

Лабораторная работа № 33-34

Тема: Добавление и редактирование данных в связанных таблицах.

Программное обеспечение: MS Visual Studio 2015

Цель: научиться реализовывать добавление и редактирование данных в связанных таблицах, используя технологию Database First Entity Framework.

Время на выполнение: 2 часа.

Задание 1: Реализовать добавление и редактирование данных в таблице «Заявления».

Порядок выполнения:

1. Откройте проект, созданный в предыдущей лабораторной работе.
2. Создайте новую форму Form2 и с помощью Конструктора форм поместите на нее все необходимые элементы: dataGridView1 для отображения таблицы, элементы GUI для поиска и кнопки добавления и редактирования записей – все как в предыдущих формах (см. Рисунок .1)

Код_заявления	Фамилия	Название	Статус
29	Иванов	АСОИУ	зачислен
30	Попова	ФИП (РЯП)	не зачислен
31	Сыроежкин	АСОИУ	зачислен
32	Шариков	АСОИУ	не зачислен
33	Каренина	АСОИУ	забрал документы
34	Трубецкой	АСОИУ	не зачислен
35	Романова	АСОИУ	не зачислен
41	Иванов	ФИП (РЯП)	не зачислен
42	Попова	ЮР	не зачислен
43	Сыроежкин	ЮР	не зачислен
44	Шариков	ЮР	не зачислен
45	Каренина	ЮР	забрал документы
46	Трубецкой	ЮР	не зачислен
49	Сыроежкин	АСОИУ	не зачислен
51	Попова	АСОИУ	не зачислен
53	Иванова	ФИП (РЯП)	не зачислен
54	sdf	ЮР	зачислен

Рисунок 1 - Форма для отображения заявлений абитуриентов

3. Для элемента dataGridView1 определите режим автоматического заполнения данными по ширине элемента:

AutoSizeColumnsMode	Fill
AutoSizeRowsMode	AllCells

4. Затем перейдите к заполнению таблицы данными. Проблема здесь заключается в том, что в базе данных сущность «Заявления» имеет поля Код_абитуриента и Код_специальности – внешние ключи, содержащие лишь id записей. Этой таблице соответствует сущностный класс Заявления:

```

public partial class Заявления
{
    ссылка: 0
    public int Код_заявления { get; set; }
    ссылка: 0
    public int Код_абитуриента { get; set; }
    ссылка: 0
    public int Код_специальности { get; set; }
    ссылка: 0
    public string Статус { get; set; }

    ссылка: 0
    public virtual Абитуриенты Абитуриенты { get; set; }
    ссылка: 0
    public virtual Специальности Специальности { get; set; }
}

```

Как можно заметить, с помощью списка, состоящего из таких объектов (List<Заявления> sheetzayav), представить данные в наглядной и удобной для пользователя форме не получится. Необходимо «выдернуть» из внешних таблиц данные о ФИО абитуриента и названии специальности и затем добавить их в дополнительные поля каждой записи в таблице. Таким образом, возникает необходимость выполнения кросс-запроса к трем таблицам и определения пользовательского типа данных (объекта) view, содержащим необходимые поля.

Далее будем использовать следующий подход: объединение таблиц с помощью LINQ-запроса, на уровне кода приложения.

В этом случае в коде формы будет следующий блок кода:

```

public partial class Form2 : Form
{
    public List<view> viewzayav = new List<view>();
    public pr116_IvanovaEntities db;

    ссылка: 1
    public Form2()
    {
        InitializeComponent();
        db = new pr116_IvanovaEntities();
    }
}

```

5. В данном коде создается список заявлений, состоящий из объектов типа **view**, который определим позже, и экземпляр контекста pr116_IvanovaEntities db.
6. Добавьте к проекту новый сущностный класс view:

```

public class view
{
    ссылка: 3
    public view(int id, string ab, string sp, string s)
    {
        this.Код_заявления = id;
        this.Фамилия = ab;
        this.Название = sp;
        this.Статус = s;
    }
    ссылка: 8
    public int Код_заявления { get; set; }
    ссылка: 3
    public string Фамилия { get; set; }
    ссылка: 3
    public string Название { get; set; }
    ссылка: 2
    public string Статус { get; set; }
}

```

7. А конструктор формы будет содержать следующие строки, реализующие инициализацию данных:

```
viewzayav = db.Абитуриенты.  
    Join(db.Заявления, а => а.Код_абитуриента, з => з.Код_абитуриента, (а, з) =>  
        new { а, з })  
.Join(db.Специальности, зс => зс.з.Код_специальности, с => с.Код_специальности, (зс, с) =>  
    new { зс, с })  
.AsEnumerable()  
.Select(x=> new view (  
    x.зс.з.Код_заявления,  
    x.зс.а.Фамилия,  
    x.с.Название,  
    x.зс.з.Статус))  
.OrderBy(o=>o.Код_заявления).ToList();  
dataGridView1.DataSource = viewzayav.ToList();
```

8. В данном коде приводим анонимный тип, возвращенный в результате выполнения Join() к «перечисляемому» с помощью метода AsEnumerable(). Во-вторых, проецируем элементы полученной последовательности в список объектов типа view с помощью Select(). И, так как поля объекта view уже расположены в нужном порядке, остается только настроить названия столбцов в сетке:

```
dataGridView1.DataSource = viewzayav.ToList();  
dataGridView1.ReadOnly = true;  
if (dataGridView1.RowCount == 0) label1.Visible = true;  
else label1.Visible = false;  
dataGridView1.Columns[0].HeaderText = "№";  
dataGridView1.Columns[0].Width = 30;  
dataGridView1.Columns[1].HeaderText = "Фамилия абитуриента";  
dataGridView1.Columns[2].HeaderText = "Специальность";
```

9. Добавьте по аналогии с формами, созданными в предыдущей лабораторной работе, код кнопок добавления и редактирования записи:

```
private void button2_Click(object sender, EventArgs e)  
{  
    if (Application.OpenForms.Count == 1)  
    {  
        FormAddZayav addZayav = new FormAddZayav();  
        addZayav.Owner = this; addZayav.Show();  
    }  
    else Application.OpenForms[2].Focus();  
}  
  
private void Редактировать_Click(object sender, EventArgs e)  
{  
    if (dataGridView1.SelectedCells.Count == 1)  
    {  
        if (Application.OpenForms.Count == 1)  
        {  
            view item = viewzayav  
                .First(w => w.Код_заявления.ToString() == dataGridView1  
                    .SelectedCells[0].OwningRow.Cells[0].Value.ToString());  
            FormEditZayav edZayav = new FormEditZayav(item);  
            edZayav.Owner = this;  
            edZayav.Show();  
        }  
        else Application.OpenForms[2].Focus();  
    }  
}
```

10. Для редактирования записи создайте форму edRec с уже привычным набором элементов: четыре текстовых поля с поясняющими надписями, кнопка и label1 для отображения сообщения об ошибке. Важно отметить, что в любом заявлении основная информация – это связанные с ним абитуриент и специальность. статус заявления – скорее пояснение, чем свойство. Поэтому, при изменении основной информации, фактически произойдет удаление старого статуса и добавление нового, чего по условиям задачи (хранение всех записей «в архиве») быть не должно. Таким образом, редактировать возможно только статус. Дизайн формы редактирования показаны на рисунке 2.

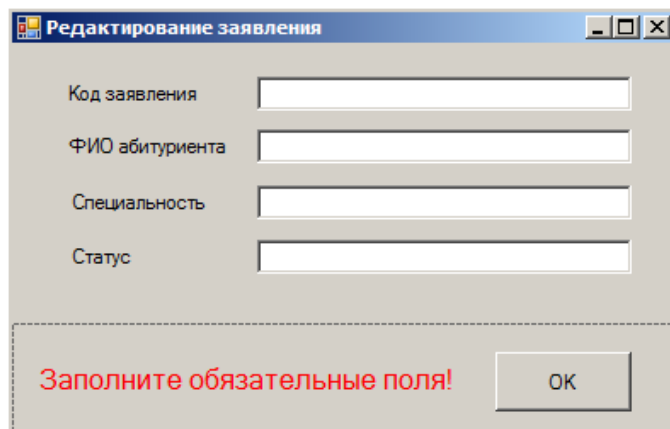


Рисунок 2 - Форма для редактирования заявления

11. Код формы выглядит следующим образом:

```
public partial class FormEditZayav : Form
{
    view item;
    ссылка: 1
    public FormEditZayav(view itm)
    {
        InitializeComponent();
        item = itm;
        textBox1.Text = item.Код_заявления.ToString();
        textBox2.Text = item.Фамилия.ToString();
        textBox3.Text = item.Название.ToString();
        textBox4.Text = item.Статус.ToString();
    }
}
```

12. Код кнопки ОК выглядит следующим образом:

```

private void button1_Click(object sender, EventArgs e)
{
    if (textBox4.Text.Trim() == "")
    {
        label7.Visible = true;
        label7.Text = "Заполните поле \'Диагноз\'";
    }
    else
    {
        var result = ((Form2)Owner).db.Заявления.SingleOrDefault(w =>
            w.Код_заявления == item.Код_заявления);
        result.Статус = textBox4.Text.ToString();
        ((Form2)Owner).db.SaveChanges();
        ((Form2)Owner).viewzayav =
            ((Form2)Owner).db.Абитуриенты
                .Join(((Form2)Owner).db.Заявления,
                    a => a.Код_абитуриента,
                    з => з.Код_абитуриента,
                    (a, з) => new { a, з })
                .Join(((Form2)Owner).db.Специальности,
                    зс => зс.з.Код_специальности,
                    с => с.Код_специальности,
                    (зс, с) => new { зс, с })
                .AsEnumerable()
                .Select(x => new view(
                    x.зс.з.Код_заявления,
                    x.зс.а.Фамилия,
                    x.с.Название,
                    x.зс.з.Статус))
                .OrderBy(o => o.Код_заявления).ToList();
        ((Form2)Owner).dataGridView1.DataSource =
            ((Form2)Owner).viewzayav;
        ((Form2)Owner).dataGridView1.Refresh();
        this.Close();
    }
}

```

Здесь важно понимать, что хоть данные отображаются в таблице с помощью viewzayav, но все изменения необходимо производить с таблицей Заявления нашей базы данных. И при изменении данных в ней потребуется и обновление списка, следовательно, и выполнение сложного запроса (в предыдущих формах, к слову, он был просто Select()). Так как меняем (или далее добавляем) всего один элемент, и точно знаем, какой, когда и где, то можно обойтись без кросс-запроса, обновляя параллельно и список viewzayav, что гораздо проще с точки зрения кода. Однако это создает потенциальный «разрыв» между контекстом (БД) и списком, что чревато ошибками при последующих модификациях кода. Вместо этого лучше вынести LINQ-запрос и сопутствующую логику в отдельный метод, подобно тому, как это описывалось в предыдущем разделе.

С добавлением заявления не все так просто. Пользователь должен выбрать одного абитуриента, одну специальность и указать статус. Понятно, что выбирать лучше всего из списка, для чего пользователю потребуется наличие в нем функции поиска. Пользуясь написанным ранее кодом, сделать это легко.

13. С этой целью создадим новую форму и добавим на неё два элемента типа dataGridView для просмотра списка абитуриентов и списка специальностей. Под ними разместим необходимые для поиска элементы (два «комплекта», по одному на таблицу), поле для ввода статуса и кнопку (см. Рисунок 3).

Рисунок 3 - Форма для добавления заявления

В класс формы и в конструктор формы добавим следующий код:

```
public partial class FormAddZayav : Form
{
    public pr116_IvanovaEntities db;
    public List<Абитуриенты> sheetabiturients;
    public List<Специальности> sheetspecs;

    ссылка: 1
    public FormAddZayav()
    {
        InitializeComponent();
        db = new pr116_IvanovaEntities();
        //настройка таблицы абитуриентов
        sheetabiturients = db.Абитуриенты.OrderBy(o => o.Код_абитуриента).ToList();
        dataGridView1.DataSource = sheetabiturients;
        dataGridView1.ReadOnly = true;
        if (dataGridView1.RowCount == 0) label1.Visible = true;
        else label1.Visible = false;
        dataGridView1.Columns[0].HeaderText = "№";
        dataGridView1.Columns[4].Visible = false;
        dataGridView1.Columns[5].Visible = false;
        dataGridView1.Columns[6].Visible = false;
        dataGridView1.Columns[7].Visible = false;
        //настройка таблицы специальностей
        sheetspecs = db.Специальности.OrderBy(o => o.Название).ToList();
        dataGridView2.DataSource = sheetspecs;
        dataGridView2.ReadOnly = true;
        if (dataGridView2.RowCount == 0) label1.Visible = true;
        else label1.Visible = false;
        dataGridView2.Columns[0].Visible = false;
        dataGridView2.Columns[1].HeaderText = "Специальность";
        dataGridView2.Columns[2].Visible = false;
        dataGridView2.Columns[3].Visible = false;
        dataGridView2.Columns[4].Visible = false;
    }
}
```

14. Важно также помнить, что за «пустоту» списка абитуриентов «отвечает» label1, а за список специальностей – label2. Интерфейс формы представлен на рисунке ниже.

15. Что касается кода добавления записи, то он следующий:

```

private void button1_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedCells.Count == 1 && dataGridView2.SelectedCells.Count == 1)
    {
        if (textBox3.Text.Trim() != "")
        {
            try {...}
            catch (Exception err)
            {
                label1.Visible = true;
                label1.Text = "Ошибка ввода! " +
                    err.Message;
            }
        }
        else
        {
            label1.Visible = true;
            label1.Text = "Укажите статус!";
        }
    }
    else
    {
        label1.Visible = true;
        label1.Text = "Выберите ровно одного абитуриента и ровно одну специальность!";
    }
}

```

16. Внутри оператора try будет следующий код, по структуре своей схожий с аналогичными блоками в предыдущих формах добавления записей:

```

Заявления result = new Заявления();
result.Код_абитуриента = Int32.Parse(dataGridView1
.SelectedCells[0].OwningRow.Cells[0].Value.ToString());
result.Код_специальности = Int32.Parse(dataGridView2
.SelectedCells[0].OwningRow.Cells[0].Value.ToString());
result.Статус = textBox3.Text;
((Form2)Owner).db.Заявления.Add(result);
((Form2)Owner).db.SaveChanges();
// блок кода для обновления таблицы
((Form2)Owner).viewzayav = ((Form2)Owner).db.Абитуриенты
.Join(((Form2)Owner).db.Заявления,
a => a.Код_абитуриента,
з => з.Код_абитуриента,
(a, з) => new { а, з })
.Join(((Form2)Owner).db.Специальности,
зс => зс.з.Код_специальности,
с => с.Код_специальности,
(зс, с) => new { зс, с })
.AsEnumerable()
.Select(x => new view(
    x.зс.з.Код_заявления,
    x.зс.а.Фамилия,
    x.с.Название,
    x.зс.з.Статус))
.OrderBy(o => o.Код_заявления).ToList();
((Form2)Owner).label1.Visible = false;
((Form2)Owner).dataGridView1.DataSource =
((Form2)Owner).viewzayav;
((Form2)Owner).dataGridView1.Refresh();
this.Close();

```

17. Изменение видимости label1 на форме Form3 здесь необходимо потому, что запись может добавляться в пустую таблицу Заявления, следовательно, в таблице dataGridView1 формы Form3 появятся данные, и отображение этой надписи уже не требуется. Остается снова напомнить, что вынесение повторяющихся блоков кода с их «универсализацией» (таких, как кросс-запрос) будет правильным решением с точки зрения архитектуры приложения.

18. Проверьте разработанный функционал приложения: редактирование и добавление данных в таблицу «Заявления».

Задание 2 для самостоятельного выполнения по вариантам:

Вариант 1: Экзамены, Дисциплины и Специальности

Вариант 2: Оценки, Экзамены, Абитуриенты

- 1) Реализуйте в клиентском приложении отображение таблицы с помощью формирования объекта на основе содержимого нескольких таблиц – кросс-запрос.
- 2) Спроектируйте дизайн графического интерфейса приложения. Добавьте на форму необходимые таблицы, кнопки и текстовые поля. Свяжите их с соответствующими данными и событиями.
- 3) Реализуйте возможность навигации (поиска) по таблицам в приложении.
- 4) Реализуйте функционал редактирования, добавления и удаления данных для тех таблиц, где это необходимо. Учтите проверку данных, вводимых пользователем.