Borland®

Delphi и базы данных

Borland®

Введение в базы данных

Базой данных (БД) называется электронное хранилище информации, доступ к которому имеет один или несколько компьютеров.

Таблицы имеющие связи между собой, называют ***реляционными***, и *базы данных*, в которых имеются взаимосвязанные таблицы, также называются ***реляционными***.

Часто пользовательские приложения не работают с базами данных напрямую. Имеются специальные программы, называемые **системы управления базами данных** (***СУБД***), которые служат посредниками между базой данных и пользовательским приложением. Такой подход называют архитектурой клиент-*сервер*, а такие *СУБД* часто называют серверами баз данных.

Основой любой БД является таблица. **Таблица** - это файл определенного формата с данными, представленными в табличном виде, например:
Такая таблица состоит из полей и записей.

№	Фамилия	Имя	Отчество
1	Иванов	Иван	Иванович
2	Петров	Петр	Петрович
3	Сидоров	Сидор	Сидорович

Поле - столбец таблицы, имеющий название, *тип данных* и размер. *Поле* предназначено для описания отдельного *атрибута записи*. Например, *поле* "№" имеет целочисленный *тип данных*, а *поле* "Фамилия" - *строковый*.

Запись - строка таблицы, описывающая какой-то *объект*, или иначе, набор атрибутов какого-то объекта. Например, строка под номером 1 описывает человека - Иванова Ивана Ивановича.

Первичный ключ - это *поле* или набор *полей*, однозначно идентифицирующих *запись*. В ключевом *поле* не может быть двух записей с одинаковым значением.

Индекс - это *поле* или набор *полей*, которые часто используются для сортировки или поиска данных. Индексные поля еще называют вторичными ключами. В отличие от первичных ключей, поля для индексов могут содержать как уникальные, так и повторяющие значения.

Связи (отношения)

Реляционные связи (отношения) между таблицами предназначены для разбивки таблиц на самодостаточные части. Рассмотрим пример.

№	Фамилия	Имя	Отчество	Экзамен
1	Иванов	Иван	Иванович	Математика
2	Иванов	Иван	Иванович	Физика
3	Петров	Петр	Петрович	Рус. Яз.
4	Петров	Петр	Петрович	Литература
5	Сидоров	Сидор	Сидорович	Сопр.мат.
6	Сидоров	Сидор	Сидорович	Теор.вер.

Разобьём эту таблицу на две разных таблицы, имеющие релятивную *связь*:

№	Фамилия	Имя	Отчество
1	Иванов	Иван	Иванович
2	Петров	Петр	Петрович
3	Сидоров	Сидор	Сидорович

Студент	Экзамен
1	Математика
1	Физика
2	Рус. Яз.
2	Литература
3	Сопр.мат.
3	Теор.вер.

При создании связей, как правило, одна *таблица* называется **главной (master)**, другая - **подчиненной (details)**. *Связь*, представленная на предыдущем рисунке называется ***отношением один-ко-многим***.

Отношение ***один-к-одному*** подразумевает, что одной записи в главной таблице соответствует одна *запись* в подчиненной таблице.

№	Студент	Группа
1	Иванов И.И.	11
2	Петров П.П.	22
3	Сидоров С.С.	32

№	Дом. адрес	Телефон
1	Москва, Ленина 3/10	111-11-11
2	С-Петербург, Гагарина 5/14	222-22-22
3	Москва, пр-т Строителей 20/1	333-33-33

Отношение один-к-одному

№	Покупатель
1	Иванов И.И.
2	Петров П.П.
3	Сидоров С.С.

Код	Товар	Едизм	Цена р.
А-01	Макароны	кг.	100
А-02	Сахар	кг.	150
С-34	Хлеб	шт.	50

Отношение многие-ко-многим

Ссылочная целостность

Ссылочной целостностью называют особый механизм, осуществляемый средствами *СУБД* или программистом, ответственный за поддержание непротиворечивых данных в связанных релятивными отношениями *таблицах*. ***Ссылочная целостность*** подразумевает, что в *таблицах*, имеющих релятивные связи, нет ссылок на несуществующие записи.

Существует несколько видов изменений данных, которые могут привести к нарушению ссылочной целостности:

1. Удаляется запись в *родительской таблице*, но не удаляются соответствующие связанные записи в *дочерней таблице*.
2. Изменяется запись в *родительской таблице*, но не изменяются соответствующие ключи в *дочерней таблице*.
3. Изменяется ключ в *дочерней таблице*, но не изменяется значение связанного поля родительской таблицы.

Нормализация базы данных

Существуют некоторые правила, помогающие улучшить проектируемую *БД*. Такие правила носят рекомендательный характер, и называются *нормализацией базы данных*.

Процесс нормализации данных заключается в устранении избыточности данных в таблицах.

Существует несколько *нормальных форм*:

Первая нормальная форма требует, чтобы каждое *поле* таблицы *БД* было неделимым (атомарным) и не содержало повторяющихся групп. *Неделимость* означает, что в таблице не должно быть *полей*, которые можно разбить на более мелкие поля. Рисунок:

№	Студент 1	Студент 2	Студент 3
1	Иванов И.И.	Петров П.П.	Сидоров С.С.

Повторяющиеся группы

Вторая нормальная форма (*2НФ*) требует, чтобы *таблица* удовлетворяла всем требованиям первой нормальной формы, и чтобы любое не ключевое *поле* однозначно идентифицировалось полным набором ключевых *полей*. Рассмотрим пример: некоторые студенты посещают спортивные платные секции, и ВУЗ взял на себя оплату этих секций.

№ студента	Секция	Плата
100	Плавание	100
110	Скейтборд	150
112	Теннис	175
254	Плавание	100
260	Теннис	175

Нарушение второй нормальной формы

№ студента	Секция
100	Плавание
110	Скейтборд
112	Теннис
254	Плавание
260	Теннис

Ключ: № студента

Секция	Плата
Плавание	100
Скейтборд	150
Теннис	175

Ключ: Секция

Правильная вторая нормальная форма

Третья нормальная форма (*3НФ*) требует, чтобы в таблице не имелось транзитивных зависимостей между не ключевыми полями, то есть, чтобы *значение* любого поля, не входящего в *первичный ключ*, не зависело от другого поля, также не входящего в *первичный ключ*. Допустим, в нашей студенческой базе данных есть *таблица* с расходами на спортивные секции:

Секция	Плата	Кол-во студентов	Общая стоимость
Плавание	100	2	200
Скейтборд	150	1	150
Теннис	175	2	350

Нарушение третьей нормальной формы

Механизм доступа к данным - это программный инструмент, позволяющий получить *доступ* к базе данных и ее таблицам. Как правило, это *драйвер* в виде *.dll файлов, который устанавливается на ПК разработчика (и клиента), и который используется программой для связи с *БД*.

Сравнение BDE и ADO

Borland Database Engine (BDE) - этот *механизм доступа к данным* позволяет обращаться к локальным и *файл-серверным* форматам баз данных dBase, FoxPro и Paradox, к различным серверам SQL и ко многим другим источникам данных, *доступ* которых поддерживался при помощи драйверов *ODBC*. Например, с помощью *BDE* можно напрямую работать с табличными файлами MS Excel. Но механизм доступа *BDE* признается устаревшим.

В данный момент многие инструменты Delphi являются кросс - платформенными.

ActiveX Data Object (ADO) - это *механизм доступа к данным*, разработанный корпорацией Microsoft. Если точнее, то *ADO* - это надстройка над технологией **OLE DB**, посредством которой можно связываться с различными данными приложений Microsoft. Технология *ADO*, как и *BDE*, независима от конкретного сервера *БД*, имеет поддержку как локальных баз данных различных типов, так и некоторых клиент-серверных *БД*. Плюсов у этой технологии много. Драйверы, разработанные корпорацией Microsoft для собственных нужд, более надежные, чем драйверы сторонних производителей. Поэтому если вам требуется работать с базами данных *MS Access* или для архитектуры клиент-*сервер* использовать MS SQL Server, то использование *ADO* будет наиболее предпочтительным. Кроме того, имеется плюс и в вопросе распространения программ - во всех современных Windows встроены драйверы *ADO*.

Основными компонентами являются:

- *TADOConnection* (для подключения к *БД*);

-*TADOTable* (аналог *TTable* из *BDE*);

-*TADOQuery* (аналог *TQuery* из *BDE*, для выполнения запросов и получения набора данных);

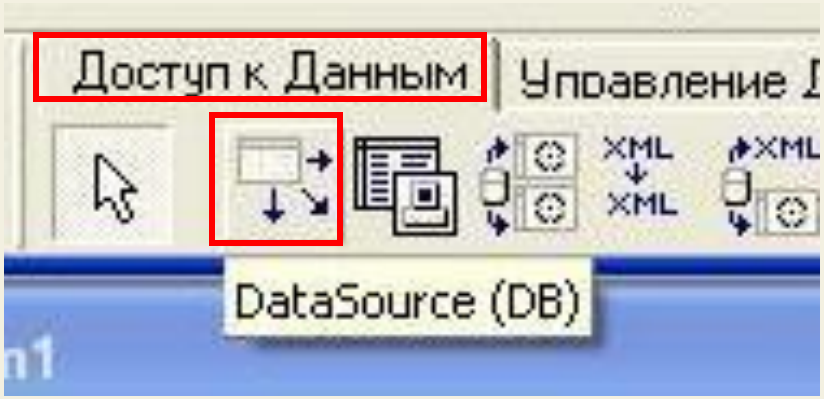
-*TADODataSet* (предназначенный для набора данных, полученных через *SQL-запрос*).

Borland®

Использование компонент Delphi для работы с базами данных

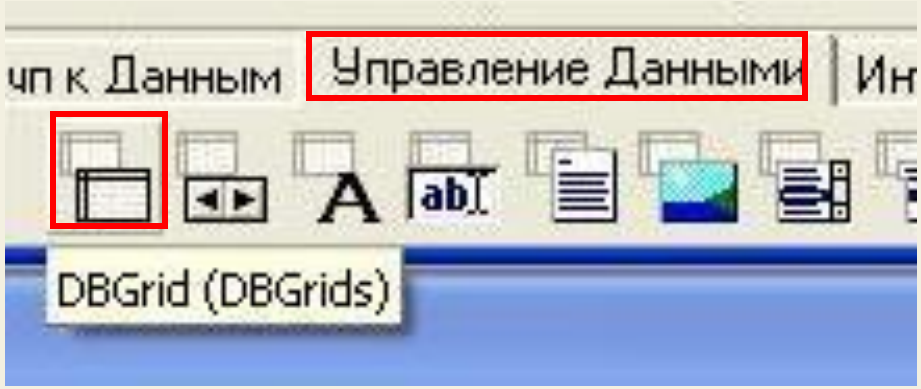
Начнем с обзора некоторых компонент, которые понадобятся нам для создания приложения, использующего локальную базу данных

1. Компонент **TTable** – основной компонент базы, через который идет обращение к конкретной таблице конкретной базы данных. Находится он на вкладке **BDE**

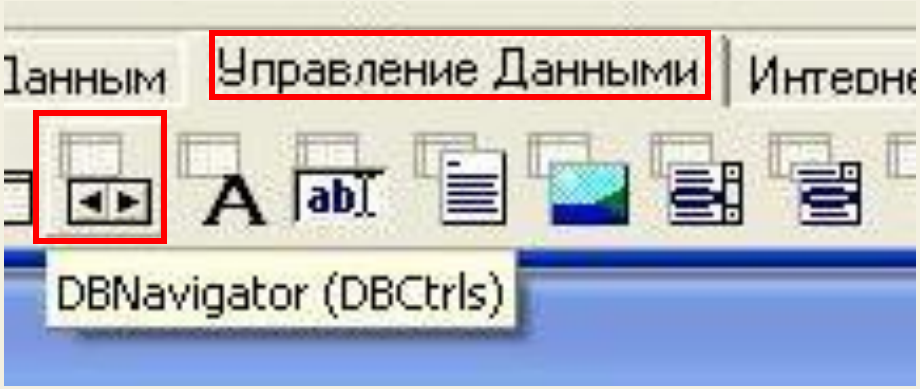


2. Компонент **TData Sourse** – он связывает наш компонент **TTable** с конкретной таблицей на нашем диске. Находится на вкладке **Доступ к Данным**

Начнем с обзора некоторых компонент, которые понадобятся нам для создания приложения, использующего локальную базу данных



3. Компонент **TDBGrid** – визуальный компонент, который отображает таблицу и ее содержимое на форме (без него мы не сможем увидеть таблицу, хотя программно можем с ней работать). Находится на вкладке **Управление данными**



2. Компонент **TDBNavigator** – визуальный компонент, который позволяет осуществлять навигацию по базе. Находится на вкладке **Управление данными**

После знакомства с компонентами начнем создание приложения, причем в качестве базы данных используем готовую базу с описанием и фото рыбок, имеющуюся в Delphi в качестве демонстрационной базы

Начнем по шагам

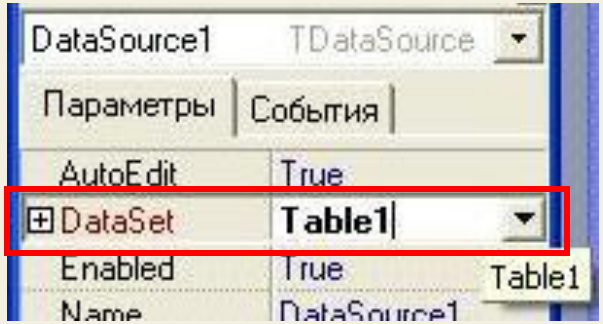
ШАГ 1

Положим на форму компонент **TTable** с вкладки BDE. Сейчас присоединим его к конкретной базе данных на нашем диске

Для этого раскрываем свойство **DataBaseName** и выбираем базу данных **DBDEMOS** – это демонстрационная база, входящая в состав Delphi

Дальше раскроем свойство **TableName** и выберем среди нескольких входящих таблиц файл **biolife.db** – это и есть таблица, которая содержит описание и фото рыбок

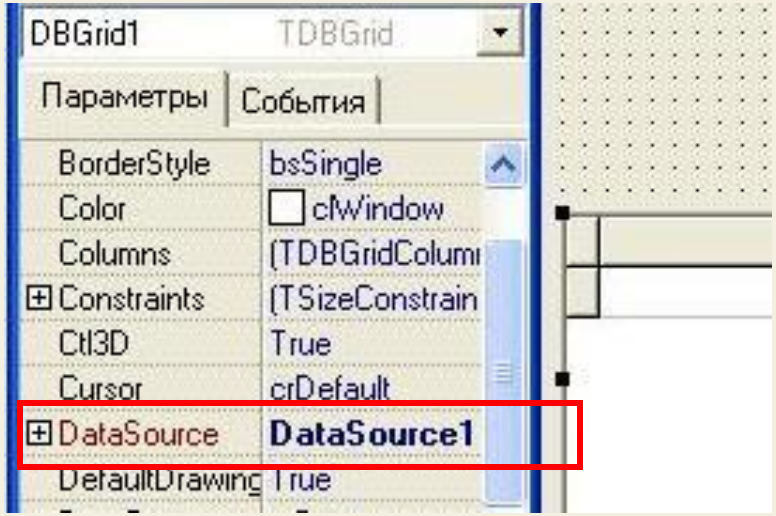
ШАГ 2



Ложим на форму компонент **TDataSourse** – он будет связывать визуальные компоненты, которые отображают содержимое таблицы с компонентом **TTable**

Находим свойство **DataSet** у этого компонента и в выпадающем списке указываем на **Table1**

ШАГ 3



Ставим на форму компонент **DBGrid** – он и будет отображать нашу таблицу с рыбками

В свойстве **DataSourse** выбираем источник данных – **DataSourse1**

ШАГ 4

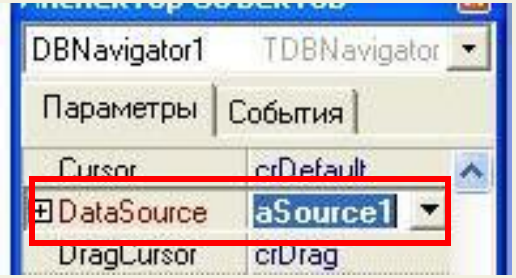
А сейчас **делаем таблицу активной**: у компонента **TTable** устанавливаем свойство **Active** в **True**.

В результате в **DBGrid** мы видим содержание таблицы :

Species No	Category	Common_Name	Species Name	Length (cm)	Length_In	Notes	Graphic
90030	Snapper	Red Emperor	Lutjanus sebae	60	0472440945	(MEMO)	(GRAPHIC)
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus	229	4803149606	(MEMO)	(GRAPHIC)
90070	Angelfish	Blue Angelfish	Pomacanthus nauarchus	30	0236220472	(MEMO)	(GRAPHIC)
90080	Cod	Lunartail Rockcod	Variola louti	80	6062992126	(MEMO)	(GRAPHIC)

Можно откомпилировать приложение и поработать с таблицей – мы можем просматривать и редактировать эту базу

ШАГ 5



Для удобства работы с таблицей поместим на форму элемент **DBNavigator** с вкладки **Управление данными** и в инспекторе объектов поставим его свойство **DataSource** указывающим на тот же **DataSource1**, что и для **DBGrid** - сейчас работать с таблицей стало гораздо удобнее

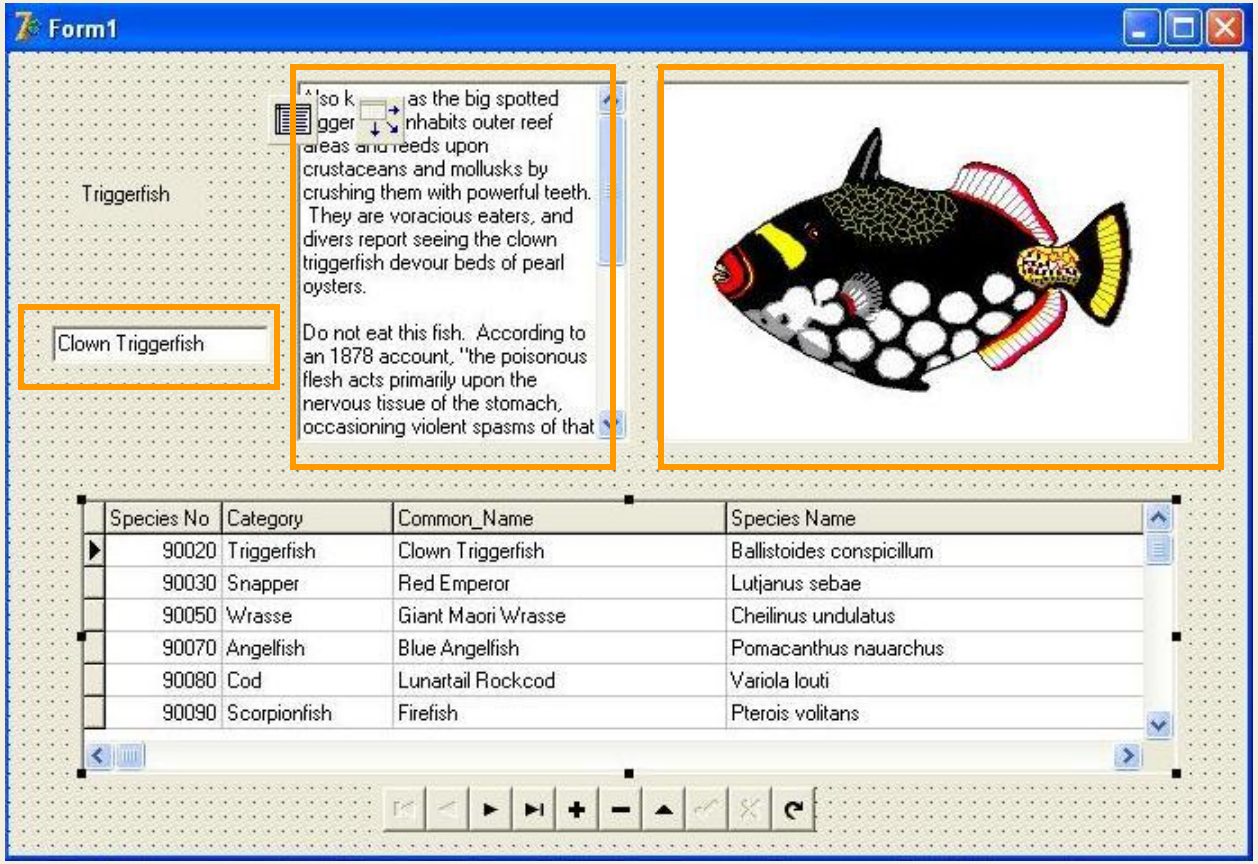
ШАГ 6

В Delphi имеется ряд компонент для отображения содержания отдельных ячеек – **DBEdit, DBMemo, DBImage** и др. Все они находятся на вкладке Управление данными . Поместим их на форму и свяжем с определенными столбцами таблицы: для каждого из этих компонентов укажем свойство **DataSourse** в **DataSourse1**, а свойство **DataField**, следующим образом:

DBEdit - ассоциируем с полем Common_name

DBMemo - ассоциируем с полем Notes

DBImage - ассоциируем с полем Graphic



DBImage, отображающий ячейку с рисунком рыбки

DBMemo, отображающий ячейку с описанием

DBEdit , отображающий ячейку с именем

Сейчас можно откомпилировать программу и поработать с нашей базой данных: в компонентах **DBEdit, DBMemo, DBImage** отображаются соответственно имя, описание и рисунок рыбки

Итак, мы создали программу для работы с демонстрационной (с готовой) базой данных, но сами базу мы не создавали

Для создания базы данных (таблицы) существует программа **Database Desktop**, входящая в состав Delphi

Borland®

Создаем приложение с базой данных MS Access

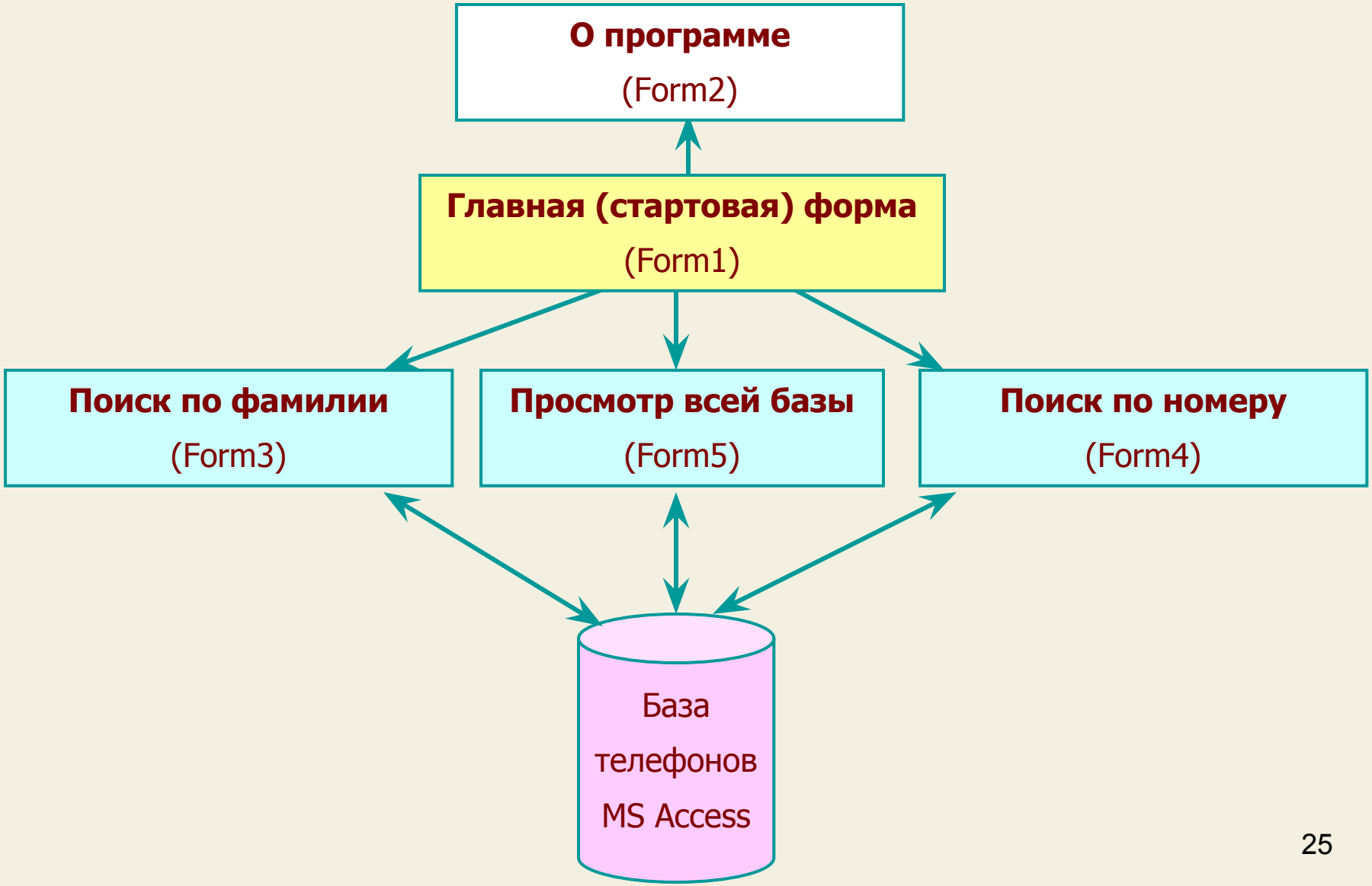
Для работы с базой данных **сначала создадим ее в MS Access**. Пусть это будет телефонный справочник с полями ФИО, ТЕЛЕФОН и АДРЕС

Таблица1 : таблица

ФИО	телефон	Адрес
Абрамов	50212	
Аскаров	51021	
Астахов	51236	
Ахтемзянов	51010	
Бандалетов	52441	
Бахвалов	52361	
Быков	52330	
Димиев	51787	
Домнин	51252	
Дукалис	51332	
Закиров	52110	
Зарипов	51233	
Захватаев	52447	
Иванов	51002	

Заполним таблицу произвольными значениями и сохраним ее. Сейчас можно приступать к созданию приложения. Нашим приложением будет **электронный телефонный справочник с функциями поиска по номеру или по фамилии**

Рассмотрим структуру нашей программы:



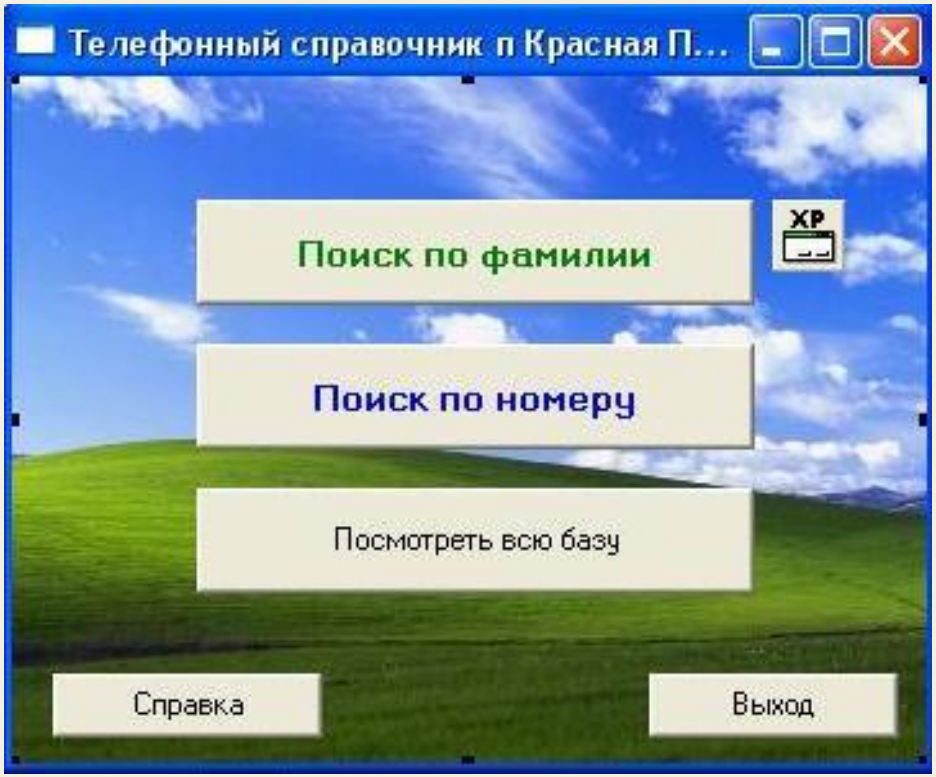
Как видно из схемы, наше приложение содержит 5 форм и опирается на базу Ms Access

ШАГ 1

Создадим 5 форм, познакомим их, назовем заголовки форм, выберем размеры и стили форм

ШАГ 2

На **главной** (стартовой) **форме** (Form1) расположим компоненты:

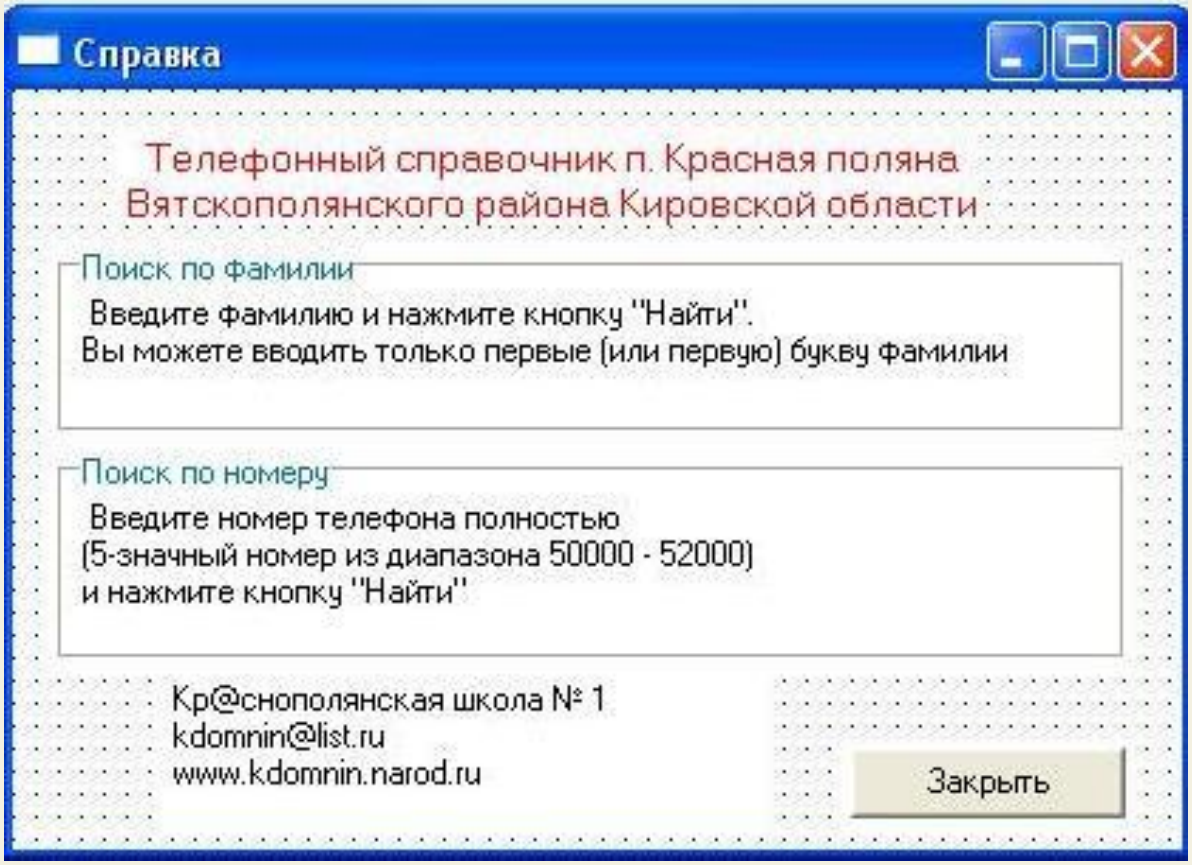


1. Кнопка – **Поиск по фамилии**
2. Кнопка – **Поиск по номеру**
3. Кнопка – **Посмотреть всю базу**
4. Кнопка – **Справка**
5. Кнопка – **Выход**
6. Манифест XP

Для всех кнопок запишем соответствующий код (открытие соответствующей формы – **ShowModal**, выход – **close**)

ШАГ 3

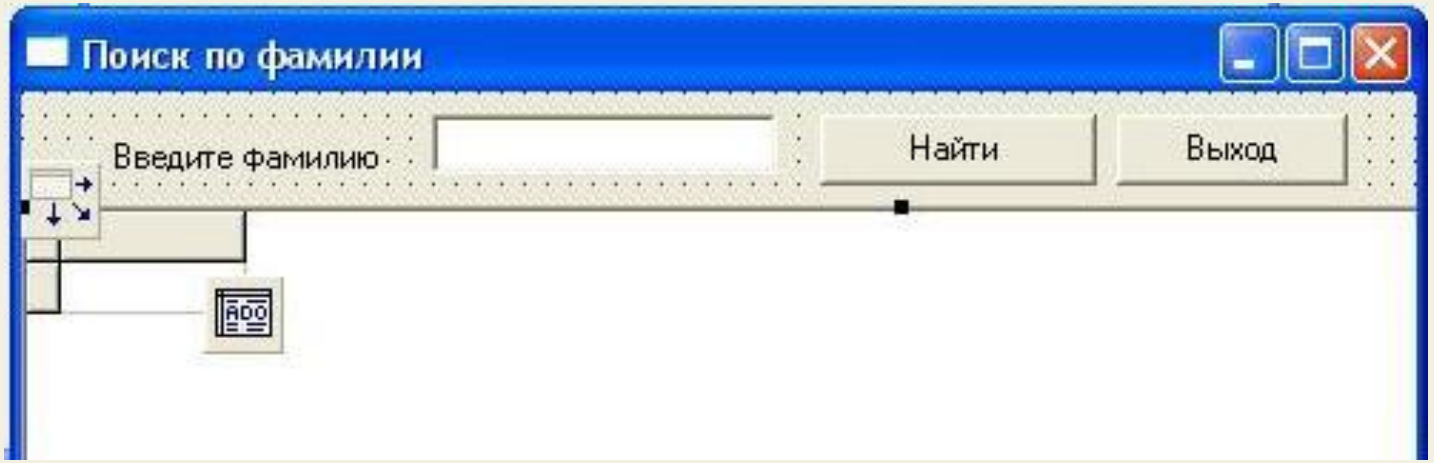
Разработаем дизайн формы **Справка**



Расположим на ней информацию по работе с программой с помощью соответствующих компонент и кнопку **Закрыть**, для которой запишем код выхода

ШАГ 4

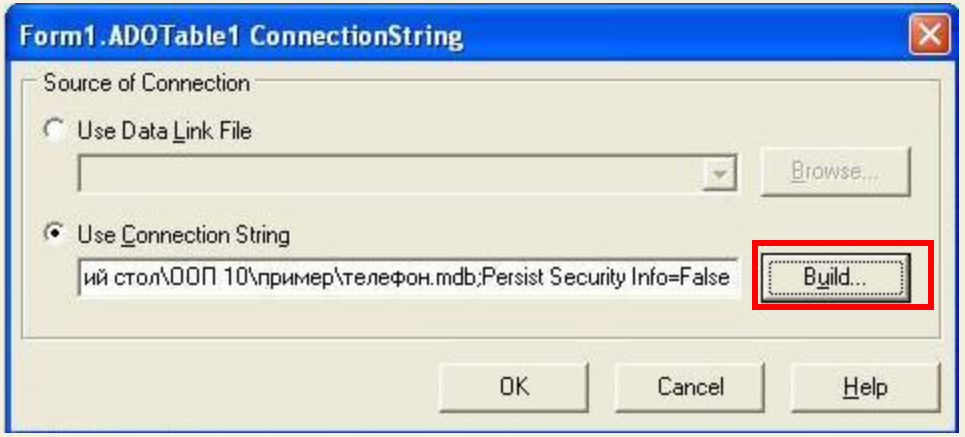
Разрабатываем форму **Поиск по фамилии**. Эта форма должна быть связана с базой телефонов MS Access



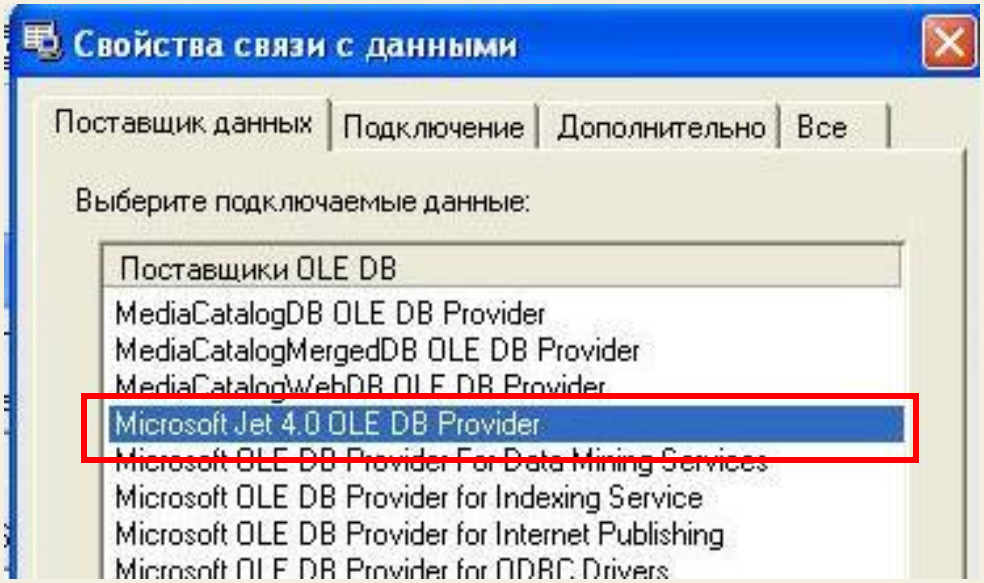
Для связи формы с Access используем следующие компоненты:

1. Вместо **TTable**, которую мы использовали в первом примере, для связи с Access служит «свой» компонент – **ADOTable**, который находится на вкладке ADO.

   Помещаем его на форму и привязываем к таблице телефонных номеров. Для этого раскрываем свойство **ConnectionString** и нажимаем кнопку **Build**

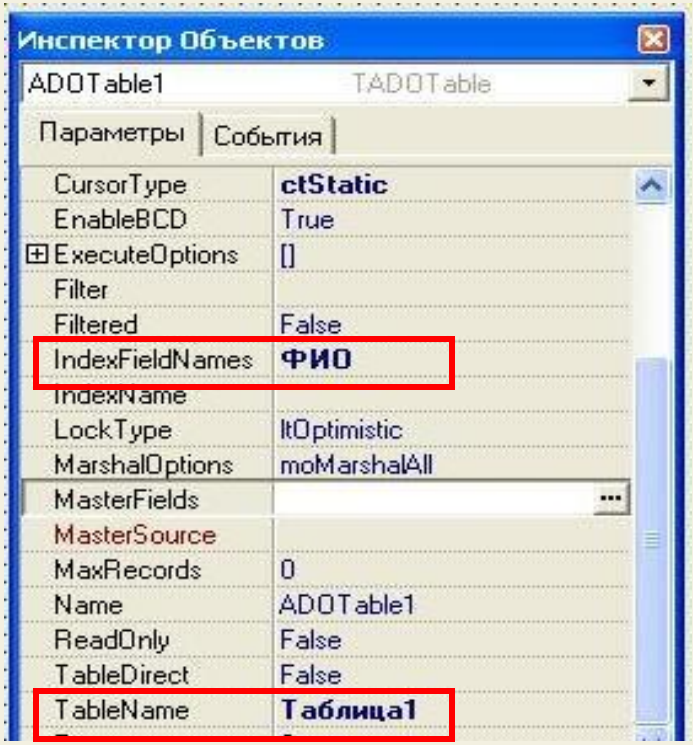


2. Открывается окно связи с данными, в котором на вкладке **Поставщик** данных выбираем **Microsoft Jet 4.0 Ole DB Provider**



3. Переходим на вкладку **Подключение** и выбираем через кнопку обзора нашу базу (**телефон.mdb**)

Здесь же можно проверить подключение, задать пароль и права доступа к базе

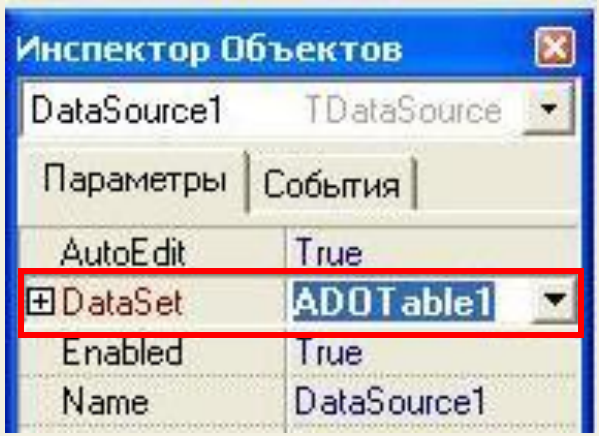


4. Устанавливаем свойства **ADOTable**:

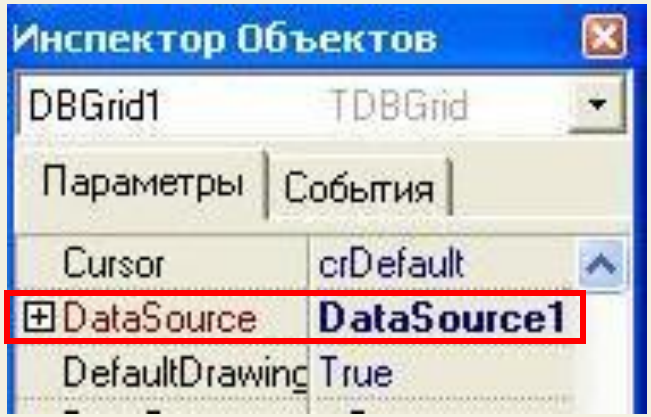
TableName – в раскрывающемся списке выбираем нашу таблицу (у меня она названа **Таблица1**

IndexFieldName – **ФИО** (данные будут сортированы по полю ФИО)

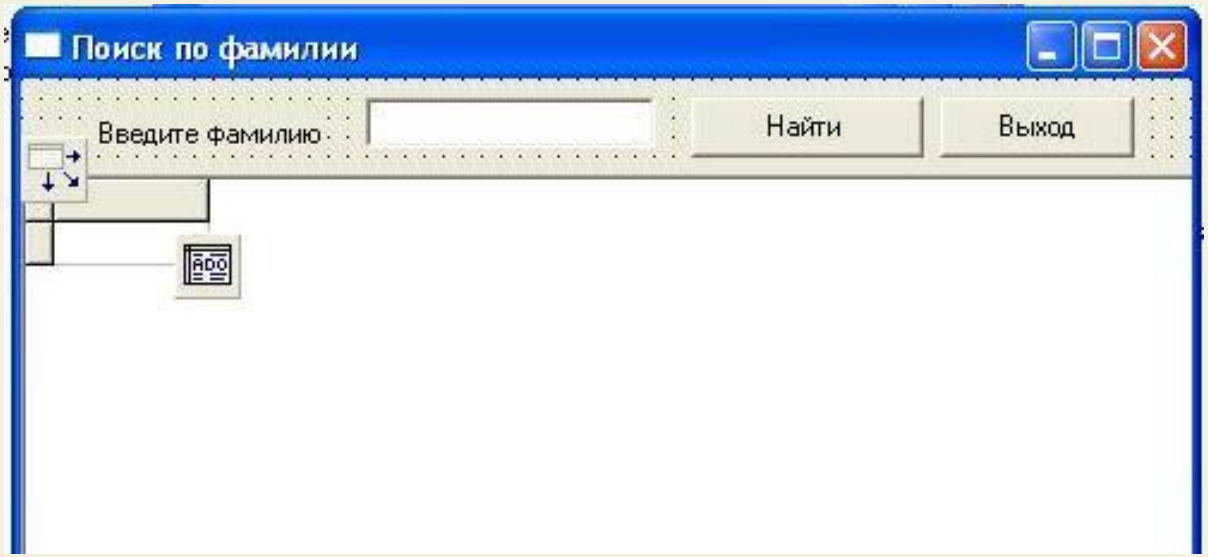
Свойство **Active** ставим в **True**



5. Помещаем на форму элемент **DataSourse** и его свойство **DataSet** устанавливаем через раскрывающийся список в **ADOTable1**



6. Ложим на форму компонент для отображения данных таблицы – **DBGrid**, у которого источником данных выбираем **DataSourse1** – и мы сразу видим, что в **DBGrid появились данные из нашей таблицы**



7. Помещаем на форму кнопки **Найти** и **Выход**, Label «**Введите фамилию**» и поле **Edit** для ввода фамилии

8. В результате мы имеем следующую форму

Свойство **Visible** у **DBGrid** – а сделаем **False**, чтобы при открытии формы поиска не было видно содержимого таблицы

Сейчас запишем код кнопки **Найти**

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if edit1.Text='' then
showmessage('Нербходимо ввести хотя бы часть фамилии')
else
begin
dbgrid1.Visible:=true;
Adotable1.Active:=true;
AdoTable1.Locate('ФИО',Edit1.text,[loPartialKey])
end;
end;
```

Если фамилия не введена, то выводим об этом сообщение

Если фамилия введена, то видимость грида делаем True, активизируем ADOTable и осуществляем поиск по полю ФИО с помощью метода Locate

Ищем по полю ФИО

Параметр поиска, позволяющий искать по первым буквам фамилии

Образец для поиска берем из Edit – а, в который вводится фамилия для поиска

И последнее – код кнопки **Выход**

```
procedure TForm1.Button2Click(Sender: TObject);
begin
edit1.Text:='';
  dbgrid1.Visible:=false;
form1.Close
```

Очищаем Edit для ввода фамилии

Делаем грид невидимым

Закрываем форму

ШАГ 5

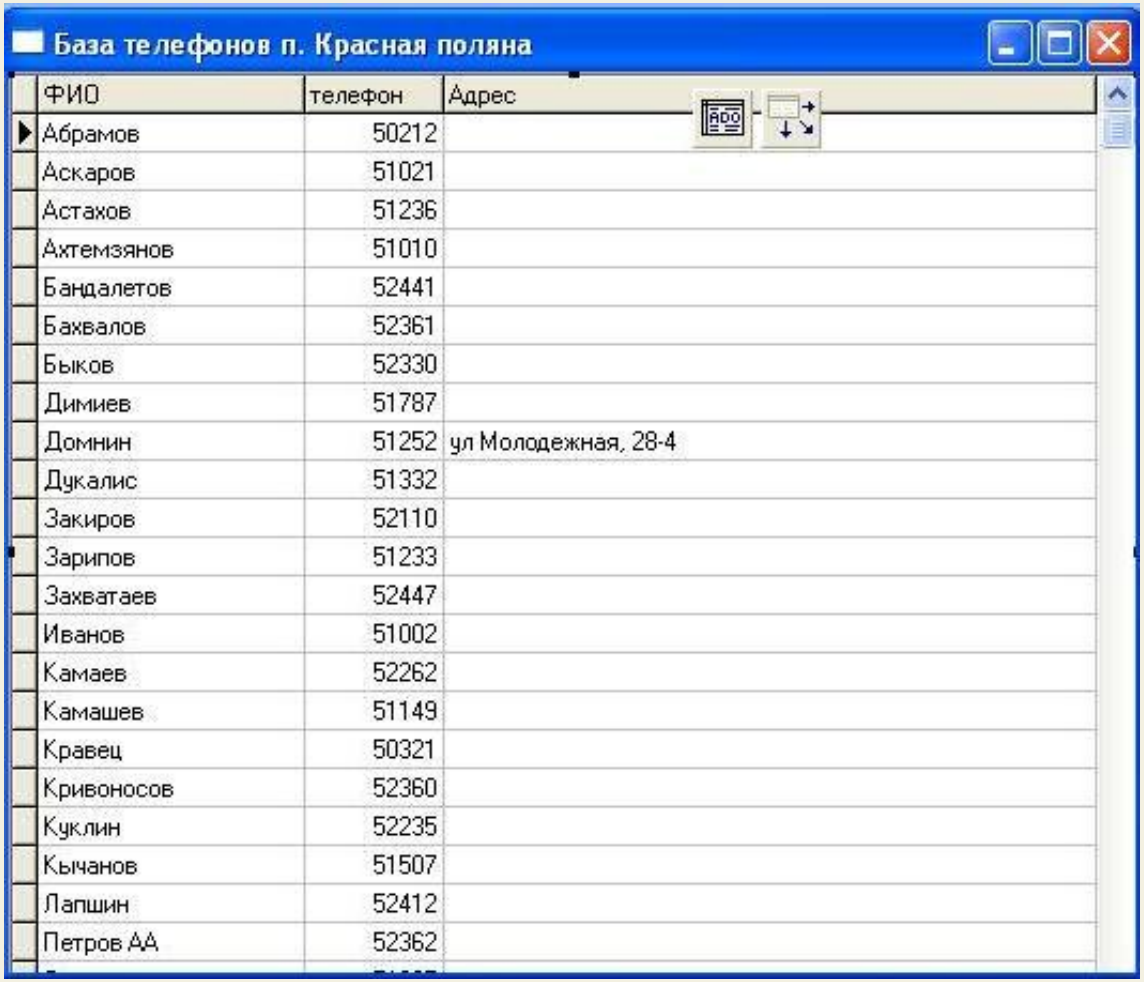
Разрабатываем форму **Поиск по номеру.**

Для этой формы размещаем точно такие же компоненты, связываем их с таблицей – т.е выполняем те же шаги, что и с формой Поиск по фамилии (смотри шаг 4)

Аналогичны и коды кнопок Найти и Выход. Отличие в коде кнопки Найти в том, что в качестве поля поиска указываем поле «Телефон»

ШАГ 6

Последняя форма отображает всю базу, поэтому **DBGrid** на ней можно развернуть на всю форму, а саму форму сделать побольше. Естественно те же компоненты для отображения данных (ADOTable, DataSourse и DBGrid, но кнопок никаких не ставим



ШАГ 7

Все сохраняем, компилируем и пробуем работу приложения