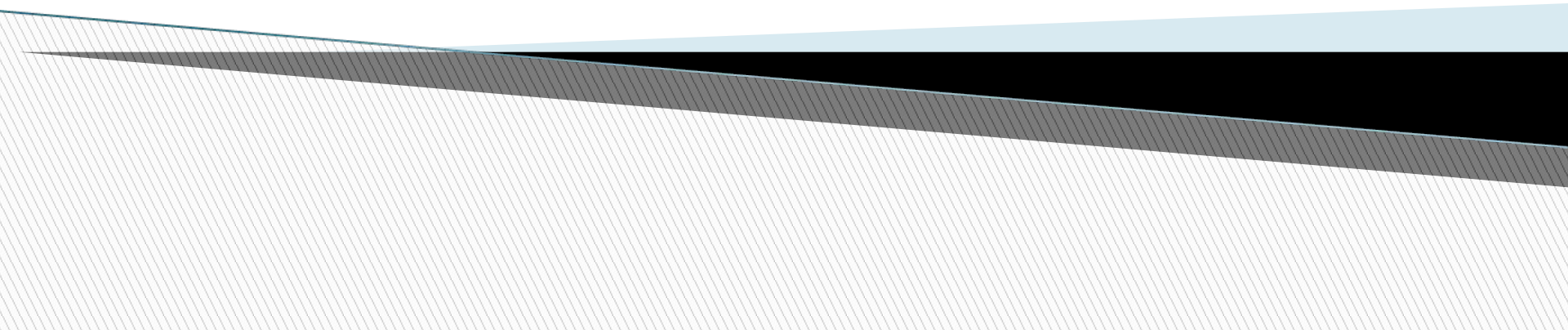


Система визуального объектно- ориентированного программирования Delphi



Технология BDE для работы с базами данных в Delphi



Создание таблицы Paradox

Paradox и DBF – это таблицы, а не базы данных. Если в одной базе Access могло храниться несколько таблиц, то у Paradox и DBF в одном файле храниться одна таблица.

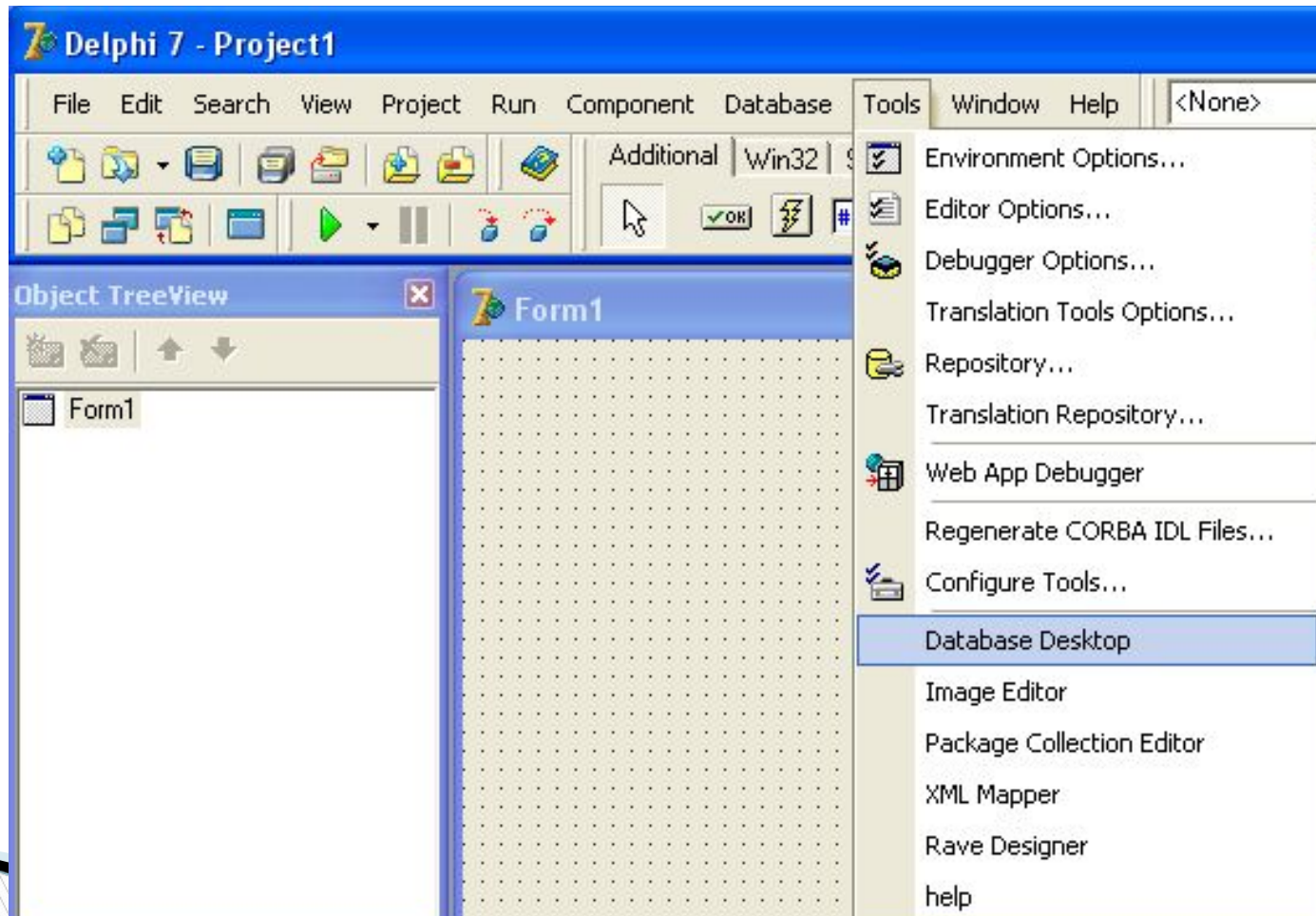
Создание и работа с таблицами Paradox и DBF одинаковы, поэтому рассмотрим на примере Paradox.

Для создания базы необходимо запустить отдельную программу Database Desktop.

Tools- Database Desktop.

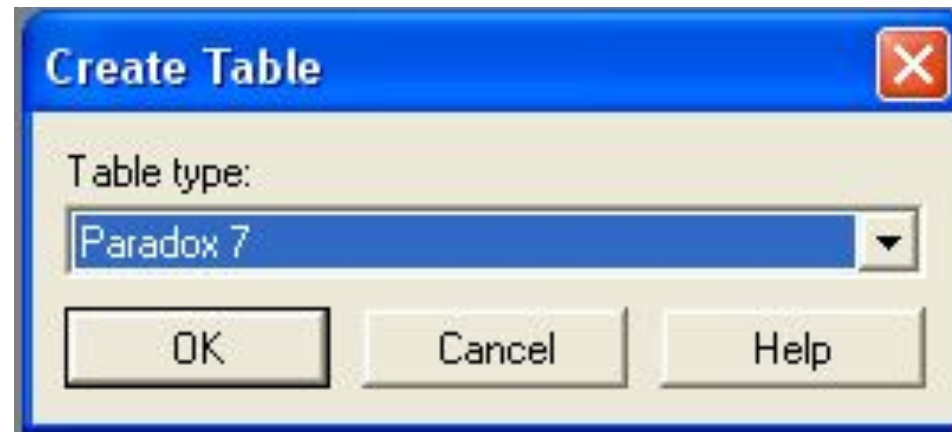
Построение с помощью DataBase Desktop

Программа предназначена для создания новых таблиц баз данных и редактирования уже существующих, а также для работы с визуальными и SQL-запросами и псевдонимами баз данных.

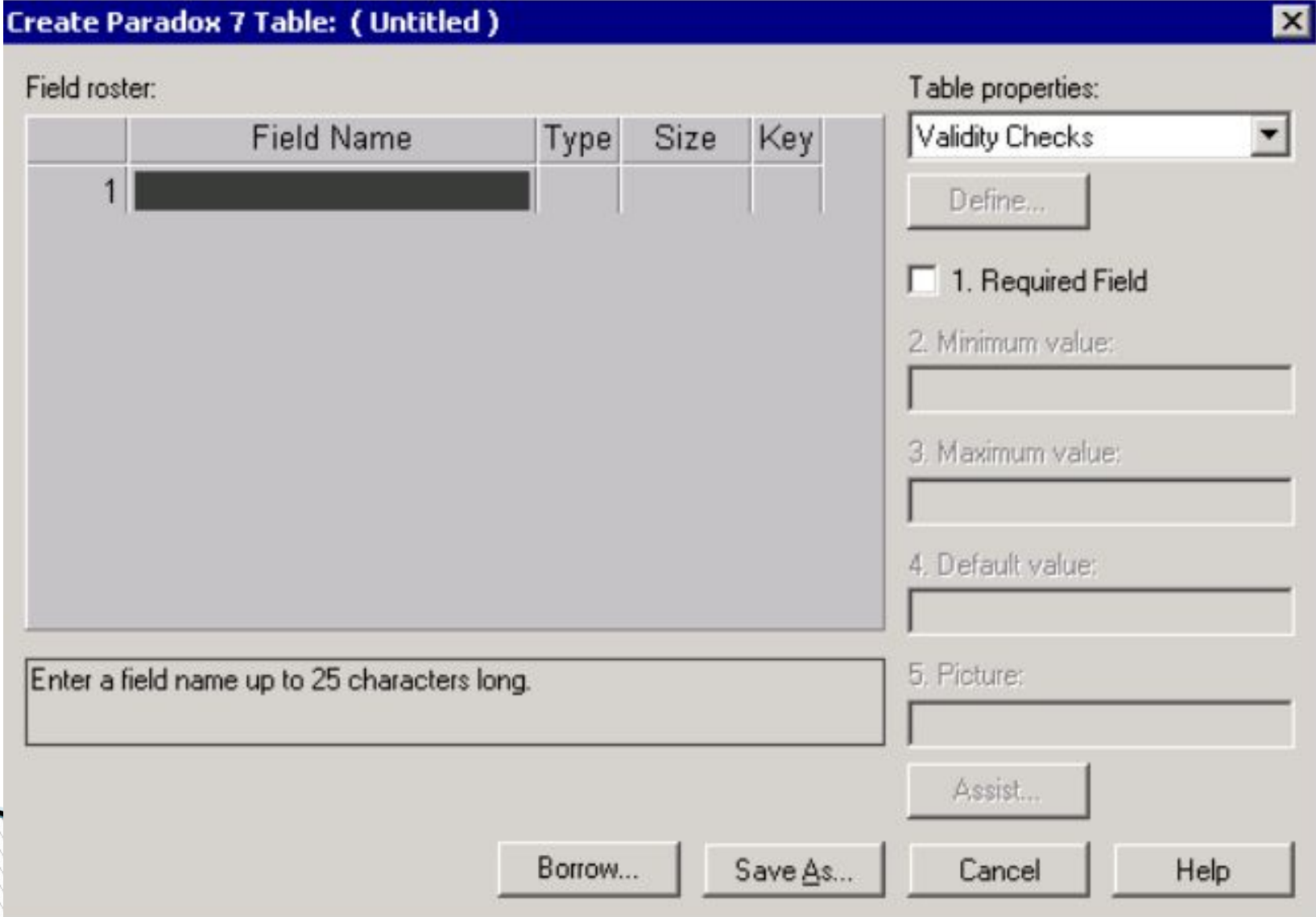


Создание новой базы данных

File->New->Table... и выбираем тип таблицы



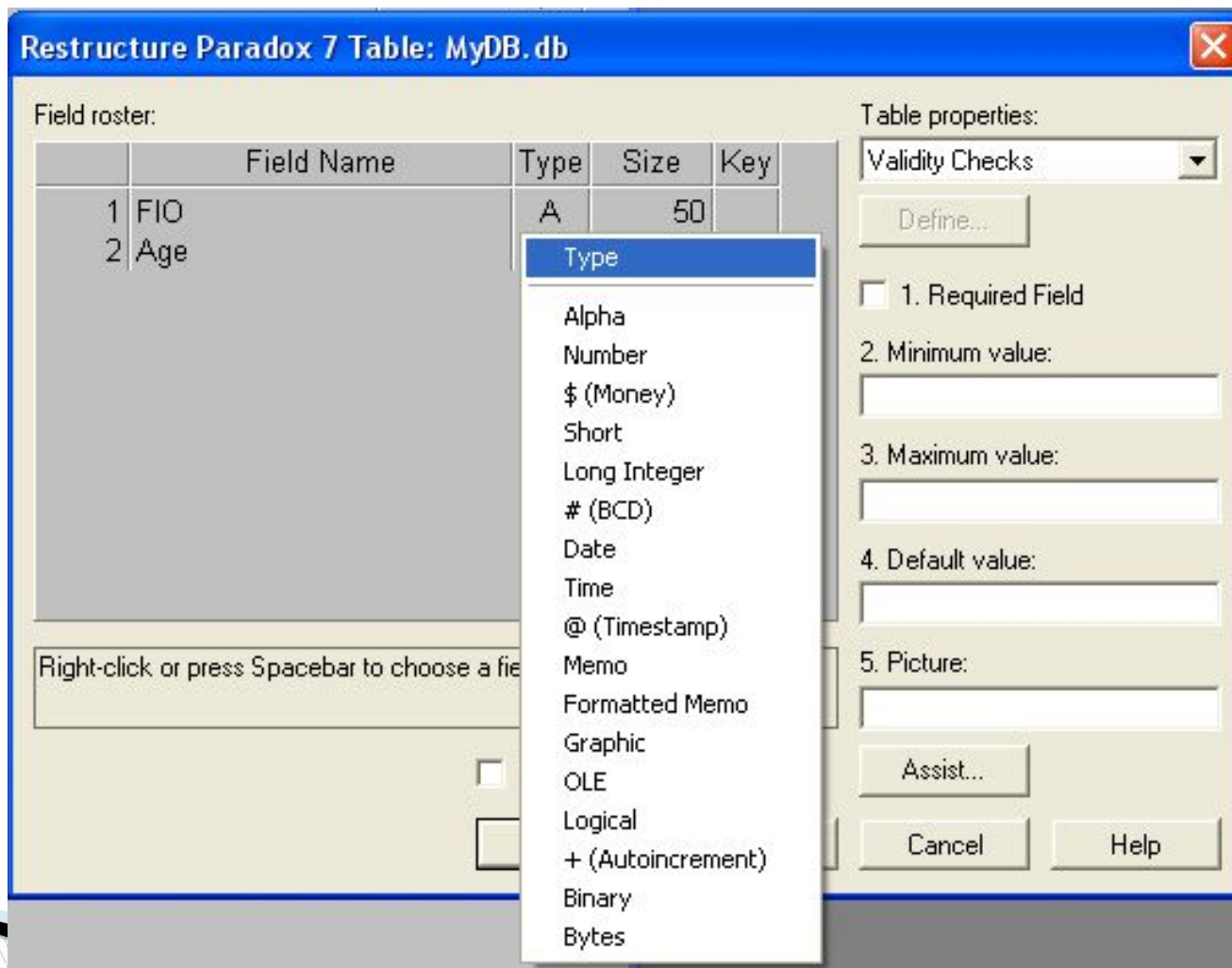
База данных готова. Теперь необходимо заполнить ее поля. Рассмотрим появившийся диалог.



1. Номер. Генерируется автоматически и изменять его нельзя.
2. Field Name (Имя поля). Называть поля можно только английскими буквами и нельзя использовать пробелы.
3. Type (Тип поля). Если щёлкнуть в этой колонке правой кнопкой мыши, то появиться меню со всеми допустимыми типами, необходимо только выбрать нужный.
4. Size (Размер поля). Не у всех типов полей можно менять размер.
5. Ключ. Если дважды щёлкнуть по этой колонке, то текущее поле станет ключевым. Ключевыми могут быть только первые поля, второе поле сможет быть ключевым только вместе с первым. Без ключевого поля невозможно добавлять новые записи в таблицу.

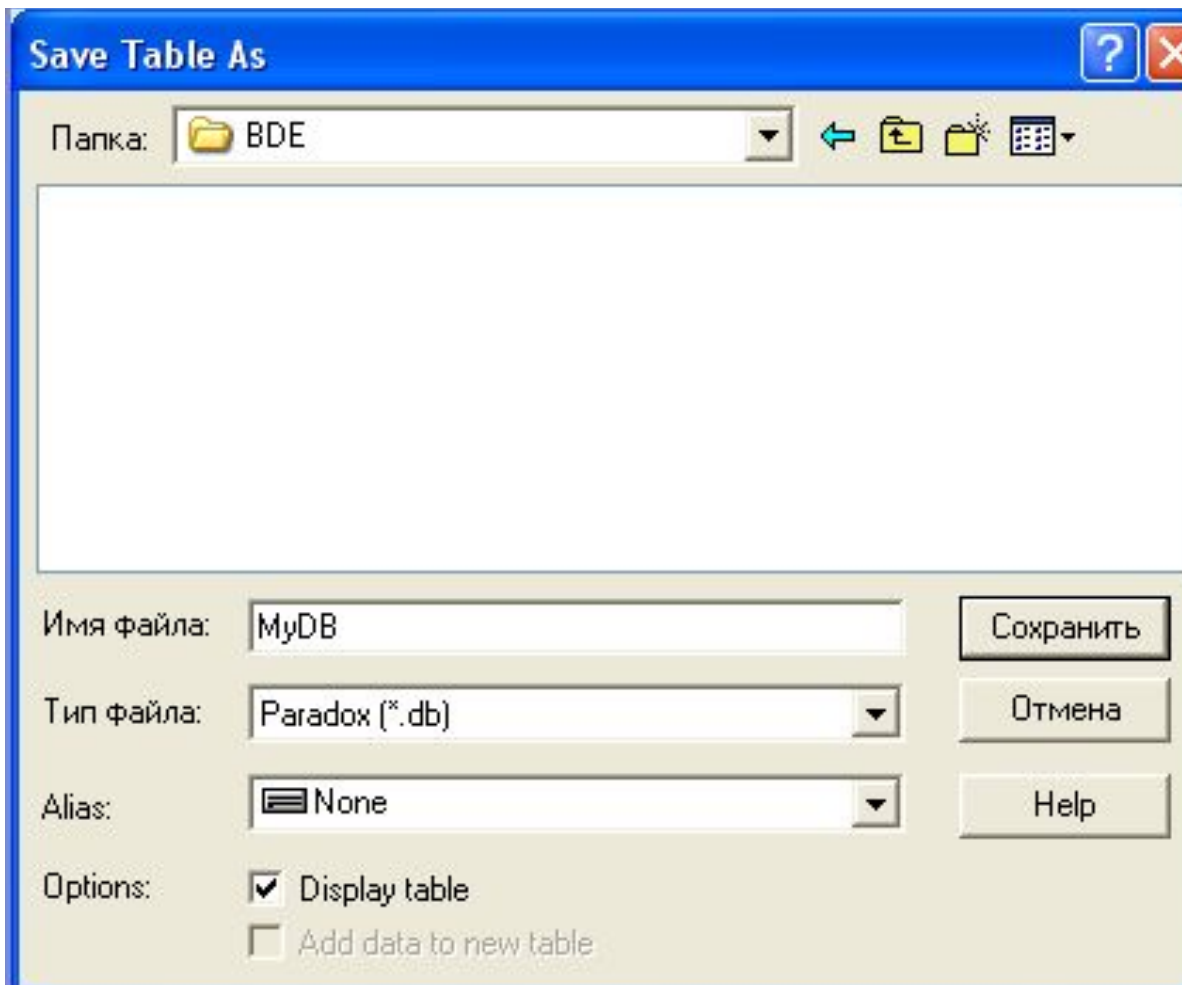
Окно создания новой таблицы

Задаем поля, их типы и размеры. Alpha - строковый, Number - числовой.

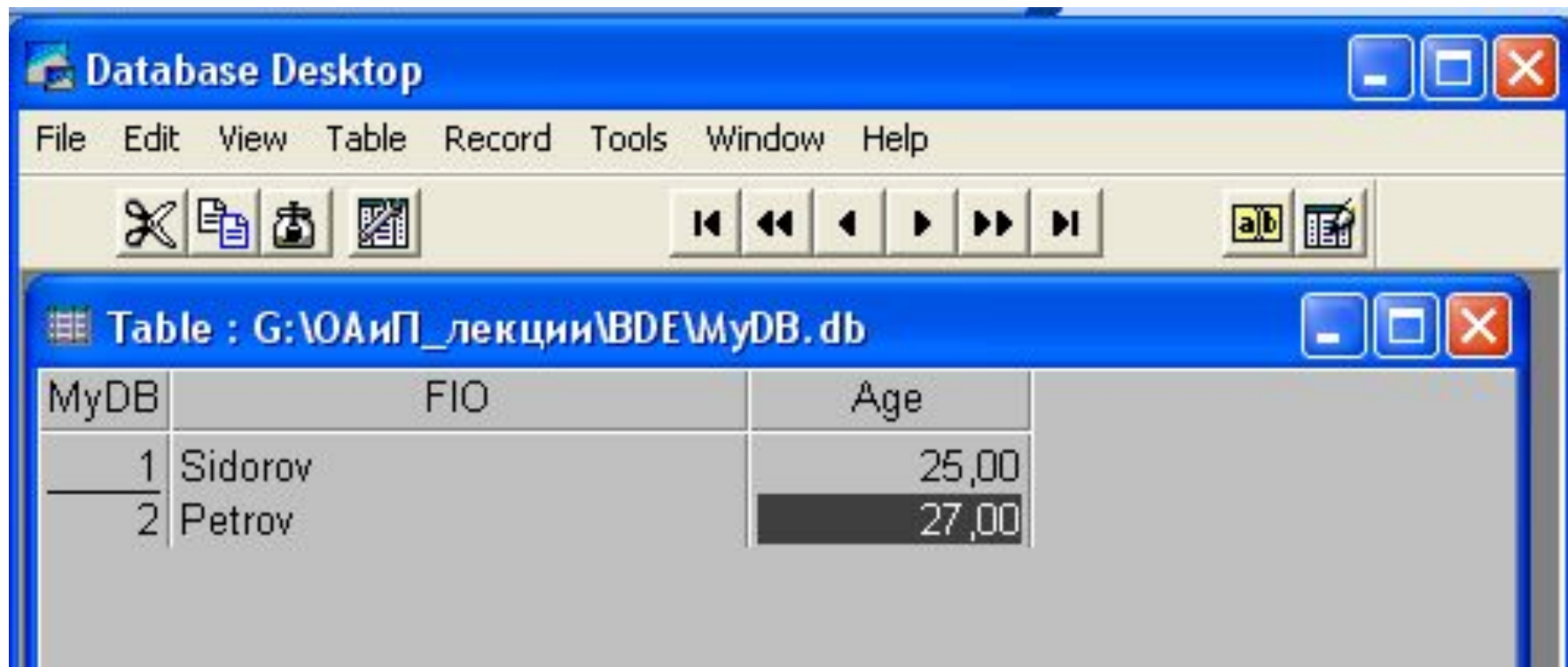


Сохраняем таблицу

Сохраняем таблицу в папку с программой под осмысленным именем

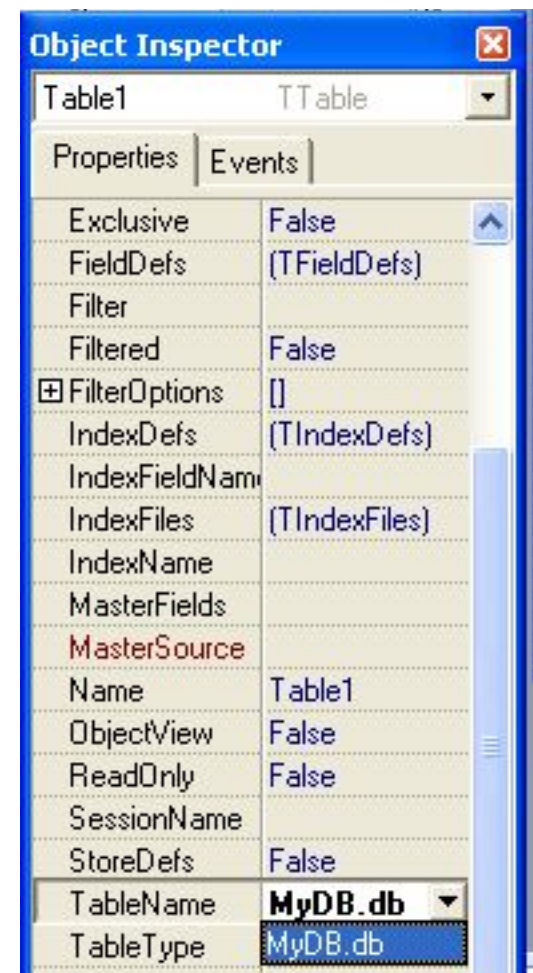
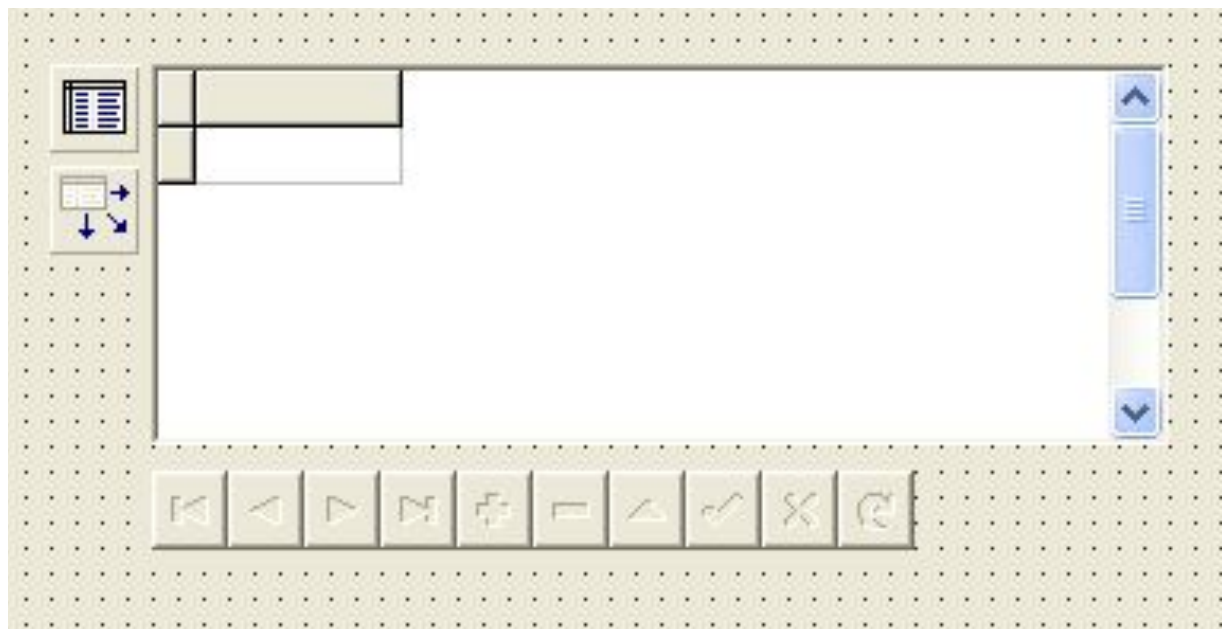


Заполняем таблицу значениями



Подключение таблицы к проекту

На форме располагаем все необходимые компоненты

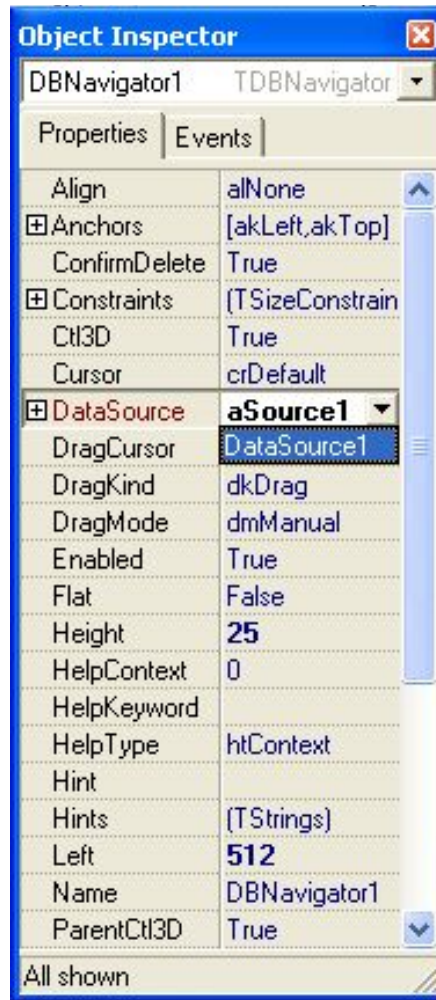


Подключаем БД и все компоненты



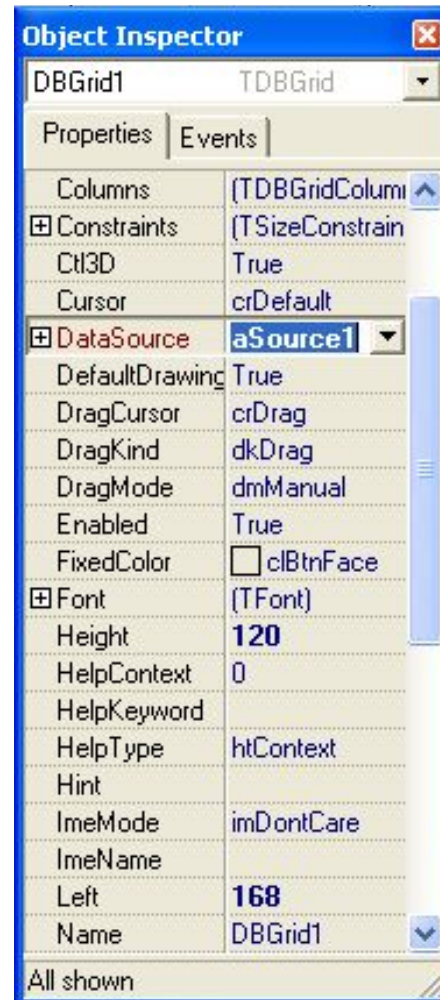
Object Inspector window for DataSource1 (TDataSource). The DataSet property is set to Table1. The Name property is DataSource1 and the Tag property is 0.

Property	Value
AutoEdit	True
DataSet	Table1
Enabled	Table1
Name	DataSource1
Tag	0



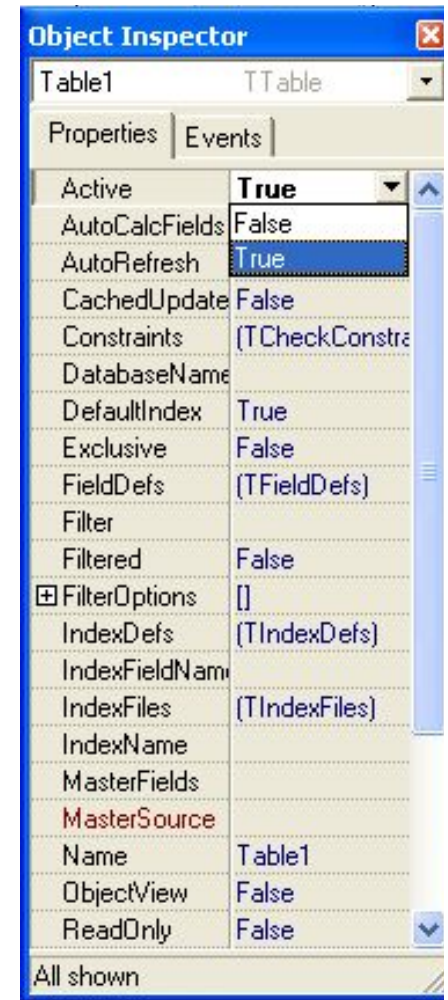
Object Inspector window for DBNavigator1 (TDBNavigator). The DataSource property is set to aSource1. The Name property is DBNavigator1 and the ParentCtl3D property is True.

Property	Value
Align	alNone
Anchors	[akLeft,akTop]
ConfirmDelete	True
Constraints	(TSizeConstrain
Ctl3D	True
Cursor	crDefault
DataSource	aSource1
DragCursor	DataSource1
DragKind	dkDrag
DragMode	dmManual
Enabled	True
Flat	False
Height	25
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
Hints	(TStrings)
Left	512
Name	DBNavigator1
ParentCtl3D	True



Object Inspector window for DBGrid1 (TDBGrid). The DataSource property is set to aSource1. The Name property is DBGrid1 and the Left property is 168.

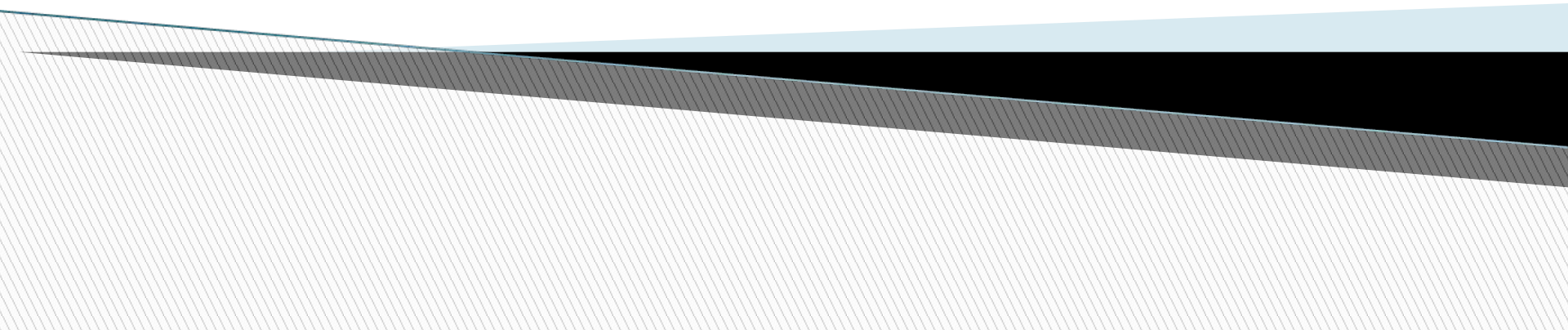
Property	Value
Columns	(TDBGridColumn
Constraints	(TSizeConstrain
Ctl3D	True
Cursor	crDefault
DataSource	aSource1
DefaultDrawing	True
DragCursor	crDrag
DragKind	dkDrag
DragMode	dmManual
Enabled	True
FixedColor	<input type="checkbox"/> clBtnFace
Font	(TFont)
Height	120
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
ImeMode	imDontCare
ImeName	
Left	168
Name	DBGrid1



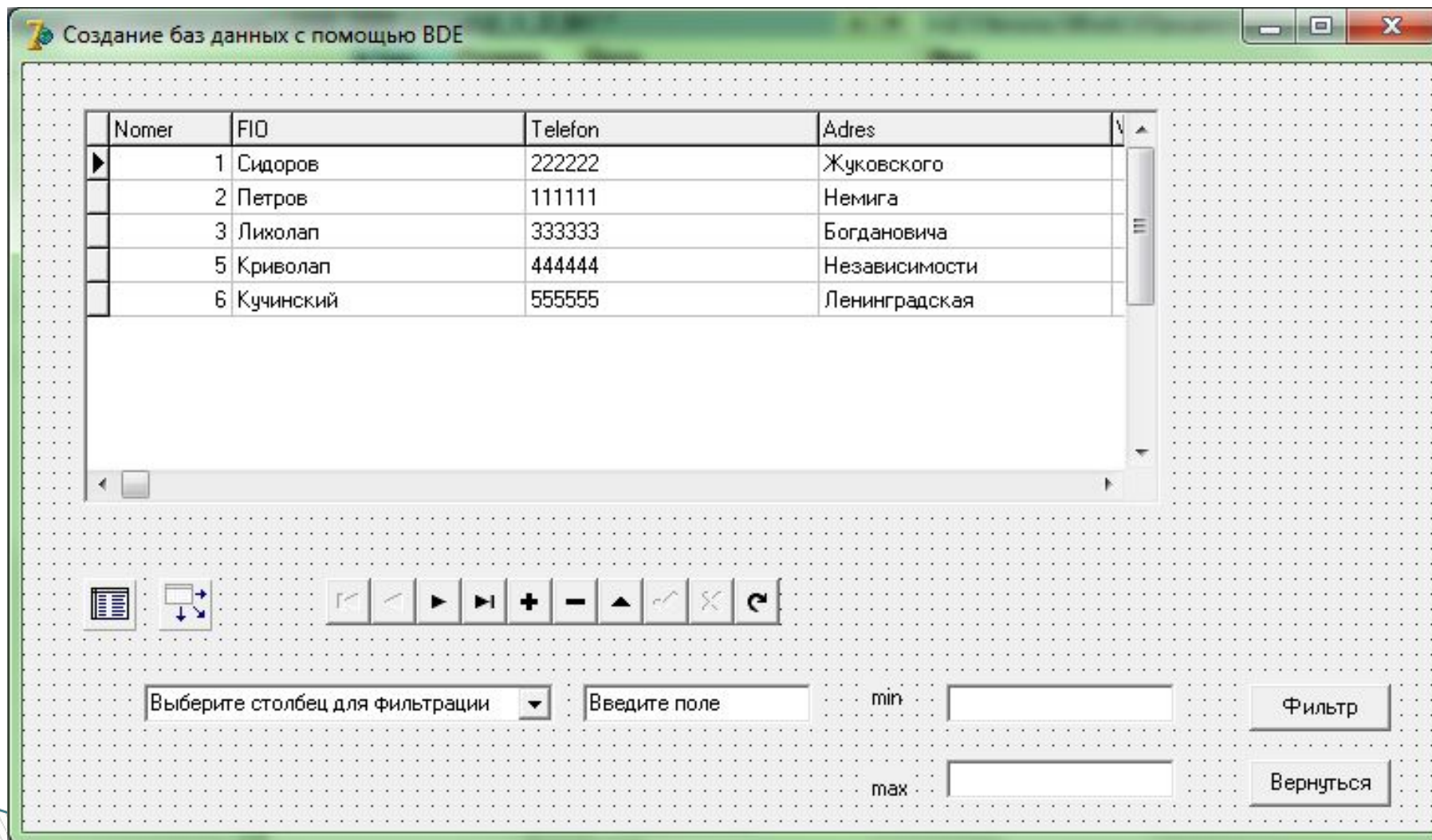
Object Inspector window for Table1 (TTable). The Active property is True and the AutoRefresh property is True. The Name property is Table1 and the ReadOnly property is False.

Property	Value
Active	True
AutoCalcFields	False
AutoRefresh	True
CachedUpdate	False
Constraints	(TCheckConstra
DatabaseName	
DefaultIndex	True
Exclusive	False
FieldDefs	(TFieldDefs)
Filter	
Filtered	False
FilterOptions	[]
IndexDefs	(TIndexDefs)
IndexFieldName	
IndexFiles	(TIndexFiles)
IndexName	
MasterFields	
MasterSource	
Name	Table1
ObjectView	False
ReadOnly	False

Программа База данных (BDE)



Основное окно



Фильтры (не используя Query)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ComboBox1.ItemIndex=0
  then Form1.Table1.Filter:='FIO='''+Edit1.Text+'' ' ;
  if ComboBox1.ItemIndex=1
  then Form1.Table1.Filter:='Adres='''+Edit1.Text+'' ' ;
  if ComboBox1.ItemIndex=2
  then Form1.Table1.Filter:='Telefon='''+Edit1.Text+'' ' ;
  if ComboBox1.ItemIndex=3
  then Form1.Table1.Filter:=' (Vozrast>='+Edit2.Text+') and (Vozrast<='+Edit3.Text+') ' ;
  Form1.Table1.Filtered:=true; //включить фильтр
  WidthCol; //установить ширину столбцов
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Table1.Filtered:=false; //отобразить всю таблицу
  WidthCol; //установить ширину столбцов
end;
```

//ширина столбцов таблицы

```
procedure WidthCol;
```

```
begin
```

```
Form1.DBGrid1.Columns[0].Width:=50;
```

```
Form1.DBGrid1.Columns[1].Width:=100;
```

```
Form1.DBGrid1.Columns[2].Width:=100;
```

```
Form1.DBGrid1.Columns[3].Width:=100;
```

```
Form1.DBGrid1.Columns[4].Width:=50;
```

```
Form1.DBGrid1.Width:=445;
```

```
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
```

```
Table1.Active:=true;
```

```
WidthCol;
```

```
end;
```

```
procedure TForm1.FormDestroy(Sender: TObject);
```

```
begin
```

```
Table1.Active:=false;
```

```
end;
```


Вычисляемые поля

Товары : таблица

	Имя поля	Тип данных
🔑	Key1	Счетчик
	Наименование	Текстовый
	Количество	Числовой
	Цена	Числовой

Товары : таблица

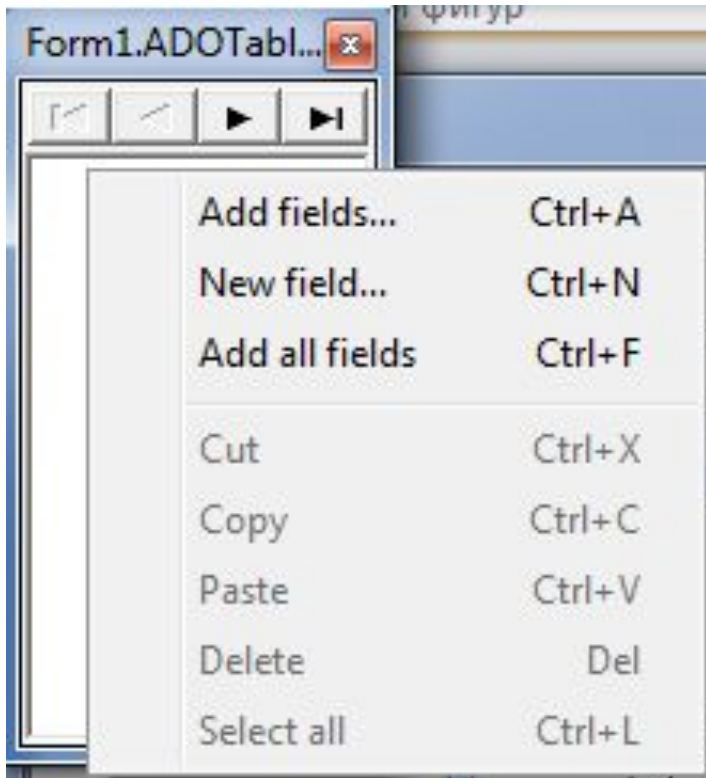
	Key1	Наименование	Количество	Цена
▶	1	Картошка	6	2
	2	Морковка	1	4
	3	Свекла	2	3
	4	Лук	1	3
	5	Зелень	3	1
*	(Счетчик)		0	0

Вычисляемые поля

The screenshot shows a Windows application window titled "Form1" with a standard Windows XP-style title bar. The main area of the window is a grid with a dotted background. At the top of the grid, there are three icons: a blue square with "ADO" and a computer icon, a blue square with "ADO" and a document icon, and a blue square with a document icon and arrows. Below these icons is a data grid with three columns: "Наименование", "Количество", and "Цена". The grid contains five rows of data. Below the grid is a toolbar with ten icons: a square with a diagonal line, a square with a diagonal line, a right-pointing triangle, a right-pointing triangle, a plus sign, a minus sign, an up-pointing triangle, a square with a diagonal line, a square with a diagonal line, and a circular arrow.

Наименование	Количество	Цена
Картошка	6	2
Морковка	1	4
Свекла	2	3
Лук	1	3
Зелень	3	1

После двойного щелчка по компоненту ADOTable1 появится редактор полей. По щелчку правой кнопкой мыши в контекстном меню необходимо выбрать Add all field.



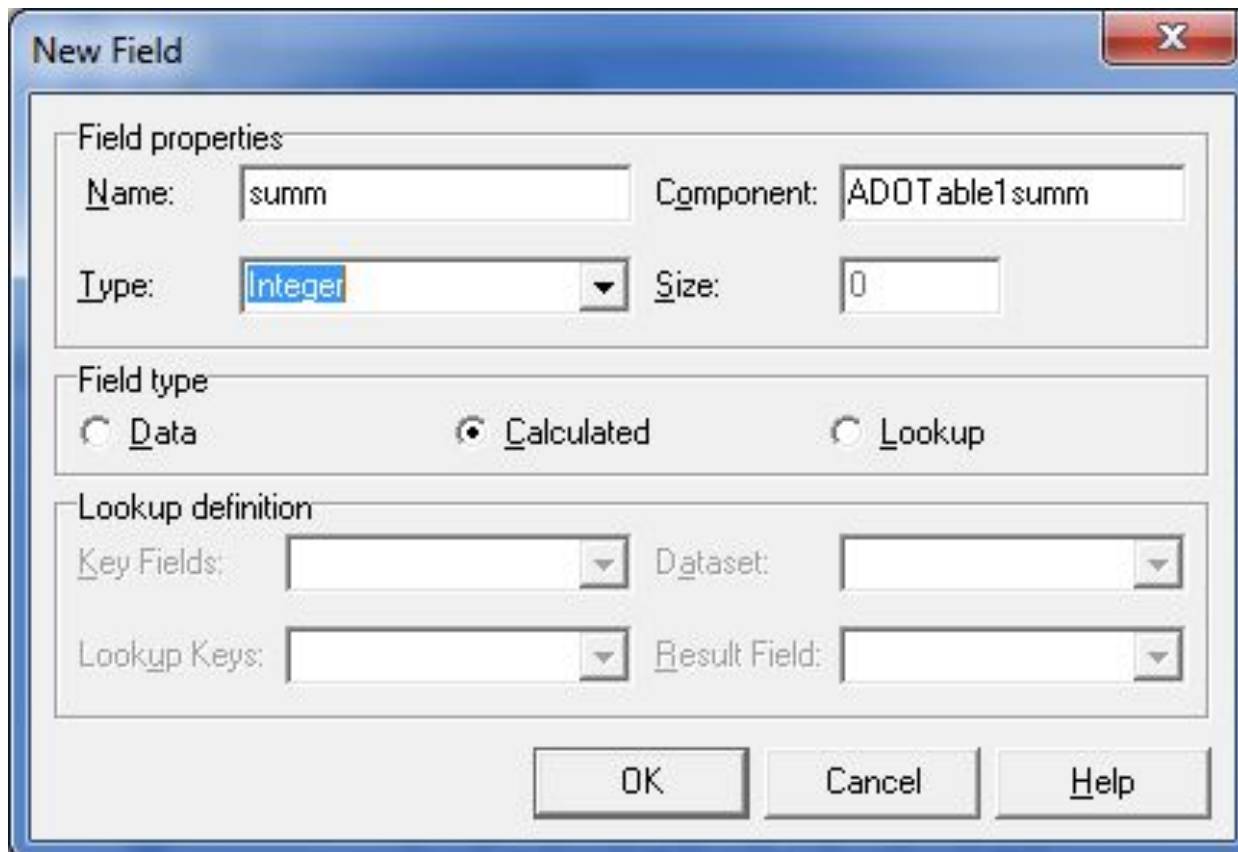
Отредактируем поля: сделаем поле Key1 невидимым и изменим ширину полей.

Кроме этого, изменим свойство name полей Количество на kol, а Цена на price.

Затем добавим новое поле в которое будет содержать стоимость каждого товара.

New field...

Но прежде чем это сделать необходимо сделать таблицу неактивной.



Появилось окно свойств нового поля. Необходимо заполнить его и изменить его свойство name на Summ. Тип поля – Calculated.

Как только поле создано таблицу вновь нужно сделать активной.

Теперь создадим обработчик события OnCalcFields. Это событие вызывается каждый раз, когда надо пересчитать вычисляемые поля. Оно будет вызываться для всех видимых пользователю записей.

```
procedure TForm1.ADOTable1CalcFields(DataSet: TDataSet);  
begin  
    summ.value:=kol.asinteger*price.asinteger;  
end;
```

Наименование	Количество	Цена	сумма
Картошка	6	2	12
Морковка	1	4	4
Свекла	2	3	6
Лук	1	3	3
Зелень	3	1	3

Добавим еще поле Edit в котором будет подсчитываться общая сумма по всем записям таблицы . И кнопку, по щелчку по которой и будет выполняться подсчет.

The screenshot shows a Visual Basic form titled "Form1" with a grid background. At the top left, there are three icons: ADO, ADO, and a directional arrow icon. Below these is a table with the following data:

Наименование	Количество	Цена	сумма
Картошка	6	2	
Морковка	1	4	
Свекла	2	3	
Лук	1	3	
Зелень	3	1	

To the right of the table is a button labeled "Итого". Below the table and button is a row of navigation and control buttons: a left arrow, a right arrow, a plus sign, a minus sign, an up arrow, a checkmark, a cross, and a refresh icon.

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var s:integer;
begin
  s:=0;
  adotable1.First;
  while not adotable1.Eof do
  begin
    s:=s+summ.asinteger;
    adotable1.Next;
  end;
  edit1.Text:=inttostr(s);
end;
```


Навигация (Перемещение по записям)

Вызов `Table1.First` перемещает к первой записи в таблице.

`Table1.Last` перемещает к последней записи.

`Table1.Next` перемещает на одну запись вперед.

`Table1.Prior` перемещает на одну запись Назад.

Можно проверять свойства `BOF` или `EOF`, чтобы понять, находится ли указатель в начале или в конце таблицы.

Процедура `MoveBy` перемещает на `N` записей вперед или назад в таблице.

Поля

Когда необходимо получить доступ из программы к индивидуальным полям записи, можно использовать одно из следующих свойств или методов:

- ▣ `property Fields[Index: Integer];`
- ▣ `function FieldByName(const FieldName: string): TField;`
- ▣ `property FieldCount;`

Свойство *FieldCount* возвращает число полей в текущей структуре записи.

Индекс передаваемый в *Fields* (начинающийся с нуля), и определяет номер поля которое становится доступным, т. е. первое поле - ноль, второе один, и так далее.

Если необходимо прочитать текущее содержание конкретного поля конкретной записи, то можно использовать свойство *Fields* или метод *FieldsByName*. Для того, чтобы найти значение первого поля записи:

```
S := Fields[0].AsString;
```

Или использовать функцию *FieldsByName* вместо свойства *Fields*:

```
S := FieldsByName('Наименование').AsString;
```

Список доступных методов:

- ▣ AsBoolean
- ▣ AsFloat
- ▣ AsInteger
- ▣ AsString
- ▣ AsDateTime .

Модуль данных

Модуль данных представляет собой хранилище объектов, которое позволяет централизованно управлять их работой.

Создать модуль данных можно командой File → New → Data Module.

В модуле данных необходимо разместить невизуальные компоненты, которые обеспечат доступ к самой БД:
AdoConnection, Adotable, DataSource.

- 1. Бросить на форму все необходимые компоненты: ADOConnection, ADOTable или ADOQuery, DataSource, DBGrid, DBNavigator, DBEdit, DBImage.
- 2. Для ADOConnection в Object Inspector дважды щелкнуть напротив ADOConnectionString и установить соединение с базой данных., указав поставщика данных Microsoft Jet 4.0 Ole DB Provider, LoginPrompt – False.
- Для ADOTable выставить Active – False, выбрать Connection и TableName.
- Для DataSource выбрать Dataset.
- Для DBGrid, DBNavigator выбрать DataSource
- Для DBEdit, DBImage выбрать DataSource, DataField.
- Для ADOTable выставить Active – True.

Простая выборка данных

Для простой выборки данных используется сокращенный синтаксис оператора SELECT:

SELECT [ALL | DISTINCT] *СписокВыборки*
FROM *ИмяТаблицы*
WHERE *УсловиеОтбора*
ORDER BY *ИмяПоля* [,...n] [ASC | DESC]

СписокВыборки определяет поля, включаемые в итоговый набор данных, *ИмяТаблицы* указывает таблицу БД, из которой возвращаются записи, а *УсловиеОтбора* позволяет ограничить число возвращаемых записей с помощью логических операторов.

По умолчанию команда `SELECT` возвращает все записи, включая дубликаты, что определяется ключевым словом `ALL`, которое может быть опущено. Для получения набора уникальных неповторяющихся записей необходимо указывать ключевое слово `DISTINCT`.

Для выборки всех полей из таблицы в списке выборки необходимо указать звездочку (*), в противном случае через запятую перечисляются имена полей.

Сортировка указывается в разделе `ORDER BY ИмяПоля [,...n] [ASC | DESC]`.

По умолчанию сортировка осуществляется по возрастанию, что соответствует зарезервированному слову `ASC`, которое может опускаться, для сортировки в убывающем порядке указывается – `DESC`.

Условие отбора

Условие отбора определяет критерий отбора записей, включаемых в итоговый набор. В результат будут включены только те строки, которые соответствуют наложенным условиям.

Условие может включать выражения, образованные с помощью операторов сравнения или логических операторов. Условия могут также объединяться и с помощью логических операндов AND, OR и NOT. Кроме в SQL можно использовать поиск по шаблону с использованием оператора LIKE:

В шаблоне могут использоваться следующие универсальные символы:

- ▣ **%** – подразумевает любую строку, состоящую из 0 и более СИМВОЛОВ;
- ▣ **_** – ровно один СИМВОЛ;
- ▣ **[]** – любой символ из заданного множества (например, [adfh]) или диапазона (например, [0-9]),
- ▣ **[^]** – любой символ, не попадающий в заданный диапазон или множество.

```
select * from uchenik where fio like 'K%'
```

```
select * from uchenik where fio like '_o%'
```

```
select * from uchenik where fio like '[K]_[p]%'
```

```
select * from uchenik where fio like '[^П]%'
```


Для определения соответствия выражения одному из перечисленных в заданном списке значений применяется **логический оператор IN**. Данный оператор всегда может быть записан и в виде группы условий, объединенных операндом OR, and.

Однако в список значений нельзя включать неопределенное значение NULL, для работы с такими значениями используется функция выборки IS NULL.

```
select * from uchenik where pol is null
```

Bcë!

