Практическая работа №26

Тема: Создание приложения с БД, запросы к БД

Цель работы: изучение механизма работы с базами данных в Visual Studio

Задачи:

– изучение механизма связывания источника данных с элементом управления dataGridView;

– формирование компетенций при построении приложений по обработке информационных массивов с применением элементов управления: DataSet, DataGridView, BindingSourse, BindingNavigator

– Создание пользовательского интерфейса и модулей для обработки кнопок ввода, редактирования и поиска данных

Материально-техническое обеспечение:

Место проведения: Компьютерный класс.

Время на выполнение работы: 2 часа.

Оборудование: ПК

Средства обучения: операционная система, текстовый процессор MS Word, программные средства определенного вида Visual Studio

Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

Краткие теоретические сведения:

DataAdapter является средством связи между **Dataset** и **Connection**. С **Dataset** он связан посредством команд:

Fill - загрузить данные,

Update - записать изменения.

В момент записи создаётся список ошибок (Error Collection). DataAdapter работает с Connection с помощью реляционного метода. Команды взаимодействия с БД (Command) можно создавать как вручную, так и автоматически.

Вывод таблицы базы данных Microsoft Access в компоненте dataGridView

Рассмотрим пример1:

Пусть имеется база данных, созданная в приложении Microsoft Access, сохраненная в файле с именем “mydb.mdb”. Файл размещается на диске по следующему пути:

C:\Programs\C_Sharp\WindowsFormsApplication1\mydb.mdb

База данных имеет несколько таблиц, одна из которых имеет название “Order”.

Задача состоит в том, чтобы с помощью средств языка C# осуществить подключение к базе данных и вывести таблицу с именем «Order» на форму.

Приложение реализовать как Windows Forms Application.

Общий вид таблиц и связей между ними изображен на рисунке 1.

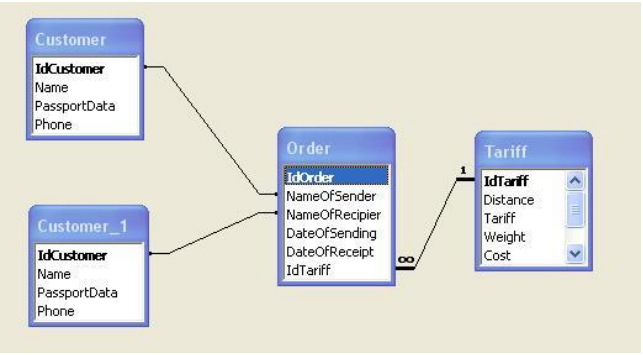


Рис. 1. Связи между таблицами базы данных

Ключ к выполнению данной задачи

1. Создание приложения. Загружаем MS Visual Studio.
2. Подключение к базе данных. Чтение строки подключения Connection String.

Осуществим подключение базы данных MS Access к нашему приложению.

В итоге получаем строку подключения к базе данных ***Connection String***. Эта строка в дальнейшем будет использована в нашем приложении.

Чтобы получить корректную строку подключения к базе данных, нужно выделить базу данных в панели Server Explorer (mydb.mdb) и в окне “Properties” прочитать (скопировать) значение свойства “Connection String” (рис. 2, красное выделение). Следует учесть, что слеш ‘\’ в строке на C# нужно заменить на ‘\\’ (два слеша) согласно синтаксису языка.

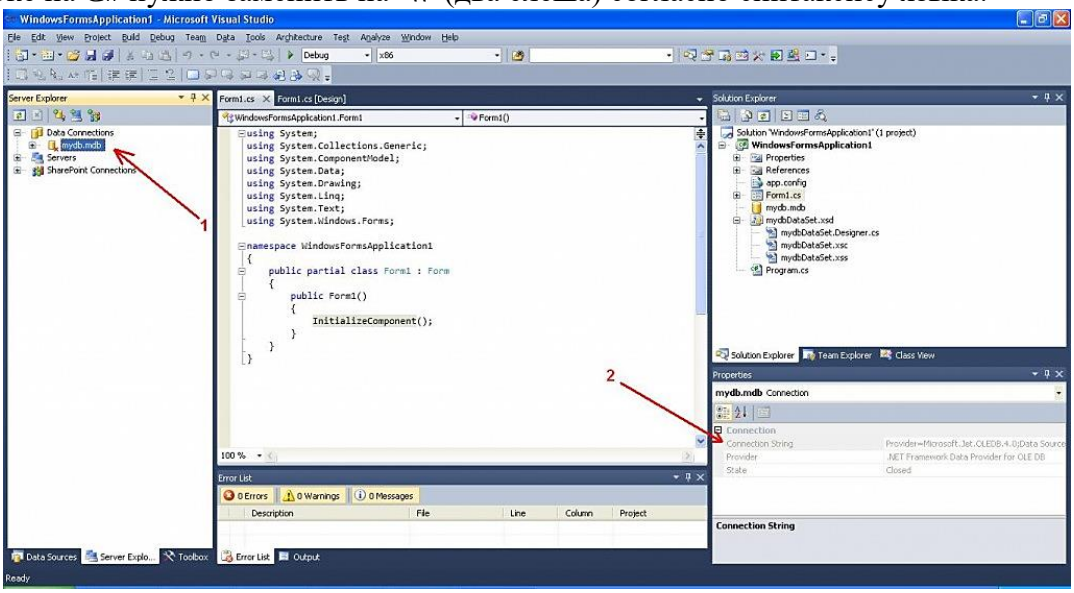


Рис. 2. Чтение свойства Connection String

3. Размещение компонента типа **dataGridView**.

Выносим на форму компонент dataGridView (рис. 3), представляющий компонент-таблицу, в которой будет выведена наша таблица “Order” из базы данных. Получаем объект-переменную под названием dataGridView.

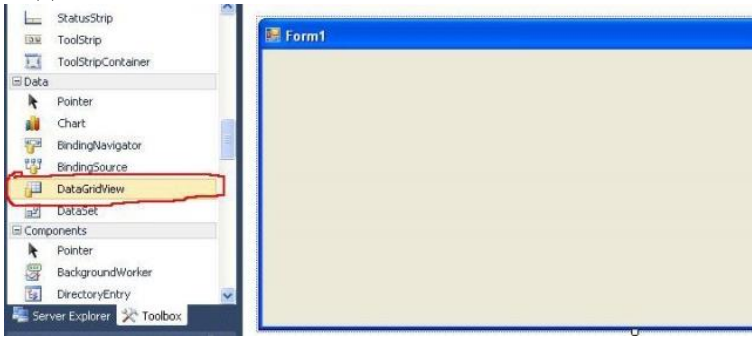


Рис. 3. Компонент DataGridView на панели Toolbox

Размещение компонента dataGridView на форме изображено на рисунке 4.

Рис. 4. Компонент dataGridView на главной форме приложения

4. Изменение программного кода.

*4.1. Добавление переменных **SQL-запроса** и строки подключения к базе данных.*

Активируем текст модуля Form1.cs (главная форма) с помощью Solution Explorer. В программный код формы вводим дополнительные переменные CmdText и ConnString.

Переменная CmdText будет содержать строку SQL-запроса для вывода всех записей таблицы “Order”. Переменная ConnString представляет собой строку подключения к базе данных (см. п. 2).

Общий вид программного кода класса формы следующий:

```
public partial class Form1 : Form
{
 public string CmdText = "SELECT * FROM [Order]";
 public string ConnString = "Provider=Microsoft.Jet.OLEDB.4.0;
DataSource=C:\\Programs\\C_Sharp\\WindowsFormsApplication1\\mydb.mdb";
  public Form1()
  {
```

```
    InitializeComponent();
  }
}
```

4.2. Подключение пространства имен ***OleDb****.*

В Microsoft Visual Studio взаимодействие с файлом данных Microsoft Access осуществляется с помощью поставщика данных OLE DB или ODBC. Поставщик данных OLE DB обеспечивает доступ к данным, находящимся в любом хранилище данных, если оно поддерживает классический протокол OLE DB на основе технологии COM. Этот поставщик состоит из типов, которые определены в пространстве имен **System.Data.OleDb**.

В последующих шагах мы будем использовать методы из этого пространства имен. Поэтому, вначале файла Form1.cs после строки

using System.Windows.Forms;

нужно добавить строку подключения пространства имен OleDb:

using System.Data.OleDb;

4.3. Создание объекта типа ***OleDbDataAdapter****.*

В конструкторе формы после вызова

InitializeComponent();

Добавляем строку создания объекта типа OleDbDataAdapter:

OleDbDataAdapter dA = new OleDbDataAdapter(CmdText, ConnString);

Объект типа OleDbDataAdapter организовывает пересылку наборов данных с вызываемым процессом. Адаптеры данных содержат набор из четырех внутренних объектов команд. Это команды чтения, вставки, изменения и удаления информации. Как видно из программного кода, конструктор объекта получает входящими параметрами строку запроса на языке SQL (переменная CmdText) и строку подключения к базе данных (переменная ConnString). Таким образом, после выполнения данного кода, объект адаптера уже связан с нашей базой данных.

4.4. Создание объекта набора данных ***DataSet****.*

После создания адаптера данных (OleDbDataAdapter) создаем объект типа DataSet (набор данных):

DataSet ds = new DataSet();

Набор данных представляет что-то вроде промежуточного буфера для данных, которые могут отображаться. Набор данных представляет удобный механизм чтения и обновления данных а также инкапсулирует множество таблиц и связей между ними.

*4.5. Заполнение таблицы "****Order****" на основе* ***SQL-запроса****.*

Следующая команда – это заполнение набора данных (переменная ds) значениями записей из базы данных на основе SQL-запроса, содержащегося в адаптере данных dA с помощью метода Fill():

dA.Fill(ds, "[Order]");

4.6. Визуализация данных в ***dataGridView1****.*

На данный момент данные из таблицы "Order" считаны в объекте ds (типа DataSet), представляющем собой набор данных.

Для их отображения необходимо чтобы свойство DataSource компонента dataGridView1 ссылалось на первую таблицу (в нашем случае одна таблица) набора данных ds. Программный код этой операции имеет следующую реализацию:

dataGridView1.DataSource = ds.Tables[0].DefaultView;

После этого данные из таблицы "Order" отобразятся на форме (рис. 5).

5. Весь программный код.

Общий листинг класса главной формы приложения имеет следующий вид:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
using System.Data.OleDb;
namespace WindowsFormsApplication1
{
  public partial class Form1 : Form
  {
    public string CmdText = "SELECT * FROM [Order]";
    public string ConnString =
"Provider=Microsoft.Jet.OLEDB.4.0;DataSource=C:\\Programs\\C_Sharp\\WindowsFormsApplic
ation1\\mydb.mdb";
    public Form1()
    {
      InitializeComponent();
      OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CmdText, ConnString);
      // создаем объект DataSet
      DataSet ds = new DataSet();
      // заполняем таблицу Order
      // данными из базы данных
      dataAdapter.Fill(ds, "[Order]");
      dataGridView1.DataSource = ds.Tables[0].DefaultView;
    }
  }
}
```

Результат выполнения приложения изображен на рис. 5.

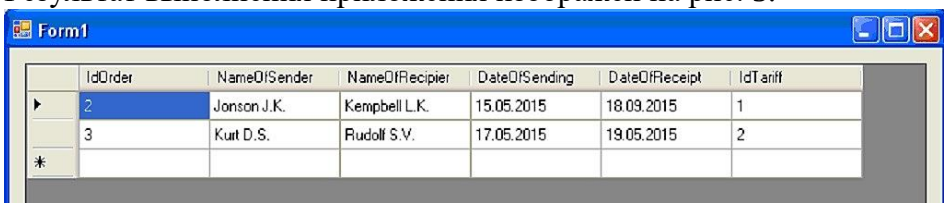


Рис. 5. Результат выполнения приложения

6. Схема взаимодействия.

Общая схема взаимодействия между объектами изображена на рис. 6.

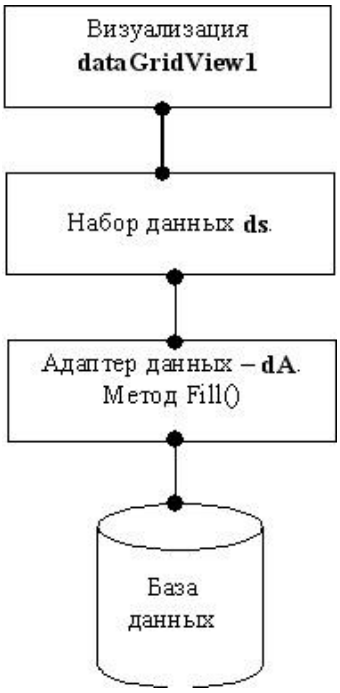


Рис. 6. Схема взаимодействия между объектами для доступа к базе данных

Таким образом, можно выводить на форму любую таблицу базы данных. Условия выведения данных из базы данных задаются в строке SQL-запроса в переменной CmdText.

Например, если в CmdText задать следующую строку:

CmdText = "SELECT * FROM [Order] WHERE [NameOfSender] LIKE 'I%'";

то в результате из базы данных будут извлекаться записи, начинающиеся с символа '**I**'.

Пример2. Рассмотрим выполнение запросов SELECT, INSERT, UPDATE и DELETE к базе данных Microsoft Access из программы на языке C#, на примере проекта Windows Forms в Visual Studio.

в Microsoft Access база данных Workers (рабочие) с одной таблицей Worker (рабочий). Таблица содержит следующие столбцы:

- w_id (идентификатор записи) — тип данных Счетчик;
- w_name (имя) — тип данных Короткий текст;
- w_position (должность) — тип данных Короткий текст;
- w_salary (зарплата) — тип данных Числовой.

Пример написания кода на C# для выполнения запросов к MS Access

Добавим в класс формы строковое поле string — строку подключения к БД. В строке подключения в значении параметра Data Source указывается путь к файлу с базой данных.

```
// строка подключения к MS Access
// вариант 1
public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Workers.mdb;";
// вариант 2
//public static string connectString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Workers.mdb;";
```

Далее добавим поле класса типа OleDbConnection. Экземпляр данного класса понадобится для установления соединения с базой данных MS Access.

```
// поле - ссылка на экземпляр класса OleDbConnection для соединения с БД
private OleDbConnection myConnection;
```

В конструкторе класса создадим экземпляр OleDbConnection и установим соединение с БД. Т.е. при запуске программы будет устанавливаться соединение с Access.

```
// конструктор класса формы
public Form1()
{
    InitializeComponent();

    // создаем экземпляр класса OleDbConnection
    myConnection = new OleDbConnection(connectString);

    // открываем соединение с БД
    myConnection.Open();
}
```

В обработчике события закрытия формы добавим код, закрывающий соединение с базой данных. Таким образом при выходе из программы соединение с БД будет закрываться.

```
// обработчик события закрытия формы
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // заркываем соединение с БД
    myConnection.Close();
}
```

Напишем в обработчик события нажатия кнопки SELECT1 код, выполняющий запрос выборки данных одного столбца одной строки. Вывод данных производится в TextBox.

```
// обработчик события нажатия кнопки SELECT1
private void selectButton1_Click(object sender, EventArgs e)
{
    // текст запроса
    string query = "SELECT w_name FROM Worker WHERE w_id = 1";

    // создаем объект OleDbCommand для выполнения запроса к БД MS Access
    OleDbCommand command = new OleDbCommand(query, myConnection);

    // выполняем запрос и выводим результат в textBox1
    textBox1.Text = command.ExecuteScalar().ToString();
}
```

При нажатии кнопки SELECT2 будет выполняться запрос на получение многострочных данных из нескольких столбцов (по сути таблицы). Данные выводятся в ListBox.

```
// обработчик события нажатия кнопки SELECT2
private void selectButton2_Click(object sender, EventArgs e)
{
    // текст запроса
    string query = "SELECT w_name, w_position, w_salary FROM Worker ORDER BY w_salary";

    // создаем объект OleDbCommand для выполнения запроса к БД MS Access
    OleDbCommand command = new OleDbCommand(query, myConnection);

    // получаем объект OleDbDataReader для чтения табличного результата запроса SELECT
    OleDbDataReader reader = command.ExecuteReader();

    // очищаем listBox1
    listBox1.Items.Clear();

    // в цикле построчно читаем ответ от БД
    while(reader.Read())
    {
        // выводим данные столбцов текущей строки в listBox1
        listBox1.Items.Add(reader[0].ToString() + " " + reader[1].ToString() + " " + reader[2].ToString() + " ");
    }

    // закрываем OleDbDataReader
    reader.Close();
}
```

Далее запрос вставки данных (INSERT) в MS Access на C#:

```
// обработчик события нажатия кнопки INSERT
private void insertButton_Click(object sender, EventArgs e)
{
    // текст запроса
    string query = "INSERT INTO Worker (w_name, w_position, w_salary) VALUES ('Михаил', 'Водитель', 20000)";

    // создаем объект OleDbCommand для выполнения запроса к БД MS Access
    OleDbCommand command = new OleDbCommand(query, myConnection);

    // выполняем запрос к MS Access
    command.ExecuteNonQuery();
}
```

Метод ExecuteNonQuery() класса OleDbCommand выполняет запрос и возвращает целое число типа int — количество строк затронутых выполняемым запросом. Данный метод подходит для запросов вставки INSERT, обновления UPDATE и удаления DELETE. Т.е. для тех, которые не возвращают данные.

Запрос обновления данных UPDATE:

```
// обработчик события нажатия кнопки UPDATE
private void updateButton_Click(object sender, EventArgs e)
{
    // текст запроса
    string query = "UPDATE Worker SET w_salary = 123456 WHERE w_id = 3";

    // создаем объект OleDbCommand для выполнения запроса к БД MS Access
    OleDbCommand command = new OleDbCommand(query, myConnection);

    // выполняем запрос к MS Access
    command.ExecuteNonQuery();
}
```

Пример SQL-запроса удаления данных (DELETE) из БД Access с помощью языка C#:

```
// обработчик события нажатия кнопки DELETE
private void deleteButton_Click(object sender, EventArgs e)
{
    // текст запроса
    string query = "DELETE FROM Worker WHERE w_id < 3";

    // создаем объект OleDbCommand для выполнения запроса к БД MS Access
    OleDbCommand command = new OleDbCommand(query, myConnection);

    // выполняем запрос к MS Access
    command.ExecuteNonQuery();
}
```

Пример создания объекта подключения к БД SQL и запроса на выборку данных из 4-х таблиц:

```
// создать объект подключения
SqlConnection conn = new SqlConnection(Connection.con);
try
{
    conn.Open();//открыть подключение
    string query = "SELECT [dbo].[Student].[FIO], [dbo].[Group].[Name_group],"+
        " [dbo].[Subject].[Name_subject], [dbo].[Ball].[ball], [dbo].[Ball].[vid_control]"+
        "FROM [dbo].[Student], [dbo].[Group], [dbo].[Subject], [dbo].[Ball] " +
        "WHERE ([dbo].[Student].[id_group]=[dbo].[Group].[Id_group]) AND " +
        "([dbo].[Ball].[id_student]=[dbo].[Student].[Id_student]) AND " +
        "([dbo].[Ball].[id_subject]=[dbo].[Subject].[Id_subject])";

    SqlCommand cmd = new SqlCommand(query, conn);

    SqlDataReader reader = cmd.ExecuteReader();

    if (reader.HasRows == false)
    { MessageBox.Show("данные не найдены!"); }
    else
    {
        while (reader.Read())
        {
            dataGridView1.Rows.Add(reader[0], reader[1], reader[2], reader[3], reader[4]);
        }

    }
    reader.Close();
    conn.Close();//закрыть подключение
}
catch
{
    MessageBox.Show("Ошибка!");
}
```

Элемент управления dataGridView

В Microsoft Visual Studio элемент управления dataGridView разработан для использования в приложениях, созданных по шаблону Windows Forms Application. Данный элемент управления позволяет организовывать данные в виде таблицы. Данные могут быть получены из базы данных, коллекции, внутренних переменных — массивов или других объектов программы. После размещения на форме, система создает объект (переменную) с именем dataGridView1. С помощью этого имени можно программно оперировать методами и свойствами этого элемента управления. В DataGridView данные могут быть получены из базы данных, коллекции, внутренних структур данных (массивов, структур и т.д.).

Виды данных, которые могут быть представлены в ячейках dataGridView:

- dataGridViewButtonColumn. Ячейки представлены в виде кнопок типа Button;
- dataGridViewCheckBoxColumn. Ячейки представлены элементами управления типа CheckBox, которые позволяют выбирать несколько вариантов (опций) из набора предложенных;
- dataGridViewComboBoxColumn. Ячейки представлены элементами управления типа ComboBox, предназначенных для выбора одного из нескольких вариантов;
- dataGridViewImageColumn. Ячейки таблицы есть изображениями типа Image;
- dataGridViewLinkColumn. Ячейки таблицы представлены ссылками;
- dataGridViewTextBoxColumn. Этот вариант предлагается по умолчанию при добавлении (создании) нового столбца. В этом случае ячейки таблицы представлены в виде полей ввода. Это позволяет вводить данные в таблицу как в матрицу.

Добавление столбца программным путем

Добавить столбец в dataGridView можно:

- с помощью специального мастера;
- программным путем.

Столбцы в dataGridView организованы в виде коллекции Columns типа DataGridViewColumnCollection.Чтобы добавить столбец программным путем используется метод (команда) Add из коллекции Columns.

Метод Add имеет 2 варианта реализации:

int DataGridViewColumnCollection.Add(DataGridViewColumn dataGridViewColumn);

int DataGridViewColumnCollection.Add(string ColumnName, string HeaderText); где

- DataGridViewColumn – тип System.Windows.Forms.Column который добавляется;
- ColumnName – название, по которому будет осуществляться обращение к столбцу из других методов;
- HeaderText – текст, который будет отображаться в заголовке столбца.

Фрагмент кода, который добавляет два произвольных столбца следующий:

```
// Добавить столбец с именем column-1, заголовок столбца - "Header column - 1"
dataGridView1.Columns.Add("column-1", "Header column - 1");
// Добавить столбец с именем column-2
dataGridView1.Columns.Add("column-2", "Header column - 2");
```

В реальных программах название столбца и его заголовка получаются из других элементов управления, например TextBox.

Для вставки столбца используется метод Insert, который имеет следующее объявление

void DataGridViewColumnCollection.Insert(int columnIndex, DataGridViewColumn dataGridViewColumn);

Вызов этого метода из программного кода аналогичен методу Add.

Как программно реализовать удаление столбца? Методы Remove() и RemoveAt()

Чтобы удалить столбец используется один из двух методов из коллекции Columns:

- метод RemoveAt() – удаляет столбец по заданному индексу в коллекции;
- метод Remove() – удаляет столбец по его имени.

Общий вид метода RemoveAt():

void DataGridViewColumnCollection.RemoveAt(int index);

где index – заданный индекс в коллекции. Индексы нумеруются с 0.

void DataGridViewColumnCollection.Remove(string ColumnName);

где ColumnName – название столбца (но не название заголовка столбца), которое задается в методе Add()первым параметром. Столбцы в коллекции могут иметь одинаковые значения ColumnName. Если при вызове метода Remove(), столбца с именем ColumnName нет, то генерируется исключительная ситуация.

Фрагмент кода удаления столбца с помощью метода RemoveAt():

```
// удаление столбца в позиции index
```

```
int index; // номер столбца, который удаляется
int n; // текущее количество столбцов в dataGridView
// задать номер столбца, который удаляется index = 1;
// определить текущее количество столбцов в dataGridView
n = dataGridView1.Columns.Count;
// удаление
if ((n > 0) && (index >= 0) && (index < n))
{
dataGridView1.Columns.RemoveAt(index); label1.Text = "Столбец удален";
}
else
{
label1.Text = "Столбец не удален";
}
```

Программное добавление строки. Метод Add()

Добавлять строку можно одним из двух способов:

– путем непосредственного ввода с клавиатуры;
– программным путем.

Строки в DataGridView организованы в виде коллекции Rows типа dataGridViewRowCollection. Ниже приведен фрагмент метода, добавляющего 2 произвольные строки в таблицу

```
// Добавить строки в таблицу
if (dataGridView1.Columns.Count <= 0)
{
label1.Text = "Строки не добавлены";
return;
}
dataGridView1.Rows.Add("Ivanov I.I.", 25, "New York");
dataGridView1.Rows.Add("Petrenko P.P.", 38, "Moscow"); l
abel1.Text = "Строки добавлены";
```

Программное удаление строки. Методы Remove() и RemoveAt()

Для удаления строки используется один из двух методов:

– метод RemoveAt() – удаляет строку по заданному индексу;
– метод Remove() – удаляет строку, которая есть входным параметром типа DataGridViewRow.

Фрагмент кода удаления строки имеет вид:

```
// Удалить строку
int nr, nc;
nc = dataGridView1.Columns.Count;    // количество столбцов
 nr = dataGridView1.RowCount;
if ((nc > 0) && (nr > 1))
{
dataGridView1.Rows.RemoveAt(0);                // удалить первую строку
label1.Text = "Строка удалена";
}
else
{
label1.Text = "Строка не удалена";
}
```

Задание текста заголовка в заданном столбце программным путем

Чтобы задать текст заголовка в заданном столбце используется свойство HeaderText. Фрагмент кода установки текста заголовка в столбце с индексом 0 имеет вид:

```
// задать текст в заголовке
int nc = dataGridView1.ColumnCount; if (nc > 0)
{
// задать новый текст заголовке первого столбца
dataGridView1.Columns[0].HeaderText = "Header - 1";
label1.Text = "Текст задан";
}
else
{
label1.Text = "Текст не задан";
```

```
}
```
Установка выравнивания заголовка в заданном столбце программным путем

Выравнивание заголовка в столбце задается с помощью
свойства HeaderCell.Style.Alignment.

Фрагмент кода установки выравнивания в заголовке столбца с индексом 0:

```
// выравнивание заголовка int nc;
nc = dataGridView1.ColumnCount;
if (nc > 0)
{
// задать выравнивание по центру (по горизонтали и по вертикалу)
dataGridView1.Columns[0].HeaderCell.Style.Alignment =DataGridViewContentAlignment.MiddleCenter;
label1.Text = "Выравнивание выполнено";
}
else
{
label1.Text = "Выравнивание не выполнено";
}
```
Установка шрифта заголовка в столбцах программным путем

Для установки шрифта в заголовках столбцов используется свойство ColumnHeadersDefaultCellStyle. В этом свойстве используется свойство Font.

Во фрагменте кода создается шрифт Arial, имеющий размер 12 и курсивное начертание.

```
// задать шрифт в заголовке int nc;
nc = dataGridView1.ColumnCount;
// создать шрифт "Arial", размер 12, начертание - "курсив"
Font F = new Font("Arial", 12, FontStyle.Italic);
if (nc > 0)
{
// установить шрифт заголовка
dataGridView1.ColumnHeadersDefaultCellStyle.Font = F;
label1.Text = "Шрифт задан";
}
else
{
label1.Text = "Шрифт не задан";
}
```
Установка цвета шрифта заголовков программным путем

Чтобы задать цвет шрифта заголовков программным путем нужно использовать свойство ColumnHeaderDefaultCellStyle. В этом свойстве есть свойства ForeColor и BackColor.

```
int nc;
nc = dataGridView1.ColumnCount;
if (nc > 0)
{
// создать системный шрифт
Font F = new Font("Arial", 14);
// задать цвет в заголовках столбцов
dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.Purple;
// задать шрифт
dataGridView1.Columns[0].DefaultCellStyle.Font = F;
label1.Text = "Цвет заголовка изменен";
}
else
{
label1.Text = "Цвет не изменен";
}
```
Установка размеров dataGridView1 программным путем

```
// задать размер dataGridView1
dataGridView1.Width = 600;
dataGridView1.Height = 150;
```

Установка ширины заданного столбца dataGridView1

```
// задать ширину столбца int nc;
nc = dataGridView1.ColumnCount;
if (nc > 0)
{
// задать ширину столбца с индексом 0
dataGridView1.Columns[0].Width = 70;
label1.Text = "Ширина столбца задана";
}
else
{
label1.Text = "Ширина столбца не задана";
}
```

Установка высоты заданной строки dataGridView1

```
// задать высоту строки int nc, nr;
nc = dataGridView1.ColumnCount;
nr = dataGridView1.RowCount;
if ((nc > 0) && (nr > 1))
{
dataGridView1.Rows[0].Height = 50;
label1.Text = "Высота строки задана";
}
else
{
label1.Text = "Высота строки не задана";
}
```

Установка выравнивания в заданном столбце и строке

```
// выравнивание в строках
int nc, nr;
nc = dataGridView1.ColumnCount;
nr = dataGridView1.RowCount;
if ((nc > 0)&&(nr>1))
{
// выравнивание для всех строк
dataGridView1.RowsDefaultCellStyle.Alignment =DataGridViewContentAlignment.BottomRight;
// выравнивание для строки с индексом 0
dataGridView1.Rows[0].DefaultCellStyle.Alignment=DataGridViewContentAlignment.MiddleCenter;
// выравнивание для столбца с индексом 0
dataGridView1.Columns[0].DefaultCellStyle.Alignment=DataGridViewContentAlignment.BottomLeft;
}
```

Установка шрифта, цвета символов и фона в первом столбце

Чтобы задать шрифт, цвет символов и фона в первом столбце, используется свойство DefaultCellStyleстолбца с индексом 0. В этом свойстве есть свойства Font, BackColor, ForeColor.
Ниже приведен фрагмент кода, который задает цвет шрифта, символов и фона в dataGridView1.

```
// шрифт и цвет в первом столбце
int nc, nr;
nc = dataGridView1.ColumnCount;
nr = dataGridView1.RowCount;
if ((nc > 0) && (nr > 1))
{
// создать шрифт
Font F = new Font("Times New Roman", 10, FontStyle.Bold);
// цвет символов и фона в первом столбце
dataGridView1.Columns[0].DefaultCellStyle.BackColor = Color.Red;
dataGridView1.Columns[0].DefaultCellStyle.ForeColor = Color.Blue;
// шрифт в первом столбце
dataGridView1.Columns[0].DefaultCellStyle.Font = F;
label1.Text = "Шрифт и цвет в 1-м столбце изменен";
}
else
{
label1.Text = "Шрифт не изменен";
}
```

Определение количества столбцов

```
// определить количество столбцов int n;
n = dataGridView1.Columns.Count;
label1.Text = n.ToString();
```

Определение ширины заданного столбца в пикселях

```
// ширина столбца в пикселях
int w;
int nc;
nc = dataGridView1.Columns.Count;
if (nc > 0)
{
w = dataGridView1.Columns[0].Width;
label1.Text = w.ToString();
}
```

Определение количества строк

```
// определить количество строк без строки заголовка int n;
n = dataGridView1.Rows.Count;
label1.Text = (n - 1).ToString();
```

Определение высоты заданной строки в пикселях

```
// определить высоту строки в пикселях
 int h;
int nr, nc;
nc = dataGridView1.Columns.Count;
 nr = dataGridView1.RowCount;
if ((nr>1)&&(nc>0))
{
h = dataGridView1.Rows[0].Height;
label1.Text = h.ToString();
}
```

Ход работы:

Требования к содержанию отчета:

- Номер и название практической работы.
- Цель работы.
- По каждому заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

Порядок выполнения работы:

Все проекты практической работы размещать в своей папке в новой **Пр25_Фамилия** (копия в сетевой папке).

В начале каждого файла проекта установить комментарии: пр.р.№_____ (указать номер), свою Фамилию. Формулировку задания

Задание 1. Создать приложение (проект **Windows Forms приложения (платформа .NET Framework)** и назовите его **Пр25_Фамилия), позволяющее извлекать данные с помощью объекта DataSet.**

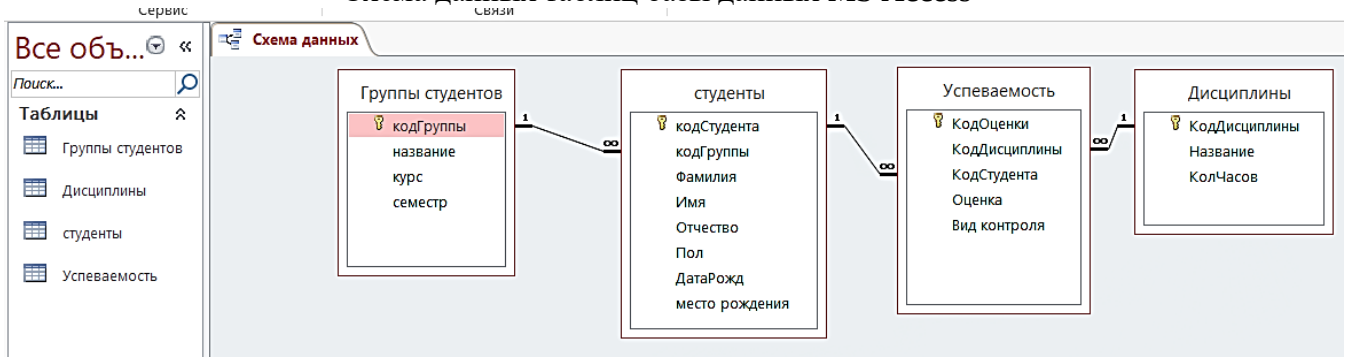
Примечание: в качестве альтернативы для заполнения элемента управления вручную, можно задать свойства для привязки DataGridView в данных источника и автоматически заполняет ее данными.

Алгоритм создания приложения:

В качестве источника данных будем использовать базу данных **dekanatSQL.mdf**, предварительно перенесенную из базы данных из MS Access в MS SQL Server. Скопируйте файл базы данных в свою папку.

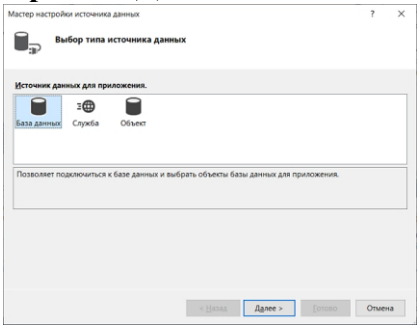
Для обработки данных в приложении необходимо использовать компоненты *DataGridView, BindingSourse, BindingNavigator, DataSet*.

Схема данных таблиц базы данных MS Access



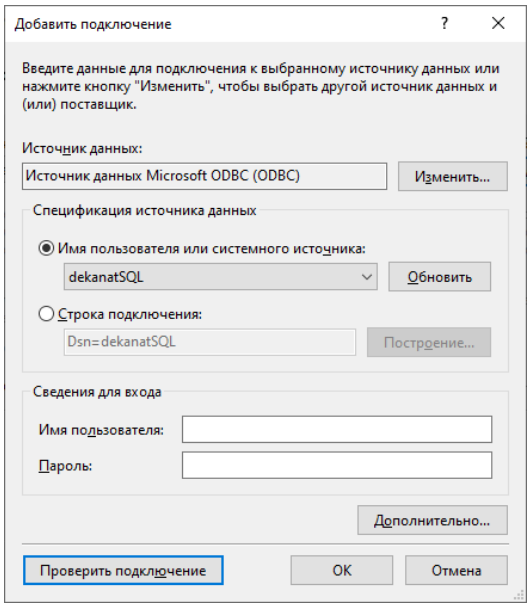
1. Выбрать источник данных

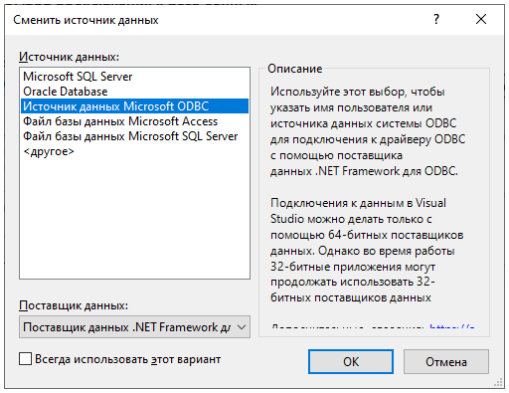
Проект / Добавить новый источник данных…



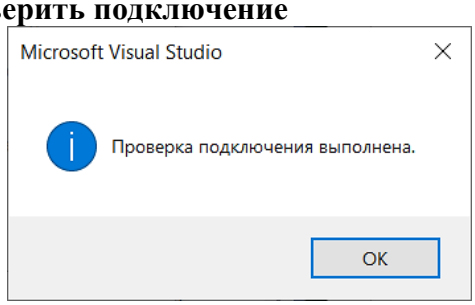
в окне Мастер настройки источника данных **выбрать База данных, Далее, Набор данных**

Создать подключение / Изменить… / Файл базы данных Microsoft Access / через Обзор задать путь к файлу БД



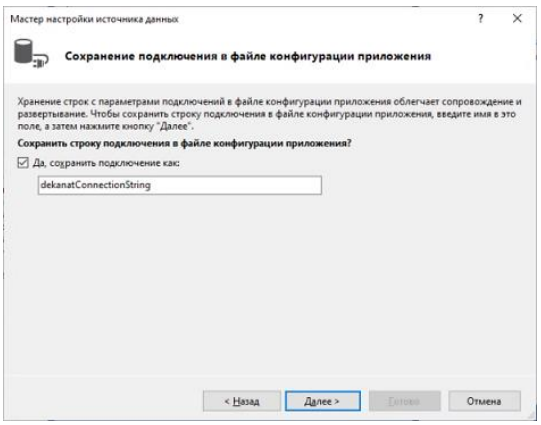


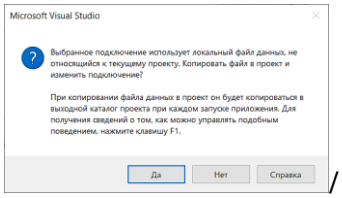
Проверить подключение



Ок.

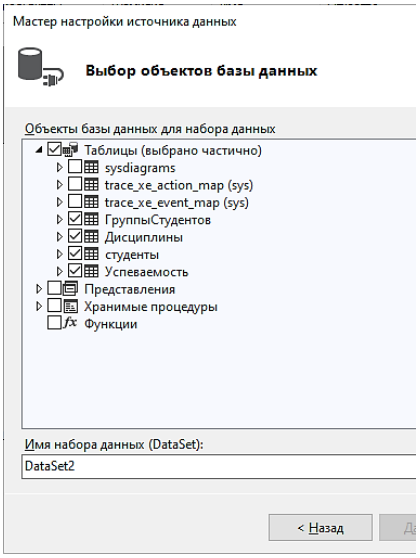
Просмотрите строку подключения (ее необходимо знать для подключения БД программно) **/ Далее / Да**



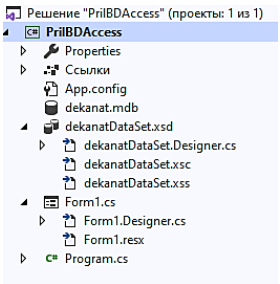


/ Далее/ Далее

Развернуть список **Таблицы** и поставить галочки только напротив 4-х таблиц БД: *Группы студентов, Студенты, Дисциплины, Успеваемость*



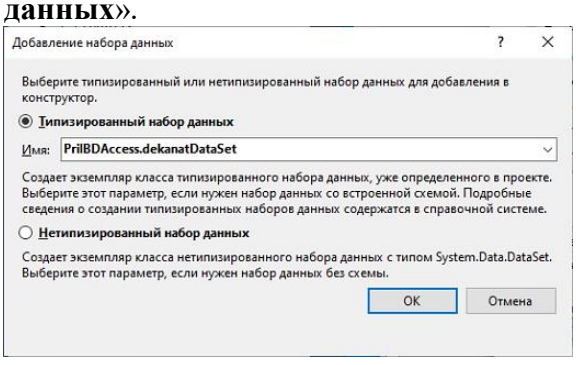
Готово



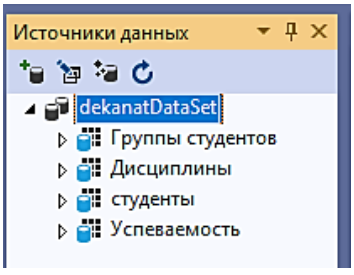
Получим структуру проекта

2. Добавить к своему проекту компонент **DataSet** из категории **Данные**

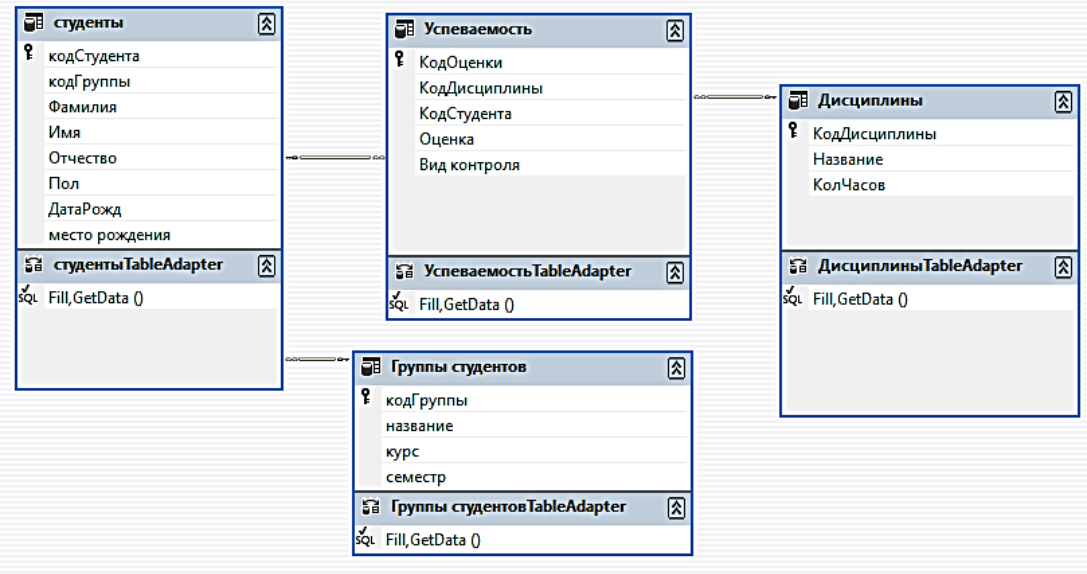
При добавлении **Dataset**, в мастере необходимо выбрать пункт «**Типизированный набор данных**».



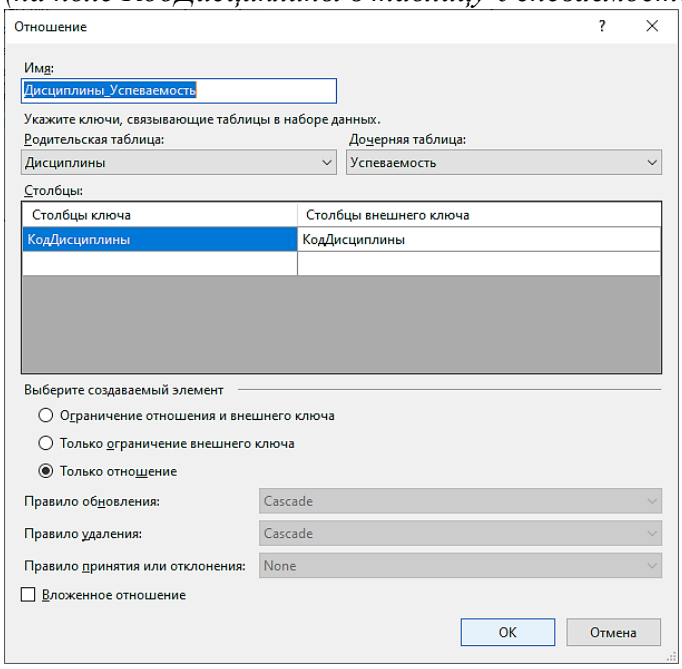
Если все шаги были проделаны верно, то появится **Название БДDataSet** в окне **Источник данных** (развернуть в левой части окна приложения или выбрать в меню **Вид** –



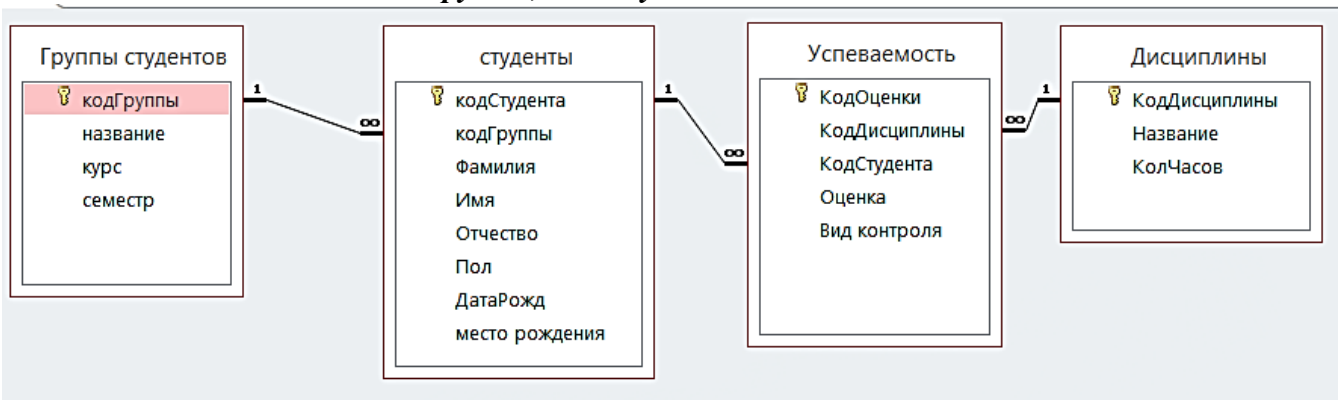
Другие окна)
В окне **Обозреватель решений** дважды щелкнув на файле с **названиемБДDataSet.xsd**, можно увидеть схему данных.



***Примечание**: Обратите внимание, существуют ли связи в созданной схеме. Если связи отсутствуют, то их необходимо создать. Для этого перенесите ключ из одной таблицы (например, КодДисциплины из таблицы Дисциплины) на одноименное поле второй таблицы (на поле КодДисциплины в таблицу Успеваемость) и в появившемся окне нажмите **ОК**:*



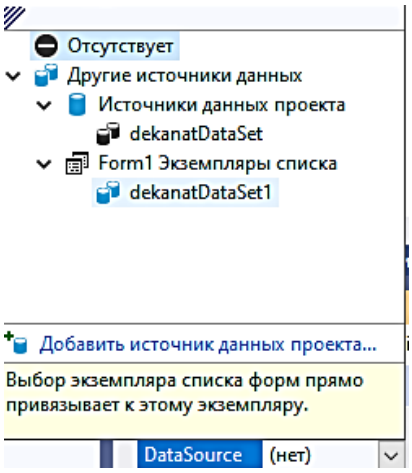
*Таким образом, необходимо создать остальные связи для полей **кодГруппы, кодСтудента***



При необходимости сохраните изменения в схеме.
Закройте схему

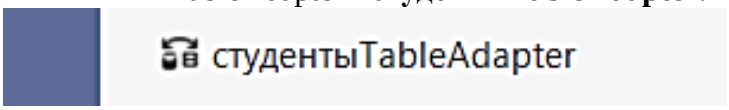
3. На форму из вкладки «**Данные**» панели элементов добавить компоненты **BindingSource, DataGridView, BindingNavigator** (два последних являются визуальными компонентами).
4. Настроить свойства компонента **bindingSource1**

1) в элементе **bindingSource1** выбрать в свойстве **DataSource** именно тот **Dataset**, который расположен на одной форме с этим компонентом



2) переименовать в **bindingSourceDekan**
3) в поле **DataMember** нужно выбрать имя таблицы, связанной с этим компонентом - **Студенты**.

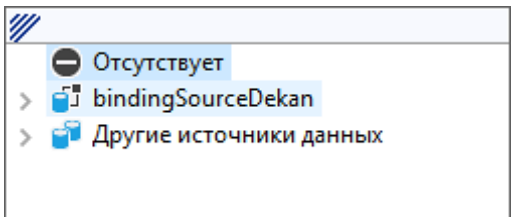
Если всё сделано, верно, то в списке невидимых компонентов должен появиться TableAdapter - **студентыTableAdapter.**

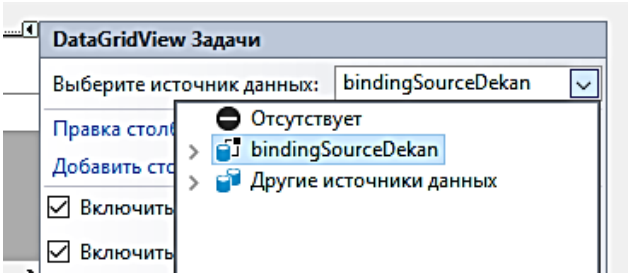


4) перейдите в редакторе кода формы и найдите строку, которая загружает данные в адаптер таблиц. Этот код был сгенерирован при установке привязки данных. Код должен иметь следующий вид:

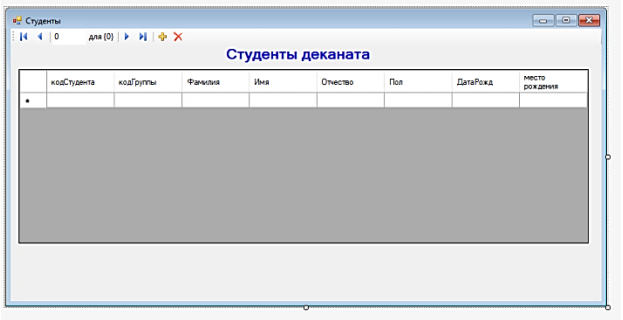
```
private void Form1_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные в таблицу "dekanatDataSet1.студенты"
    // При необходимости она может быть перемещена или удалена.
    this.студентыTableAdapter.Fill(this.dekanatDataSet1.студенты);

}
```

Он находится в событии формы **Form1_Load**.
5. Далее необходимо настроить свойства компонента **dataGridView1**
Переименовать в **dataGridViewDekan**
свойству **DataSource** установить значение **bindingSourceDekan** или в области задач **dataGridViewDekan** выбрать источник.

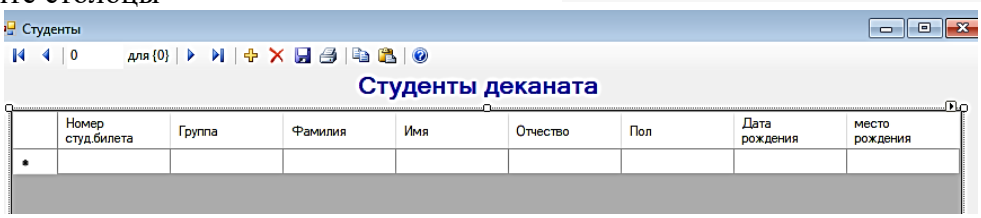




Появятся заголовки столбцов таблицы.
6. Настроить свойства компонента **bindingNavigator1**
Установить свойству **BindingSource** соответствующий компонент формы - **bindingSourseDekan**
Оформите форму, проверьте результат работы



Переименуйте столбцы



Задание 2. Запрограммировать кнопки навигационной панели.
Примечание: все операции над записями в программе происходят через BindingSource.

1. Для этого нужно на панель **bindingNavigator1** добавить **Стандартные элементы** (в окне свойств данного элемента или ПКМ по панели и далее выбрать необходимое)

Приведите панель к следующему виду:

2. Создайте обработчик события для кнопки «**Сохранить**»

*В некоторых случаях компонент BindingNavigator уже содержит кнопку **Сохранить**, но код, сгенерированный конструктором Windows Forms, при этом отсутствует. В этом случае приведенный код можно поместить в Click обработчик событий кнопки, вместо создания полностью новой кнопки ToolStrip.*

Кнопка по умолчанию отключена, свойству **Enabled** кнопки необходимо присвоить значение **true**, чтобы кнопка работала правильно.

Синтаксис команды: TableAdapterName.Update(DataSetName.TableName);

Добавьте код:

```
private void сохранитьToolStripButton_Click(object sender, EventArgs e)
{
    this.студентыTableAdapter.Update(this.dekanatDataSet1);
}
```

Задание 3. Защита от случайного удаления

Чтобы пользователь случайно не удалил запись в нашей базе данных нужно сделать так чтобы перед удалением приложение спрашивало об удалении записи.

Для этого перейдем к компоненту **DataGridView**, откройте список событий и установите обработчик для события **UserDeletingRow**. Нажмите два раза на пустую строчку возле события и перейдете к коду. Вписываем код:

```
private void dataGridViewDekan_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)
{
    DialogResult dr = MessageBox.Show("Удалить запись?", "Удаление", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2);
    if (dr == DialogResult.Cancel)
    {
        e.Cancel = true;
    }
}
```

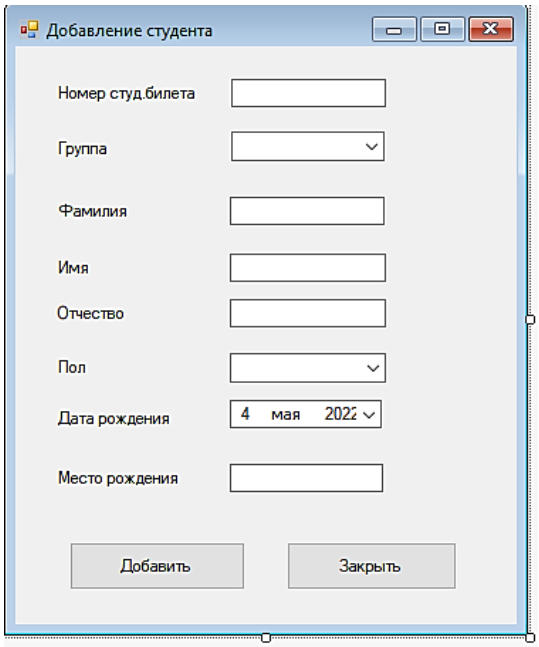
После этого проверьте работоспособность данного кода. Запустите приложение, выделите строку и попробуйте удалить запись.

*Как только пользователь выделит строчку и нажмет кнопку **“Delete”** сработает наше событие **“dataGridView1_UserDeletingRow”** и появится окно с вопросом об удалении.*

Задание 4. Добавление данных

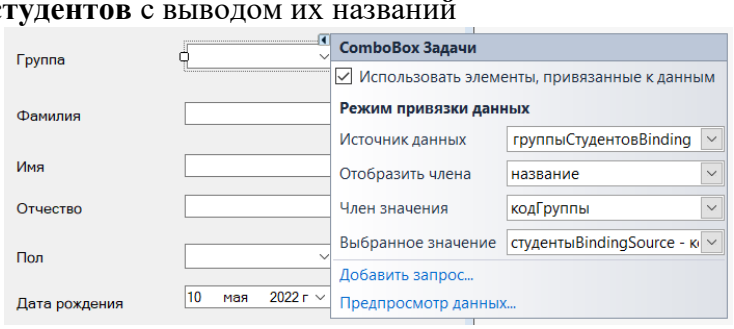
1) Добавьте новую форму в проект и назовите ее **«AddStud»**:

2) Измените заголовок формы и добавьте на форму компоненты:



- 5 TextBox и меняем свойство «Name» на соответствующие;
- 8 label;
- 2 Button (Добавить, Закрыть) и меняем свойство «Name» на соответствующие ();
- 2 comboBox (Группа и Пол)
- 1 dateTimePicker (Дата рождения)

3) Для групп установите связь с таблицей **Группы студентов** с выводом их названий



*В панели действий под опцией " Использовать связанные с данными элементы списка (**Use data bound items")** расположены следующие параметры:*

- ***Data Source (Источник данных)** - определяет таблицу или запрос из которого заполняется список;*
- ***Display Member (Член отображения)** - определяет поле значениями которого заполняется список;*
- ***Value Member (Член значений)** - определяет значения какого поля подставляются в связанное с выпадающим списком поле;*
- ***Selected Value (Выбранное значение)** - определяет связанное с выпадающим списком поле.*

4) Для поля **Пол** создать выпадающий список с обозначением «м» и «ж»

5) Перейдите на главную форму и добавьте кнопку **«Добавить запись»** для добавления нового студента. Дважды щелкните на нее и напишите код для вызова созданной новой формы:

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    AddStud af = new AddStud();
    af.Owner = this;
    af.Show();

}
```

6) Проверьте работоспособность приложения

7) Вернитесь на форму **«Добавления записи»**. Напишите обработчик событий для кнопки **«Закрыть»**:

```
private void buttonClose_Click(object sender, EventArgs e)
{
    Close();
}
```

Чтобы обработчик событий на кнопке **«Добавить»** работал как надо нужно выполнить несколько действий. Зайти в **«Form1.Designer.cs»** и изменить модификаторы доступа на **«public»** нижеуказанных файлов:

```
public dekanatDataSet dekanatDataSet1;
public System.Windows.Forms.BindingSource bindingSourceDekan;
public System.Windows.Forms.DataGridView dataGridViewDekan;
public dekanatDataSetTableAdapters.студентыTableAdapter студентыTableAdapter;
```

Это нужно для того чтобы наша база данных была доступна для выполнения разных с ней манипуляций во всех создаваемых нами формах.

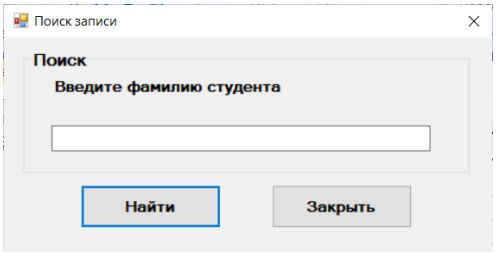
8) Обработчик событий для кнопки **«Добавить»**:

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    Form1 main = this.Owner as Form1;
    if (main != null)
    {
        DataRow nRow = main.dekanatDataSet1.Tables[2].NewRow();
        int rc = main.dataGridViewDekan.RowCount + 1;
        nRow[0] = textBoxIdStud.Text;
        nRow[1] = comboBoxGruppa.SelectedValue.ToString();
        nRow[2] = textBoxSurName.Text;
        nRow[3] = textBoxName.Text;
        nRow[4] = textBoxOtch.Text;
        nRow[5] = comboBoxPol.Text;
        nRow[6] = dateTimePickerDR.Text;
        nRow[7] = textBoxTown.Text;

        main.dekanatDataSet1.Tables[2].Rows.Add(nRow);
        main.студентыTableAdapter.Update(main.dekanatDataSet1.студенты);
        main.dekanatDataSet1.Tables[2].AcceptChanges();
        main.dataGridViewDekan.Refresh();
```

9) Проверьте работу приложения. Добавьте несколько новых записей через новую форму добавления записей (не забывайте при проверке добавления записи заполнять все поля и выбирать дату рождения)
10) Разберите, что выполняется при обработке события и установите комментарии к строкам программного кода.

Задание 5. Поиск данных

Реализуйте самый простой поиск. Данный поиск используется в небольших приложениях. Создайте новую форму с именем «**SearchForm**», вызываемую с главной формы при нажатии соответствующей кнопки, и добавьте на форму компоненты:



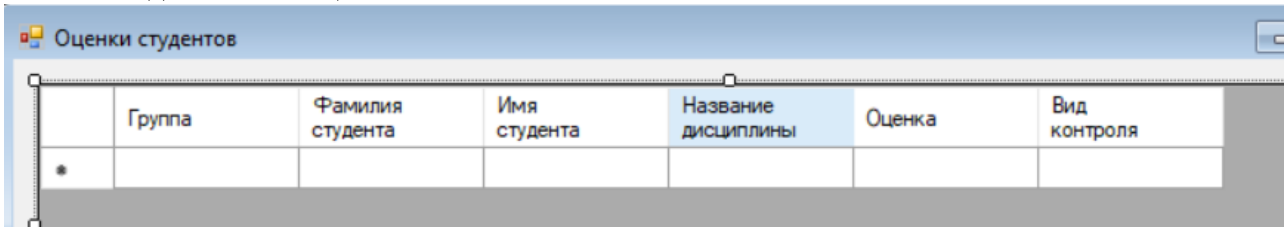
- 1 GroupBox (Поиск);
- 1 label;
- 2 button (Найти и Закрыть);
- 1 TextBox и меняем в свойствах «Name» на «textBoxPoisk».

```
private void buttonPoisk_Click(object sender, EventArgs e)
{
    Form1 main = this.Owner as Form1;
    if (main != null)
    {
        for (int i = 0; i < main.dataGridViewDekan.RowCount; i++)
        {
            main.dataGridViewDekan.Rows[i].Selected = false;
            for (int j = 0; j < main.dataGridViewDekan.ColumnCount; j++)
                if (main.dataGridViewDekan.Rows[i].Cells[j].Value != null)
                    if (main.dataGridViewDekan.Rows[i].Cells[j].Value.ToString().Contains(textBoxPoisk.Text))
                    {
                        main.dataGridViewDekan.Rows[i].Selected = true;
                        break;
                    }
        }
```

Добавьте обработчик событий на форме поиска для кнопки **«Закрыть»** и Обработчик события вызова формы поиска на главной табличной форме. Проверьте работу приложения.

Задание 6. Создание запроса

1) Добавьте на главную форму кнопку вызова таблицы оценок студентов.
2) Создайте новую форму **StudOcenki** с компонентом **DataGridView**, в который добавьте необходимые столбцы:



3) Реализуйте вывод данных успеваемости студентов. Для этого:

- в классе **class StudOcenki : Form** опишите добавление переменных SQL-запроса и строки подключения к базе данных:
  - объявление переменной со строкой подключения к базе данных
  - объявление поля - ссылка на экземпляр класса OleDbConnection (в зависимости от источника данных OdbcConnection) для соединения с БД
  - опишите подключение пространства имен **OleDb (**Odbc**)**
- Функциональность по чтению данных из БД и их последующему выводу в **DataGridView** реализуйте в методе **StudOcenki_Load()**:
  - опишите создание экземпляра класса **OleDbConnection**
  - опишите открытие соединение с БД
  - опишите **SQL-запрос** получение данных из таблиц (**обратите внимание!!! на правильность задания названий таблиц и их полей в соответствии с вашей БД**)

```
// текст запроса
string query =
"SELECT " +
 "[Группы студентов].[Название], " +
 "[студенты].[Фамилия], " +
 "[студенты].[Имя], " +
 "[Дисциплины].[Название], " +
 "[Успеваемость].[Оценка], " +
 "[Успеваемость].[Вид контроля] " +
"FROM " +
 "[студенты], " +
 "[Группы студентов], " +
 "[Дисциплины], " +
 "[Успеваемость] " +
"WHERE " +
 "([Успеваемость].[кодСтудента]=[студенты].[КодСтудента]) AND " +
 "([студенты].[кодГруппы] = [Группы студентов].[КодГруппы]) AND " +
 "([Успеваемость].[КодДисциплины] = [Дисциплины].[КодДисциплины]) ";
```

- опишите создание объекта (**command**), выполняющего запрос к БД
- опишите визуализацию данных в **dataGridView1** с учетом вашего **поставщика данных**

```
// получаем объект OleDbDataReader для чтения табличного результата запроса SELECT
OleDbDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

// в цикле построчно читаем ответ от БД
while (reader.Read())
{
    data.Add(new string[6]);

    data[data.Count - 1][0] = reader[0].ToString();
    data[data.Count - 1][1] = reader[1].ToString();
    data[data.Count - 1][2] = reader[2].ToString();
    data[data.Count - 1][3] = reader[3].ToString();
    data[data.Count - 1][4] = reader[4].ToString();
    data[data.Count - 1][5] = reader[5].ToString();
}
// закрываем OleDbDataReader
reader.Close();

foreach (string[] s in data)
    dataGridView1.Rows.Add(s);
```

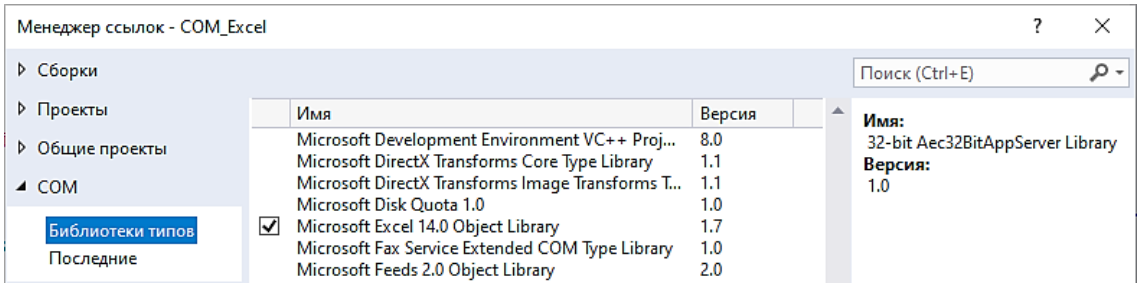
Опишите вызов метода формы:

```
public StudOcenki()
{
    InitializeComponent();
    StudOcenki_Load();
}
```

- Проверьте работу приложения
- Разберите, что выполняется при обработке события и установите комментарии к строкам программного кода.

Задание 7. Выполните экспорт данных в рабочую книгу приложения Excel. Для этого:

1) Создайте на форме Оценки кнопку для экспорта данных
2) Далее прежде чем писать код выгрузки данных, необходимо добавить ссылку на библиотеку объектов Microsoft Excel. Щелкните правой кнопкой мыши свой проект или меню **Проект** и выберите меню **Добавить ссылку…**. После этого перейдите на вкладку **COM** и выберите и добавьте библиотеку объектов Microsoft Excel, например ***Microsoft Excel 14.0 Object Library***



3) Далее создайте обработчик события нажатия кнопки **Экспорт**, в котором необходимо создать объект и документ Excel, получить данные из ***DataGridView*** и добавить строки и столбцы в документ Excel.

Образец кода

- В файле кода формы добавьте:

using Excel = Microsoft.Office.Interop.Excel;

Теперь будет доступен класс для запуска Excel из программы.

- Создайте объект класса Excel и его структурные элементы:

```
//Создание приложения Excel
Excel.Application xlApp = new Excel.Application();
xlApp.Visible = true;
```

- Формируем рабочую книгу и лист (ширину столбцов подберите самостоятельно)

```
//Книга
Excel.Workbook wBook;
Excel.Worksheet xlSheet;
wBook = xlApp.Workbooks.Add();
xlApp.Columns.ColumnWidth = 15;
//Лист
xlSheet = (Excel.Worksheet)wBook.Sheets[1];
//Присвоение имени листа
xlSheet.Name = "Оценки";
```

Задать значение ячейки можно так: **ExcelApp.Cells[1, 1]= "№п/п";**
где **ExcelApp** – имя объекта класса

- Аналогично данному примеру, сформируйте заголовки столбцов нашей таблицы листа Excel
- Для переноса данных применим цикл по строкам и столбцам таблицы оценок, сформированной в компоненте *DataGridView* (обратите внимание на имя вашего компонента ***DataGridView*** в форме Оценки):

```
//Цикл для заполнения ячеек листа
for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
{
    for (int j = 0; j < dataGridView1.Columns.Count; j++)
    {
        xlApp.Cells[i + 2, j + 1] = dataGridView1.Rows[i].Cells[j].Value.ToString();
    }
}
```

i + 2, потому что первая строка отведена для подписей столбцов!

- Далее можно установить некоторые параметры листа (например, выравнивание содержимого ячеек) и отобразить полученный результат:

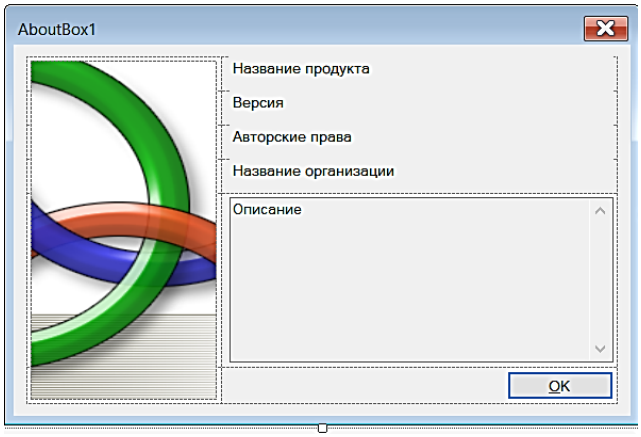
```
// выравнивание: 1- по значению; 2 - влево; 3- посередине; 4 - вправо;
xlSheet.Cells.HorizontalAlignment = 3;
xlApp.Visible = true;
```

- Проверьте результат работы программы.

Задание 8. Создание пользовательского интерфейса.

Создайте новую форму, содержащую основное меню:

1) Данные – Студенты… (подпункты Добавить…, Удалить), Группы, Дисциплины
2) Просмотр – Студенты, Оценки
3) Справка – Вызов справки…, О программе
4) Выход)



Запрограммируйте элементы меню. Реализуйте необходимые элементы пользовательского интерфейса. Для оформления интерфейса пользователя используйте:

- ✓ строку статуса, всплывающие подсказки, выполняющий необходимые действия.
- ✓ иконки для пунктов меню

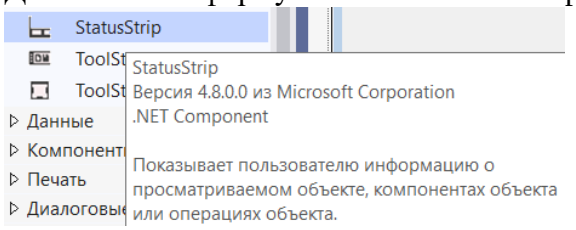
Примечание: окно о программе создается через меню ***«Проект» - «Добавить компонент»…*** *- элемент форма* ***Окно «О программе»****.*

Задание 9. На табличной форме **Студенты** создайте фильтрацию и сортировку данных.

- Самостоятельно допишите в обработчик события кнопки *Экспорт* сохранение файла с заданными форматами фильтра файлов Excel и выводом диалогового окна, подтверждающее успешный экспорт при сохранении файла и дальнейшее закрытие приложения Excel. В данном случае удобно добавить конструкцию try-catch-finally и задать соответствующие блоки описания

Примечание*: Чаще всего catch и finally применяются вместе для получения и использования ресурсов в блоке try, устранения исключительных обстоятельств в блоке catch и освобождения ресурсов в блоке finally.*

- Добавьте на форму элемент StatusStrip для отображения количества записей в таблице



и напишите самостоятельно код строки подсчета количества строк в таблице (за исключением заголовка) в обработчик загрузки табличной формы.

При необходимости запрограммируйте остальные необходимые элементы разработанного приложения

Контрольные вопросы:

1) Назначение DataAdapter и команд Fill, Update.
2) Опишите технологию созданию приложения с БД:
   а) Подключение к базе данных. Чтение строки подключения Connection String.
   б) Размещение компонента типа dataGridView.
   в) Добавление переменных SQL-запроса и строки подключения к базе данных.
   г) Подключение пространства имен OleDb.
   д) Создание объекта типа OleDbDataAdapter.
   е) Заполнение таблицы на основе SQL-запроса (привести примеры запросов).
   ж) Визуализация данных в dataGridView.
3) Как осуществить экспорт данных из таблицы в приложение Excel
4) Элемент управления dataGridView: назначение, основные операции с данными и свойства