

Лабораторная работа № 13

Тема: Разработка приложения для модификации данных.

Программное обеспечение: MS Visual Studio 2015

Цель: научиться создавать приложение, позволяющее добавлять, изменять и удалять строки в таблице базы данных.

Время на выполнение: 2 академических часа.

Задание: Разработать приложение для добавления данных в таблицу «Абитуриенты» базы данных «Kollege».

Для обновления данных используется метод **ExecuteNonQuery**, который выполняет запрос, не создавая объект **SqlDataReader** для приема его результатов. Значения возвращаемых параметров и параметров вывода становятся доступными по завершении вызова метода **ExecuteNonQuery**. Метод **ExecuteNonQuery** возвращает целое число, соответствующее количеству записей, затронутых запросом.

Порядок выполнения:

1. Откройте проект, созданный в предыдущей лабораторной работе (в данной работе будет рассматриваться случай с использованием элемента управления **TabControl**).
2. На главной форме разместите элемент управления **ToolStrip**, добавьте на нем 3 элемента **Button**, как показано на рисунке 13.1.

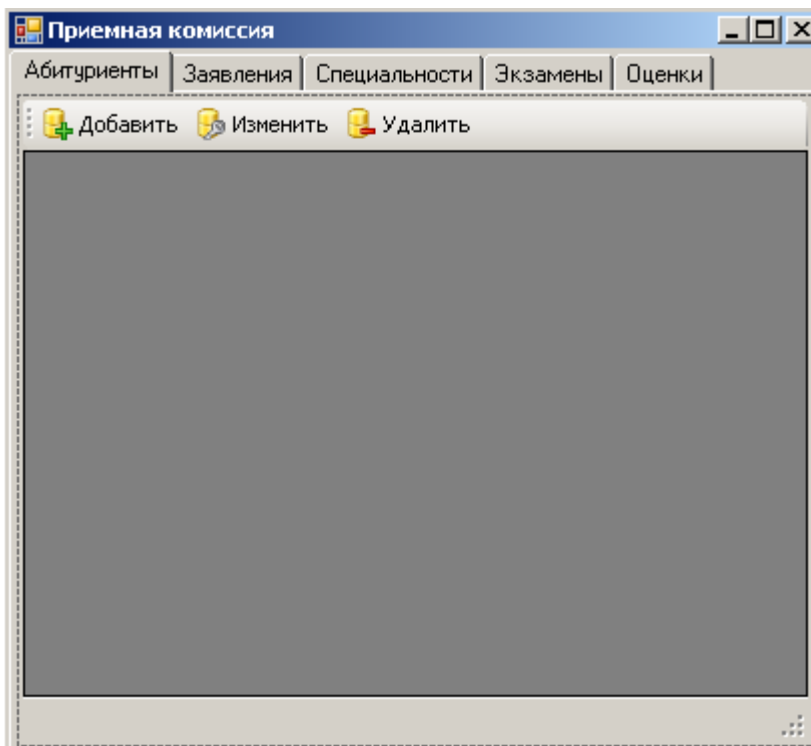


Рисунок 13.1 — Дизайн формы приложения

3. Для добавления и модификации записей необходимо добавить новую форму. Добавьте новую форму с помощью меню **Проект –> Добавить форму Windows**.
4. На новой форме разместите элементы управления, как показано на рисунке 13.2.

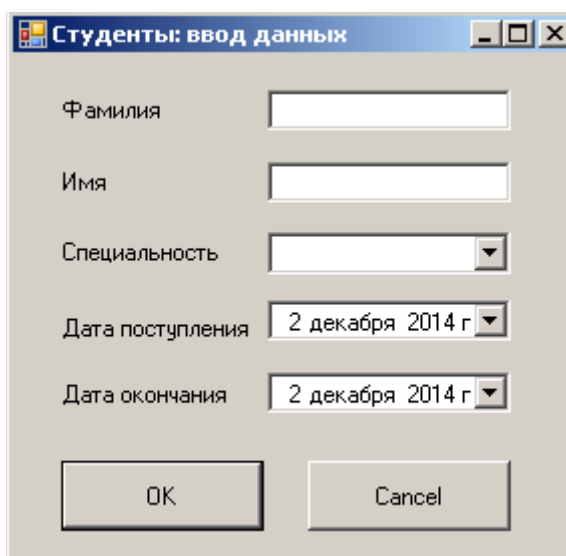


Рисунок 13.2 — Форма для ввода и модификации данных

При создании экземпляра DataTable столбцам таблицы присваиваются имена полей из объекта SqlDataReader.

5. В конструкторе формы необходимо задать два параметра: модифицируемая запись из базы данных, передаваемая как DataRow, и переменная типа bool, указывающая на действие — обновление или создание новой записи. Для этого добавьте код, представленный ниже.

Код класса формы представлен ниже:

```
using System.Data.SqlClient;
namespace Приемная_комиссия
{
    public partial class Abiturient : Form
    {
        private DataRow row;
        private bool isUpdate;

        public Abiturient(DataRow inputRow, bool update)
        {
            InitializeComponent();
            row = inputRow;
            isUpdate = update;
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            if (isUpdate)
            {
                this.Text = "Изменение данных об абитуриенте";
                textBox1.Text = row["Фамилия"].ToString();
                textBox2.Text = row["Имя"].ToString();
                textBox3.Text = row["Отчество"].ToString();
                comboBox1.Text = row["Статус"];
                textBox4.Text = row["Город"].ToString();
            }
        }
    }
}
```

```

    }

    public DataRow Row
    {
        get {return row;}
        set { row=value;}
    }

    private void button1_Click(object sender, EventArgs e)
    {
        row["Фамилия"] = textBox1.Text;
        row["Имя"] = textBox2.Text;
        row["Отчество"] = textBox3.Text;
        row["Статус"] = comboBox1.Text;
        row["Город"] = textBox3.Text;
    }

```

В приведенном примере необходимо заменить названия параметров в соответствии с таблицей Абитуриенты (*FirstName* → *Фамилия* и т.д.).

- На главной форме загрузку данных в DataGridView выделите в отдельный метод:

```

private void LoadData()
{
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = Properties.Settings.Default.ConnectionString;
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = "Select * from абитуриенты";
    conn.Open();
    SqlDataReader rdr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    for (int i = 0; i < rdr.FieldCount; i++)
    {
        dt.Columns.Add(new DataColumn(rdr.GetName(i), rdr.GetFieldType(i)));
    }
    while (rdr.Read())
    {
        DataRow row = dt.NewRow();
        for (int i = 0; i < rdr.FieldCount; i++)
        {
            row[i] = rdr.GetValue(i);
        }
        dt.Rows.Add(row);
    }
    conn.Close();
    dataGridView1.DataSource = dt;
    lbRecordCount.Text = "Количество записей: " + String.Format("{0}",
dt.Rows.Count);
}

```

- Тогда обработчик события загрузки формы примет вид:

```

private void Form1_Load(object sender, EventArgs e)
{
    LoadData();
}

```

- На главной форме создайте обработчик события на кнопку **Добавить**:

В приведенном ниже коде необходимо заменить названия таблиц и параметров в соответствии с хранимой процедурой *Абитуриент Insert*.

```
private void sb_insert_Click(object sender, EventArgs e)
{
    Form2 form = new Form2(((DataTable)dataGridView1.DataSource).NewRow(), false);
    if (form.ShowDialog() == DialogResult.OK)
    {
        SqlConnection conn = new
SqlConnection(Kollege.Properties.Settings.Default.ConnectionString);
        SqlCommand com = new SqlCommand();
        com.Connection = conn;
        com.CommandType = CommandType.StoredProcedure;
        com.CommandText = "Student_Insert";
        com.Parameters.Add(new SqlParameter("@firstName", SqlDbType.NVarChar));
        com.Parameters["@firstName"].Value = form2.Row["FirstName"];
        com.Parameters.Add(new SqlParameter("@lastName", SqlDbType.NVarChar));
        com.Parameters["@lastName"].Value = form2.Row["LastName"];
        com.Parameters.Add(new SqlParameter("@facultyId", SqlDbType.Int));
        com.Parameters["@facultyId"].Value = form2.Row["FacultyId"];
        com.Parameters.Add(new SqlParameter("@datefrom", SqlDbType.Date));
        com.Parameters["@datefrom"].Value = form2.Row["DateFrom"];
        conn.Open();
        if (com.ExecuteNonQuery() > 0)
        {
            MessageBox.Show("Студент успешно добавлен");
        }
        conn.Close();
        LoadData();
        dataGridView1.Rows[dataGridView1.Rows.Count-1].Selected=true;
    }
}
```

9. Для кнопок **OK** и **Cancel** задайте в свойстве **DialogResult** соответствующее значение OK, Cancel.
10. Запустите приложение, добавьте нового студента, измените данные любой строки (см. рисунки 13.3 — 13.4).

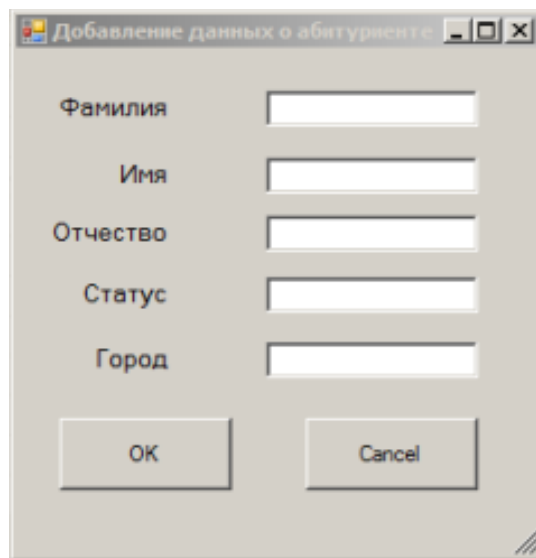


Рисунок 13.3 — Форма ввода и обновления в режиме отладки

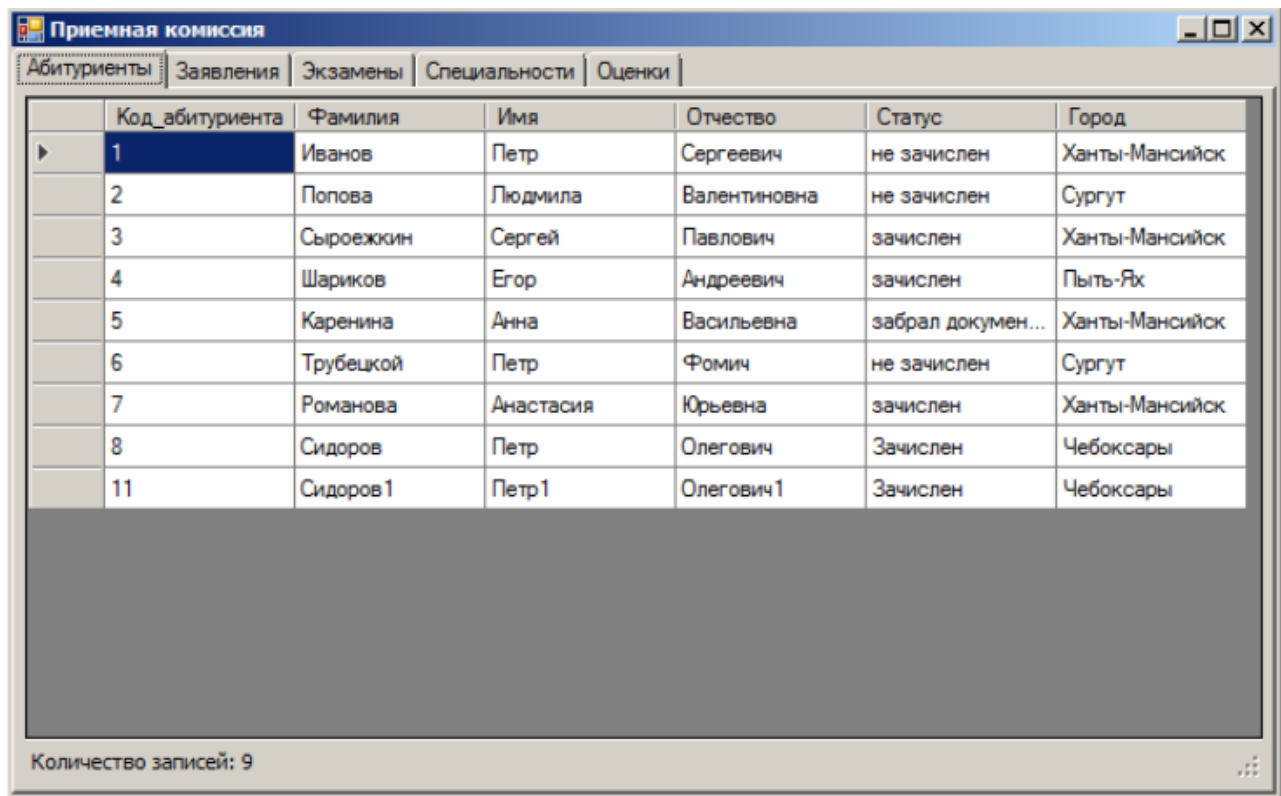


Рисунок 13.4 — Главная форма в режиме отладки

Задание 3: Разработать приложение для обновления данных в таблице «Абитуриенты» базы данных «Kollege».

Порядок выполнения:

1. Измените хранимую процедуру Абитуриент_Update следующим образом:

```
ALTER procedure [dbo].[Абитуриент_Update]
@Код_абитуриента int,
@Фамилия varchar(30),
@Имя varchar(20),
@Отчество varchar(20),
@Статус varchar(20),
@Город varchar(30)
as
begin

UPDATE Абитуриенты
SET Фамилия=@Фамилия, Имя=@Имя, Отчество=@Отчество,
Статус=@Статус, Город=@Город
WHERE Код_абитуриента=@Код_абитуриента
end
```

2. На главной форме создайте обработчик события на кнопку **Обновить**:

В приведенном ниже коде необходимо заменить названия таблиц и параметров в соответствии с Вашей предметной областью (Students → Абитуриенты, StudentId → Код_абитуриента и т.д.).

```

private void toolStripButton2_Click(object sender, EventArgs e)
{
    int id = (int)dataGridView1.SelectedRows[0].Cells["StudentId"].Value;
    DataTable dt = (DataTable)dataGridView1.DataSource;
    DataRow[] rows = dt.Select(String.Format("StudentId={0}", id));
    Form2 form2 = new Form2(rows[0], true);
    if (form2.ShowDialog() == DialogResult.OK)
    {
        SqlConnection conn = new SqlConnection(Kollege.Properties.Settings.Default.ConnectionString);
        SqlCommand com = new SqlCommand();
        com.Connection = conn;
        com.CommandType = CommandType.StoredProcedure;
        com.CommandText = "Student_Update";
        com.Parameters.Add(new SqlParameter("@studentId", SqlDbType.NVarChar));
        com.Parameters["@studentId"].Value = form2.Row["StudentId"];
        com.Parameters.Add(new SqlParameter("@firstName", SqlDbType.NVarChar));
        com.Parameters["@firstName"].Value = form2.Row["FirstName"];
        com.Parameters.Add(new SqlParameter("@lastName", SqlDbType.NVarChar));
        com.Parameters["@lastName"].Value = form2.Row["LastName"];
        com.Parameters.Add(new SqlParameter("@facultyId", SqlDbType.Int));
        com.Parameters["@facultyId"].Value = form2.Row["FacultyId"];
        com.Parameters.Add(new SqlParameter("@datefrom", SqlDbType.Date));
        com.Parameters["@datefrom"].Value = form2.Row["DateFrom"];
        com.Parameters.Add(new SqlParameter("@dateto", SqlDbType.Date));
        com.Parameters["@dateto"].Value = form2.Row["DateTo"];
        conn.Open();
        com.ExecuteNonQuery();
        conn.Close();
        LoadData();
    }
}

```

3. Протестируйте приложение, выполнив отладку приложения, для изменения строки сначала необходимо выделить строку в таблице, и затем щелкнуть по кнопке **Изменить**.

Задание 4: Разработать приложение для удаления данных в таблице «Абитуриенты» базы данных «Kollege».

Порядок выполнения:

На главной форме создайте обработчик события на кнопку **Удалить**:

В приведенном ниже коде необходимо заменить название хранимой процедуры и параметров в соответствии с Вашей предметной областью (StudentId → Код_абитуриента).

```

private void toolStripButton3_Click(object sender, EventArgs e)
{
    int id = (int)dataGridView1.SelectedRows[0].Cells["StudentId"].Value;
    if (MessageBox.Show("Удалить выделенную запись", "Удаление", MessageBoxButtons.YesNo) == DialogResult.OK)
    {
        SqlConnection conn = new SqlConnection(Kollege.Properties.Settings.Default.ConString);
        SqlCommand com = new SqlCommand();
        com.Connection = conn;
        com.CommandType = CommandType.StoredProcedure;
        com.CommandText = "Student_Delete";
        com.Parameters.Add(new SqlParameter("@studentId", SqlDbType.Int));
        com.Parameters["@studentId"].Value = id;
        conn.Open();
        com.ExecuteNonQuery();
        conn.Close();
        LoadData();
    }
}

```

4. Протестируйте приложение, выполнив отладку приложения, для удаления строки сначала необходимо выделить строку в таблице, и затем щелкнуть по кнопке **Удалить**.