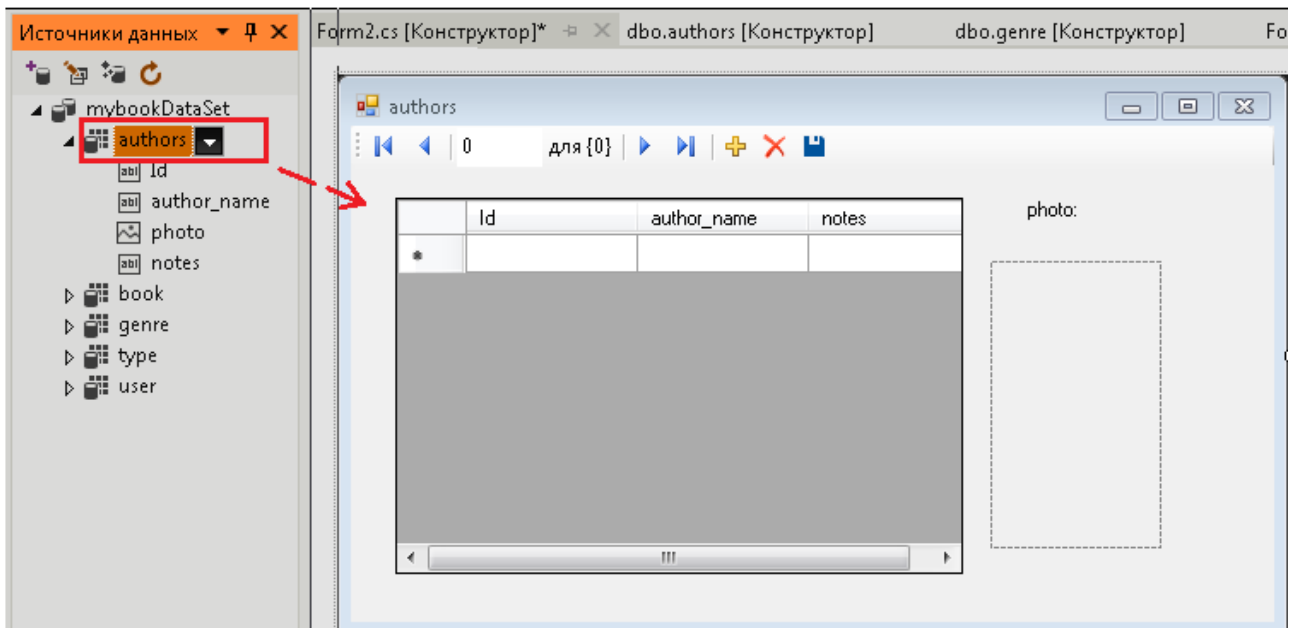


Рис. 3. Розміщення таблиці **authors** на новій формі

Після цього можна прив'язати вікно для редагування даних таблиці до відповідного пункту меню. Для цього слід двічі натиснути мишею на відповідному пункту меню та додати до функції оброблення відповідної події такий код:

```
private: System::Void groupsToolStripMenuItem_Click
(System::Object^ sender, System::EventArgs^ e)
{
    Authors authors = new Authors();
    authors.Show();
}
```

Для таблиці **genre** потрібно аналогічно додати до проекту нову форму та налаштувати відповідні елементи.

Оскільки нові вікна мають власні копії об'єкта DataSet, то, відповідно, після внесення змін до бази даних необхідно оновити дані на основній формі програми. Для цього потрібно розмістити компонент Button на основній формі програми та двічі клікнути вказівником мишки на ній. У

автоматично згенерованій функції оброблення події необхідно вписати код для оновлення даних.

```
private: System::Void button1_Click
(System::Object^ sender, System::EventArgs^ e)
{
    libraryDataSet.Books.Clear();
    libraryDataSet.Authors.Clear();
    libraryDataSet.Genre.Clear();
    Form1_Load(this, e);
}
```

При оновленні даних передовсім необхідно у визначеній послідовності очистити таблиці об'єкта DataSet, а потім знову заповнити. При заповненні даних повторно викликається функція оброблення події завантаження форми програми Form1_Load().

У кінцевому результаті багатовіконна програма-клієнт має виглядати як на рис. 4, де додаткові вікна викликаються з меню основного вікна.

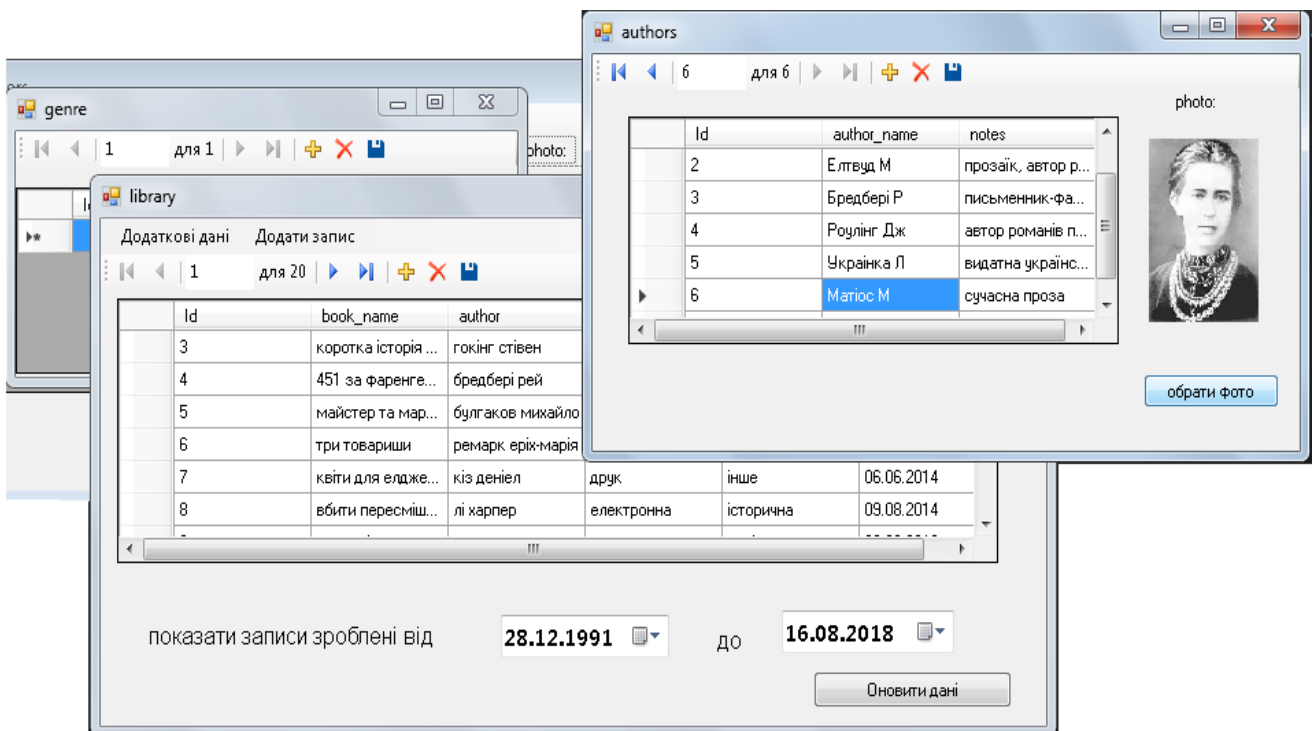


Рис. 4. Розроблена багатовіконна програма

Завдання для самостійного виконання

1. Для створеної в попередніх роботах багатотабличної реляційної БД в середовищі MS SQL Server згідно зі зразком, описаним у теоретичних відомостях, розробіть клієнтську програму засобами C#/C++ (проект Windows Forms у середовищі MS Visual Studio). Можна додати таблицю, що містить BLOB-дані.

2. Згідно з описаним алгоритмом додайте до проекту додаткові елементи Windows Form та налаштуйте їх з метою можливості обробки даних, що містяться в додаткових таблицях.

3. Розробіть програму-клієнт, що забезпечує зручну роботу з даними, які містяться в усіх таблицях БД (додаткові вікна викликаються з меню основного вікна).

4. Перевірте коректність роботи додатку.

Контрольні запитання

1. Що таке багатовіконна програма?
2. Як організувати обробку даних зі зв'язаних таблиць БД в різних вікнах?
3. Який пункт меню середовища MS Visual Studio дає змогу переглянути зв'язки між таблицями підключеної до проекту БД?
4. Чи мають нові вікна власні копії об'єкта DataSet?
5. Чому потрібно реалізувати оновлення об'єкта DataSet головної форми програми після роботи з даними додатковому вікні?
6. Який компонент можна використати для роботи з даними типу «дата-час»?

7. Як налаштувати компонент BindingNavigator для роботи з записами таблиці?
8. Яким пунктом середовища MS Visual Studio слід скористатись, щоб переглянути вміст підключеної до проекту БД?
9. Як забезпечити коректне оновлення даних в головному вікні програми?

Реєстрація та авторизація користувачів у клієнтській програмі, розподіл ролей

МЕТА: закріплення основ роботи з базами даних, вивчення механізмів реалізації реєстрації та авторизації користувачів у клієнтській програмі.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Реєстрація нових користувачів – внесення їхніх даних до наявної бази даних користувачів. Такий процес є актуальним в багатокористувацьких інформаційних системах, які потребують ідентифікації користувачів з певною метою (наприклад, навчальні системи різного типу – для фіксації досягнень користувача та інші). Перелік полів, які має заповнити користувач під час реєстрації визначається окремо, залежно від мети авторизації користувачів у системі та відповідно до призначення самої системи. Зазвичай, дані про користувачів зберігаються в окремій таблиці БД.

Для збереження даних користувачів можна застосувати будь-яку СКБД, налагодивши її зв'язок з додатком. Фрагмент коду, що відповідає за додавання відомостей про нових користувачів до відповідної таблиці з

даними, наведено нижче.

```
con.Open();
SqlCommand com = new SqlCommand();
com.Connection = con;
com.CommandText = "INSERT INTO Users ('Login',
'Password') VALUES ('" + textBox1.Text + "', '" +
textBox1.Text + "')";
com.ExecuteNonQuery();
    MessageBox.Show("Вітаємо,      "+ textBox1.Text + "
Реєстрація пройшла успішно!" ); }
else
    MessageBox.Show("Перевірте коректність заповнення полів.
Не всі дані введені");
}
```

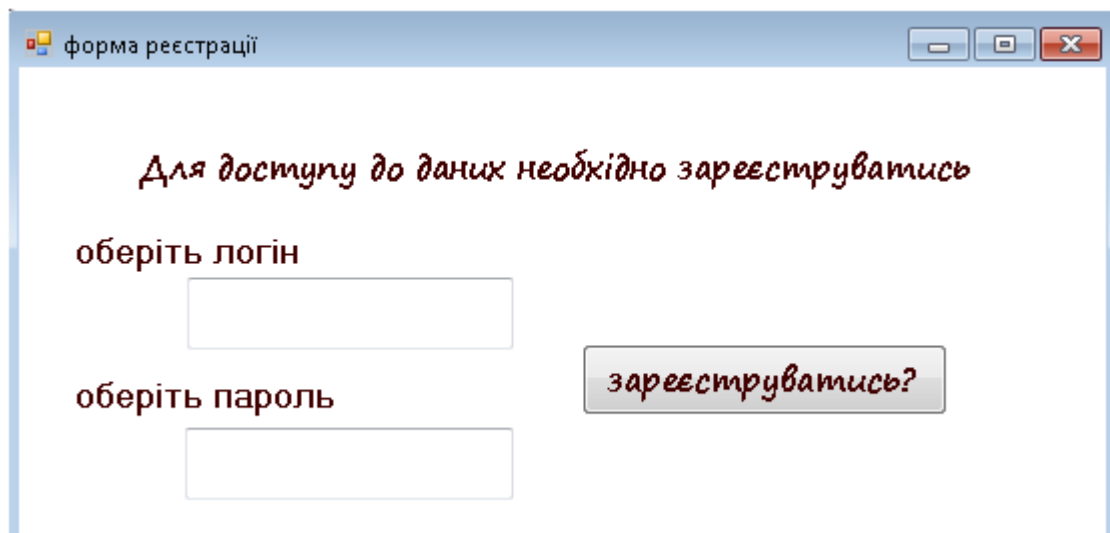


Рис. 1. Форма для реєстрації користувачів

У деяких випадках дані про користувачів заносить безпосередньо адміністратор системи чи користувач, що має такі повноваження. У цьому випадку користувачу додатка потрібно окремо повідомити дані, за якими він може авторизуватися в системі (зазвичай – логін та пароль).

Авторизація користувачів

Розглянемо детальніше реалізацію механізму авторизації зареєстрованих користувачів в клієнтській програмі. Нехай для авторизації в програмі користувач має ввести логін і пароль, а дані про користувачів зберігаються у таблиці users, що має таку структуру: (рис. 2)

	Имя	Тип данных	Допустимы значения NULL
PK	Id	int	■
	nik	nchar(10)	■
	password	nchar(10)	■
	level	int	■

Рис. 2. Таблица з даними користувачів додатка

Нехай маємо два типи користувачів: адміністратор (nik = "admin", password = "admin", level=1) та користувач (nik = "user", password = "user", level=2).

Змінимо попередній проект таким чином, щоб окрім реєстрації нових користувачів було передбачено надання доступу до головної форми проекту лише зареєстрованим користувачам. На формі, що призначена для авторизації користувачів і забезпечення різнорівневого доступу до даних (ідентифікації користувачів), розміщуємо такі компоненти (рис. 3):

форма авторизації

для доступу до системи введіть логін та пароль

логін

пароль

вхід

зареєструватись?

Рис. 3. Форма авторизації в проекті з відповідними компонентами

На головній формі проекту, доступ до якої отримує користувач після успішної авторизації, розташовуємо кнопку "вихід" та напис, що буде містити інформацію про рівень доступу користувача (рис. 4.).

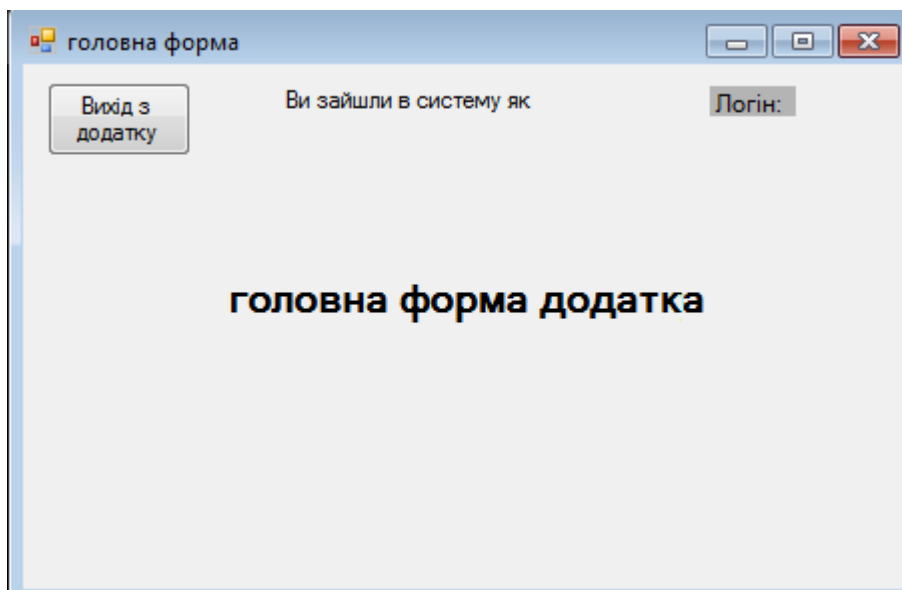


Рис. 4. Головна форма проекту з відповідними компонентами

Реалізуємо з'єднання з таблицею, що містить дані користувачів, перевірку введених користувачем даних і доступ до головної форми проекту у випадку успішної авторизації. Під'єднання до БД, що містить таблиці з даними зареєстрованих користувачів, можна реалізувати при завантаженні форми або при натисканні на кнопку "вихід".

Розглянемо код для кнопки "вихід".

```
private void button1_Click(object sender, EventArgs e)
{
    If (textBox1.Text != "" && textBox2.Text != "")
    {
        string CommandText = "SELECT * FROM [dbo].[users] WHERE nik
        =' " + textBox1.Text + "' AND password = ' " + textBox2.Text +
        " '";

        sqlCommand1.CommandText = CommandText;
        sqlConnection2.Open();
        SqlDataReader rd = sqlCommand1.ExecuteReader();
```

```

string check_login = "";
string check_password = "";
int i = 0;
while (rd.Read())
    {i++;
        check_login = rd.GetString(1);
        check_password = rd.GetString(2);}
//якщо запис існує в базі то викликаємо дочірню форму
if (i == 1)
    { this.Hide();
        Form2 form2 = new Form2();
        form2.Show();    }
else
    {MessageBox.Show("Невірний логін або пароль!");
        textBox1.Clear();
        textBox2.Clear();
        }
    sqlConnection2.Close();}
else
    {MessageBox.Show("Заповніть всі поля!"); }
    }

```

Для забезпечення різнорівневого доступу до даних у головній формі проекту можна додати перевірку значення поля `level`.

```

if (p==1) {label2.Text = label2.Text + " адміністратор";}
else if (p==2) {label2.Text = label2.Text + " користувач";}

```

Завдання для самостійного виконання

1. Розглянути наведені в теорії засоби організації механізмів реєстрації та авторизації користувачів у додатку.

2. Створити базу даних для атрибутів авторизації засобами будь-якої реляційної СКБД.

3. Засобами Visual Studio (мова C#/C++) розробити проект, що передбачає можливість реєстрації користувачів і збереження їхніх даних в базі даних користувачів.

4. Реалізувати процес авторизації користувачів та надання різноманітного доступу до даних.

5. Занотувати у звіт результати реєстрації користувача та наступної його авторизації в проекті (протестувати всі випадки, можливі під час авторизації користувачів, відобразити їх у звіті).

Контрольні запитання

1. Наведіть приклади додатків, в яких потрібно реалізувати лише авторизацію користувачів.
2. Наведіть приклади додатків, в яких потрібно реалізувати реєстрацію користувачів з подальшою авторизацією.
3. Як можна реалізувати збереження даних користувача, отриманих під час реєстрації?
4. Що передбачає авторизація користувача в додатку?
5. Як реалізувати процес авторизації користувачів?
6. Що таке рівень доступу до даних?
7. Наведіть приклади додатків, що передбачають різні права для різних типів користувачів.
8. На що варто звернути увагу при організації різноманітного доступу до даних?
9. Як реалізувати наявність різних прав доступу до даних для двох типів користувачів в додатку?

10. Яку СКБД Ви обрали для роботи з базою даних користувачів додатка? Чому?

Опрацювання збережуваних процедур засобами MS Visual Studio

МЕТА: розроблення засобами MS Visual Studio клієнтської програми, що передбачає виклик збереженої процедури з запитом до БД та перевірку введених даних згідно з деяким критерієм.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Збережена процедура – це поіменована збережена сукупність операторів Transact-SQL, яка зберігається на сервері і може приймати та повертати параметри. Такі процедури діють на рівні конкретної БД та призначені для багатократного використання.

При роботі з клієнт-серверним СУБД зазвичай постає необхідність виклику збереженої процедури з клієнтської програми. Розглянемо особливості опрацювання збережуваних процедур MS SQL Server засобами Visual Studio

Видалення записів з БД

Нехай у БД MS SQL Server створено збережену процедуру, що при кожному звертанні до неї видаляє певний запис з таблиці Books БД згідно з деяким критерієм. Наприклад, процедура `delete_from_table` під час кожного виклику видаляє запис з ID, що вкаже користувач:

```
CREATE PROCEDURE delete_from_table
    @idd int
as
```



```
begin
  delete from Books
  where ID = @idd
end
GO
```

Розробимо для БД клієнтську програму, що надає користувачу можливість видалення записів шляхом виклику вказаної збереженої процедури.

Створимо новий проект у Visual Studio. Для підключення необхідної БД MS SQL Server до проекту можна скористатись командою Add New Data Source пункту меню Data. У діалоговому вікні Data Sources Configuration Wizard (рис. 1), окрім пункту Tables, обов'язково слід відмітити галочкою пункт Stored Procedures та натиснути на кнопку готово. Створена БД MS SQL Server буде підключена до проекту разом з необхідними Вам збережуваними процедурами.

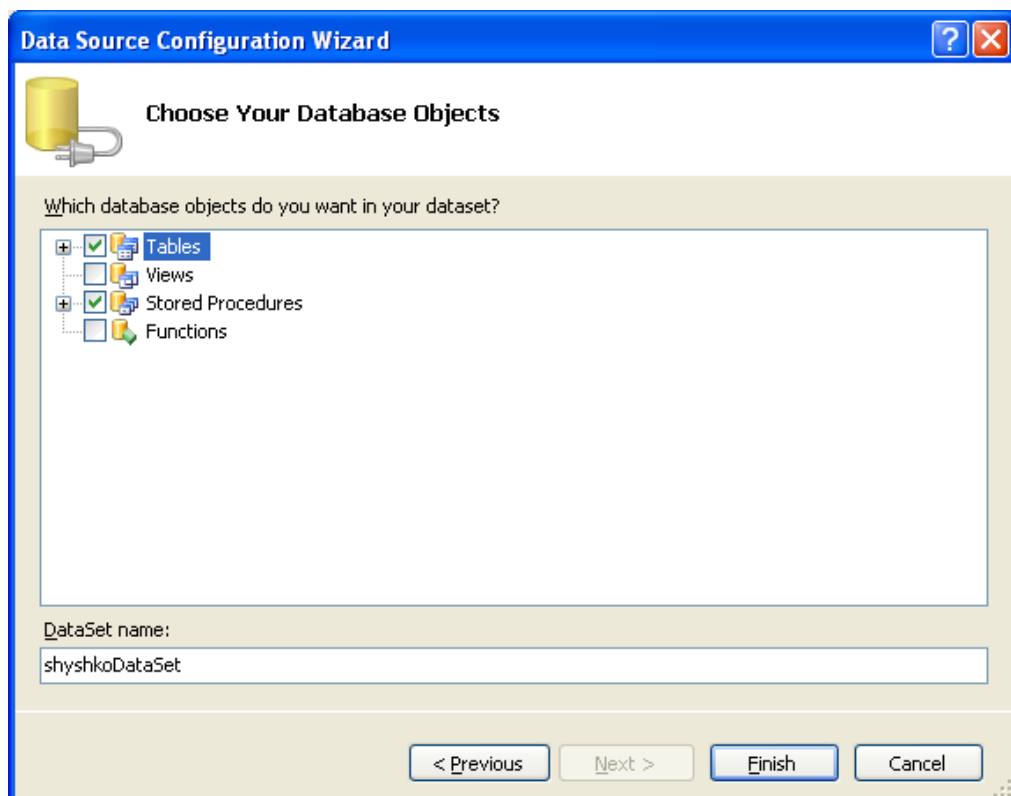


Рис. 1. Вибір вмісту бази даних, що підключається до проекту

Для того, щоб переглянути всі підключені до проекту збережувані процедури, їхній вміст й запустити, за потреби, процедуру на виконання, можна скористатись вкладкою Server Explorer (рис. 2).

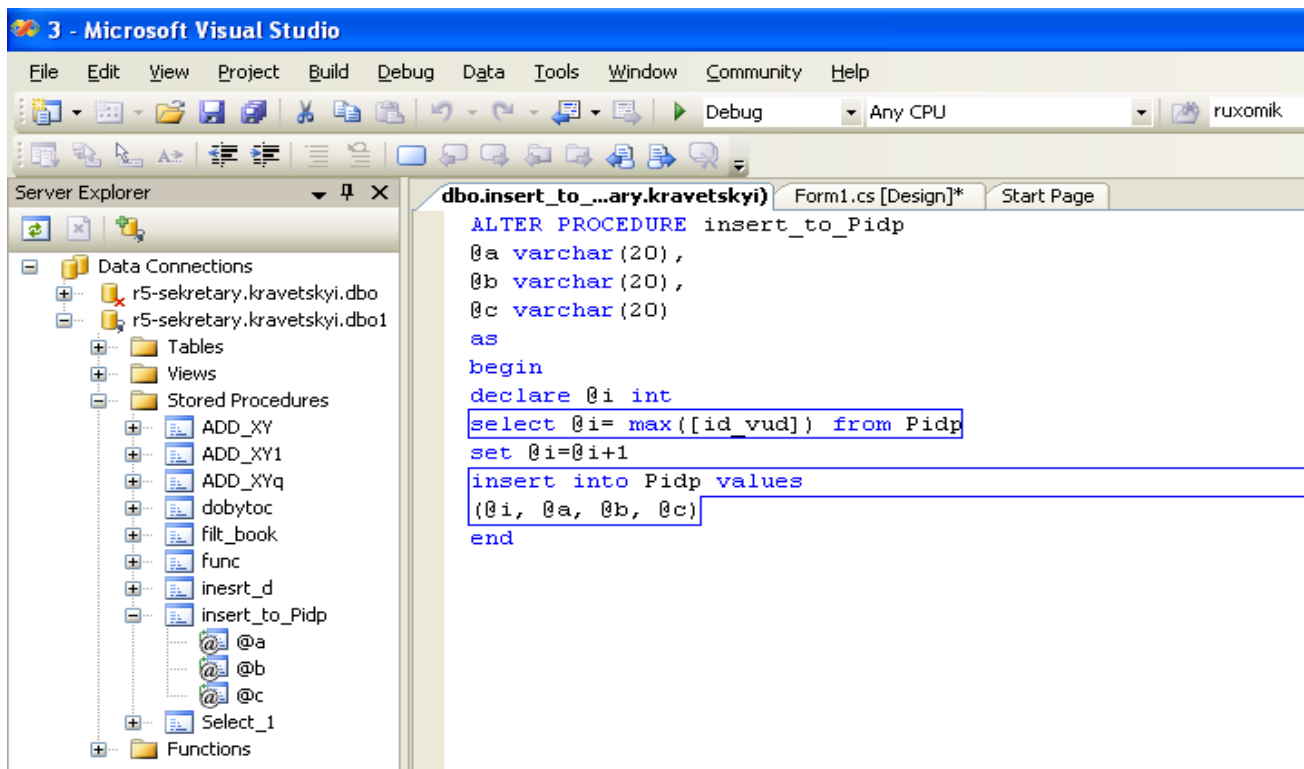


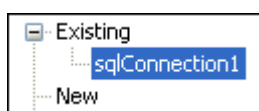
Рис. 2. Перегляд збережуваних процедур, підключених до проекту

Для того, щоб запустити процедуру під час виконання програми, можна скористатись об'єктами SqlConnection та SqlCommand (якщо цих компонентів немає на вкладці Toolbox, потрібно додати їх за допомогою вкладки Choose Toolbox Items).

Спочатку на форму треба додати об'єкт SqlConnection.

У властивості ConnectionString вказуємо шлях до бази даних. Далі додаємо до проекту об'єкт SqlCommand. Властивості цього компонента слід змінити так:

- властивість Connection – обираємо sqlConnection1



- властивість CommandType – обираємо тип

StoredProcedure

Text
StoredProcedure
TableDirect

- властивість CommandText – обираємо необхідну збережену

процедуру з переліку всіх підключених процедур

(None)
add_into_avtors
delete_from_books
filt_po_vud

За допомогою вхідних параметрів можна передати необхідну інформацію у процедуру із місця її виклику. Це місце виклику, як правило, є в клієнтській програмі.

Якщо потрібна нам процедура містить вхідні параметри, потрібно їх описати, скориставшись властивістю Params, пункт SqlParameter Collection Editor (рис. 3).

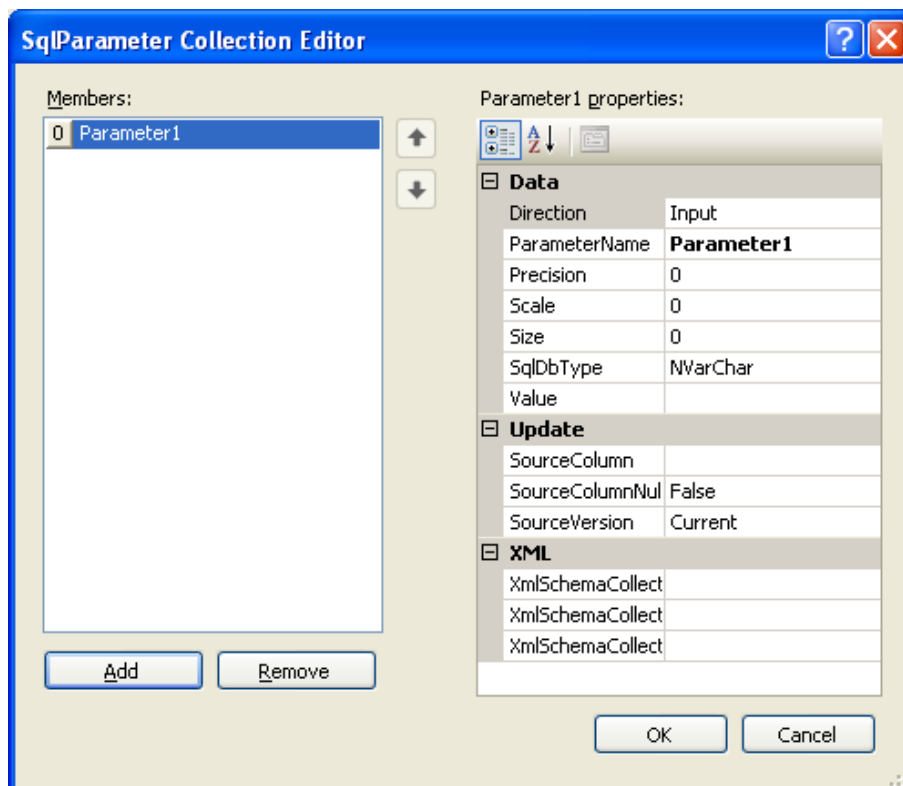


Рис. 3. Діалогове вікно для опису параметрів процедури.

У полі Direction слід обрати тип параметра (для вхідного Input, для вихідного Output). У полі ParametrName вводимо назву параметра (для описаної раніше процедури delete_from_table це параметр @idd).

Зверніть увагу, що назва параметра, вказана у властивості Parametr об'єкта SqlCommand, має збігатися з назвою параметра в процедурі!

У поле Precision вводимо кількість знаків, що мають відобразитись після коми. У поле Size вводимо розмір змінної; в полі SqlDbTypeType вказуємо тип параметра (має збігатися з типом відповідного параметра, що вказаний при створенні збереженої процедури); в поле властивостей Value – початкове значення параметра (рис. 4).

Для відображення інформації, що міститься в таблиці БД, додаємо на форму компонент dataGridView, у властивості Table обираємо таблицю, робота з якою передбачає в нашому випадку виклик збереженої процедури. Для того, щоб користувач міг вводити значення вхідного параметра (в нашому випадку ID відповідного запису, що має бути видалений після виклику процедури), додаємо на форму проекту поле textBox1.

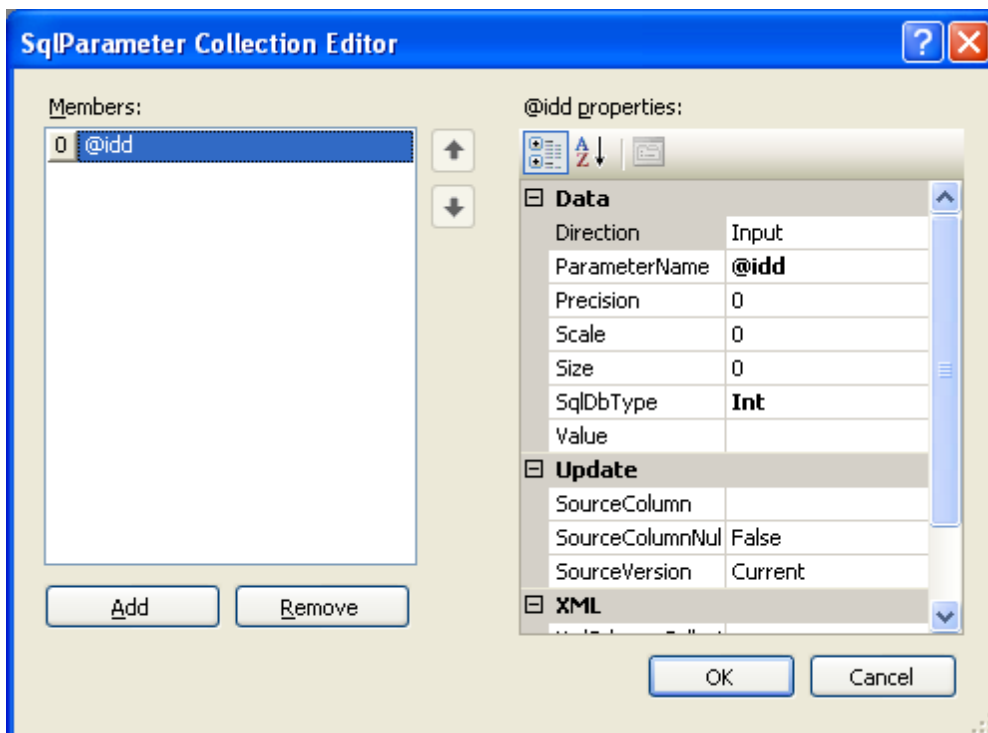


Рис. 4. Налаштування параметрів об'єкту SqlCommand.

Для виклику процедури додаємо на форму кнопку. В методі Click кнопки прописуємо такий код:

```

this.sqlConnection1.Open();
int ID = System.Convert.ToInt32(textBox1.Text);
this.sqlCommand1.Parameters["@idd"].Value = ID;
this.sqlCommand1.ExecuteReader();
this.sqlConnection1.Close();
this.booksTableAdapter.Fill(this.DataSet.Books);
this.textBox1.Text = "";

```

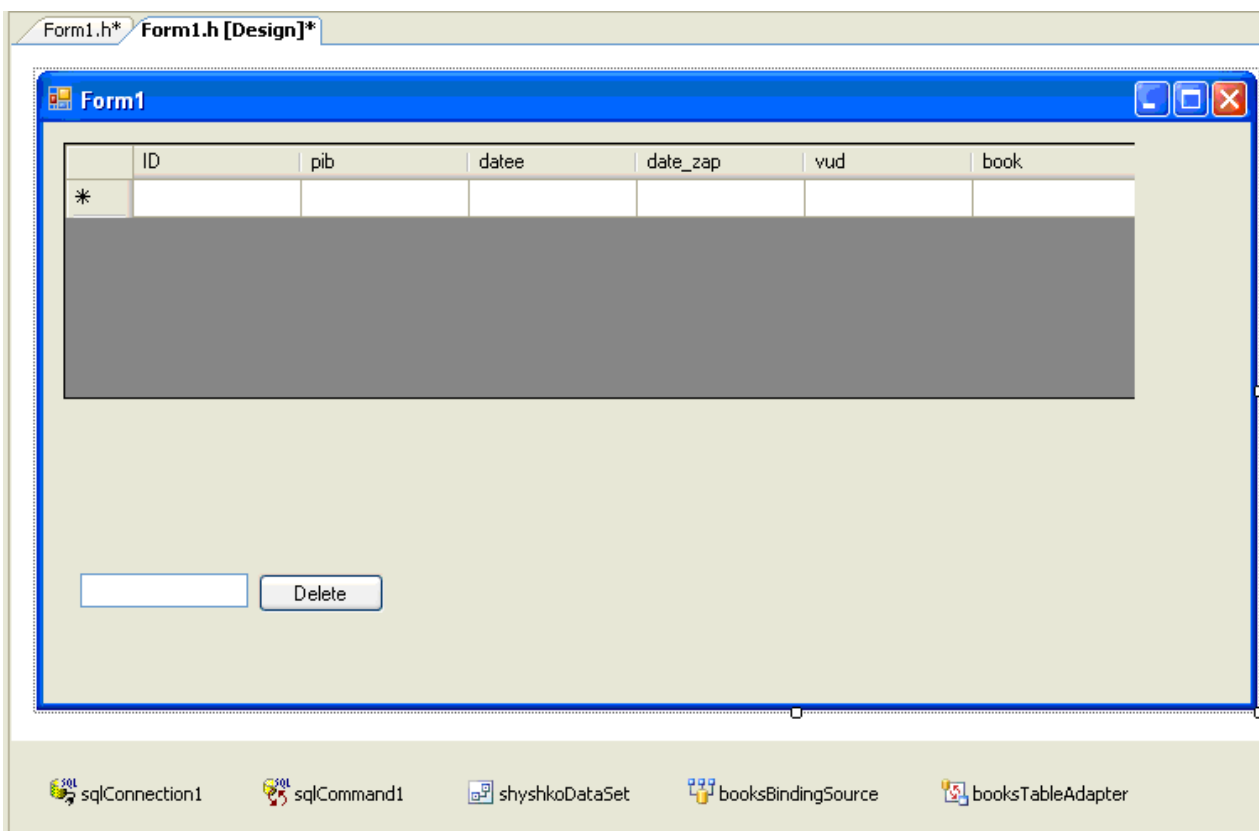


Рис. 5. Форма проекту з відповідними компонентами

Для роботи проекту потрібно підключити бібліотеку:

```
using System.Data.SqlClient;
```

Запускаємо проект на виконання та перевіряємо, чи коректно викликається та виконується необхідна збережена процедура.

Додавання записів до БД

Нехай у БД MS SQL Server створено збережувану процедуру insert_from_table, що при кожному звертанні до неї додає запис до нашої БД, причому вміст запису формує користувач:

```

CREATE PROCEDURE insert_from_table
    @a varchar(20),
    @b int(4),
    @c varchar(20)
as
begin
    declare @i int
    select @i = max(ID) from Pidp
    set @i =@i+1
    insert into Pidp values (@i, @a, @b, @c)
end
GO

```

Розробимо для БД клієнтську програму, що надає користувачу можливість додавання записів шляхом виклику вказаної збереженої процедури.

Для підключення БД разом з описаною збереженою процедурою скористаємось алгоритмом, описаним у попередньому прикладі. Далі слід додати до проекту об'єкти `SqlConnection` та `SqlCommand`, налаштувати відповідні властивості (рис.6).

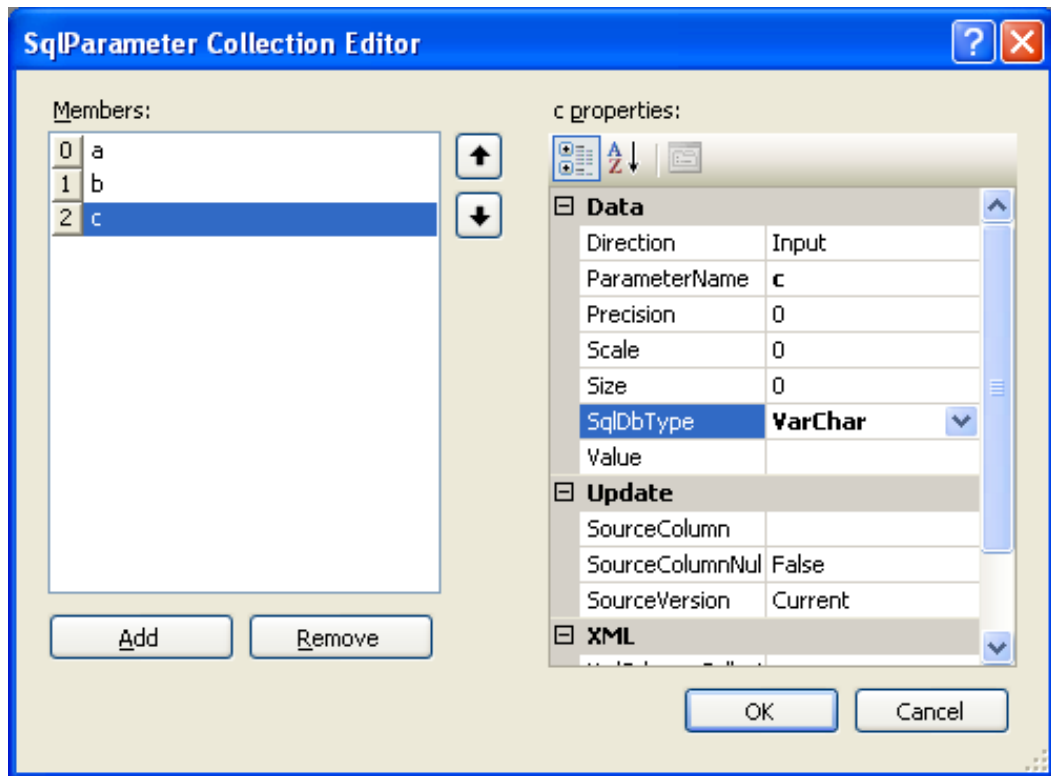


Рис. 6. Налаштування параметрів для процедури insert_from_table

Для відображення інформації, що міститься в таблиці БД, додаємо на форму компонент dataGridView. Для того, щоб користувач міг вводити значення вхідних параметрів (в нашому випадку – значення трьох полів запису, що додається до БД після виклику процедури), додаємо на форму проекту три компоненти textBox та кнопку для виклику процедури.

У методі Click кнопки прописуємо код:

```
private void button1_Click(object sender, EventArgs e)
{
    String ^nazva = (this.textBox1.Text);
    int kilk = System.Convert.ToInt32(this.textBox2.Text);
    String ^profesia = (this.textBox3.Text);
    this.sqlCommand1.Parameters["a"].Value = nazva;
    this.sqlCommand1.Parameters["b"].Value = kilk;
    this.sqlCommand1.Parameters["c"].Value = profesia;
    this.sqlConnection1.Open();
    this.sqlCommand1.ExecuteReader();
    this.sqlConnection1.Close();
}
```

```

this.textBox1.Text = "";
this.textBox2.Text = "";
this.textBox3.Text = "";
this.pidpTableAdapter.Fill(this.DataSet.Pidp);
}

```

На рис. 7 наведена форма проекту з відповідними компонентами до виклику (рис. 7а) та після виклику збереженої процедури (рис. 7б) відповідно.

The screenshot shows a Windows application window with a data table and three input fields. The table has columns for id_vud, nazva_pid, kilkist_prac, and profesia. The input fields are labeled 'назва підприємства', 'кількість працівників', and 'професія'. The 'додати' button is visible below the input fields.

	id_vud	nazva_pid	kilkist_prac	profesia
	25	Color	23	менеджер
	28	Авторемонт	12	автослюсар
*				

назва підприємства: Ватра кількість працівників: 20 професія: продавець

додати

Рис. 7а. Форма проекту з відповідними компонентами

The screenshot shows the same Windows application window after a save operation. The data table now includes a new row with id_vud 30, nazva_pid 'Ватра', kilkist_prac 20, and profesia 'продавець'. The input fields are now empty, and the 'додати' button is highlighted with a dashed border.

	id_vud	nazva_pid	kilkist_prac	profesia
	26	Color	23	менеджер
	28	Авторемонт	12	автослюсар
	30	Ватра	20	продавець
*				

назва підприємства: кількість працівників: професія:

додати

Рис. 7б. Форма проекту після виклику збереженої процедури

Збережувані процедури з запитом та перевіркою

Під час роботи з БД (додавання та вилучення записів, оновлення інформації, запити на вибірку певних даних тощо) часто виникає потреба перед виконанням операцій провести їх перевірку на відповідність деякого критерію. Так, наприклад, база даних студентів або співробітників не може містити однакові значення полів «телефон» чи «номер студентського квитка». Розглянемо особливості роботи із збереженими процедурами, що передбачають попередню перевірку даних.

Нехай у БД MS SQL Server створено збережувану процедуру `insert_to_books`, за допомогою якої можна додавати до таблиці лише ті записи, в яких значення поля `book` є автентичним (таблиця не має містити користувачів з однаковим значенням вказаного поля). Тобто процедура накладає обмеження на записи, що додає користувач. В зазначеному випадку процедура може мати такий вигляд:

```
CREATE PROCEDURE insert_to_books
    @name varchar(50),
    @date datetime(10),
    @date_vst datetime(10),
    @typ varchar(50),
    @book varchar(50),
    @date_vud datetime(10),
    @ret bit out
AS
begin
declare @book_insert int
select @book_insert= [ID] from books where @book = book
if @book_insert <> NULL
    set @ret=0
else
    begin
```

```

declare @i int
select @i= max([ID]) from books
set @i=@i+1
insert into books values
    (@i, @name, @date, @date_vst,@typ,@book,@date_vud)
set @ret=1
end
return
end

```

Ця процедура містить вихідний параметр `@ret`, який передає інформацію про результат перевірки запису, що додається в таблицю, на відповідність зазначеній умові. Для виклику цієї процедури з клієнтської програми слід додати до проекту об'єкти `SqlConnection` та `SqlCommand`, налаштувати відповідні властивості, як зазначено. Для вихідного параметра `@ret` цієї роцедури потрібно встановити відповідне значення властивості `Direction` (рис. 8).

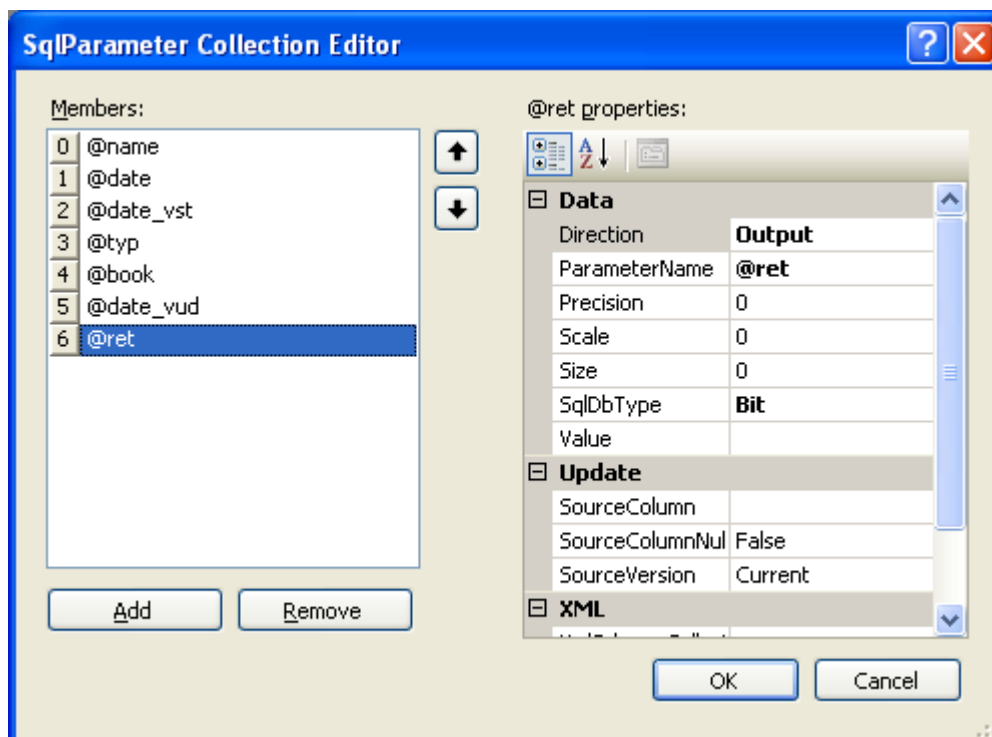


Рис. 8. Налаштування вихідного параметра для процедури `insert_to_books`

Для того, щоб користувач міг вводити значення вхідних параметрів (значення шести полів запису, що додається до БД після виклику процедури), додаємо на форму проекту компоненти `textBox` та кнопку для виклику процедури. Код для методу `Click` кнопки виклику процедури буде загалом аналогічним описаному в попередньому прикладі, однак потрібно передбачити випадок, коли запис не може бути додано до таблиці та вивести користувачу відповідне інформаційне повідомлення.

```
int tmp =  
System.Convert.ToInt32(sqlCommand1.Parameters["@ret"].Value);  
if (tmp == 0)  
    {MessageBox.Show("Така книга вже видана!"); }  
else  
    {MessageBox.Show("Запис додано до бази даних!"); }
```

На рис. 9 наведена форма проекту з відповідними компонентами.

ID	pib	datee	date_zap	vud	book	date_book
*						

ПІБ :

Дата народж. :

Дата запису :

Вид :

Книжка :

Дата видачі :

Рис. 9. Форма проекту з відповідними компонентами

Завдання для самостійного виконання

1. Розробити в середовищі MS SQL Server просту збережену процедуру, що видаляє записи з таблиці БД, створеної в попередніх

роботах, згідно з введеним користувачем значенням деякого поля. Запустити процедуру на виконання. Занотувати у звіт результат її виконання.

2. Розробити в середовищі MS SQL Server просту збережену процедуру, що додає записи до Вашої БД (вміст запису формує користувач за допомогою вхідних параметрів). Запустити процедуру на виконання. Занотувати у звіт результат її виконання.

3. Засобами Visual Studio (мова C++/ C#) розробити клієнтську програму для створеної в попередніх роботах БД MS SQL Server.

4. Використовуючи наведені відомості, передбачити виклик створених у середовищі MS SQL Server зберезуваних процедур. Запустити проект на виконання та перевірити коректність виконання процедур. Занотувати у звіт результат їхнього виконання.

5. Розробити в середовищі MS SQL Server збережену процедуру з розгалуженням, що містить запити до вашої БД та передбачає перевірку значення полів запису за деяким критерієм (процедура має вхідні і вихідні параметри). Запустити процедуру на виконання. Занотувати у звіт результат її виконання за різних значень вхідних параметрів.

6. Засобами MS Visual Studio (мова C++/ C#) розробити клієнтську програму для створеної в попередніх роботах БД MS SQL Server.

7. Використовуючи наведені відомості, передбачити виклик створеної в середовищі MS SQL Server зберезуваної процедури, що передбачає перевірку значення полів запису за деяким критерієм. Запустити проект на виконання та перевірити коректність виконання процедур. Занотувати у звіт результат їхнього виконання за різних значень вхідних параметрів.

Контрольні запитання

1. Що таке збережувана процедура? Як і з якою метою вона створюється?
2. Яку структуру мають збережувані процедури, розроблені засобами Transact-SQL?
3. Яким чином підключити БД разом зі збережуваними процедурами, що містяться в ній?
4. Як можна переглянути всі підключені до проекту збережувані процедури та їхній вміст?
5. Яке призначення об'єкта SqlCommand?
6. Які значення може приймати властивість CommandType?
7. Які властивості об'єкта SqlCommand слід налаштувати для роботи зі збережуваною процедурою?
8. Для чого призначений метод ExecuteReader об'єкта SqlCommand?
9. Яке значення властивості Direction об'єкта SqlCommand потрібно встановити для вихідного параметра збережуваної процедури?
10. Скільки об'єктів SqlCommand потрібно використати для роботи з трьома збережуваними процедурами?

Опрацювання баз даних SQLite. Обробка та пошук даних

МЕТА: вивчення основ обробки даних, що зберігаються в базі даних SQLite, засобами MS Visual Studio

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

З точки зору користувача, SQLite – це одиночний файл на диску. Цей файл не вимагає для свого використання запуску сторонніх процесів, наявності СУБД і додаткових засобів адміністрування.

Протокол обміну забезпечують виклики функцій (API) бібліотек SQLite, що не вимагає від програміста додаткових знань структури БД та особливостей організації взаємодії.

SQLite можна назвати вбудованою реляційною базою даних, оскільки вона (як MySQL і PostgreSQL) реалізує основні можливості стандарту SQL (включаючи тригери і транзакції) та має досить велику швидкість обробки даних.

SQLite забезпечує підтримку обсягу даних до 2-х терабайт, а розмір запису (при використанні полів BLOB) обмежений тільки пам'яттю комп'ютера. Підтримується невелика, але цілком достатня для більшості завдань, кількість типів даних.

SQLite варто використовувати для прикладних програм з невеликими базами даних, додатків, що не вимагають адміністрування баз даних, для створення тимчасової бази даних в процесі роботи додатків, в ролі бази даних доступу та паролів на невеликих Web-порталах тощо.

Зручним є процес розгортання і перенесення бази даних з одного комп'ютера на інший. Сама база даних може зберігатись разом з додатком і копіюватися іншими користувачами в аналогічні програми.

Створення бази даних SQLite

Для створення бази даних можна скористатись різними менеджерами SQLite. Розглянемо роботу з базами даних з використанням додатка SQLite Maestro (додаток не потребує встановлення; достатньо розпакувати архів та запустити його). Для створення бази даних обираємо пункт «Створити нову базу даних». У

діалоговому вікні, що відкриється (рис. 1), необхідно ввести ім'я бази даних з розширенням .db.

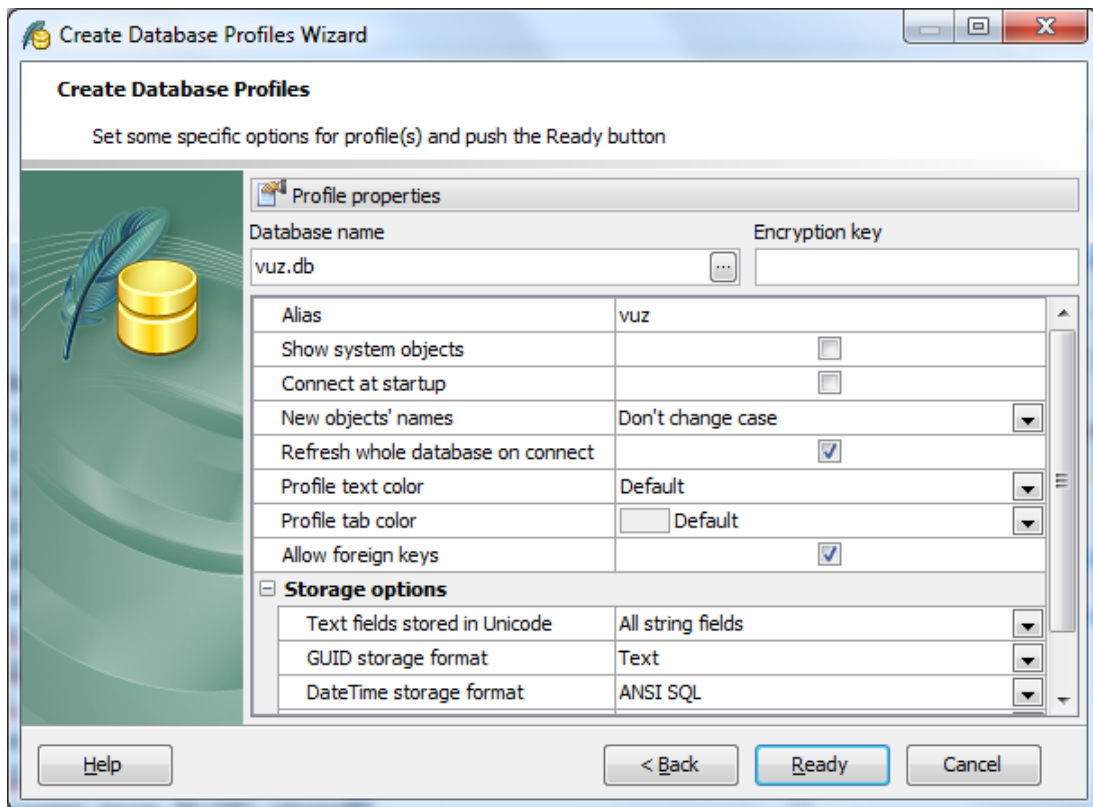


Рис. 1. Створення нової бази даних SQLite

Після цього можна безпосередньо працювати з даними у зручному візуальному режимі. Для створення нової таблиці потрібно скористатись командою Create New Table контекстного меню пункту Table (рис. 2).

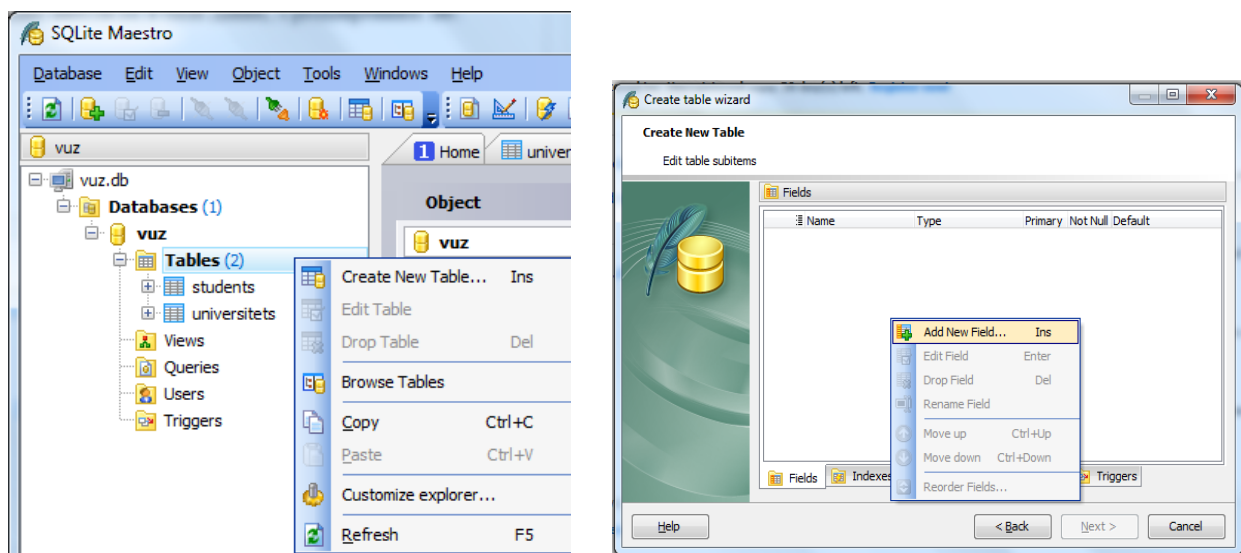


Рис. 2. Робота з таблицями в SQLite Maestro

Командою Add new field можна додавати нові поля до створеної таблиці. У діалоговому вікні, що відкривається, можна задати назву та обрати тип поля, встановити параметри поля, значення за замовчуванням та інше (рис. 3).

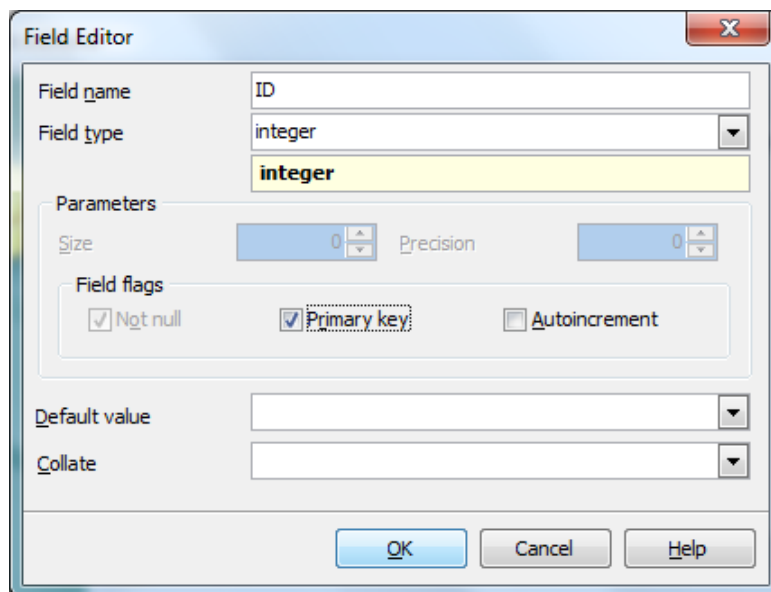


Рис. 3. Налаштування полів таблиці у вікні Field Editor

Після налаштування всіх полів потрібно підтвердити створення бази даних. Система автоматично згенерує відповідний SQL код (рис. 4).

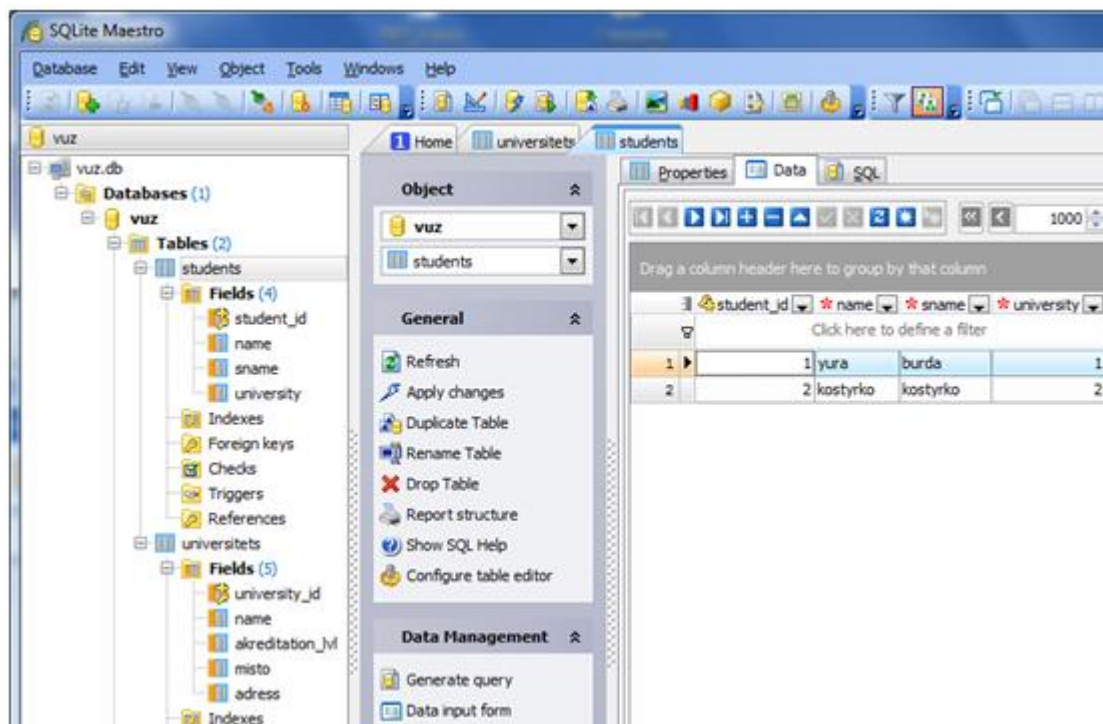


Рис. 4. Створені таблиці

Робота з даними з БД SQLite в середовищі Visual Studio

Розглянемо як приклад особливості розроблення програми-клієнта у середовищі MS Visual Studio, що демонструє основи роботи з даними, які містяться у таблицях БД SQLite.

Створимо новий Windows Forms проект у середовищі MS Visual Studio. В папку ...\Назва\Назва\bin\Debug додаємо бібліотеку System.Data.SQLite (знаходиться на сервері в папці SQLite або скачати з офіційного сайту). Далі потрібно додати посилання (Add References). У наступному діалоговому вікні обираємо файл System.Data.SQLite.dll).

Після цього слід підключити до проекту бібліотеку:

```
using System.Data.SQLite;
```

Нехай маємо БД «vuz.db» з двох таблиць **students** та **universitets**, структуру яких наведено на рис. 4.

Розробимо клієнтський додаток для роботи з даними. Для цього створимо новий Windows Forms проект в MS VisualStudio. Для опрацювання даних розміщуємо компоненти на формі так, як показано на рис. 5.

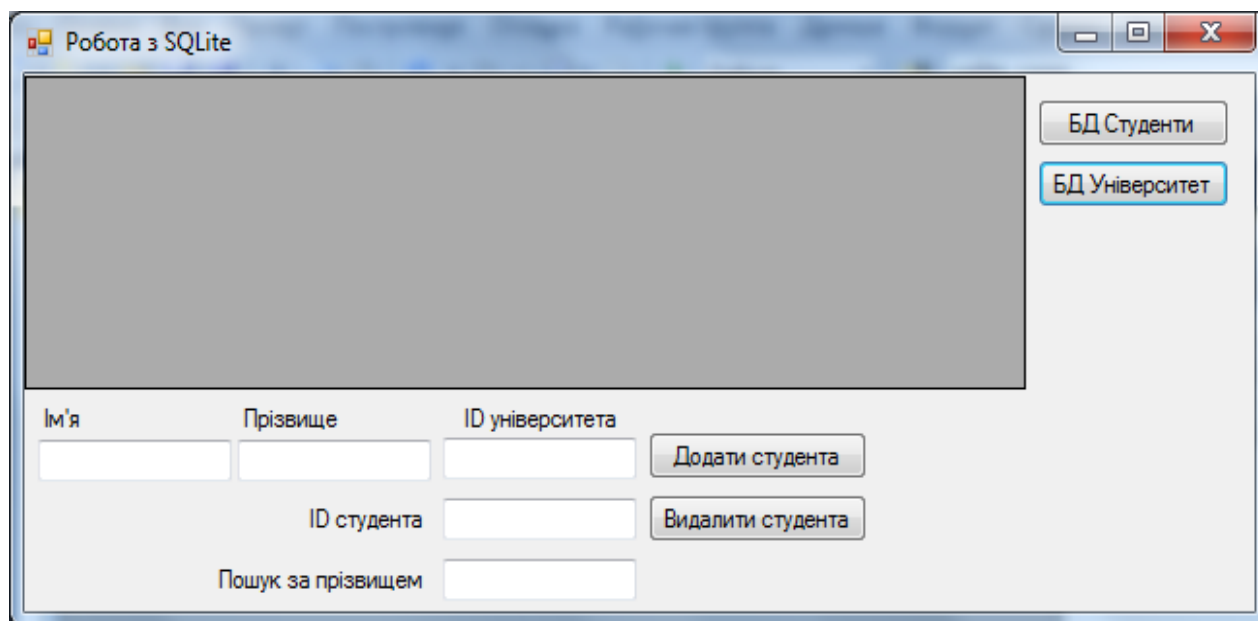


Рис. 5. Форма проекту з доданими компонентами

Підключаємо створену за допомогою SQLite Maestro БД до Windows

Forms проекту (файл бази даних з розширенням .db потрібно скопіювати в папку проекту)

Програмуємо кнопку «Університет» так:

```
sqlite_conn = new SQLiteConnection("Data Source=vuz.db");
const string sql = "select * from universitets;";
try
{
    sqlite_conn.Open();
    DataSet ds = new DataSet();
    var da = new SQLiteDataAdapter(sql, sqlite_conn);
    da.Fill(ds);
    dataGridView1.DataSource = ds.Tables[0].DefaultView;
}
catch (Exception)
{
    throw;
}
```

Аналогічно програмуємо кнопку «Студенти»:

```
sqlite_conn = new SQLiteConnection("Data Source=vuz.db");
const string sql = "select * from students;";
try
{
    sqlite_conn.Open();
    DataSet ds = new DataSet();
    var da = new SQLiteDataAdapter(sql, sqlite_conn);
    da.Fill(ds);
    dataGridView1.DataSource = ds.Tables[0].DefaultView;
}
catch (Exception)
{
    throw;
}
```

```
sqlite_conn.Close();
```

Кнопка «Додати»:

```
sqlite_conn = new SQLiteConnection("Data Source=vuz.db");
    sqlite_conn.Open();
    if
    (universityCheck(Convert.ToInt32(textBox3.Text)) == false)
    {
        textBox3.Text = "Bad id";
        return;
    }
    string sql = "INSERT INTO students (name, sname,
university) VALUES('" + textBox1.Text + "', '" +
textBox2.Text + "', " + textBox3.Text + ")";
    try
    {
        DataSet ds = new DataSet();
        var da = new SQLiteDataAdapter(sql, sqlite_conn);
        da.Fill(ds);
        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
    }
    catch (Exception)
    {
        sqlite_conn.Close();
        throw;
    }
    sqlite_conn.Close();
```

Для реалізації видалення записів з таблиці в коді процедури для кнопки «Видалення» потрібно використати відповідний SQL-запит на видалення, вказавши його як параметр об'єкта `SQLiteDataAdapter`:

Для реалізації пошуку програмуємо подію TextChanged для відповідного компонента TextBox:

```
private void textBox5_TextChanged(object sender, EventArgs e)
{
    sqlite_conn = new SQLiteConnection("Data
Source=vuz.db");
    string sql = "SELECT * FROM students WHERE sname LIKE
'" + textBox5.Text + "%'";
    try
    {
        sqlite_conn.Open();
        DataSet ds = new DataSet();
        var da = new SQLiteDataAdapter(sql,
sqlite_conn);
        da.Fill(ds);
        dataGridView1.DataSource =
ds.Tables[0].DefaultView;
    }
    catch (Exception)
    {
        throw;
        sqlite_conn.Close();
    }
    sqlite_conn.Close();
}
```

Будуємо проект, запускаємо на виконання та перевіряємо, чи коректно додаються, оновлюються, видаляються дані (рис. 6).

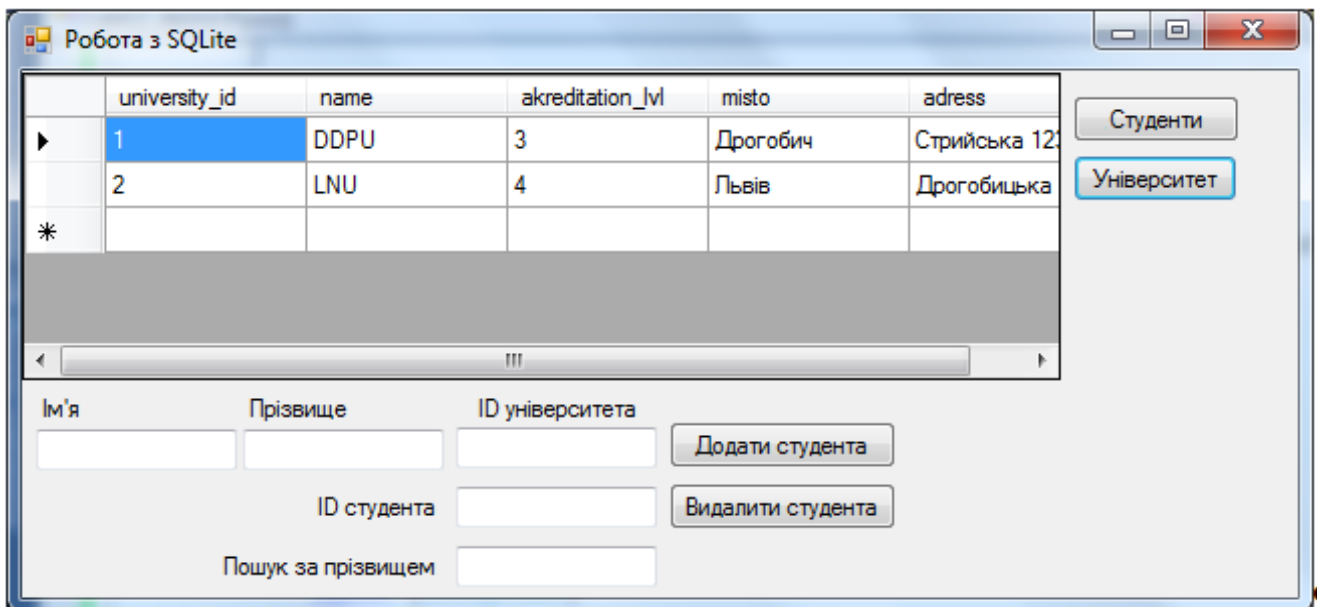


Рис. 6(а). Робота з даними, що містяться в таблиці **universitets**

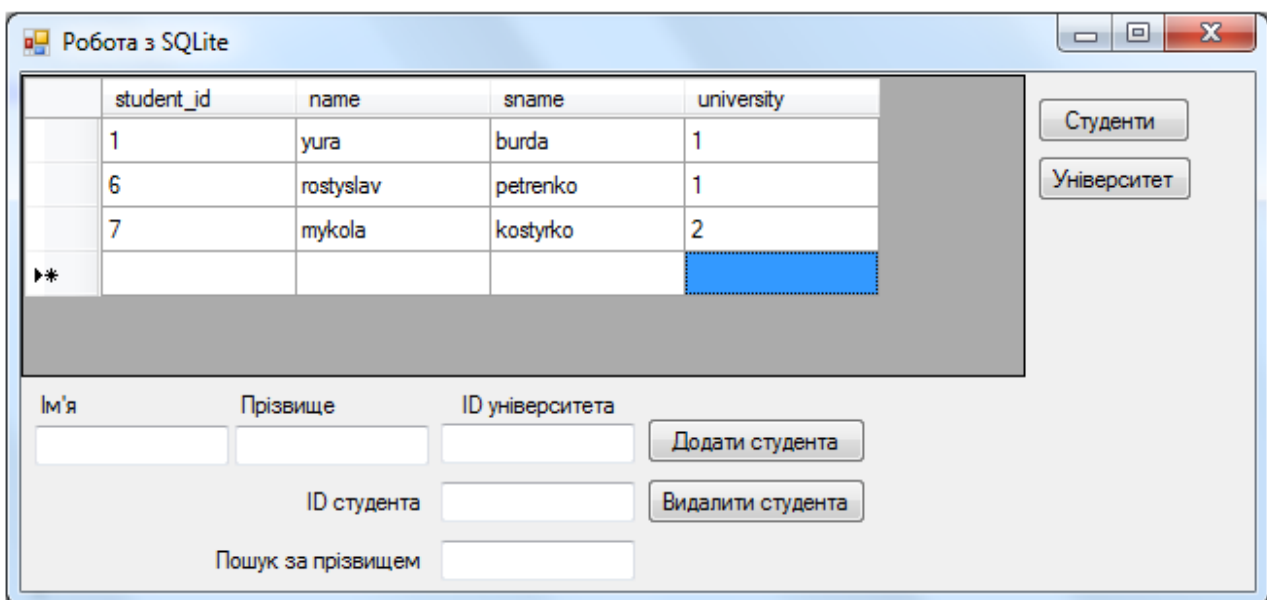


Рис. 6(б). Робота з даними, що містяться в таблиці **students**

Завдання для самостійного виконання

1. Створити нову БД згідно з Вашим варіантом за допомогою SQLite Maestro (або будь-якого іншого SQLite менеджера), що складається з 2 - 3 таблиць. Установити зв'язки між таблицями. Додати записи до таблиць за допомогою відповідних SQL-запитів.

2. Створити новий Windows Forms проект у середовищі MS Visual Studio та підключити розроблену БД до проекту. Протестувати підключення.

3. Розглянути засоби відображення даних, що містяться в БД. Реалізувати виведення даних на форму при запуску проекту.

4. Забезпечити можливість додавання та видалення даних, оновлення та фільтрування даних, пошук за деяким критерієм. Занотувати у звіт результат виконання проекту за різних значень вхідних параметрів.

Контрольні запитання

1. До якого типу належать бази даних, розроблені засобами SQLite?
2. Які SQLite менеджери Ви знаєте?
3. Назвіть призначення та особливості додатку SQLite Maestro?
4. Які типи даних передбачено у SQLite Maestro?
5. Як установити зв'язки між таблицями в SQLite Maestro?
6. Яку бібліотеку потрібно додати до проекту для роботи з БД, розробленої засобами SQLite?
7. Який режим роботи з даними описано в теоретичних відомостях?
8. Для чого використовуються об'єкти класу SQLiteDataAdapter?
9. Об'єкти якого класу використовуються для встановлення з'єднання з джерелом даних SQLite?
10. Як реалізувати з клієнтської програми видалення запису з таблиці?

Опрацювання баз даних MongoDB. Робота з даними в клієнтській програмі

МЕТА: ознайомлення зі структурою нетабличних документно-орієнтованих СКБД MongoDB та основами роботи з даними, що зберігаються в них (створення БД та колекцій, додавання, оновлення та видалення даних запити на вибірку); розроблення клієнтської програми для опрацювання даних.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

MongoDB – документно-орієнтована система керування базами даних з відкритим вихідним кодом, що не вимагає опису схеми таблиць. На відміну від реляційних баз даних, MongoDB не використовує табличну структуру з чітко заданою кількістю стовпців і типів даних.

Усередині MongoDB може бути нуль або більше баз даних, кожна з яких є контейнером для інших сутностей.

Якщо в реляційних БД вміст складають таблиці, то в MongoDB база даних складається з колекцій. Кожна колекція має своє унікальне ім'я – довільний ідентифікатор, що складається з не більше ніж 128 різних алфавітно-цифрових символів і знака підкреслення. Колекції можуть бути проіндексовані, що поліпшує продуктивність вибірки і сортування.

Колекції складаються з нуля або більше «документів». Документ можна уявити як об'єкт, який зберігає деяку інформацію. Будь-який документ всередині колекції може мати власний унікальний набір полів, дечим подібний до рядків в реляційних СКБД, однак може містити набагато більше інформації.

Документ – це набір пар ключ-значення. Значення можуть відрізнятися за типом даних. Є такі типи значень:

- **String** – рядковий тип даних (для рядків використовується кодування UTF-8);
- **Array** – тип даних для зберігання масивів елементів;
- **Binary data** – тип для зберігання даних в бінарному форматі;
- **Boolean** – зберігає логічні значення TRUE або FALSE;
- **Date** – зберігає дату в форматі часу Unix;
- **Double** – числовий тип даних для зберігання чисел з плаваючою крапкою;
- **Integer** – використовується для зберігання цілочисельних значень;
- **JavaScript** – тип даних для зберігання коду JavaScript;
- **Null** – тип даних для зберігання значення Null;
- **Object** – рядковий тип даних;
- **ObjectID** – тип даних для зберігання id документа;
- **Regular expression** – застосовується для зберігання регулярних виразів;
- **Timestamp** – застосовується для зберігання часу.

Об'єктна мова запитів у стилі JSON дає змогу проводити великий набір операцій над даними (умовний пошук, складна вставка/оновлення і таке інше) та має підтримку різних типів даних (підтримка масивів зокрема).

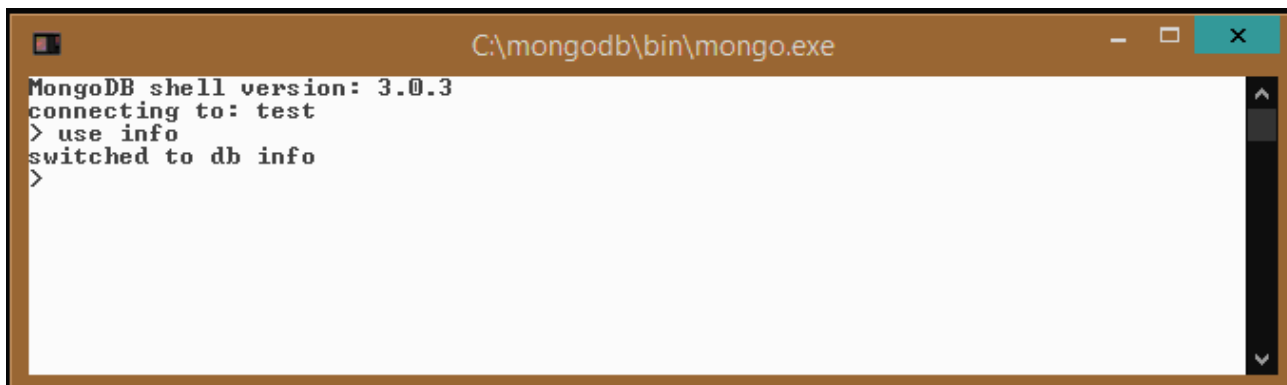
«Індекси» в MongoDB майже ідентичні індексам в реляційних базах даних. Результатом запиту до MongoDB є повернення курсору, з якими можна працювати, не завантажуючи при цьому самі дані.

Робота з даними в MongoDB

Для роботи з даними потрібно запустити сервер (C:\mongodb\bin\

mongod.exe), а потім запустити mongo.exe (C:\mongodb\bin\mongo.exe). Усі дані передаються у форматі JSON.

Перед початком роботи з даними потрібно встановити потрібну нам БД як поточну, щоб потім її використовувати, командою Use <назва_БД> (рис. 1). Якщо такої БД не існує, то MongoDB автоматично створить її при додаванні до неї даних.



```
C:\mongodb\bin\mongo.exe
MongoDB shell version: 3.0.3
connecting to: test
> use info
switched to db info
>
```

Рис. 1. Використання команди Use

Для того, щоб переглянути всі наявні БД в консолі використовують команду show dbs. Список всіх колекцій в поточній БД можна подивитися за допомогою команди show collections. Переглянути всі доступні команди для роботи з БД в MongoDB можна командою db.help() (щоб переглянути команди для роботи з колекціями в БД info потрібно скористатись командою db.info.help()).

Команда db.stats() дає змогу **переглянути статистику** щодо поточної БД. Щоб переглянути статистику за окремою колекцією, потрібно ввести команду db.<назва_колекції>.stats()

Для **створення колекції** застосовують метод db.createCollection (name, options), де name – назва колекції, а options – необов'язковий об'єкт з додатковими налаштуваннями ініціалізації. Наприклад:

```
db.createCollection ("student")
```

Ім'я колекції не повинно починатися з префікса system, позаяк він

зарезервований для внутрішніх колекцій (наприклад, колекція system.users містить всіх користувачів бази даних). Також ім'я колекції не повинно містити знака долар – \$.

Щоб **перейменувати** існуючу колекцію, слід використовувати функцію renameCollection:

```
db.student.renameCollection("нова назва")
```

Для **додавання** до колекції даних використовується функція insert:

```
db.student.insert ({ "name": "Alex", "age": 18,  
languages_p: [ "php", "c++", "python"]})
```

У цьому випадку в колекцію student додається простий об'єкт. В результаті вдалого додавання консоль виводить вираз:

```
WriteResult ({ "nInserted": 1})
```

Є другий спосіб додавання в БД документа, який містить два етапи: визначення документа (document = ({...})) і власне його додавання:

```
document = ({ "name": " Alex", "age": 18, languages_p:  
[ "php", "c++", "piton"]})
```

```
db.student.insert (document)
```

Щоб **переглянути** додані дані, використовують функцію find()(рис. 2) або find().pretty() – для форматного виведення даних.



```
C:\mongodb\bin\mongo.exe  
> db.users.find(<>  
< { "_id" : ObjectId<"556a388224e649b9e4666c35">, "name" : "Tom", "age" : 28, "lan  
guages" : [ "english", "spanish" ] }  
< { "_id" : ObjectId<"556a38e624e649b9e4666c36">, "name" : "Bill", "age" : 32, "la  
nguages" : [ "english", "french" ] }  
> -
```

Рис. 2. Запит на виведення даних

Якщо потрібно вивести дані згідно з певним критерієм, потрібно вказати відповідні поля та їхні значення. Так, наприклад, команда `db.student.find({name: "Olia"})` виведе всі документи, в яких значення поля `name = Olia`:

Для видалення з БД документів за певним значенням поля використовується команда `db.db_name.remove({"field": "value"})`.

Щоб оновити документи згідно з певним критерієм – команда `db.db_name.update({"field1": "criteria"}, {"field1": "updated", "field2": "updated"})`

Підключення MongoDB до проекту MS Visual Studio (C#)

Для розробки програми-клієнта опрацювання зазначених даних, що містяться в БД MongoDB, створимо новий Windows Forms проект у середовищі MS Visual Studio.

Додаємо бібліотеки для роботи з MongoDB. Для цього відкриваємо вкладку «Проект»-> «Керування пакетами NuGet» (рис. 2)

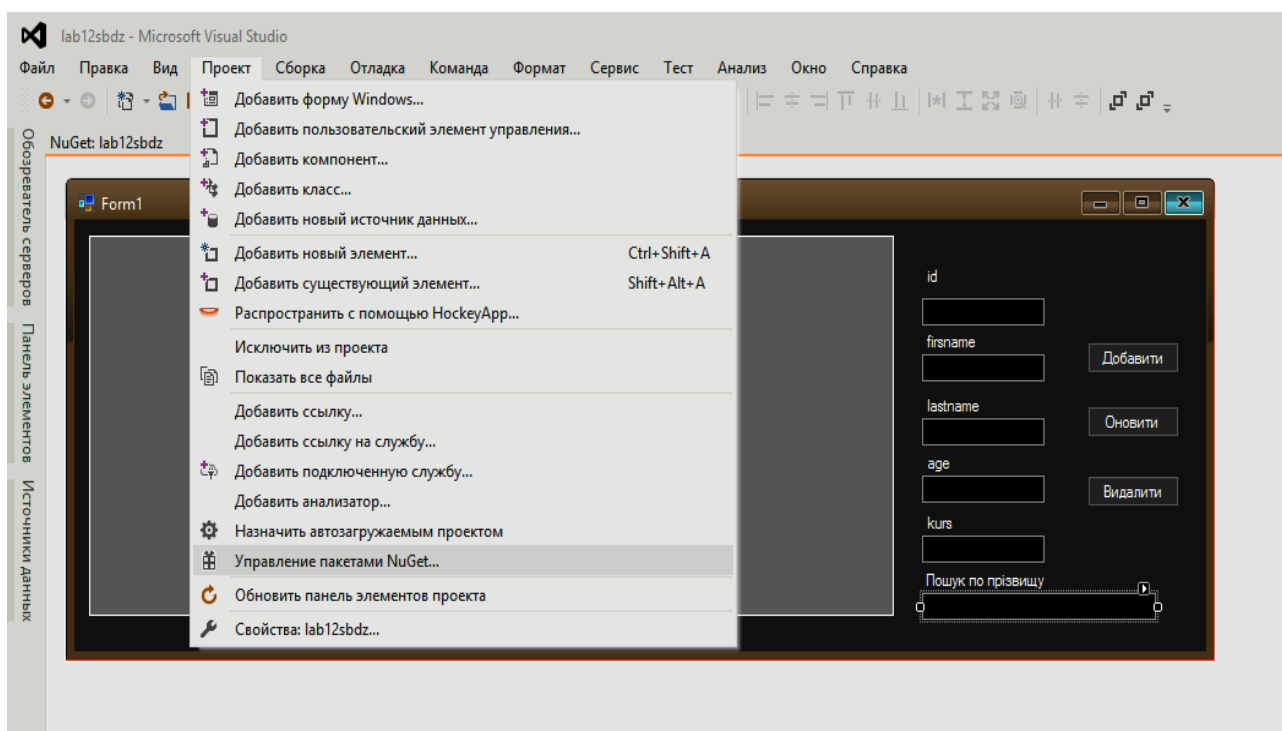


Рис. 2. Додавання NuGet пакета

Уведемо в пошук mongodb та встановимо пакет, як показано на рис 3.

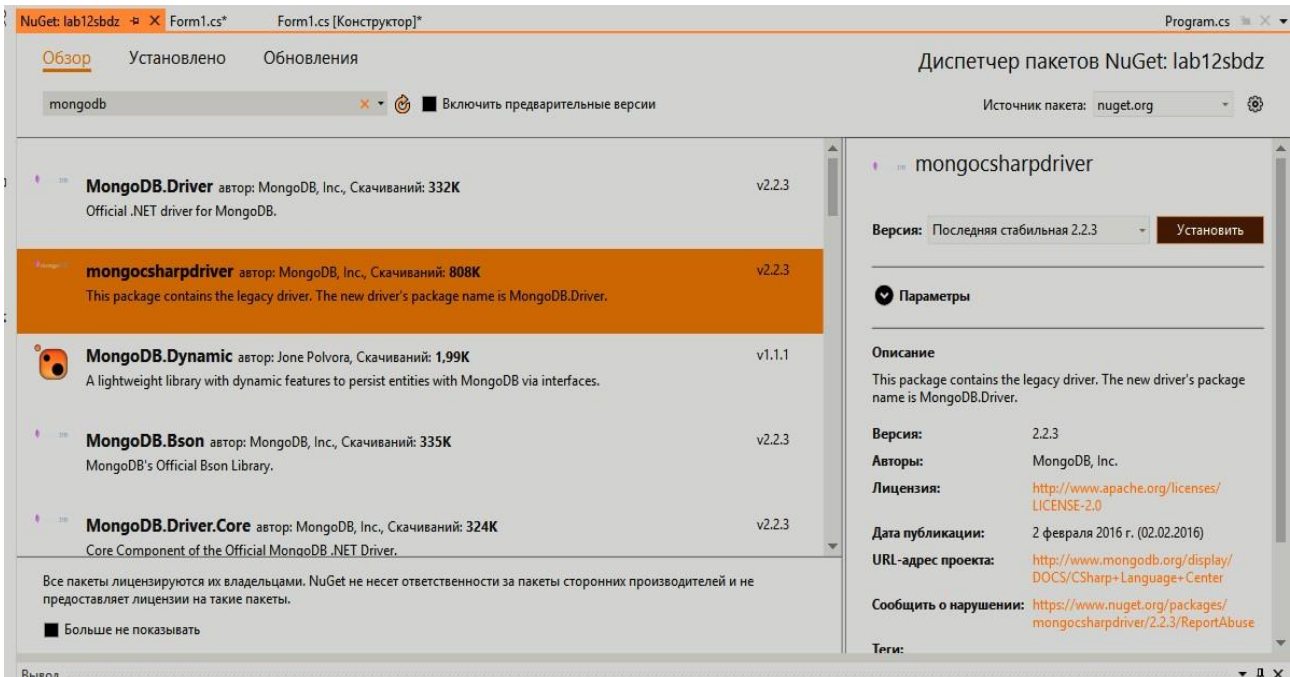


Рис. 3. Додавання NuGet пакета

Після успішно встановлення в пункті «Посилання» повинно з'явитись 4 dll файли (рис 4)

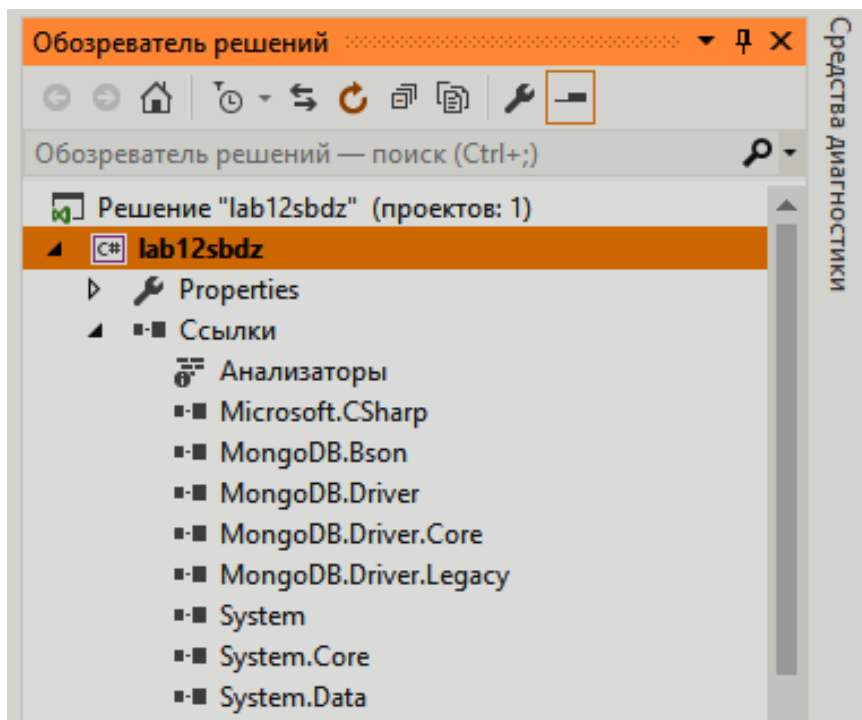


Рис. 4. Додавання NuGet пакетів

Тепер підключимо бібліотеки до проекту:

```
using MongoDB.Driver;
using MongoDB.Driver.Builders;
```

Зверніть увагу: для того, щоб проект працював стабільно, потрібно Visual Studio версії не нижче 2012 і встановлений .Net Framework версії не нижче 4.5. Реалізуємо процес підключення до бази даних MongoDB. Для початку опишемо наш клас *student*.

```
public class student
{
    public object id { get; set; }
    public string firstname { get; set; } public string
lastname { get; set; } public int age { get; set; }
    public int kurs { get; set; }

    public student(object id, string firstname, string
lastname, int age, int kurs)
    {this.id = id; this.firstname = firstname; this.lastname =
lastname; this.age = age;
    this.kurs = kurs; }
}
```

Створимо функцію `connect()` та змінну для роботи з колекцією

```
MongoCollection<student> collection; public void
connect()
{
    const string url = "mongodb://localhost:27017"; var
client = new MongoClient(url);
    var server = client.GetServer();
    var mongoDatabase=server.GetDatabase("test");
    //де test - назва БД
    collection=mongoDatabase.GetCollection("students");
```

```
//де students - назва колекції  
}
```

Для того, щоб зчитати дані після запуску проекту та вивести їх в компонент dataGridView1 потрібно прописати обробник Form_Load():

```
private void Form1_Load(object sender, EventArgs e)  
{  
    connect();  
    dataGridView1.DataSource = collection.FindAll().ToList();  
}
```

Рис. 9. Завантаження даних з колекцій у dataGridView1

Для додавання записів у БД можна скористатись функцією collection.Insert(). Опишемо подію для кнопки **Додати**.

```
private void button1_Click(object sender, EventArgs e)  
{  
    student st = new  
student(ObjectId.GenerateNewId(), textBox2.Text,  
textBox3.Text, int.Parse(textBox4.Text),  
int.Parse(textBox5.Text));  
    var query = Query<student>.EQ(x => x.id, st.id); //шукаємо  
чи id існує  
    var result = collection.Find(query).ToList();  
    if (result.Count > 0)  
        //Перевірка чи існує даний id в колекції, якщо існує  
result.Count буде більше нуля  
    {  
        MessageBox.Show("Даний id вже існує", "Помилка"); return;}  
    else  
    {  
        collection.Insert(st); //Додаємо запис в БД  
    }  
    dataGridView1.DataSource = collection.FindAll().ToList();  
    //оновлення таблиці
```

```
dataGridView1.Columns[0].Visible = false;
//приховуємо 1-ий стовпець із згенерованим MongoDB id.
}
```

Для видалення даних можна скористатись функцією `collection.Remove()` а для пошуку даних – `collection.Find(query).ToList()` відповідно.

Наприклад, пошук за прізвищем можна реалізувати так:

```
private void textBox6_TextChanged(object sender, EventArgs e)
{
    var query = Query<student>.Matches(x => x.lastname, "^"
    + textBox6.Text); dataGridView1.DataSource =
    collection.Find(query).ToList();
}
```

Завдання для самостійного виконання

1. Створити нову БД засобами MongoDB згідно з Вашим варіантом, що складається з декількох колекцій.
2. Наповнити БД даними згідно з Вашою предметною областю за допомогою відповідних команд (за можливості використати різні типи даних).
3. Розглянути наведені в теоретичних відомостях команди для роботи з БД, колекціями та документами, що містяться в БД.
4. Підключити розроблену БД до проекту MS Visual Studio (C#/C++). Протестувати підключення.
5. Забезпечити коректне відображення даних з БД.
6. Розглянути наведені засоби роботи з даними, що містяться в БД MongoDB Реалізувати можливість додавання, видалення та оновлення даних, пошук даних згідно з деяким критерієм.

Контрольні запитання

1. Яку структуру має БД у MongoDB?
2. Яку команду використовують для того, щоб переглянути всі наявні БД в MongoDB?
3. Яким чином видалити колекцію з БД в MongoDB?
4. Які типи даних підтримуються в MongoDB?
5. Як можна додати нові дані до колекції?
6. Для чого використовують систему керування пакетами NuGet?
7. Яку функцію можна використати для додавання даних з клієнтської програми у БД MongoDB?
8. Яким чином можна реалізувати пошук даних згідно деякого критерію з клієнтської програми?
9. Які компоненти MS Visual Studio можна використати для відображення даних, що містяться у колекціях?
10. За допомогою яких команд можна реалізувати видалення даних з колекції?

Варіанти завдань для розробки власної бази даних

	Предметна область	Орієнтовний вміст БД
1.	Облік студентів на факультеті ЗВО	Особисті дані, рік народження, адреса, дата зарахування, номер наказу, факультет, група, куратор групи, форма навчання, успішність, наявність стипендії
2.	Облік пацієнтів у поліклініці	Номер, прізвище, ім'я, по батькові, дата народження пацієнта; прізвище, ім'я, по батькові лікаря, посада та спеціалізація лікаря; діагноз, поставлений даним лікарем даному пацієнту, чи необхідно амбулаторне лікування, строк втрати працездатності, чи є на диспансерному обліку, примітка
3.	Облік товарів на складі	Найменування товару, тип товару, кількість на складі, дата надходження, виробник, постачальник, опис товару, вартість.
4.	Облік відвідувачів спортивної школи	Ім'я та прізвище вихованця, дата народження, адреса, телефон, секція, команда, особисті дані тренерів, розклад занять, облік змагань та досягнень за роками, примітка
5.	Облік товарів та продаж у магазині спортоварів	Наявні на складі товари, товари під замовлення, назва товару, вартість товару, дата продажу, відповідальний менеджер, опис товару, тип товару.

6.	Облік автомобілів в автосалоні	Модель авто, вартість, колір кузова, наявність або відсутність автоматичної коробки передач, кількість одиниць, клієнти та менеджери салону.
7.	Облік товарів і продаж у продуктовому магазині	Наявні на складі товари, товари, що є на полицях, назва товару, тип товару, вартість товару, час та дата продажу, відповідальний менеджер, упаковка
8.	Книжкове видавництво	Автори, назва, розділ (технічна, суспільно-політична і т.п.), рік видання, тираж, відповідальний за друк, кількість сторінок, кількість томів; номер тома, ціна
9.	Облік рейсів в аеропорту	Назва аеропорту, марка літака, розклад рейсів, номер рейсу, пункт призначення, дата рейсу, час вильоту, час у дорозі, чи є маршрут міжнародним, пілоти, стюардеси, кількість місць, вартість квитків, кількість проданих квитків
10.	Облік книжок і читачів у бібліотеці	Прізвища читачів, номер читацького квитка, рік народження, дата запису, вид читача (студент, аспірант, викладач тощо), назви взятих книг, ціна, чи є новим виданням, коротка анотація, дати їх видачі, дати повернення, працівники бібліотеки

11.	Облік робіт у мережі автомайстерень	Особисті дані водія, номер прав водія, адреса та телефон власника автомобіля; номер, марка, потужність і колір автомобіля особисті дані механіка, кваліфікація механіка, назва, адреса та телефон ремонтної майстерні
12.	Облік товарів, що виробляються на підприємстві	Код виробу, назва виробу, адреса та телефон підприємства, кількість співробітників, обсяг продукції, що випускається, тип продукції, рік випуску та обсяг випуску певного виробу, замовлення, клієнти
13.	Облік науково-викладацьких кадрів на кафедрі та їх навантаження	Прізвище та ім'я викладача, рік народження, адреса, телефон, кафедра, посада, науковий ступінь, стажування, навчальне навантаження, дисципліни, закріплені за викладачем, наукові праці
14.	Облік місць у поїздах у касах продажу	Тип потягу, сполучення, кількість місць, вартість квитка, кількість проданих квитків, заброньовані квитки
15.	Облік тварин і рослин, занесених в Червону книгу України	Нава тварини, вид тварини, рід, сімейство, дата занесення в книгу, чисельність популяції, живе чи ні в Україні, необхідні для порятунку заходи, назва рослини, вид, ареал поширення, дата занесення в книгу

16.	Облік рослин та робіт у лісовому господарстві	Найменування зеленого масиву, площа, основна порода, чи є заповідником, дата останньої перевірки; прізвище обслуговуючого лісника, примітка
17.	Облік авто та замовлень у автотранспортному підприємстві	Номерний знак автомобіля, марка автомобіля, технічний стан авто, вантажопідйомність, середня швидкість, витрата палива; табельний номер водія, особисті дані водія, стаж роботи, оклад; дата виїзду, дата прибуття, місце призначення, відстань, витрата пального, маса вантажу
18.	Облік працівників у відділі кадрів	Прізвище, ім'я, по батькові, домашня адреса працівника, телефон, дата народження, посада працівника, дата зарахування, стаж роботи, оклад; найменування підрозділу, кількість штатних одиниць, фонд заробітної плати за місяць і за рік
19.	Облік замовлень та турів в турфірмі	Наявні тури, час поїздки, вартість туру, тривалість туру, кількість турів, дані про клієнтів, замовлені тури, менеджери турфірми

20	Облік замовлень на роботи в будівельній фірмі	Прізвище, ім'я, по батькові клієнта, номер рахунку, адреса, телефон клієнта; номер замовлення, дата виконання, вартість замовлення, витрачені будівельні матеріали, їх ціна та кількість, працівники, що виконували замовлення, примітка
21.	Облік товарів в аптеці	Наявні на складі товари, назва товару, місце збереження товару (холодильник, сейф, склад), тип товару (ліки, косметичні засоби, супутні товари), лікарська форма, вартість товару, термін придатності, час та дата надходження, відповідальний менеджер, упаковка, примітки

Література

1. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань : навч. посібник. – Львів : «Магнолія-2006», 2012. – 584 с.
2. Павельчак А.Г., Самотий В.В., Дзелендзяк У.Ю. Бази даних та знань : конспект лекцій [для студентів базового напрямку 050201 “Системна інженерія”] – Львів : Львівська політехніка. – 2011. – 115 с.
3. Лосєв М. Ю., Тарасов О. В., Федько В. В. Організація баз даних і знань (ADO.NET) : конспект лекцій / М. Ю. Лосєв, О. В. Тарасов, В. В. Федько – Х. : Вид. ХНЕУ, 2011. – 108 с.
4. Хантер Д. XML. Работа с XML, 4-е издание / Д. Хантер, Д. Рафтер, Д. Фаусетт та ін. – СПб. : Діалектика, 2010. – 1344 с.
5. Садаладж П.Дж. NoSQL: Новая методология разработки нереляционных баз данных / П. Дж.Садаладж, М. Фаулер– М. : Вильямс, 2013. – 172 с.
6. Кайл Бэнкер. MongoDB в дії / переклад з англ. / Кайл Бэнкер – М. : ДМК «Пресс», 2012. – 395с.
7. Пасічник В.В., Резніченко В.А. Організація баз даних та знань. – К. : Видавнича група BHV, 2006. – 384 с.
8. Питер Роб, Карл Коронел. Системы баз данных: проектирование, реализация и управление. 5-е изд., перераб. и

доп. , пер. с англ. –СПб.: БХВ-Петербург, 2004. -1040с.: ил.

- 9.** Джеффри Д.Ульман, Дженнифер Уидом. Введение в системы баз данных. Пер. с англ. – М. : Издательство «Лори», 2000. – 374с.
- 10.** Филиппов В.А., Щукин Б.А., Богатырева Л.В. Многозначные СУБД и XML базы даннях / Филиппов В.А., Щукин Б.А., Богатырева Л.В. – «И.Д. ЛЕНАНД», 2008. – 144 с.
- 11.** Доманецька І.М. Промислові системи управління базами даних: Конспект лекцій. – К. : КНУБА, 2012. – 64с.

Навчальне видання

Ірина Шаклеїна, Надія Ших, Роман Білий

Опрацювання баз даних засобами MS Visual Studio

**Методичні рекомендації
до виконання лабораторних робіт**

**Редакційно-видавничий відділ
Дрогобицького державного педагогічного університету
імені Івана Франка**

**Головний редактор
*Ірина Невмержицька***

**Редактор
*Іванна Біблій***

**Технічний редактор
*Наталя Кізима***

**Коректор
*Наталя Кізима***

Здано до набору 31.07.2018 р. Підписано до друку 22.08.2018 р.
Формат 60x90/16. Папір офсетний. Гарнітура Times. Наклад 100 прим.
Ум. друк. арк. 6, 00. Зам. 188.

Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка. (Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції ДК № 5140 від 01.07.2016 р.) 82100, Дрогобич, вул. І. Франка, 24, к. 42, тел. 2 – 23 – 78.