МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, РАДИОТЕХНИКИ И ЭЛЕКТРОНИКИ»**

**КУРСОВАЯ РАБОТА**

**ПО ПРОГРАММИРОВАНИЮ**

**НА ТЕМУ:**

**«Отдел кадров»**

Ф.И.О.: Архипов Иван Константинович

Кафедра: КБ-9

Название кафедры: Прикладная информатика

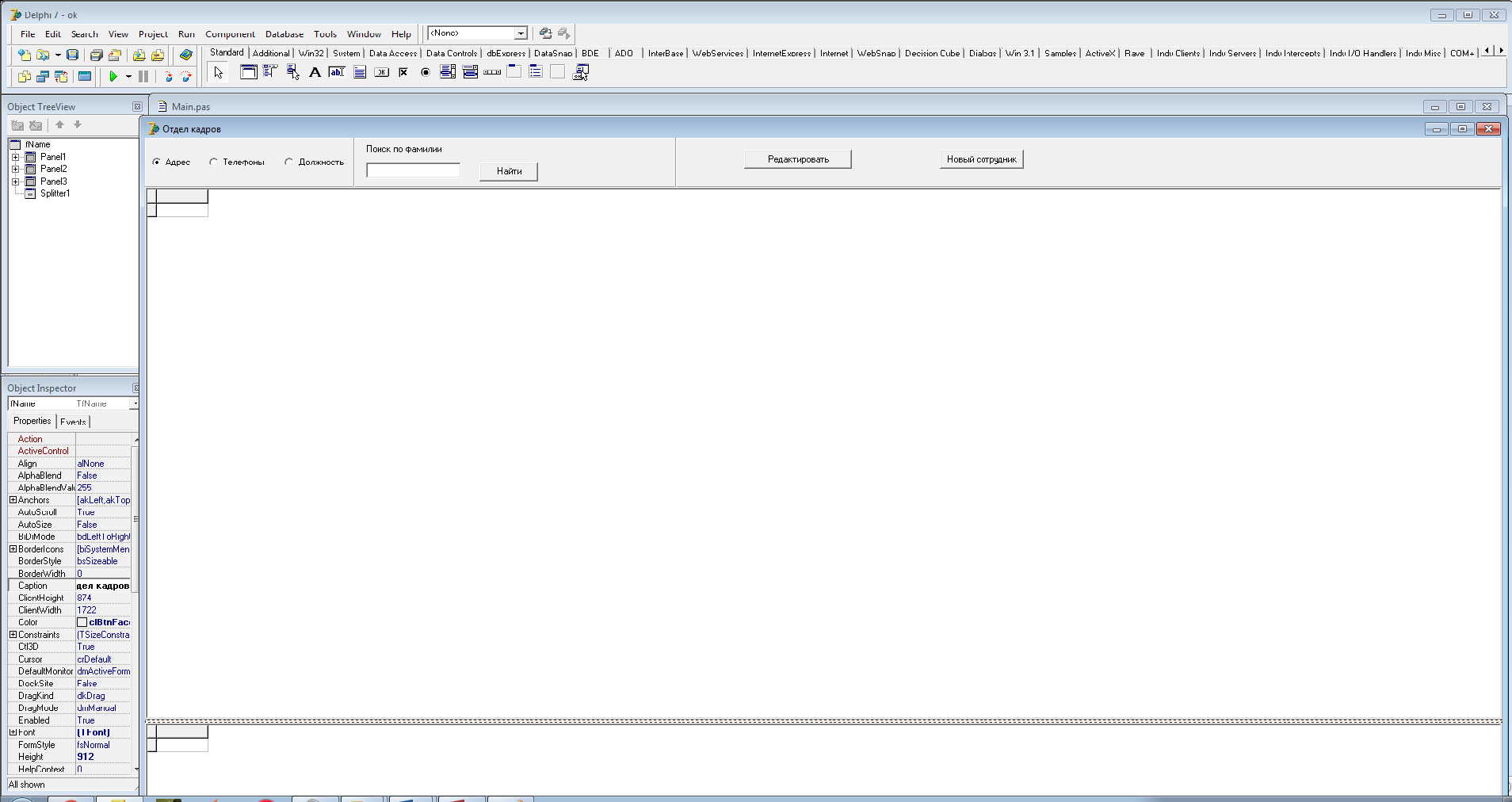
Шифр направления: 09.03.03  
Название направления: Прикладная информатика

Преподаватель: Королёв Д.А.

Москва 2015г.

# 1.  Создание клиентского приложения.

Запускаем Delphi, создаем новый проект. Добавим на форму три панели. Свойству Align верхней панели присвоим значение alTop (весь верх). Свойству Align нижней панели присвоим значение alBottom. Поместим компонент Splitter с вкладки Additional и присвоим его свойству Align также alBottom, после чего он прижмется к нижней панели. Splitter -- это разделитель между панелями, с его помощью пользователь мышью сможет передвигать нижнюю панель, меняя ее размеры. И, наконец, свойству Align средней панели присвоим значение alClient, чтобы она заняла все оставшееся место на форме. Очистим свойство Caption всех трех панелей.  
На верхнюю панель поместим три компонента RadioButton с вкладки Standard. В их свойствах Caption напишем, соответственно, "Адрес", "Телефоны" и "Должность". Переключаясь между ними, пользователь сможет выводить в нижнюю, подчиненную сетку DBGrid нужные данные. Свойству Checked первой радиокнопки присвойте значение True, чтобы включить ее. Раздел с переключателями разделим компонентом Bevel с вкладки Additional. Его ширину (свойство Width) сделаем равным 2 пикселям, превратив его в вертикальную разделительную полосу.  
Сделаем раздел поиска, поместив в него Label (с текстом «Поиск по фамилии»), Edit и кнопку Button (с текстом «Найти»).   
В последнем разделе верхней панели находятся еще две кнопки Button. Одна из них предназначена для редактирования текущей записи, другая -- для добавления новой.  
Вторая и третья панели содержат только по одному компоненту DBGrid из вкладки DataControls, свойствам Align которых присвоим значение alClient.   
Свойству Name формы присвоим значение fMain, свойство Caption формы присвоим значение "Отдел кадров", модуль сохраним под именем Main.pas, а проект в целом назовем ok (отдел кадров).



# 2.  Подключение БД.

Добавим в проект модуль данных (File\New\Data Module).

Модуль данных – это не визуальный контейнер для размещения на нем не визуальных компонентов. Также модуль данных часто используют и для хранения глобальных переменных, общих функций и процедур, которые должны быть видны по всей программе. Модуль данных не имеет формы, но сохраняется как модуль в файле \*.pas.

Свойству Name модуля данных присвоим имя fDM, а модуль сохраним как DM.pas.

Добавляем в модуль компонент ADOConnection с вкладки ADO. Этот компонент обеспечит связь других компонентов с базой данных при помощи механизма ADO. Связь обеспечивается свойством компонента ConnectionString. В общем-то, у таких компонентов, как ADOTable тоже есть это свойство, однако, имея четыре таблицы, придется четыре раза устанавливать связь. Проще один раз соединиться компонентом ADOConnection и использовать его для связи других компонентов.

Для связи с базой данных MS Access выбираем “Microsoft Jet 4.0 OLE DB Provider”.

Нажимаем на кнопку “Далее”, либо переходим на вкладку “Подключение”. Здесь необходимо выбрать или ввести базу данных. Так как файл с базой данных находится там же, где находится программа, то можно вручную вписать только имя файла с БД, без полного адреса. В этом случае вы сможете перемещать папку с клиентским приложением и базой данных и не нужно будет заново прописывать к ней путь.

Установим свойство LoginPrompt компонента ADOConnection в False. Если этого не сделать, то при каждой попытке соединиться с базой данных будет выходить запрос на пользовательское имя и пароль, нам это не нужно, наша база данных без пароля. Свойство Connected переведите в True.

Установите в модуль данных четыре компонента ADOTable, по одному на каждую таблицу из нашей базы данных. Компонент ADOTable предназначен для создания набора данных.

Выделите все четыре ADOTable (удерживая клавишу <Shift>), и в их свойстве Connection выберите связь ADOConnection1. Таким образом, все четыре ADOTable подключили к базе данных.

Выделите первый компонент ADOTable. Переименуйте его свойство Name в TLichData, а в свойстве TableName выберите главную таблицу базы – LichData. Буква “Т” в начале названия компонента укажет нам в дальнейшем, что это таблица. Рядом с компонентом установите компонент DataSource из вкладки Data Access. Компонент DataSource предназначен для организации связи с наборами данных. Свойство Name компонента DataSource переименуйте в DSLichData (DS – DataSource). В свойстве DataSet выберите таблицу TLichData.

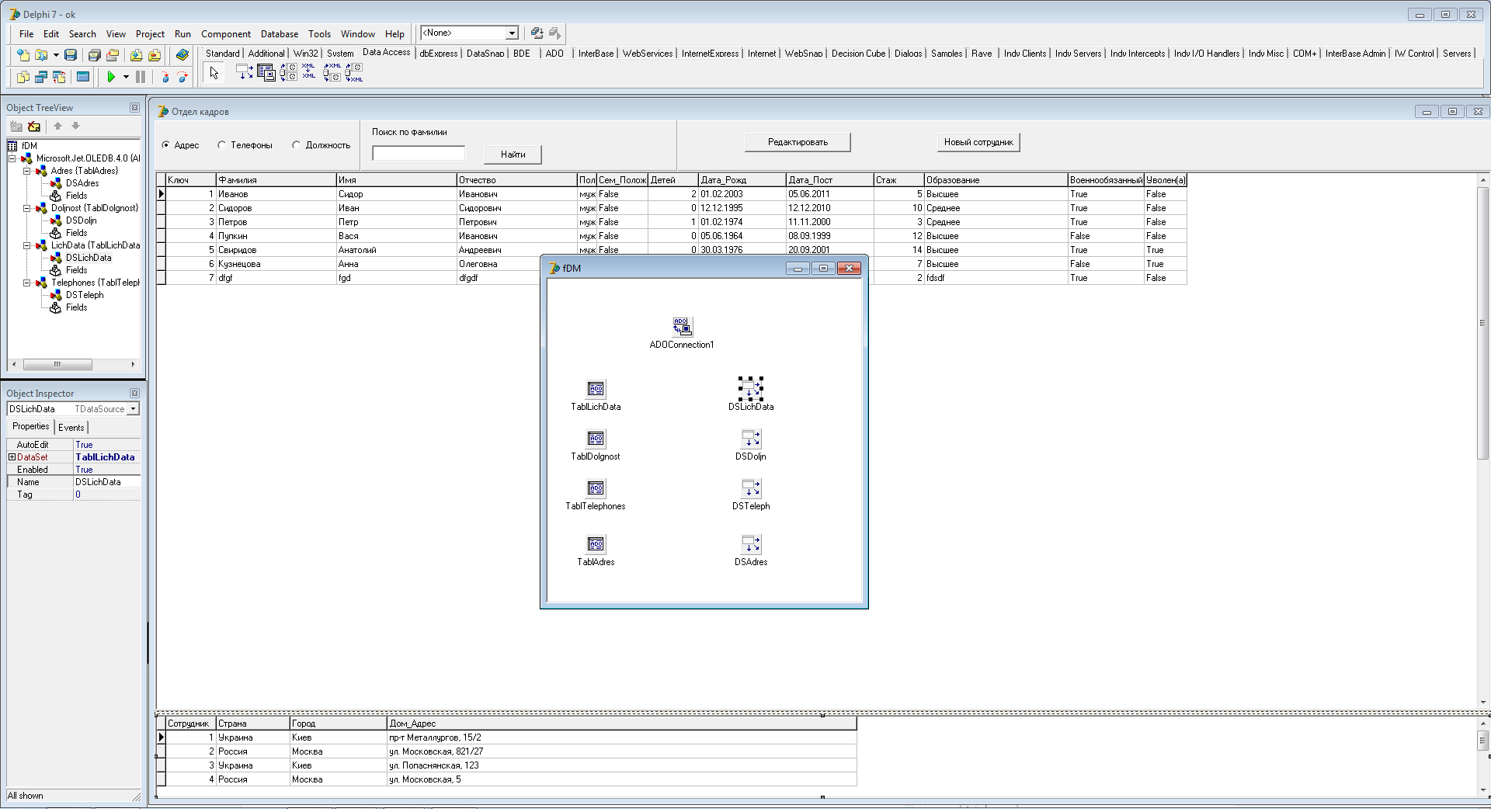
То же самое нужно проделать еще три раза:

ADOTable: TDoljnost, TTelephones, TAdres

DataSource: DSDoljnost, DSTelephones, DSAdres

Свойство Active таблиц переведите в True, открыв их.

Перейдите на главную форму. Выберите команду File\Use Unit и подключите к ней модуль Data Module (DM). Теперь выберите в компонентах DBGrid в свойстве DataSource необходимые таблицы и вы увидите содержимое этих таблиц.



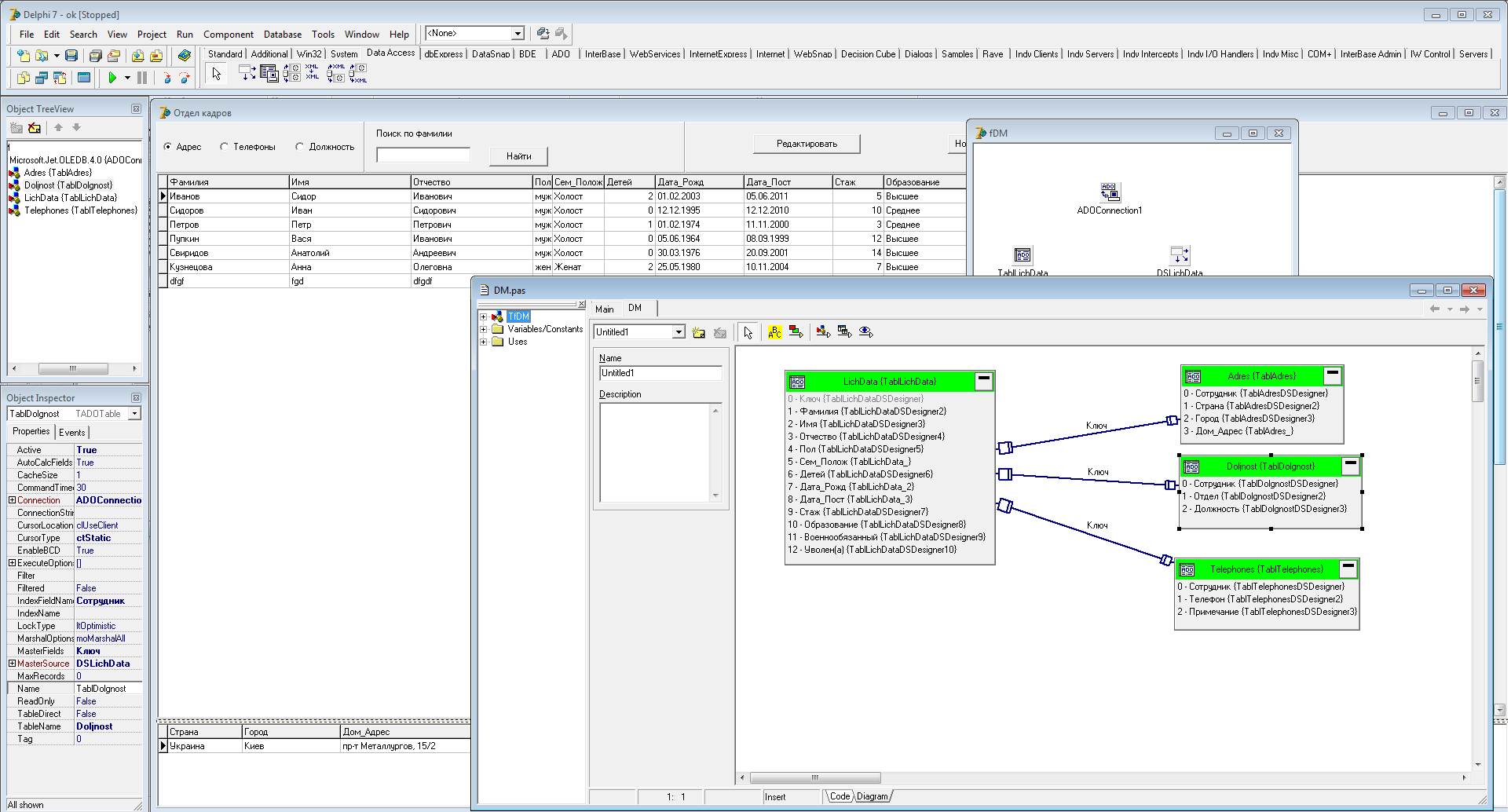
# 3. Создание связей между таблицами.

Выделите оба компонента DBGrid и установите в True их свойства ReadOnly – только чтение. Таблицы будут связаны и нам не нужно, чтобы пользователь вводил данные отдельно в каждую таблицу. Мы для этого сделаем отдельную форму, а DBGrid будем использовать только для просмотра.  
Теперь нужно установить связь между таблицами. Это требуется не только для того, чтобы в нижней сетке выходили данные только на сотрудника, выделенного в верхней сетке, но и для того, чтобы мы смогли в дальнейшем вводить связанные данные в окне редактора.  
Выделите модуль данных. Щелкните дважды по первой таблице, чтобы открыть редактор полей. Щелкните по этому редактору и выберите команду Add all fields (добавить все поля).

Редактор полей предназначен для настройки параметров каждого поля, для добавления новых полей или удаления имеющихся. Если в редакторе полей нет ни одного поля, то в компоненте DBGrid будут отображены все поля таблицы, имеющие параметры по умолчанию. Если же мы добавили в редактор полей хотя бы одно поле, то сетка DBGrid его и отобразит. В редакторе можно для каждого поля изменить различные параметры, например, ширину колонки, название колонки, видимое это поле или нет, и т.п. Кроме того, редактор полей предоставляет возможность добавлять в набор данных новые поля, например вычисляемые или просматриваемые (lookup).  
Поле “Ключ” у нас предназначено для связи с другими таблицами. Пользователю его видеть не обязательно. Выделите его, и в свойстве Visible установите False.

Здесь у нас есть два логических поля – “Сем\_Полож” и “Военнообязанный”. Чтобы True и False выходили на экране так, как нам нужно, свойству DisplayValues первого из этих полей присвойте значение “Женат;Холост”, а второго – “Да;Нет”. Первым здесь идет значение, которое будет обозначать True, вторым – False. Эти значения разделяются точкой с запятой, пробелы не нужны.  
Таким же образом добавьте все поля в остальные три таблицы. У них невидимым следует сделать поле “Сотрудник” – этому полю автоматически будет присвоено такое же число, как у поля Ключ соответствующей записи. Логических полей у них нет. Открыв редактор маски, и в поле Input Mask введите маску “#(###)-###-##-##”. Сохраните ее, нажав кнопку ОК. Для полей типа Дата в этом свойстве (в таблице LichData два таких поля) введите маску “##.##.####”.

Теперь создадим связи между таблицами. При помощи кнопки F12 перейдите в редактор кода. В нижней части окна перейдем на вкладку Diagram.   
Для начала в окно диаграмм нужно добавить наши таблицы. Найдите их в окне дерева объектов Object TreeView. Если у вас это окно закрыто, откройте его клавишами Shift+Alt+F11 либо командой меню View\Object TreeView. Ухватитесь в этом окне мышью за название главной таблицы LichData и перетащите ее в окно диаграмм. Таблица вместе с полями отобразится в окне. Точно также перетащите остальные таблицы. Связи главная – подчиненная делают кнопкой Master/Detail Connector, которую вы можете увидеть в верхней части окна диаграмм (предпоследняя).  
При установке связей главный/подчиненный важно начинать вести линию с главной таблицы к подчиненной.



# 4. Создание формы для ввода данных.

Создайте новую форму (File\New\Form). Ее свойство Name переименуйте в fEditor, а при сохранении формы дайте модулю имя Editor. Командой File\Use Unit подключите к форме модуль данных DM.  
Установим на форму четыре панели GroupBox с вкладки Standard, на каждую таблицу свой GroupBox. Займемся первой таблицей. В свойстве Caption компонента GroupBox впишите “Личные данные”, это название отразится в заголовке панели. Далее на эту панель следует установить восемь компонентов DBEdit с вкладки DataControls палитры компонентов, два DBCheckBox для редактирования логических данных, и один компонент DBComboBox для списка, а также поясняющие компоненты Label. Немного доработаем компонент DBComboBox. Щелкните дважды по его свойству Items, открыв редактор. В нем введите две строки: Муж; ЖенДля таблицы Doljnost на панели GroupBox всего два компонента DBEdit и два поясняющих Label.

Для таблицы Adres используйте три DBEdit.

А для таблицы Telephones понадобится один DBEdit, один DBComboBox, сетка DBGrid и кнопка Button. Сетка нужна для контроля введенных телефонов, ведь здесь связь один-ко-многим, и телефонов может быть несколько. В редакторе Items компонента DBComboBox введите три строки: Рабочий, Домашний, Мобильный

Теперь займемся подключением компонентов контроля. Удерживая Shift, выделите все компоненты кроме Label на первой панели. В их свойстве DataSource выберите fDM.DSLichData, подключив компоненты к нужной таблице. Выделите первый DBEdit. В его свойстве DataField выберите поле “Фамилия”. Это свойство подключает выбранный компонент к определенному полю таблицы. Таким же образом подключите к соответствующим полям остальные компоненты. Затем подключайте компоненты других таблиц, каждое к своей таблице и к соответствующему полю. Сетка DBGrid подключается к fDM.DSTelephones и не имеет поля. Она отображает все видимые поля таблицы.

В правой нижней части для удобства пользователя установите навигационный компонент DBNavigator с вкладки Data Controls. Этот компонент предназначен для перемещения по записям, включения режима редактирования записи, сохранения или отмены сделанных изменений, добавления новой записи или удаления существующей. В его свойстве DataSource  выберите fDM.DSLichData, чтобы подключить компонент к главной таблице. Нам нужна от этого компонента только возможность перехода на начало или конец таблицы, на следующую или предыдущую запись. Поэтому раскройте его свойство VisibleButtons и переключите в False все кнопки, кроме nbFirst, nbPrior, nbNext и nbLast. Нажатие на эти кнопки приведет к вызову соответствующих методов компонента ADOTable. Эти методы делают следующее:

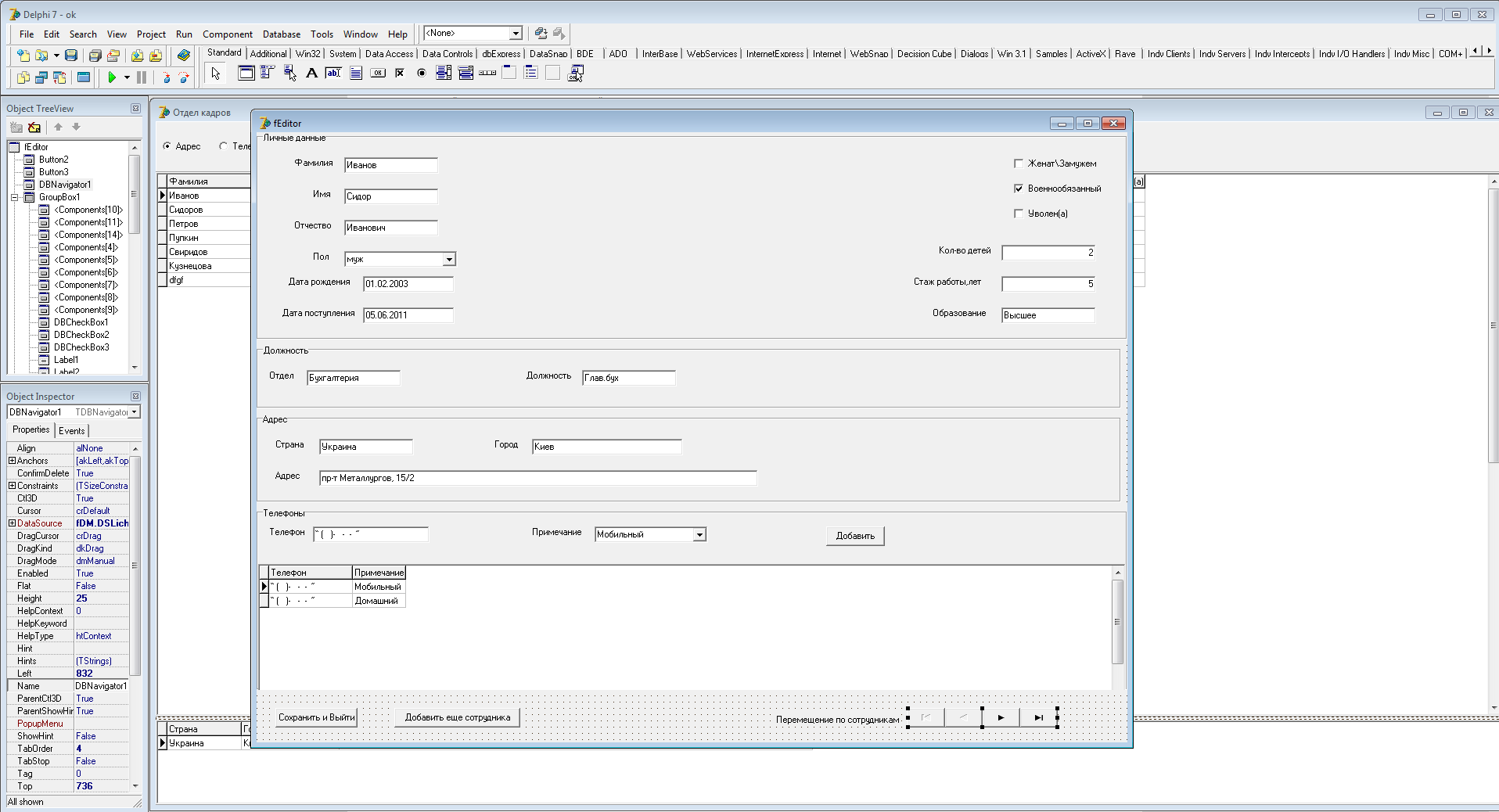
First – переход на первую запись таблицы;

Prior – переход на предыдущую запись;

Next – переход на следующую запись;

Last – переход на последнюю запись.

Когда у DBNavigator останется всего четыре кнопки, эти кнопки окажутся вытянутыми. Уменьшите ширину компонента, чтобы кнопки приняли более привычный вид.



# 5. Редактирование программного кода.

Связь между таблицами в нашей базе данных происходит по полю “Ключ”. На форме это поле отсутствует – его значение генерируется автоматически. После того, как мы внесли записи на первой панели GroupBox, их необходимо сохранить – для того, чтобы другие таблицы смогли связаться с внесенными записями.

Для сохранения изменений в таблице мы пропишем метод Post, который будет выполняться при выходе из панели GroupBox.

Выделите первый GroupBox, и дважды щелкните по событию onExit на вкладке Events инспектора объектов. Это событие происходит всякий раз, когда пользователь перейдет к другой панели GroupBox, либо к кнопкам, расположенным в нижней части окна:

procedure TfEditor.GroupBox1Exit(Sender: TObject);

begin

if fDM.TLichData.Modified then

fDM.TLichData.Post;

end;

Введенный код означает, что если запись была изменена, то следует ее сохранить. Сгенерируйте событие onExit для оставшихся панелей GroupBox и таким же образом сохраните изменения записей в соответствующих таблицах.

Далее сгенерируйте событие нажатия на кнопку “Добавить” в GroupBox с телефонными данными. Этой кнопкой мы будем добавлять новые записи в таблицу, ведь один сотрудник может иметь более одного телефона:

if fDM.TTelephones.Modified then

fDM.TTelephones.Post;

fDM.TTelephones.Append;

DBEdit14.SetFocus;

Вначале мы сохраняем измененные значения, если они были. Затем методом Append мы добавляем в таблицу новую запись. Далее мы переводит фокус ввода на DBEdit с телефонными номерами, чтобы пользователю не пришлось делать это самому.

В процедуре нажатия на кнопку “Сохранить и выйти”:

if fDM.TLichData.Modified then

fDM.TLichData.Post;

if fDM.TDoljnost.Modified then

fDM.TDoljnost.Post;

if fDM.TAdres.Modified then

fDM.TAdres.Post;

if fDM.TTelephones.Modified then

fDM.TTelephones.Post;

Close;

Для кнопки “Добавить сотрудника”:

fDM.TLichData.Append;

fDM.TDoljnost.Append;

fDM.TAdres.Append;

fDM.TTelephones.Append;

DBEdit1.SetFocus;

Здесь мы добавляем новую запись в каждую таблицу и переводим курсор в первый DBEdit, в котором редактируется фамилия.

Переходим к главной форме. Начнем с кнопки “Новый сотрудник”. Нам необходимо добавить новую запись в каждую таблицу, после чего открыть окно редактора:

fDM.TLichData.Append;

fDM.TDoljnost.Append;

fDM.TAdres.Append;

fDM.TTelephones.Append;

fEditor.ShowModal;

Сгенерируйте процедуру onClick для кнопки “Редактировать”. Тут будет лишь одна строчка кода: fEditor.ShowModal;

В результате откроется окно редактора, и компоненты будут отображать данные текущей записи. Предположим, пользователю будет удобней дважды щелкнуть по записи в верхней сетке DBGrid, чем нажимать кнопку. Поэтому выделите сетку с главной таблицей и сгенерируйте для нее событие onDBLClick. Там введите такую же строчку кода.

Перейдем к программированию радиокнопок. По нашему замыслу, при открытии программы в верхней сетке DBGrid будут отображаться данные из главной таблицы, а в нижней – из таблицы Adres. Также будет выделена радиокнопка с надписью “Адрес”. Если пользователю захочется посмотреть должность или телефоны текущего сотрудника, он будет щелкать соответствующую радиокнопку, и эти данные должны быть отображены в нижней DBGrid. Выделите первую радиокнопку с надписью “Адрес” и сгенерируйте для нее событие onClick, которое будет возникать, когда пользователь щелкнет по ней. В процедуре этого события впишите следующий код:

if RadioButton1.Checked then

DBGrid2.DataSource := fDM.DSAdres;

Здесь мы проверили, включена ли данная радиокнопка. Если да, то мы меняем связь нижней сетки DBGrid и подключаем ее к таблице Adres. Ведь связь сетки с таблицей осуществляется через соответствующий компонент DataSource, а у нас их четыре. Подключаясь то к одному, то к другому DataSource, мы можем программно менять отображенную в сетке таблицу.

Для события onClick радиокнопки с надписью “Телефоны”:

if RadioButton2.Checked then

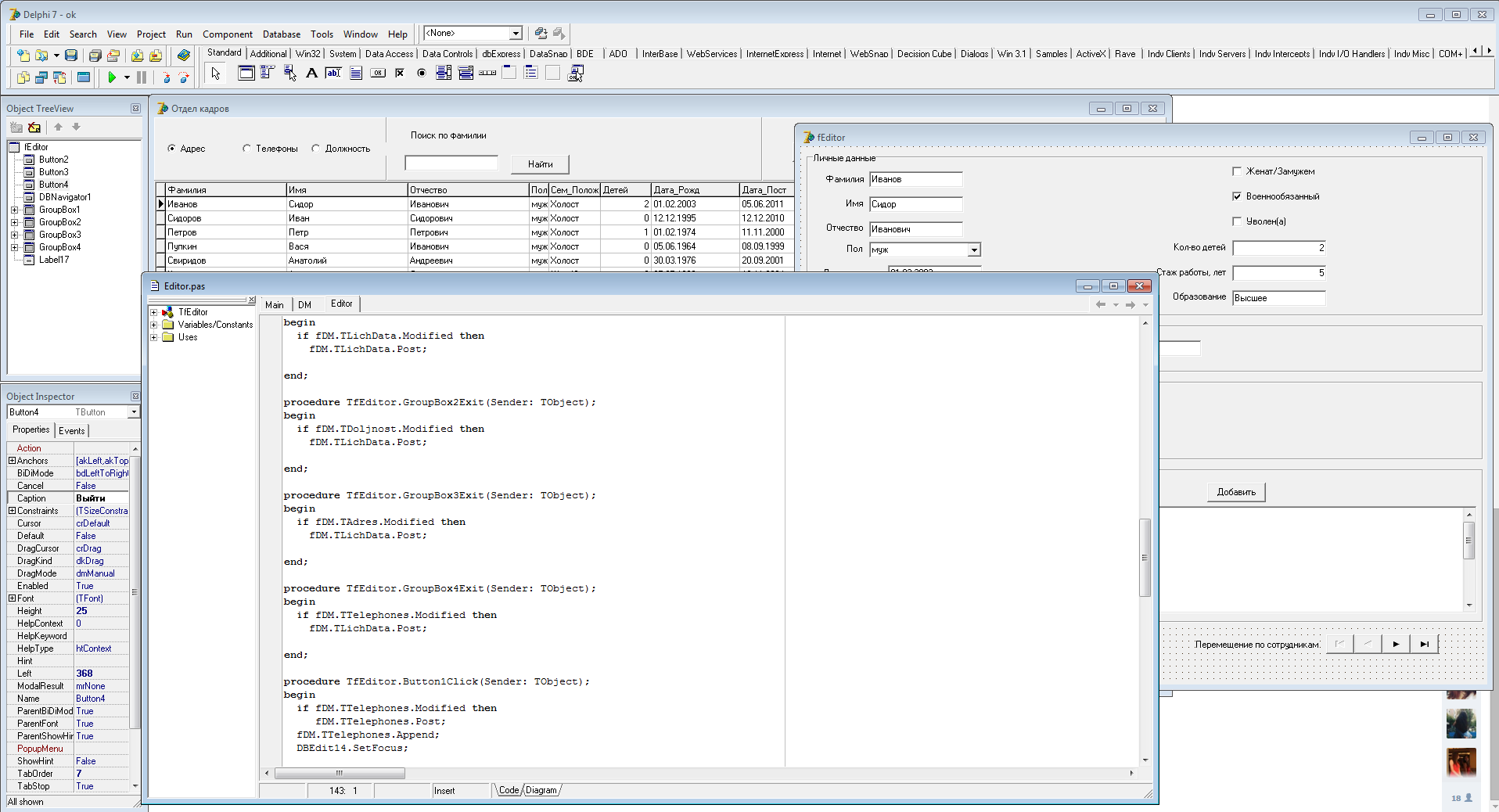
DBGrid2.DataSource := fDM.DSTelephones;

А для события onClick радиокнопки с надписью “Должность”, соответственно:

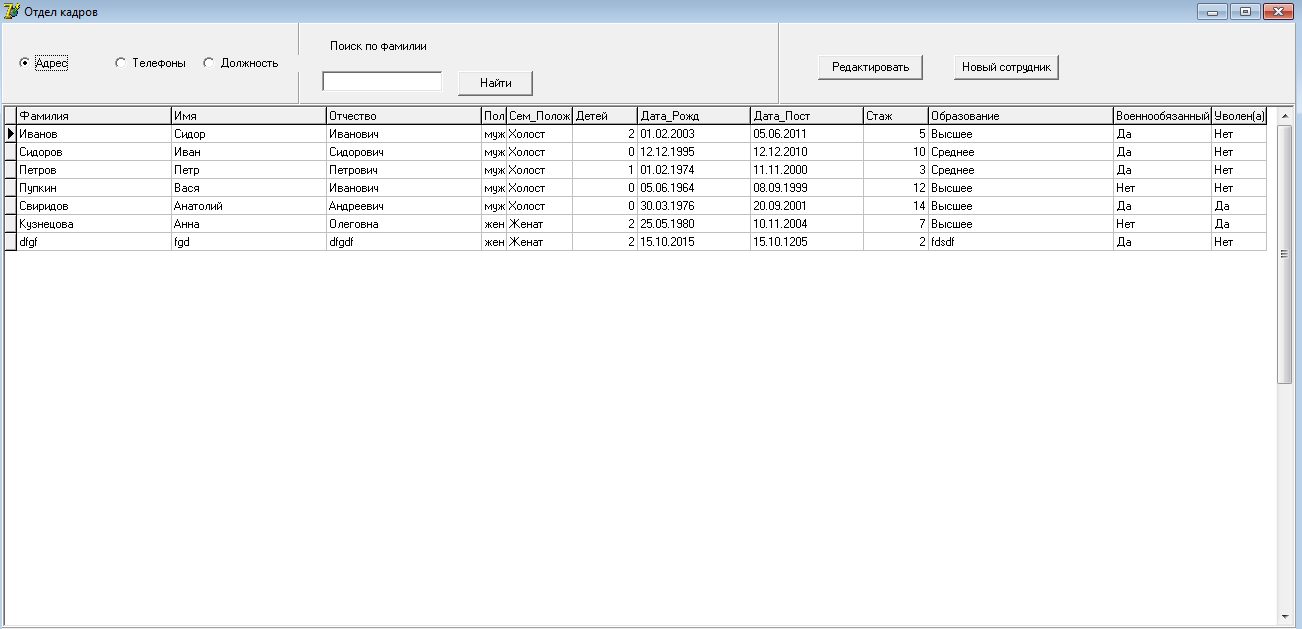
if RadioButton3.Checked then

DBGrid2.DataSource := fDM.DSDoljnost;

Таким образом, в нижней сетке мы отображаем то одну, то другую подчиненную таблицу, и всякий раз в этих таблицах будут показаны данные текущего сотрудника.



# Сотрудники

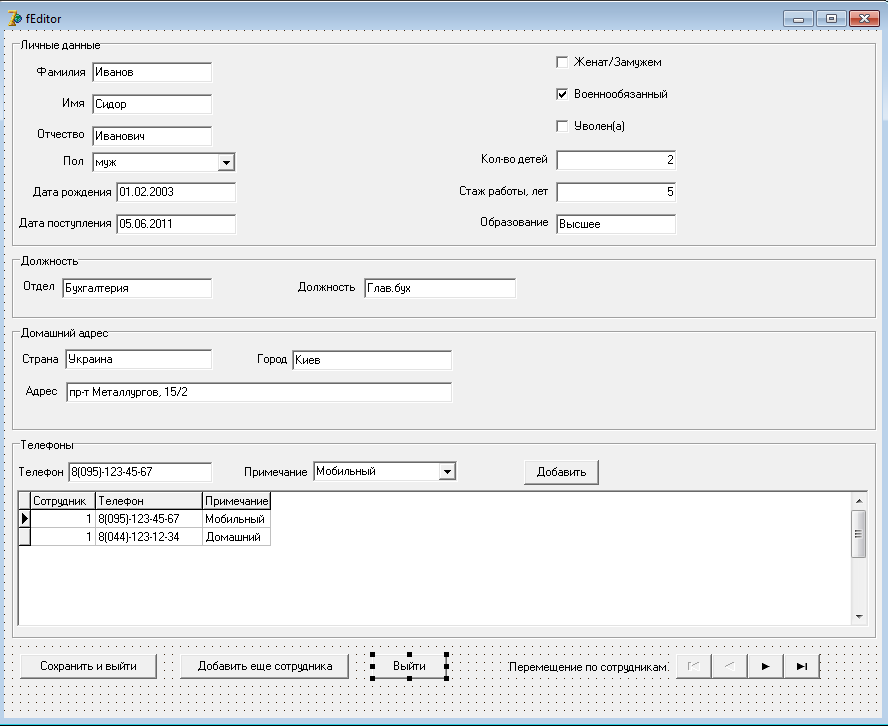


**Здесь вы сможете:**

* Просмотреть адрес сотрудника
* Просмотреть должность сотрудника
* Просмотреть телефон сотрудника
* Произвести поиск какого-либо сотрудника по фамилии
* Отредактировать данные любого из сотрудников
* Добавить нового сотрудника

## Добавление сотрудника

Для добавления сотрудника нажмите на «Новый сотрудник» или же в форме «Редактировать» нажмите «Добавить еще сотрудника». Перед вами должна появиться следующая форма:



В которой вы сможете:

- ввести фамилию, имя и отчество нового сотрудника, его родной город, а также контактный телефон.

Для закрытия данной формы без сохранения данных вы можете либо нажать крестик в правом верхнем углу, либо нажмите на кнопку «Выйти». Для сохранения данных и добавления сотрудника в базу данных нажмите кнопку «Сохранить и Выйти».

## Информация о сотруднике и редактирование

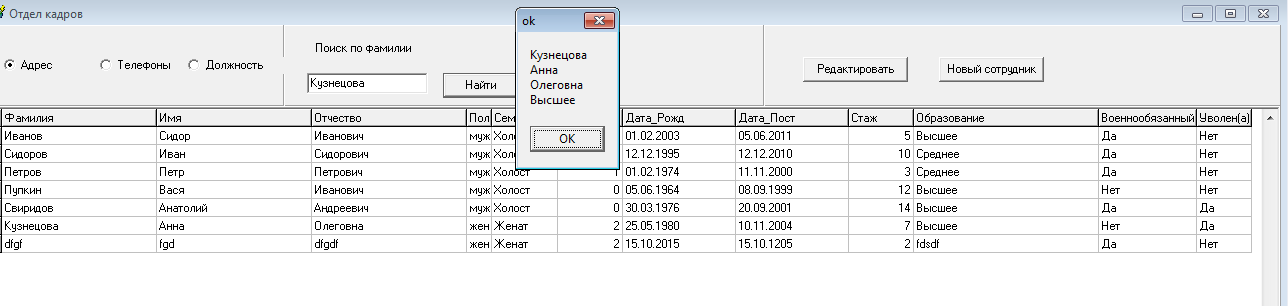
Для получения более подробной информации о сотруднике и его редактировании кликните один раз левой кнопкой мыши на соответствующую с искомым участником строку в таблице сотрудников.

Перед вами откроется та же форма, с помощью которой вы создавали нового сотрудника.

Также вы можете изменить данные о участнике, изменив их сначала в форме, а затем нажав на кнопку «Сохранить и Выйти».

## Поиск сотрудников

Чтобы найти какого-либо сотрудника вам надо ввести его фамилию в строку под «Поиск по фамилии» и потом нажать кнопку «Найти» и у вас появится отдельное окно , где вы сможете увидеть информацию о данном сотруднике.



## Код программы.

# Unit Main.

unit Main;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, DBGrids, StdCtrls, ExtCtrls;

type

TfName = class(TForm)

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

Splitter1: TSplitter;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

RadioButton3: TRadioButton;

Bevel1: TBevel;

Label1: TLabel;

Edit1: TEdit;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Bevel2: TBevel;

DBGrid1: TDBGrid;

DBGrid2: TDBGrid;

procedure Button3Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure DBGrid1DblClick(Sender: TObject);

procedure RadioButton1Click(Sender: TObject);

procedure RadioButton2Click(Sender: TObject);

procedure RadioButton3Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fName: TfName;

implementation

uses DM, Editor;

{$R \*.dfm}

procedure TfName.Button3Click(Sender: TObject);

begin

fDM.TLichData.Append;

fDM.TDoljnost.Append;

fDM.TAdres.Append;

fDM.TTelephones.Append;

fEditor.ShowModal;

end;

procedure TfName.Button2Click(Sender: TObject);

begin

feditor.showmodal;

end;

procedure TfName.DBGrid1DblClick(Sender: TObject);

begin

feditor.showmodal;

end;

procedure TfName.RadioButton1Click(Sender: TObject);

begin

if RadioButton1.Checked then

DBGrid2.DataSource := fDM.DSAdres;

end;

procedure TfName.RadioButton2Click(Sender: TObject);

begin

if RadioButton2.Checked then

DBGrid2.DataSource := fDM.DSTelephones;

end;

procedure TfName.RadioButton3Click(Sender: TObject);

begin

if RadioButton3.Checked then

DBGrid2.DataSource := fDM.DSDoljnost;

end;

procedure TfName.Button1Click(Sender: TObject);

var

myLookup: Variant;

s : String;

begin

myLookup := fDM.TLichData.Lookup('Фамилия' , Edit1.Text,

Фамилия;Имя;Отчество;Образование);

if VarType(myLookup) = varNull then

ShowMessage('Сотрудник с такой фамилией не найден!')

else if VarType(myLookup) = varEmpty then

ShowMessage(Запись не найдена!')

else if VarIsArray(myLookup) then begin

s := myLookup[0] + #13 + myLookup[1] + #13 +

myLookup[2] + #13 + myLookup[3];

ShowMessage(s);

end; //else if

end;

end.

# Unit DM.

unit DM;

interface

uses

SysUtils, Classes, DB, ADODB;

type

TfDM = class(TDataModule)

ADOConnection1: TADOConnection;

DSLichData: TDataSource;

DSDoljnost: TDataSource;

DSTelephones: TDataSource;

DSAdres: TDataSource;

TLichData: TADOTable;

TDoljnost: TADOTable;

TTelephones: TADOTable;

TAdres: TADOTable;

TLichDataDSDesigner: TAutoIncField;

TLichDataDSDesigner2: TWideStringField;

TLichDataDSDesigner3: TWideStringField;

TLichDataDSDesigner4: TWideStringField;

TLichDataDSDesigner5: TWideStringField;

TLichData\_: TBooleanField;

TLichDataDSDesigner6: TWordField;

TLichData\_2: TDateTimeField;

TLichData\_3: TDateTimeField;

TLichDataDSDesigner7: TWordField;

TLichDataDSDesigner8: TWideStringField;

TLichDataDSDesigner9: TBooleanField;

TDoljnostDSDesigner: TIntegerField;

TDoljnostDSDesigner2: TWideStringField;

TDoljnostDSDesigner3: TWideStringField;

TTelephonesDSDesigner: TIntegerField;

TTelephonesDSDesigner2: TWideStringField;

TTelephonesDSDesigner3: TWideStringField;

TAdresDSDesigner: TIntegerField;

TAdresDSDesigner2: TWideStringField;

TAdresDSDesigner3: TWideStringField;

TAdres\_: TWideStringField;

TLichDataDSDesigner10: TBooleanField;

private

{ Private declarations }

public

{ Public declarations }

//procedure MyLocate(s: String);

ed: String; //òåêñò èç Edit1

end;

var

fDM: TfDM;

implementation

uses Main;

{$R \*.dfm}

end.

# Unit Editor.

unit Editor;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, DBCtrls, StdCtrls, Grids, DBGrids, Mask;

type

TfEditor = class(TForm)

GroupBox1: TGroupBox;

GroupBox2: TGroupBox;

GroupBox3: TGroupBox;

GroupBox4: TGroupBox;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

DBEdit6: TDBEdit;

DBEdit7: TDBEdit;

DBEdit8: TDBEdit;

DBCheckBox1: TDBCheckBox;

DBCheckBox2: TDBCheckBox;

DBComboBox1: TDBComboBox;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

DBEdit9: TDBEdit;

DBEdit10: TDBEdit;

Label10: TLabel;

Label11: TLabel;

Label12: TLabel;

Label13: TLabel;

DBEdit11: TDBEdit;

DBEdit12: TDBEdit;

Label14: TLabel;

DBEdit13: TDBEdit;

Label15: TLabel;

Label16: TLabel;

DBEdit14: TDBEdit;

DBComboBox2: TDBComboBox;

DBGrid1: TDBGrid;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Label17: TLabel;

DBNavigator1: TDBNavigator;

DBCheckBox3: TDBCheckBox;

Button4: TButton;

procedure GroupBox1Exit(Sender: TObject);

procedure GroupBox2Exit(Sender: TObject);

procedure GroupBox3Exit(Sender: TObject);

procedure GroupBox4Exit(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fEditor: TfEditor;

implementation

uses DM, Main;

{$R \*.dfm}

procedure TfEditor.GroupBox1Exit(Sender: TObject);

begin

if fDM.TLichData.Modified then

fDM.TLichData.Post;

end;

procedure TfEditor.GroupBox2Exit(Sender: TObject);

begin

if fDM.TDoljnost.Modified then

fDM.TLichData.Post;

end;

procedure TfEditor.GroupBox3Exit(Sender: TObject);

begin

if fDM.TAdres.Modified then

fDM.TLichData.Post;

end;

procedure TfEditor.GroupBox4Exit(Sender: TObject);

begin

if fDM.TTelephones.Modified then

fDM.TLichData.Post;

end;

procedure TfEditor.Button1Click(Sender: TObject);

begin

if fDM.TTelephones.Modified then

fDM.TTelephones.Post;

fDM.TTelephones.Append;

DBEdit14.SetFocus;

end;

procedure TfEditor.Button2Click(Sender: TObject);

begin

if fDM.TLichData.Modified then

fDM.TLichData.Post;

if fDM.TDoljnost.Modified then

fDM.TDoljnost.Post;

if fDM.TAdres.Modified then

fDM.TAdres.Post;

if fDM.TTelephones.Modified then

fDM.TTelephones.Post;

Close;

end;

procedure TfEditor.Button3Click(Sender: TObject);

begin

fDM.TLichData.Append;

fDM.TDoljnost.Append;

fDM.TAdres.Append;

fDM.TTelephones.Append;

DBEdit1.SetFocus;

end;

procedure TfEditor.Button4Click(Sender: TObject);

begin

Close;

end;

end.