Комп'ютерний практикум №8

Тема: використання технології ADO.NET у зв’язці з MS SQL Server. Постачальник даних.

Мета: отримати представлення про методи використання ADO.NET у зв’язці з MS SQL Server, а саме при роботі з постачальником даних.

Навички що будуть здобуті: вміння отримувати доступ до створеної бази даних, виконувати команди SQL, працювати з отриманими даними.

Зміст:

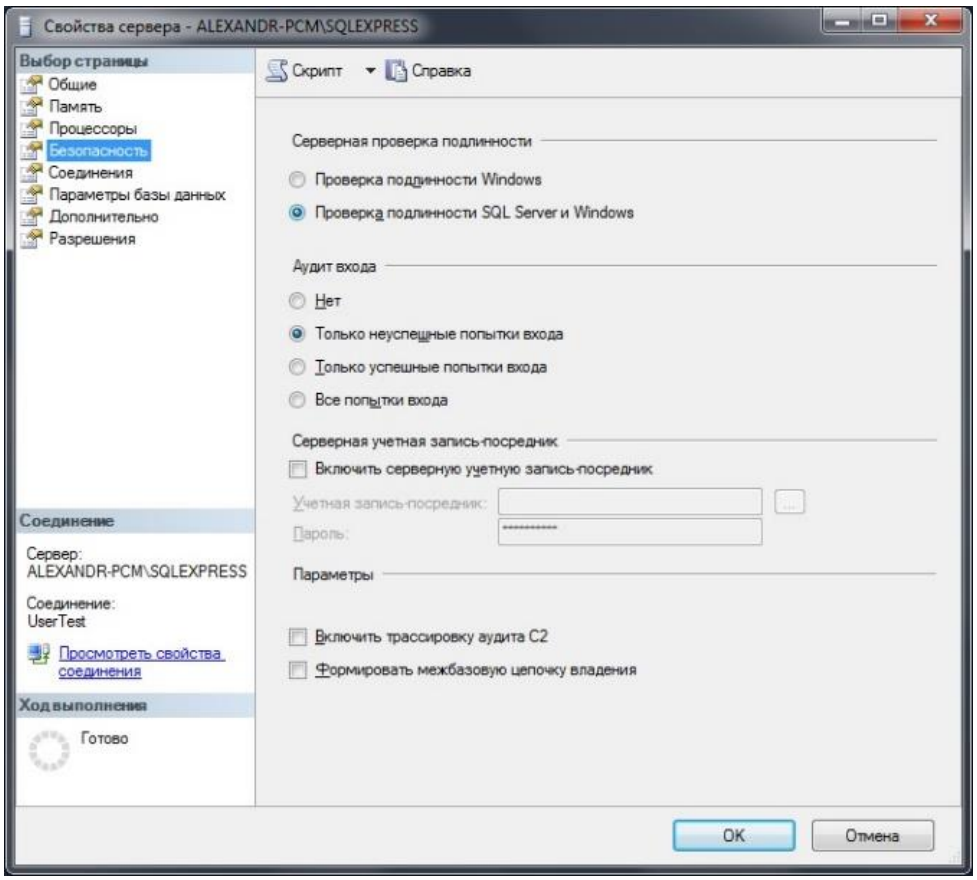
1) Підключення до бази даних.
2) Виконання основних операцій з базою даних.
3) Використання транзакцій.
4) Робота з запитами що повертають значення.
5) Робота зі збереженими процедурами.

Контрольні питання до роботи:

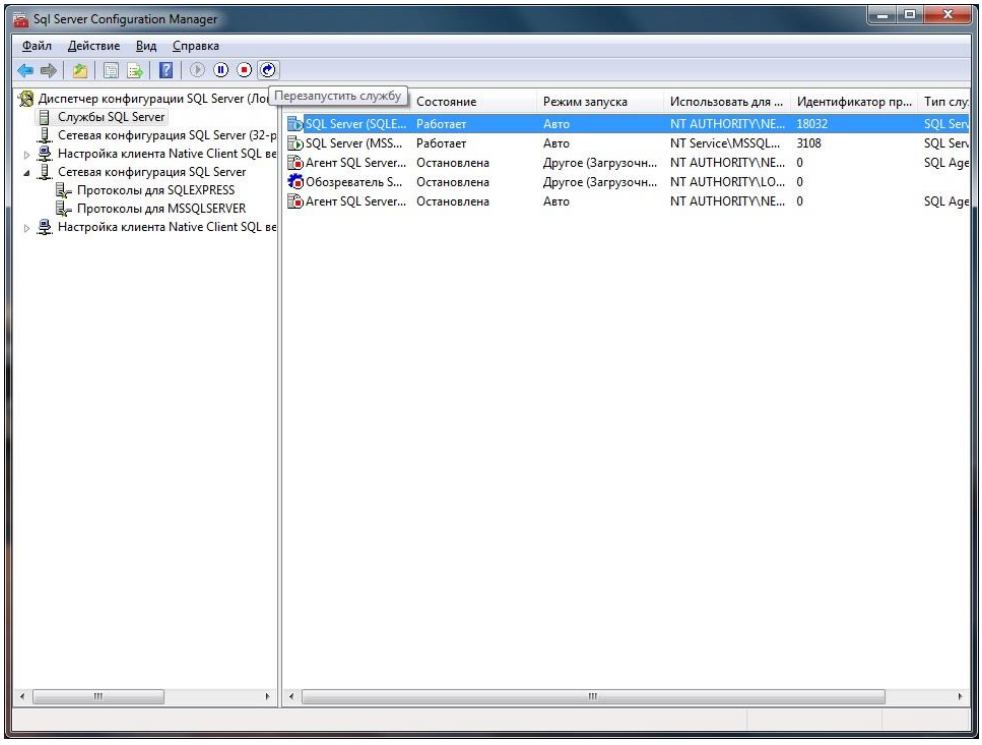
1. Архітектура ADO.NET.
2. Визначення і призначення об’єктів постачальника даних.
3. Поняття і механізм роботи пулу підключень.
4. Рядок підключення до бази даних. Його параметри.
5. Об’єкт SqlConnectionStringBuilder, його використання.
6. Об’єкт SqlConnecton, його використання.
7. Об’єкт SqlCommand, його використання.
8. Об’єкт SqlDataReader, його використання.
9. Об’єкт SqlTransaction, його використання.
10. Типізація результатів за допомого SqlDataReader. Типи C#, SQL та методи для приведення цих типів які ви пам’ятаєте.
11. Механізм параметризації запитів. Навіщо потрібен, як працює.
12. Механізм отримання скалярних значень з запитів.
13. Використання вихідних параметрів запитів.
14. Механізм роботи зі збереженими процедурами.
15. Механізм збереження файлів у базу даних.

Хід виконання роботи:

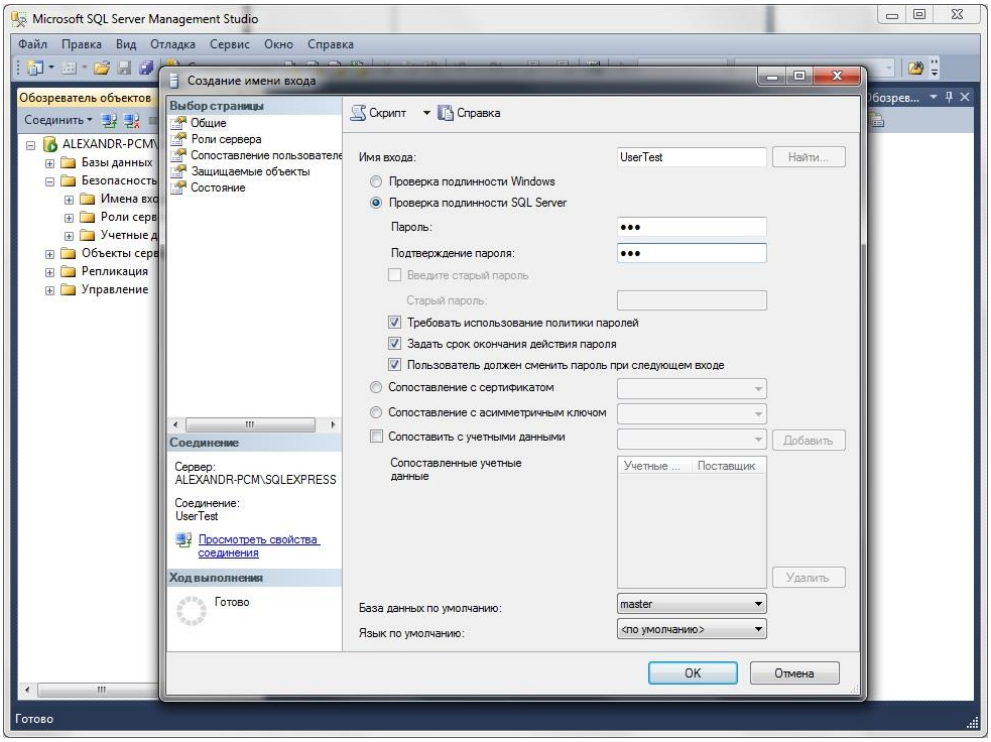
1. Перш за все розглянемо механізм підключення до серверу за допомогою технології ADO.NET. Причому для конкретного користувача. Для цього створимо цього користувача та проведемо необхідні налаштування.
2. Запустимо SQL Server Menagement Studio та з оглядача об’єктів перейдемо у властивості серверу. У них знайдемо вкладку «безпека» і там виберемо «перевірка справжності SQL Server і Windows» яке включає перевірку безпеки не тільки за допомогою аутентифікації Windows, а і за іменем та паролем користувача що буде нами зараз створений:



Також знайдемо додаток «диспетчер конфігурації SQL Server», який встановлюється разом з сервером, та включимо у мережевій конфігурації SQL Server протокол TCP/IP. Далі перезапустимо SQL Server. Це дозволить нам уникнути можливих помилок.



3. Створимо нового користувача у MS SQL Server. Для цього перейдемо у вкладку «безпека» оглядача обєктів і у іменах входу створемо нове. При цьому виберемо перевірку справжності за допомогою SQL Server, вкажемо ім’я та пароль, а також у «ролях серверу» зробимо його сісадміном.



4. Щоб переконатись що вхід за іменем та паролем працює відключимось від сервера та знову підключимось вказавши створеного користувача та його пароль:

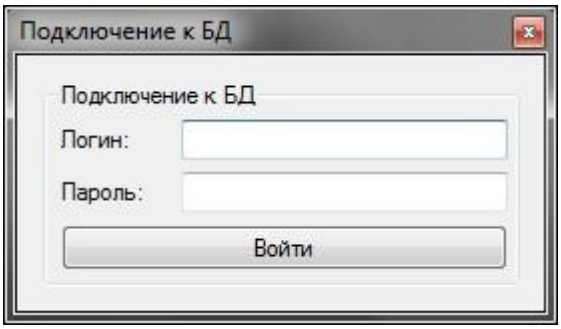
5. Далі у SQL Server Menagement Studio створимо просту базу даних що складається з однієї таблиці з якою будемо далі працювати:

```
CREATE DATABASE Laba6
GO

USE Laba6
CREATE TABLE Images
(
    Id int IDENTITY NOT NULL,
    NameFile varchar(50) NOT NULL,
    Title varchar(50) NOT NULL,
    ImageData varbinary(max) NOT NULL
```

```
)
GO
```

6. Перейдемо у Visual Studio та створимо новий проект Windows Forms, а у ньому наступну форму з ім’ям form1:



7. Далі у метод що викликається при натисканні на кнопку пропишемо наступний код:

```
string connectionString = @"Data Source=.\SQLEXPRESS; Initial
Catalog=Laba6; " +
"User ID=" + textBox1.Text + ";" +
"Password=" + textBox2.Text + ";";

using (SqlConnection connection = new
SqlConnection(connectionString))
{
    try
    {
        connection.Open();
        //Вывод на экран информации о подключении к базе данных
        MessageBox.Show("Удалось успешно подключиться к " +
connection.Database);
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message);
    }
}
```

Тепер якщо ми запустимо програму, то зможемо підключитись до створеної бази даних з використанням ім’я і пароля створеного нами користувача та побачимо відповідне повідомлення. Зверніть увагу що при використанні using ми можемо не закривати підключення до бази даних, це буде зроблено автоматично. Аутетифікація за допомогою перевірки справжності Windows вважається більш пріоритетною і тому додавання у строку підключення коду, що включає цю перевірку, зробить введені ім’я та пароль неактуальними. Причому цей код можна прописати прямо у поле вводу пароля: «; Integrated Security=True». В результаті підключення буде здійснюватися незалежно від того введено правильно логін та пароль чи ні. Очевидно що це вразливість. Щоб прибрати її замінимо рядок підключення конструктором:

```
SqlConnectionStringBuilder connectionStringBuilder = new
SqlConnectionStringBuilder();
connectionStringBuilder["Data Source"] = @".\SQLEXPRESS";
connectionStringBuilder["Initial Catalog"] = "Laba6";
connectionStringBuilder["User ID"] = textBox1.Text;
connectionStringBuilder["Password"] = textBox2.Text;
```

```
using (SqlConnection connection = new
SqlConnection(connectionStringBuilder.ConnectionString))
{
    try
    {
        connection.Open();
       //Вывод на экран информации о подключении к базе данных
        MessageBox.Show("Удалось успешно подключиться к " +
connection.Database);

    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message);
    }
}
```

8. Створимо нову форму Form2 та змінимо код підключення до бази даних у першій формі:

```
connection.Open();
Form2 form = new
Form2(connectionStringBuilder.ConnectionString);
form.Show();
this.Hide();
```

9. Додамо у нову форму наступний код:

```
string connectionString;

public Form2(string str)
{
    InitializeComponent();

    connectionString = str;
}

private void Form2_FormClosed(object sender, FormClosedEventArgs
e)
{
    Form form = Application.OpenForms[0];
    form.Close();
}
```

10. Ми отримали зв'язку з двох форм у першій з яких вводиться логін та пароль користувача після чого, якщо все введено правильно, для подальшої роботи відкривається друга форма та приховується перша. Щоб спростити наступне тестування додамо у `connectionStringBuilder` новий параметр: `connectionStringBuilder["Integrated Security"] = true;` чи взагалі приберемо першу форму, вона нам була потрібна лише як приклад.
11. Додамо на другу форму наступні єлементи управління (PictureBox та відповідні кнопки, поля вводу, випадаючі списки, тощо):

12. Перш за все додамо обробник на кнопку «Загрузить картинку на форму». Він буде завантажувати зображення у PictureBox на формі. Крім того він запише FileName у відповідне поле вводу на формі:

```
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
//Создание диалогового окна для выбора файла
    open_dialog.Filter = "Image
Files(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG|All files
(*.*)|*.*"; //Формат загружаемого файла
    if (open_dialog.ShowDialog() == DialogResult.OK) //Если в
окне была нажата кнопка "OK"
    {
        try
        {
            textBox3.Text = open_dialog.SafeFileName;
            pictureBox1.Image = new
Bitmap(open_dialog.FileName);
            pictureBox1.Invalidate();
        }
        catch
        {
            DialogResult rezult = MessageBox.Show("Невозможно
открыть выбранный файл", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
}
```

13. Додамо обробник на кнопку «Сохранить картинку в БД». Він буде підключатись до бази диних за допомогою рядка підключення створеного у першій фрмі та буде записувати

дані у базу (зображення, ім'я файлу, його заголовок). Слід також зазначити що параметризація параметрів запросу покликана вирішити ту ж саму проблему що і при авторизації коли користувач міг вбудувати у запит потрібні йому команди.

```
private void button2_Click(object sender, EventArgs e)
{
    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;
            command.CommandText = @"INSERT INTO Images VALUES
(@FileName, @Title, @ImageData)";
            command.Parameters.Add("@FileName",
SqlDbType.NVarChar, 50);
            command.Parameters.Add("@Title", SqlDbType.NVarChar,
50);
            command.Parameters.Add("@ImageData",
SqlDbType.Image, 1000000);

            //Передаем данные в команду через параметры
            ImageConverter converter = new ImageConverter();
            command.Parameters["@FileName"].Value =
textBox3.Text;
            command.Parameters["@Title"].Value = textBox4.Text;
            command.Parameters["@ImageData"].Value =
(byte[])converter.ConvertTo(pictureBox1.Image, typeof(byte[]));
            command.ExecuteNonQuery();
        }
        catch (Exception exception)
        {
            MessageBox.Show("Не получилось удалить из БД " +
exception.Message);
        }
    }
}
```

14. Отже тепер можна завантажити зображення у PictureBox на формі та звідти записати його у базу даних. Додамо метод для читання зображення з бази даних. Він буде формувати запит до бази даних, та вибирати з нього деякий рядок номер якого відповідає індексу випадаючого списку ComboBox. Крім того він запише номер, ім'я файлу, його заголовок, та власне зображення у відповідні елементи управління форми:

```
private void ReadLine(int stringN)
{
    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        try
        {
```

```
                connection.Open();
                string sql = "SELECT * FROM Images";
                SqlCommand command = new SqlCommand(sql,
connection);
                SqlDataReader reader = command.ExecuteReader();

                int i = 0;
                while (reader.Read())
                {
                    if (i == stringN)
                    {
                        int id = reader.GetInt32(0);
                        string filename = reader.GetString(1);
                        string title = reader.GetString(2);
                        byte[] data = (byte[])reader.GetValue(3);

                        textBox2.Text = id.ToString();
                        textBox3.Text = filename;
                        textBox4.Text = title;

                        //Пишем изображение в pictureBox1
                        using (MemoryStream inStream = new
MemoryStream())
                        {
                            inStream.Write(data, 0, data.Length);
                            Image image =
Image.FromStream(inStream);
                            pictureBox1.Image = new Bitmap(image);
                        }
                        break;
                    }
                    else i++;
                }
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось прочитать из БД. " +
exception.Message);
            }
        }
    }
```

15. Якщо додати у конструктор форми виклик методу ReadLine(int stringN) отримаємо найпростіший варіант читання та запису даних у БД. Ускладнимо його. Для цього додамо метод що буде зчитувати заголовки усіх файлів та записувати їх у випадаючий список:

```
    private void CreateList()
    {
        comboBox1.Items.Clear();
        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
```

```
            try
            {
                connection.Open();
                string sql = "SELECT Title FROM Images";
                SqlCommand command = new SqlCommand(sql,
connection);
                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    comboBox1.Items.Add(reader.GetString(0));
                }
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось прочитать из БД. " +
exception.Message);
            }
        }
    }
```

16. Додамо метод що буде викликатись при виборі іншого рядка з випадаючого списку:

```
    private void comboBox1_SelectedIndexChanged(object sender,
    EventArgs e)
    {
        if (comboBox1.SelectedIndex >= 0)
        {
            ReadLine(comboBox1.SelectedIndex);
        }
    }
```

17. Та методи для перемикання вибраного зображення кроком вліво та вправо:

```
    //Влево
    private void button3_Click(object sender, EventArgs e)
    {
        if (comboBox1.SelectedIndex == 0) comboBox1.SelectedIndex =
    comboBox1.Items.Count - 1;
        else comboBox1.SelectedIndex--;
    }

    //Вправо
    private void button4_Click(object sender, EventArgs e)
    {
        if (comboBox1.SelectedIndex == comboBox1.Items.Count - 1)
    comboBox1.SelectedIndex = 0;
        else comboBox1.SelectedIndex++;
    }
```

18. Змінимо також конструктор класу Form2:

```
    InitializeComponent();

    connectionString = str;
    CreateList();
    comboBox1.SelectedIndex = 0;
```

та метод що пише дані у базу. Додамо у нього роботу з випадаючим списком після того, як запис було здійснено:

```
//Обновляем список на форме
CreateList();
comboBox1.SelectedIndex = comboBox1.Items.Count - 1;
```

19. Отже на даному етапі ми маємо додаток що дозволяє завантажувати зображення на форму, зберігати його у базу даних, перемикати зображення кнопками «<» «>» та за допомогою випадаючого списку. Значення ім'я файлу, його заголовку можна змінити поки що тільки у полях вводу, без внесення їх у базу даних. Також можна змінити поточне зображення (кнопці «изменить» ставимо обробник що завантажує зображення на форму). Необхідно ще додати зміну записів, їх видалення та написати більш зручне додавання. Почнемо з останнього:

```
private void button7_Click(object sender, EventArgs e)
{
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    button1_Click(sender, e);
    flag = true;
}
```

Щоб розділити команди додавання та оновлення даних на кнопці «Сохранить» введемо змінну flag що дорівнює false за змовчанням.

20. Додамо обробник на кнопку «Сохранить». Якщо було натиснуто на кнопку «Новый» він буде створювати новий запис у базі даних, а якщо ні, то буде намагатися змінити вибраний у випадаючому списку запис.

```
private void button6_Click(object sender, EventArgs e)
{
    if (flag)
    {
        button2_Click(sender, e);
        flag = false;
    }
    else
    {
        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                SqlCommand command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = @"UPDATE Images SET
NameFile = @FileName, Title = @Title, ImageData = @ImageData
WHERE Id = @Id";
                command.Parameters.Add("@Id", SqlDbType.Int);
                command.Parameters.Add("@FileName",
SqlDbType.NVarChar, 50);
```

```
                command.Parameters.Add("@Title",
SqlDbType.NVarChar, 50);
                command.Parameters.Add("@ImageData",
SqlDbType.Image, 1000000);

                //Передаем данные в команду через параметры
                ImageConverter converter = new ImageConverter();
                command.Parameters["@Id"].Value = textBox2.Text;
                command.Parameters["@FileName"].Value =
textBox3.Text;
                command.Parameters["@Title"].Value =
textBox4.Text;
                command.Parameters["@ImageData"].Value =
(byte[])converter.ConvertTo(pictureBox1.Image, typeof(byte[]));
                command.ExecuteNonQuery();

                //Обновляем список на форме
                int id = comboBox1.SelectedIndex;
                CreateList();
                comboBox1.SelectedIndex = id;
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось сохранить
изменения в БД. " + exception.Message);
            }
        }
    }
}
```

21. Додамо обробник видалення запису з бази даних:

```
private void button8_Click(object sender, EventArgs e)
{
    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;
            command.CommandText = @"DELETE Images WHERE Id =
@Id";
            command.Parameters.Add("@Id", SqlDbType.Int);

            //Передаем данные в команду через параметры
            ImageConverter converter = new ImageConverter();
            command.Parameters["@Id"].Value = textBox2.Text;
            command.ExecuteNonQuery();

            //Обновляем список на форме
            CreateList();
```

```
                comboBox1.SelectedIndex = comboBox1.Items.Count - 1;
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось удалить из БД. " +
exception.Message);
            }
        }
    }
```

22. Технологія ADO.NET дозволяє виконувати ряд операцій єдиним пакетом – транзакцією. Якщо бодай одна з операцій буде виконана невдало, можна буде здійснити відкат виконання останніх операцій. У якості прикладу використання транзакцій перепишемо обробник збереження даних у базу – зробимо оновлення запису у базі та оновлення записів випадаючого списку на формі в одній транзакції:

```
    private void button9_Click(object sender, EventArgs e)
    {
        if (flag)
        {
            button2_Click(sender, e);
            flag = false;
        }
        else
        {
            int id = comboBox1.SelectedIndex;
            string[] list = new string[comboBox1.Items.Count];
            for (int i = 0; i < comboBox1.Items.Count; i++)
            {
                list[i] = comboBox1.Items[i].ToString();
            }
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                connection.Open();
                SqlTransaction transaction =
connection.BeginTransaction();
                SqlCommand command = new SqlCommand();
                command.Transaction = transaction;
                try
                {
                    command.Connection = connection;
                    command.CommandText = @"UPDATE Images SET
NameFile = @FileName, Title = @Title, ImageData = @ImageData
WHERE Id = @Id";
                    command.Parameters.Add("@Id", SqlDbType.Int);
                    command.Parameters.Add("@FileName",
SqlDbType.NVarChar, 50);
                    command.Parameters.Add("@Title",
SqlDbType.NVarChar, 50);
                    command.Parameters.Add("@ImageData",
SqlDbType.Image, 1000000);
```

```
                    //Передаем данные в команду через параметры
                    ImageConverter converter = new ImageConverter();
                    command.Parameters["@Id"].Value = textBox2.Text;
                    command.Parameters["@FileName"].Value =
textBox3.Text;
                    command.Parameters["@Title"].Value =
textBox4.Text;
                    command.Parameters["@ImageData"].Value =
(byte[])converter.ConvertTo(pictureBox1.Image, typeof(byte[]));
                    command.ExecuteNonQuery();

                    //Обновляем список на форме
                    comboBox1.Items.Clear();
                    command.CommandText = "SELECT Title FROM
Images";
                    SqlDataReader reader = command.ExecuteReader();
                    while (reader.Read())
                    {
                        comboBox1.Items.Add(reader.GetString(0));
                    }
                    reader.Close();

                    //Подтверждаем транзакцию
                    transaction.Commit();
                }
                catch (Exception exception)
                {
                    MessageBox.Show("Не получилось сделать
одновременно 2 транзакции. " + exception);
                    transaction.Rollback();
                    comboBox1.Items.AddRange(list);
                }
                comboBox1.SelectedIndex = id;
            }
        }
    }
```

23. Ну і на останок додамо кнопку зберігання на жорсткому диску файлу, що міститься у PictureBox:

```
    private void button10_Click(object sender, EventArgs e)
    {
        SaveFileDialog saveFileDialog = new SaveFileDialog();
    //Создание диалогового окна для выбора имени сохраняемого файла
        saveFileDialog.Filter =
    "JPEG(*.jpg;*.jpeg;*.jpe;*.jfif)|*.jpg;*.jpeg;*.jpe;*.jfif|PNG(*
    .png)|*.png|GIF(*.gif)|*.gif|BMP(*.bmp)|*.bmp|All files
    (*.*)|*.*";
        if (saveFileDialog.ShowDialog() == DialogResult.OK) //Если в
    окне была нажата кнопка "OK"
        {
            try
            {
```

```
                ImageConverter converter = new ImageConverter();
                using (System.IO.FileStream fs = new
System.IO.FileStream(saveFileDialog.FileName,
FileMode.OpenOrCreate))
                {
                    byte[] mas =
(byte[])converter.ConvertTo(pictureBox1.Image, typeof(byte[]));
                    fs.Write(mas, 0, mas.Length);
                }
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось сохранить файл на
диск. " + exception);
            }
        }
    }
```

24. Кнопку що поверне нам якесь значення з запиту:

```
    private void button11_Click(object sender, EventArgs e)
    {
        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                string sql = "SELECT COUNT(*) FROM Images";
                SqlCommand command = new SqlCommand(sql,
connection);
                MessageBox.Show($"В БД есть
{command.ExecuteScalar()} картинок");
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось прочитать вернуть
значение. " + exception.Message);
            }
        }
    }
```

25. Та кнопку що викличе збережену процедуру.

```
    private void button12_Click(object sender, EventArgs e)
    {
        string sqlExpression = "AddImage"; //Имя хранимой процедуры
        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                SqlCommand command = new SqlCommand(sqlExpression,
connection);
```

```
                //Указываем, что команда представляет хранимую
процедуру
                command.CommandType =
System.Data.CommandType.StoredProcedure;
                //Задаём параметры
                command.Parameters.Add(new SqlParameter("@NameFile",
textBox3.Text));
                command.Parameters.Add(new SqlParameter("@Title",
textBox4.Text));
                ImageConverter converter = new ImageConverter();
                command.Parameters.Add(new
SqlParameter("@ImageData",
(byte[])converter.ConvertTo(pictureBox1.Image,
typeof(byte[])));

                var result = command.ExecuteScalar();
                //Если нам не надо возвращать id
                //var result = command.ExecuteNonQuery();
                Console.WriteLine("Id добавленного объекта: {0}",
result);
            }
            catch (Exception exception)
            {
                MessageBox.Show("Не получилось запустить процедуру
добавления изображения. " + exception.Message);
            }
        }
    }
```

26. Ну і власне створимо цю процедуру. Використання збережених процедур є більш безпечним та як правило дає кращу швидкодію.

```
CREATE PROCEDURE AddImage
    @NameFile varchar(50),
    @Title varchar(50),
    @ImageData varbinary(max)
AS
    INSERT INTO Images (NameFile, Title, ImageData)
    VALUES (@NameFile, @Title, @ImageData)
    SELECT SCOPE_IDENTITY()
GO
```

27. Отже ми отримали простий учбовий додаток що дозволяє підключитись до бази даних за допомогою більш складного методу (через введення логіну і паролю користувача) та вільно працювати з зображеннями що зберігаються у базі. Можна їх переглядати, завантажувати, змінювати, видаляти. Розглянуті основні можливості. Для більшої наглядності відразу оброблено усі помилки. Використання DataSet та SqlDataAdapter, що будуть розглянуті далі, звісно зручніше, проте треба було розглянути і можливість працювати з постачальником даних напряму.
28. <u>У якості самостійної роботи:</u>
    28.1. У існуючому рішенні створити новий проект і у ньому виконати підключення до бази даних що була створена у минулих комп’ютерних практикумах.
    28.2. Виконати завдання згідно з варіантом (додаток 1).

Додаток 1

Варіанти виконання самостійної частини роботи:

Варіант 1: Замовлення у книжковому магазині.

1. За допомогою засобів постачальника даних створити форму заповнення таблиці «Книга» даними. Усі поля таблиці задати полями для вводу, усі посилання на інші таблиці випадаючими списками що містять імена та назви (жанрів, видавництва, тощо), а не номера.
2. Передбачити додавання нових записів, їх зміну та видалення. Передбачити перегляд записів таблиці.
3. Бодай десь використати транзакції.
4. Передбачити отримання інформації про кількість замовлень книг по дням та жанрам якщо замовлень було більше 1 (запит у будь-якій зручній формі).
5. Додати виклик збереженої процедури, будь-якої, на власний вибір.

Варіант 2: Ресторан.

1. За допомогою засобів постачальника даних створити форму заповнення таблиці «Склад страви» даними. Усі поля таблиці задати полями для вводу, усі посилання на інші таблиці випадаючими списками що містять імена та назви (продуктів з їх ціною, назву страви), а не номера.
2. Передбачити додавання нових записів, їх зміну та видалення. Передбачити перегляд записів таблиці.
3. Бодай десь використати транзакції.
4. Передбачити отримання інформації про кількість рахунків по стравам отриманих у день якщо їх було більше двох (запит у будь-якій зручній формі).
5. Додати виклик збереженої процедури, будь-якої, на власний вибір.

Варіант 3: Автосервіс.

1. За допомогою засобів постачальника даних створити форму заповнення таблиці «Сервіс» даними. Усі поля таблиці задати полями для вводу, усі посилання на інші таблиці випадаючими списками що містять імена та назви (автомобілів, ФІО співробітників, виду роботи), а не номера.
2. Передбачити додавання нових записів, їх зміну та видалення. Передбачити перегляд записів таблиці.
3. Бодай десь використати транзакції.
4. Передбачити отримання інформації про кількість сервісних робіт прийнятих в день кожним співробітником якщо їх біло більше одної (запит у будь-якій зручній формі).
5. Додати виклик збереженої процедури, будь-якої, на власний вибір.

Варіант 4: Музична крамниця.

1. За допомогою засобів постачальника даних створити форму заповнення таблиці «Інструмент» даними. Усі поля таблиці задати полями для вводу, усі посилання на

інші таблиці випадаючими списками що містять імена та назви (марок, класів, тощо), а не номера.
2. Передбачити додавання нових записів, їх зміну та видалення. Передбачити перегляд записів таблиці.
3. Бодай десь використати транзакції.
4. Передбачити отримання інформації про кількість чеків отриманих в день по окремим інструментам якщо покупок було більше одної (запит у будь-якій зручній формі).
5. Додати виклик збереженої процедури, будь-якої, на власний вибір.

Варіант 5: Поліклініка.

1. За допомогою засобів постачальника даних створити форму заповнення таблиці «Прийом» даними. Усі поля таблиці задати полями для вводу, усі посилання на інші таблиці випадаючими списками що містять імена та назви (ФІО пацієнтів, співробітників, назви діагнозів), а не номера.
2. Передбачити додавання нових записів, їх зміну та видалення. Передбачити перегляд записів таблиці.
3. Бодай десь використати транзакції.
4. Передбачити отримання інформації про кількість прийомів в день по співробітниках якщо прийомів було більше двох (запит у будь-якій зручній формі).
5. Додати виклик збереженої процедури, будь-якої, на власний вибір.