Лабораторная работа 2

СОЗДАНИЕ ПРОСТЕЙШЕГО КЛИЕНТСКОГО ПРИЛОЖЕНИЯ В RAD STUDIO

Цель работы: разработать клиентское приложение баз данных, позволяющее пользователям вводить/редактировать данные в таблицах БД.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями.
2. Создать приложение с контейнером DataModule
3. Организовать подключение к БД.
4. Обеспечить взаимодействие форм с наборами данных.
5. Создать главное меню приложения.
6. Ознакомится с процессом изменения данных через DBGrid.
7. Реализовать программное редактирование данных в БД.
8. Ответить на контрольные вопросы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Для того чтобы ваше будущее приложение, написанное на любом языке программирования, могло подключится к серверу **MySQL** и работать с данными необходимо установить **32**-битную версию драйвера ODBC (mysql-connector-odbc) соответствующую версии *MySQL* скачав его с официального сайта https://dev.mysql.com/downloads/connector/odbc/.

ODBC (*Open Database Connectivity*) – это программный интерфейс (*API*) доступа к базам данных, разработанный компанией *Microsoft*.

2. Создание Простейшего Клиентского Приложения

Для работы с нашей базой данных создадим новый проект. Для этого в меню **File** выберем пункт **New** и далее **Windows VCL Application – Delphi** (рис. 2.1). Появится окно с пустой формой, теперь необходимо сохранить проект в своей папке (например, **C:\User\Lab2**), снова выбрав меню **File** и нажав на **Save Project As**.

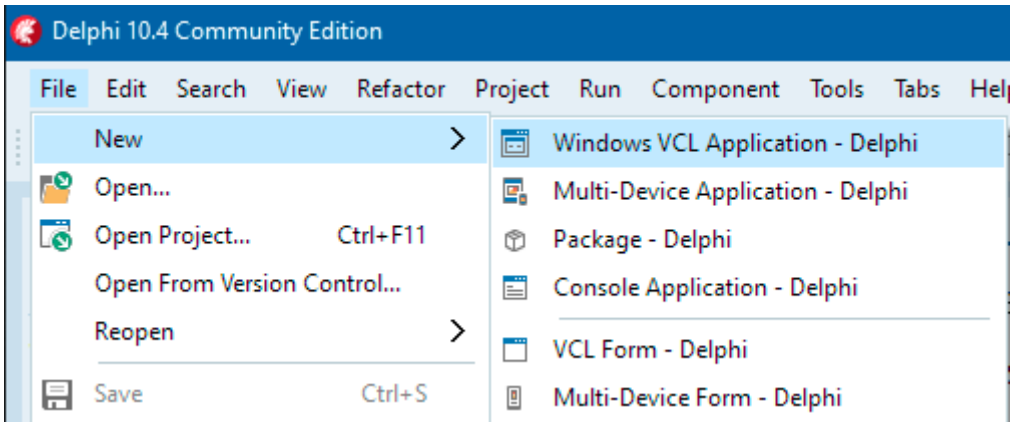


Рис. 2.1. Создание нового проекта на Delphi

Запустите проект на выполнение выбрав меню **Run** и нажав **Run** (**F9**), должно построиться приложение с пустой формой.

При разработке программ, работающих с базой данных, в которых имеется более трех таблиц, для удобства пользователя используют интерфейс на основе нескольких форм. При этом возможна различная организация обращения к полям таблиц и их значениям, необходимых для реализации бизнес-правил.

Среда разработки **RAD Studio** предлагает размещать компоненты доступа к данным на специальной форме (модуле) – контейнере **DataModule**. Следует отметить, что контейнер **DataModule** не имеет ничего общего с классической формой приложения, поскольку на нем можно размещать только невизуальные компоненты.

Данный модуль доступен на этапе разработки и в то же время не является видимым для пользователя. Основным преимуществом размещения компонентов доступа к данным в контейнере **DataModule** является то, что изменение значения любого свойства компонента проявится сразу же во всем приложении.

Контейнер **DataModule** хранит только невизуальные компоненты, а весь интерфейс будет организован в виде отдельных форм.

Добавим в проект модуль данных, зайдите в главное меню **File** щелкните **New** и далее **Other**. В появившемся окне **New Items** (рис. 2.2) в группе **Database** выберем значок **Data Module**.

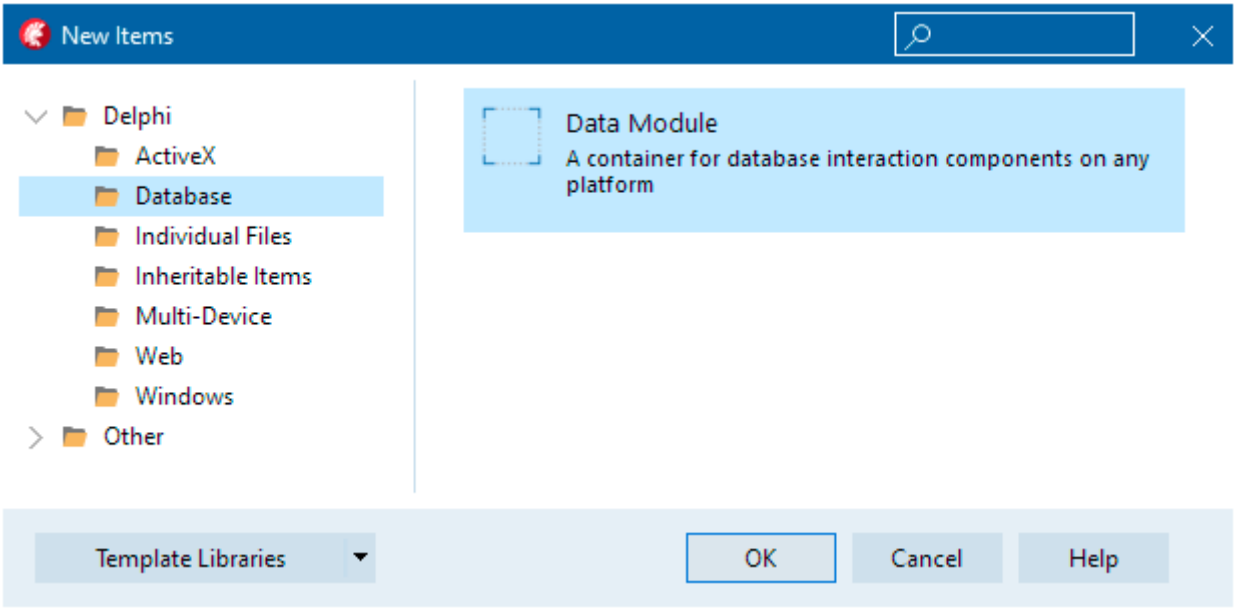


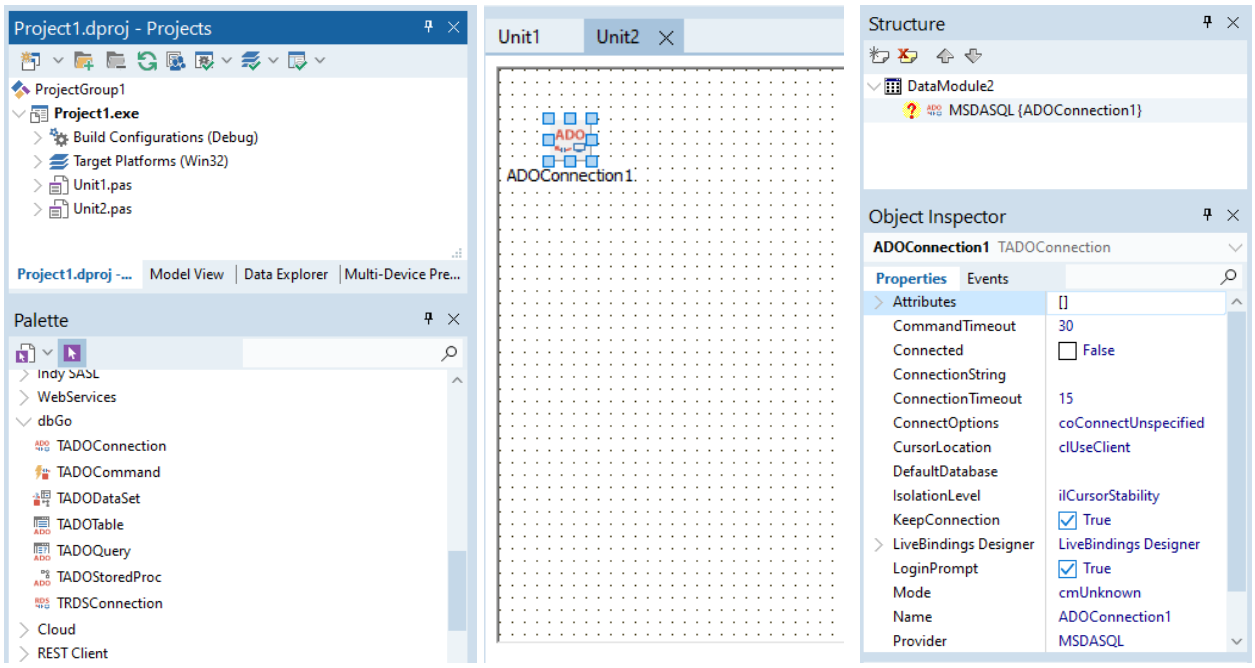
Рис. 2.2. Добавление к проекту компонента DataModule

Контейнер **DataModule**, добавленный в приложение, автоматически приобретает название с цифрой следующего модуля (в нашем случае это **DataModule2**), который при необходимости можно переименовать.

2.1 Подключение к базе данных созданной в MySQL

Для организации соединения с базой данных воспользуемся технологией **ADO**, взяв для этого невизуальный компонент **ADOConnection**, расположенный на вкладке **dbGO** палитры компонентов **Pallete** (рис. 2.3, *а*). Выберем его мышью и разместим на форме (рис. 2.3, *б*), при этом в

инспекторе объектов **Object Inspector** на вкладке **Properties** (*Свойства*) будут отображены все основные свойства выбранного компонента (рис. 2.3, *в*).



а б в

Рис. 2.3. Фрагменты рабочих областей среды разработки:
а – менеджер проекта (**Projects**) и палитра компонентов (**Palete**);
б – фрагмент формы с установленным компонентом **TADOConnection**;
в – инспектор объектов, отображающий свойства компонента **ADOConnection1**

Для настройки соединения с базой данных выберем свойство **ConnectionString** компонента **ADOConnection1** и нажмем кнопку с многоточием. При этом появится окно формирования строки подключения (рис. 2.4), в котором нажмем на кнопку **Build**.

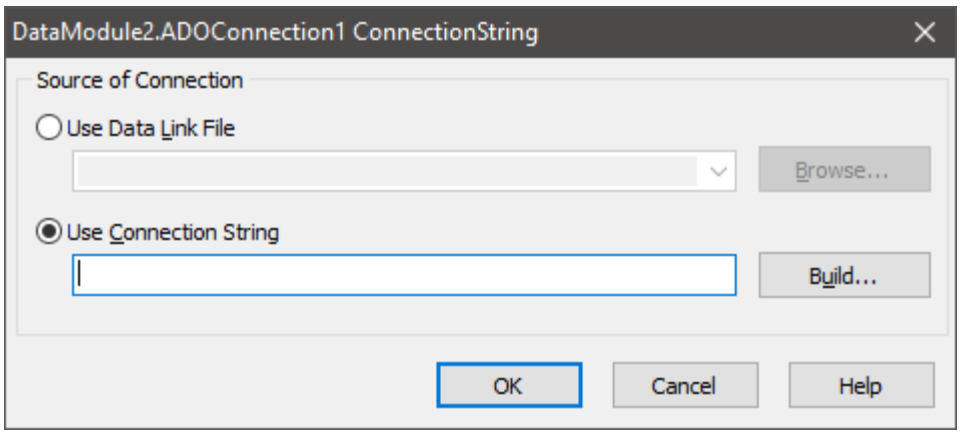


Рис. 2.4 – Создание строки подключения в БД

Появится окно настройки источника соединения. В нем мы выбираем **Microsoft OLED DB Provider for ODBC Driver**, поскольку он позво-

ляет подключаться к большинству современных СУБД (рис. 2.5). Сделав выбор, нажимаем на кнопку **Далее**. На открывшейся вкладке **Соединение** настраиваем параметр **Использовать строку соединения** и нажимаем **Сборка**.

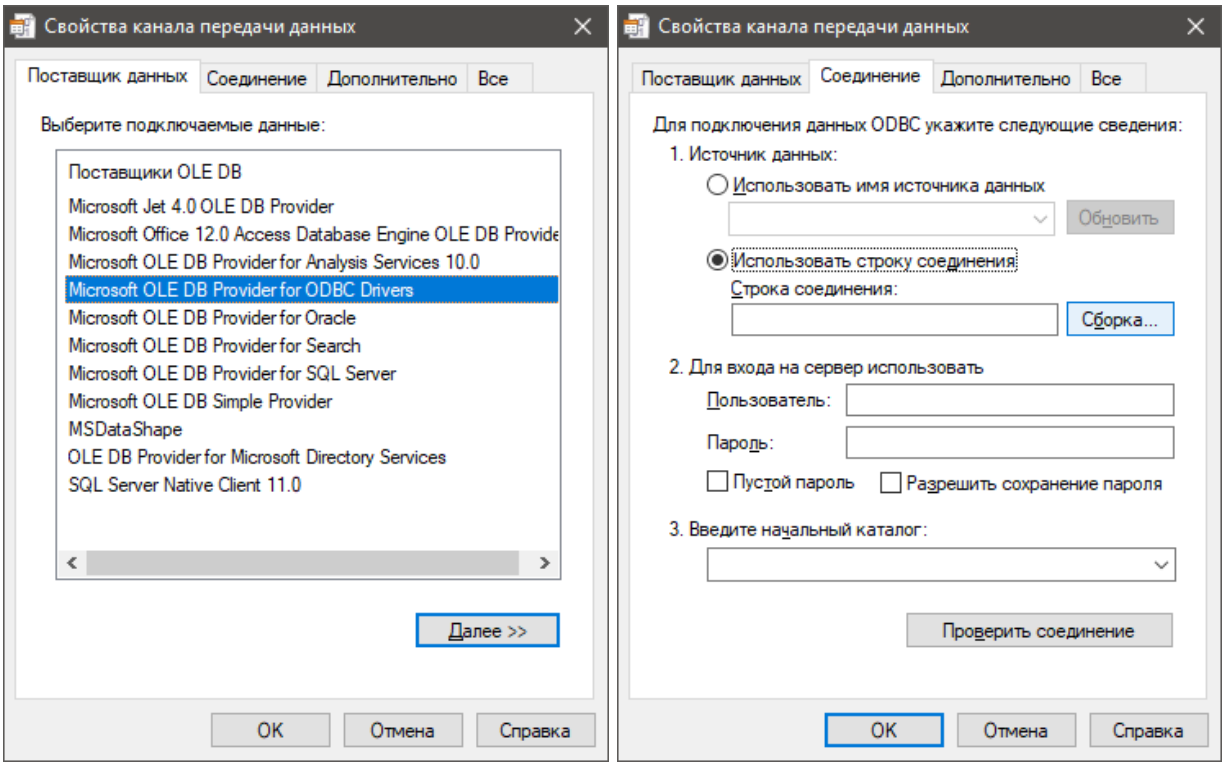


Рис. 2.5 – Создание строки подключения в БД (продолжение)

В появившемся окне можно указать путь к файлу настроек подключения к БД (в случае если файл был ранее создан) или создать новый. В нашем случае для начала необходимо создать новый файл с настройками подключения к БД. Для этого необходимо нажать на кнопку **Создать** (рис. 2.6).

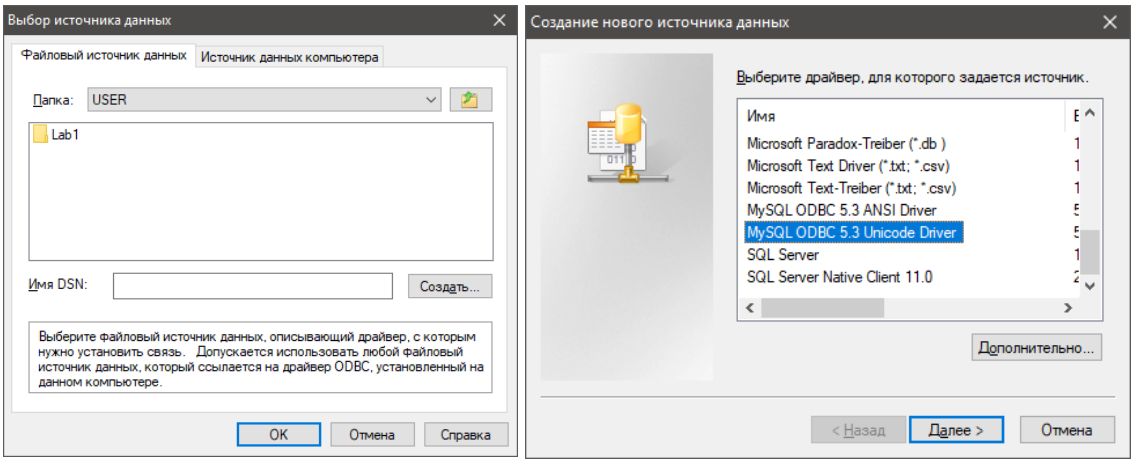


Рис. 2.6 – Выбор драйвера подключения к БД

После этого необходимо выбрать драйвер для соединения **MySQL ODBC Unicode Driver**.

После выбора драйвера нажимаем на кнопку **Далее** и в новом окне нажимаем кнопку **Обзор**, указываем путь для сохранения файла настроек. После чего нажимаем на кнопку **Готово** и появится окно подключения к БД *MySQL* (рис. 2.7):

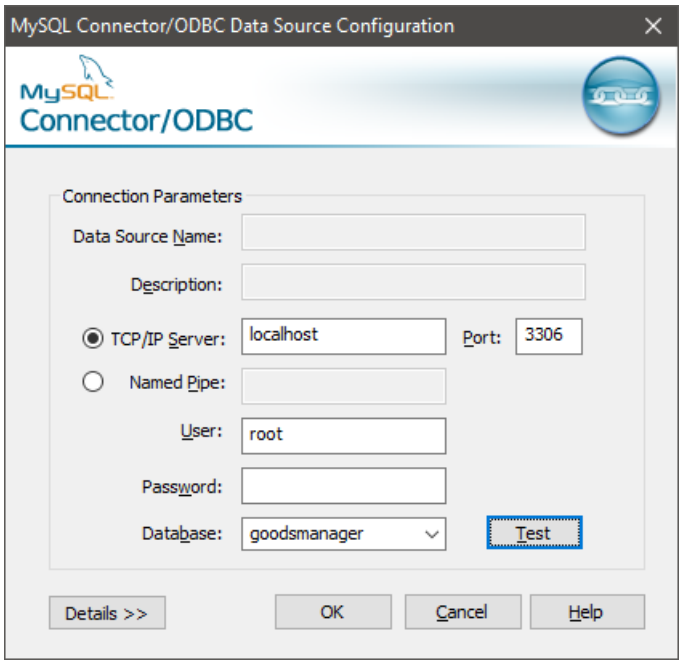


Рис. 2.7 – Данные подключения

Заполняем данные подключения и нажимаем **Test**. В случае успешного результата несколько раз подряд нажмем на кнопку **OK**.

Дале для компонента **ADOConnection1** необходимо активировать свойство **Connected** (рис. 2.8):

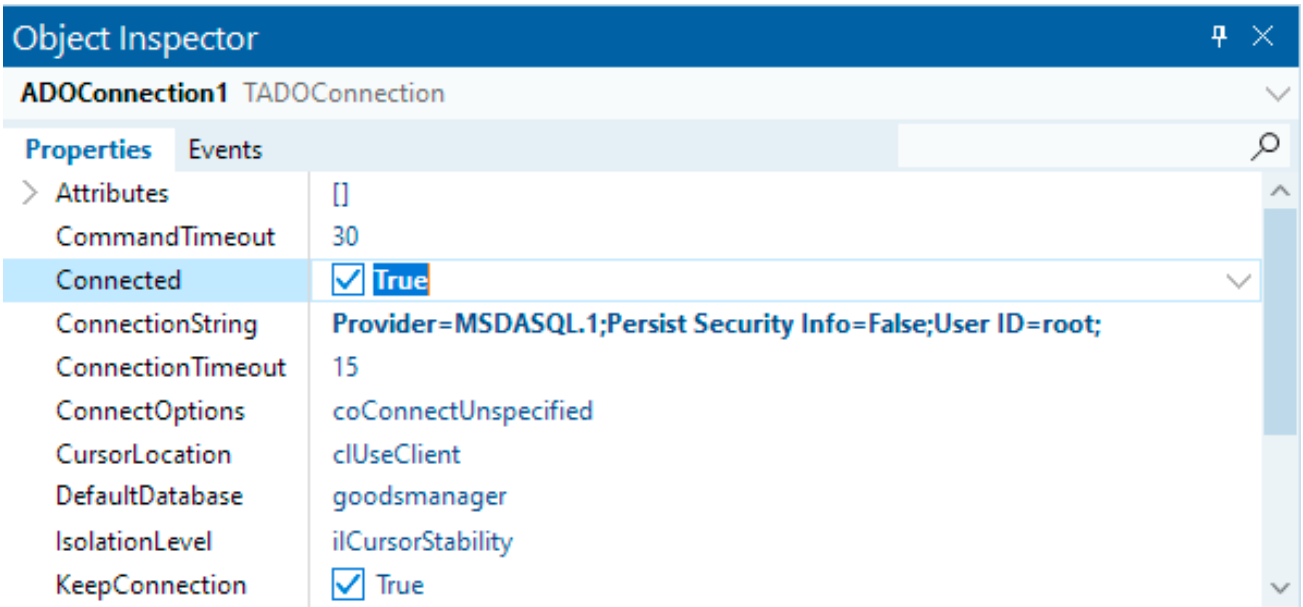


Рис. 2.8 – Активация подключения к БД

2.2 Компоненты для работы с данными

После настройки соединения разместим на **DataModule2** шесть экземпляров невизуальных компонент **ADOTable** (один компонент на одну таблицу в базе данных), два компонента **ADOQuery** и восемь компонент **DataSource** (рис. 2.9):

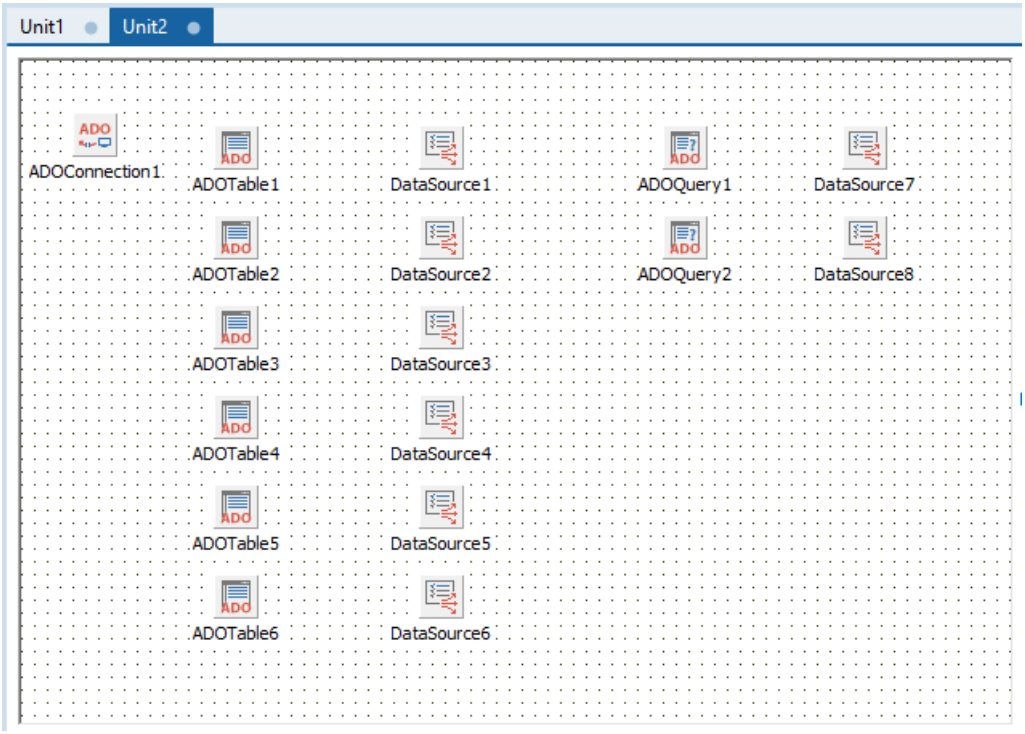


Рис. 2.9 – Пример размещения компонентов в **DataModule2**

Компоненты **ADOTable** и **ADOQuery** служат для хранения *наборов данных* (НД – данные таблиц, представлений, результатов запросов к БД).

Компонент **DataSource** (*источник данных*), можно взять на странице палитры компонентов **Data Access**. Он служит связующим звеном (передает данные) между невизуальными компонентами (в данном случае компонентом **ADOTable** или **ADOQuery**) и визуальными компонентами, которые мы добавим на формы позднее.

Чтобы избежать путаницы в именах компонентов, дадим им осмысленные имена, как показано на рис 2.10. Для настройки имени компонента используется свойство **Name** в инспекторе объектов.

Поэтому введем следующие обозначения для имен компонентов: имена компонентов для работы с таблицами начинаются с префикса **T_**, для работы с запросами с **Q_**, с наборами данных с **DS_**.

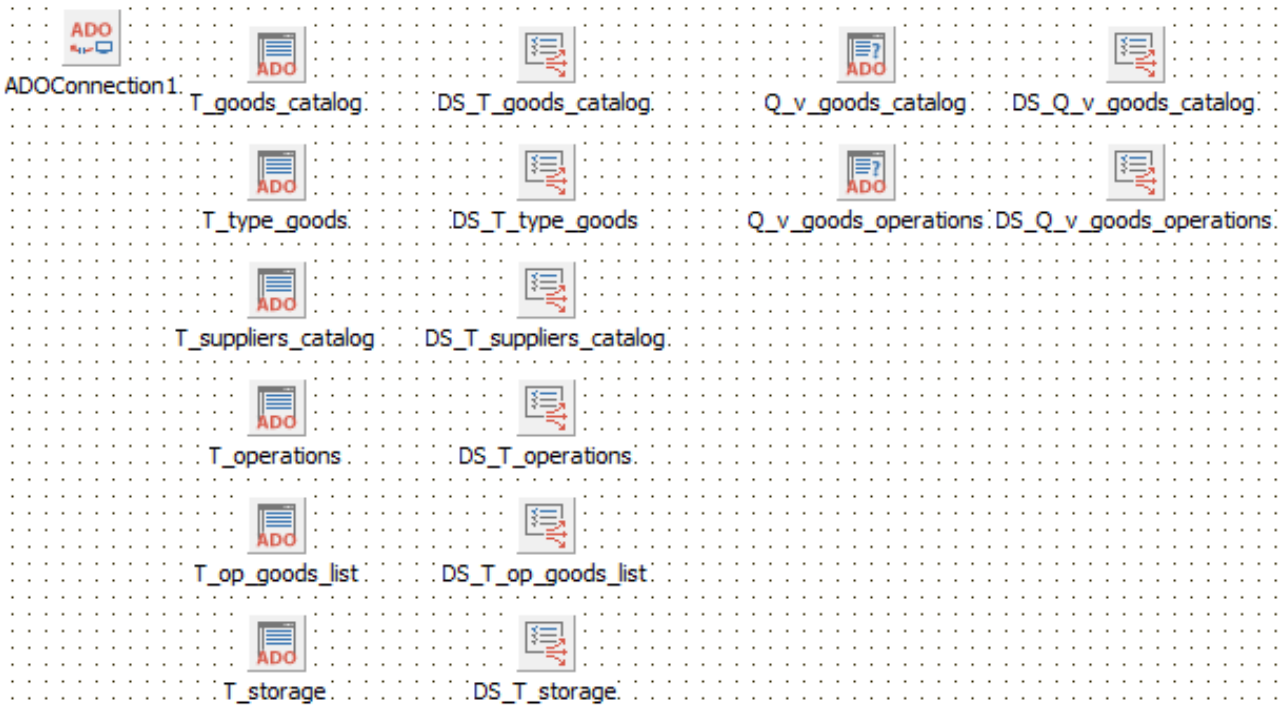


Рис. 2.10 – Размещения компонентов в **DataModule2** с измененными именами

Теперь для *всех* компонентов **ADOTable** свойству **Connection** выставите **ADOConnection1**, далее назначьте свойства **TableName** именами *соответствующих* таблиц из базы данных, а затем активируйте свойства **Active** (рис 2.11):

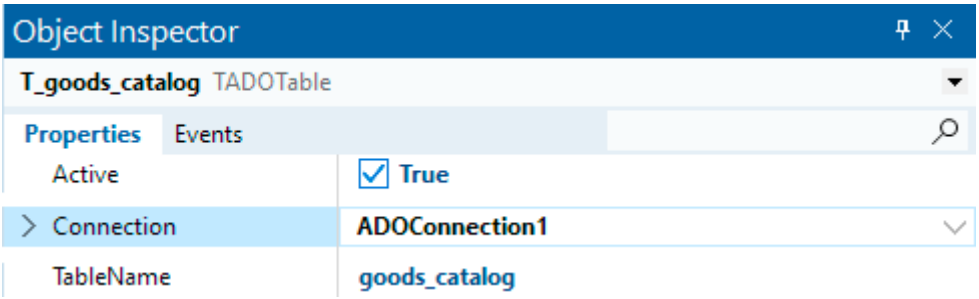


Рис. 2.11 – Настройка свойств компонента **ADOTable**

Далее привяжем компоненты **DataSource** к соответствующим компонентам **ADOTable**, установив свойства **DataSet** (рис 2.12):

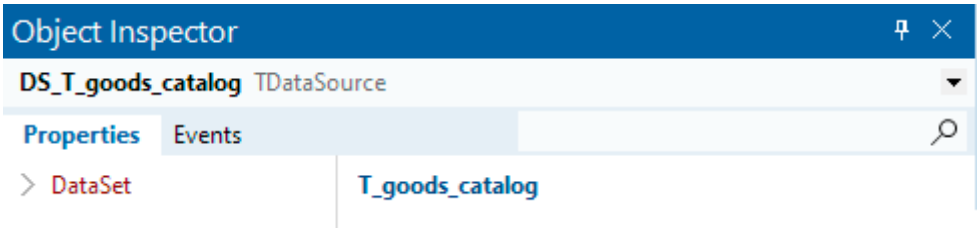


Рис. 2.12 – Настройка свойств компонента **DataSource**

Для организации работы с данными (отображение информации из нескольких таблиц и т.п.) в среде реляционных БД предпочтительным способом является использование языка **SQL** (*Structured Query Language* – язык структурированных запросов), который поддерживается всеми современными реляционными СУБД.

Для работы с запросами на языке *SQL* используются компоненты **ADOQuery**.

Рассмотрим пример настройки компонентов **ADOQuery** для выбора данных из ранее созданных двух представлений (рис 2.13). Основным свойством в данном случае является – **SQL**, оно настраивается первым.

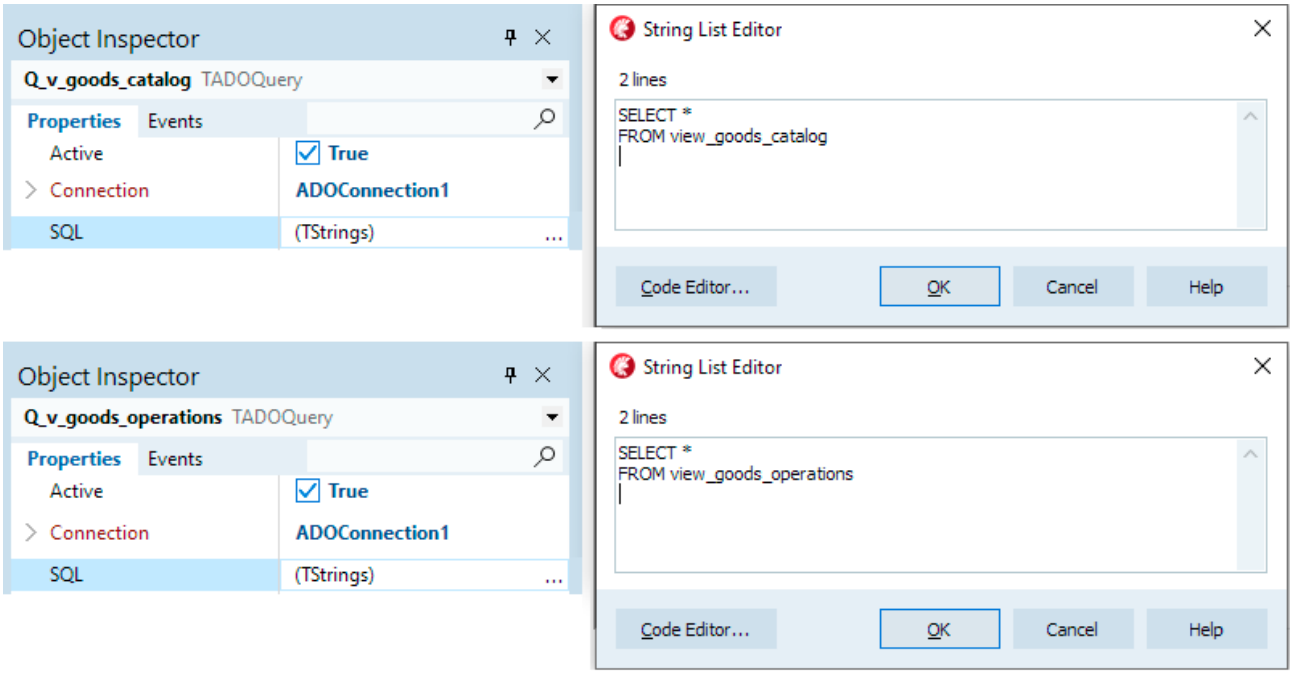


Рис. 2.13 – Настройка свойств компонентов **ADOQuery**

Теперь привяжите оставшиеся два компонента **DataSource** (*DS_Q_v_goods_catalog* и *DS_Q_v_goods_operations*) к настроенным **ADOQuery**.

2.3 Организация взаимодействия форм с наборами данных в контейнере DataModule

При создании приложения будем использовать три формы: на первой форме будет отображаться информация по операциям с товаром, на второй – сведения о поставщиках, на третьей – информация о товаре и типах товара. Поскольку первая форма была сформирована в момент создания проекта, то необходимо добавить еще *две* формы. Для этого в меню **RAD Studio** выберите **Files** затем **New** и щелкните **Form – Delphi**.

После этого в проекте должно быть 4 файла с исходным кодом (*Unit.pas - pascal*) привязанных каждый к своей форме (*Unit.dfm – delphi form*) (рис 2.14).

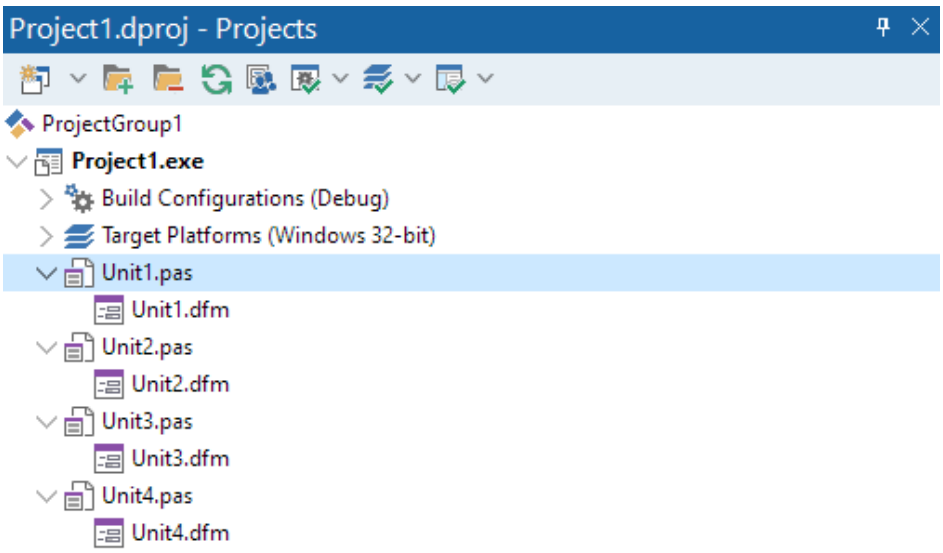


Рис. 2.14 – Состав проекта на Delphi

На визуальных формах будем размещать управляющие компоненты: кнопки, поля ввода, меню и т.д. По умолчанию главной считается первая созданная форма, она изначально появляется на экране при запуске проекта (главную форму можно поменять в настройках проекта).

Формы в проекте пока не связаны между собой, поэтому мы не можем обращаться из программного кода одной формы к компонентам других форм. Для организации взаимосвязи между формами и/или модулями, для главной формы выберите пункт меню **File** и нажмите **Use Unit**, после чего появится окно связи модулей проекта (рис. 2.15). В связи с тем, что первая форма будет вызывать все остальные, выберем все модули и нажмем кнопку **OK**.

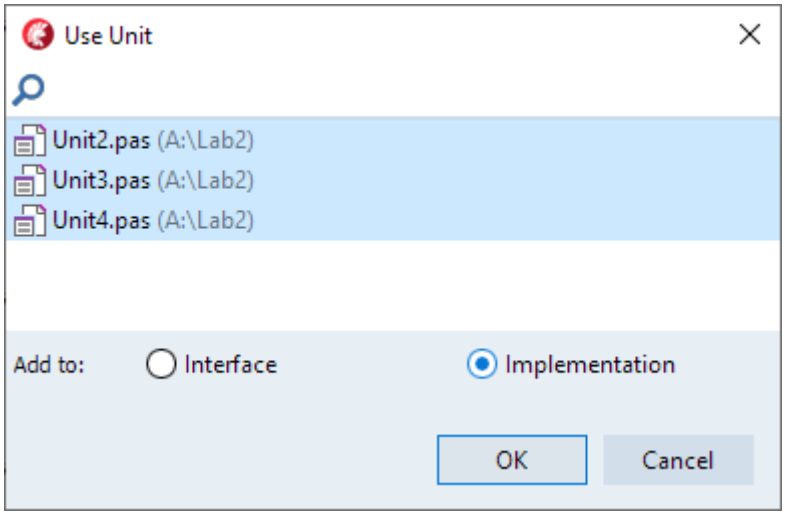


Рис. 2.15 – Создание связи между модулями проекта

Разместим на первой форме компонент **DBGrid**, который служит для отображения записей набора данных в табличной форме, взяв его из палитры компонентов на вкладке **DataControls**. Установим свойство **DataSource** компонента **DBGrid1** в значение **DS_Q_v_goods_operations** (рис. 2.16).

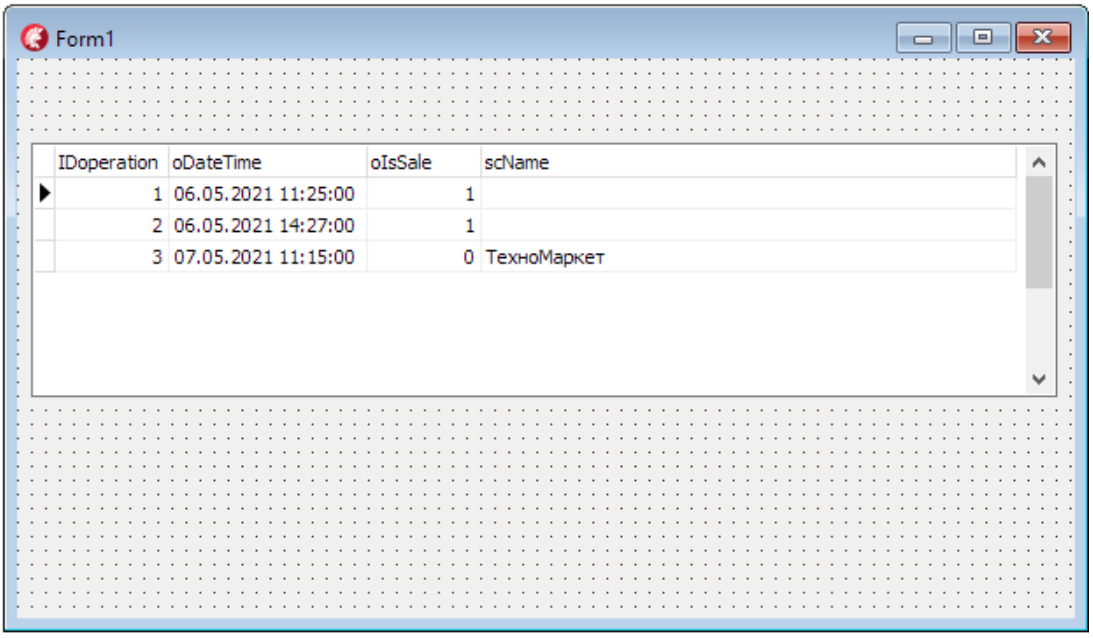


Рис. 2.16 – Отображение данных таблицы в DBGrid

2.4. Создание главного меню и вызов форм

Далее добавьте к программе меню. Разместите на главной форме в произвольном месте невизуальный компонент **MainMenu**. Опции главного меню создаются с помощью специального редактора. Редактор меню вызывается с помощью двойного щелчка по компоненту **MainMenu**. Первоначально меню пустое, но имеет один выделенный элемент. Для создания первой опции нужно перейти в инспектор объектов и свойству **Caption** присвоить нужное название. Создайте 3 пункта меню как показано на рис. 2.17.

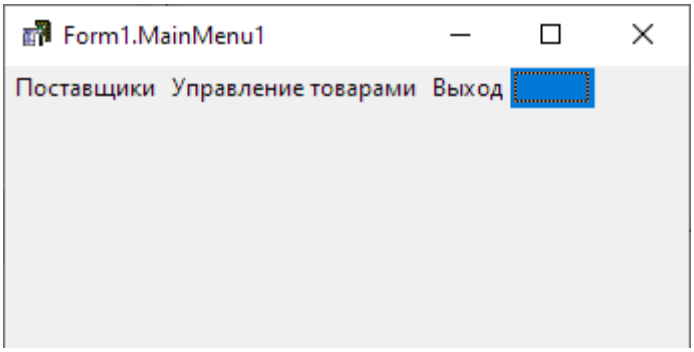


Рис. 2.17 – Создание основного меню формы

Обратите внимание на то, что среда **Delphi** автоматически создает следующий пустой пункт меню верхнего уровня.

Выберите пункт меню **Выход**, перейдите на вкладку инспектора объектов – **Events** (*События*) и щелкните дважды мышью на пустую строчку на против события **OnClick** (рис 2.18):

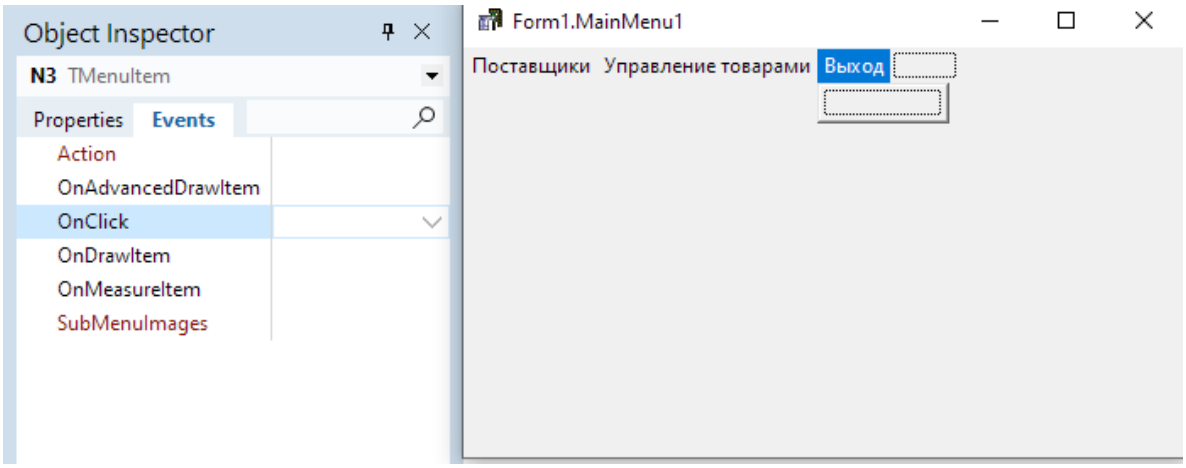


Рис. 2.18 – Назначение обработчика события

и запишем следующий текст в процедуре обработчика:

```
Close;
```

Данный код будет закрывать приложение при клики мышью на пункт меню **Выход**.

Далее привяжем к пунктам меню вызов остальных форм приложения. Для пункта **Поставщики** в обработчик **OnClick** необходимо вписать код одного из двух способов показа нужной формы:

– в случае обычной формы:

```
Form3.Show;
```

– в случае модальной формы:

```
Form3.ShowModal;
```

Разница между этими формами заключается в том, что обычная форма разрешает свободный переход между всеми формами, находящимися в данный момент на экране, а модальная форма в момент вызова блокирует переход между формами проекта до тех пор, пока не будет закрыта, и работа возможна только в ней.

Для пункта **Управление товарами** в обработчик **OnClick** добавьте вызов четвертой формы.

Теперь необходимо задать связь между формой **Form3**, отвечающей за поставщиков, и модулем с контейнером **DataModule** (**Unit2**), используйте для этого команду **Use Unit**. После этого разместите на форме компонент **DBGrid** и свяжите его с набором данных, содержащим таблицу **Поставщики** (**suppliers_catalog**) (рис 2.19).

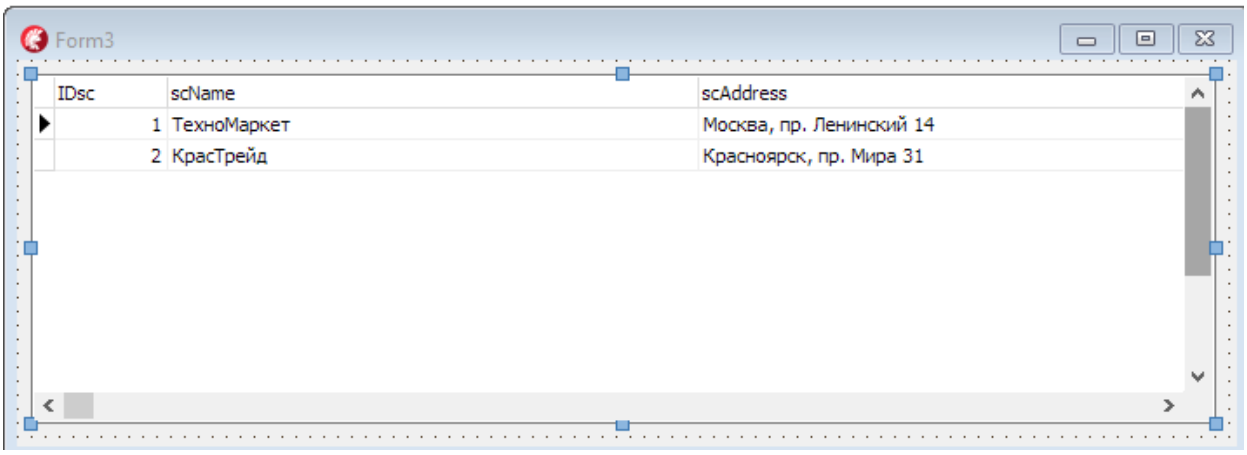


Рис. 2.19 – Отображение данных таблицы **Поставщики**

Для проверки работоспособности приложения запустим его. Сохраним проект, используя элемент меню **Save All** или одноименную пиктограмму на панели инструментов, и откомпилируем приложение с помощью элемента меню **Run** или клавиши **F9**.

Добавлять записи в набор данных и, соответственно, в таблицу **suppliers_catalog** можно непосредственно из компонента **DBGrid**. Для этого нажмем на клавиатуре клавишу **Insert** или, находясь на последней записи набора данных, клавишу управления курсором ↓ (рис 2.20). Набор данных автоматически перейдет в режим добавления новой записи. После ввода значений в поля записи можно запомнить запись в наборе данных, перейдя на другую запись при помощи клавиш управления курсором. Отказаться от запоминания записи можно, нажав клавишу **Esc**. Следует отметить, что в поля автоинкрементного типа нельзя вводить данные, тем не менее при сохранении записи им будет автоматически присвоено новое значение.

Для изменения записи переместим указатель текущей записи в нужное место и изменим значения там, где это необходимо. Набор данных автоматически перейдет в режим редактирования. Для удаления записи следует установить на нее указатель текущей записи и нажать комбинацию клавиш **Ctrl + Del**.

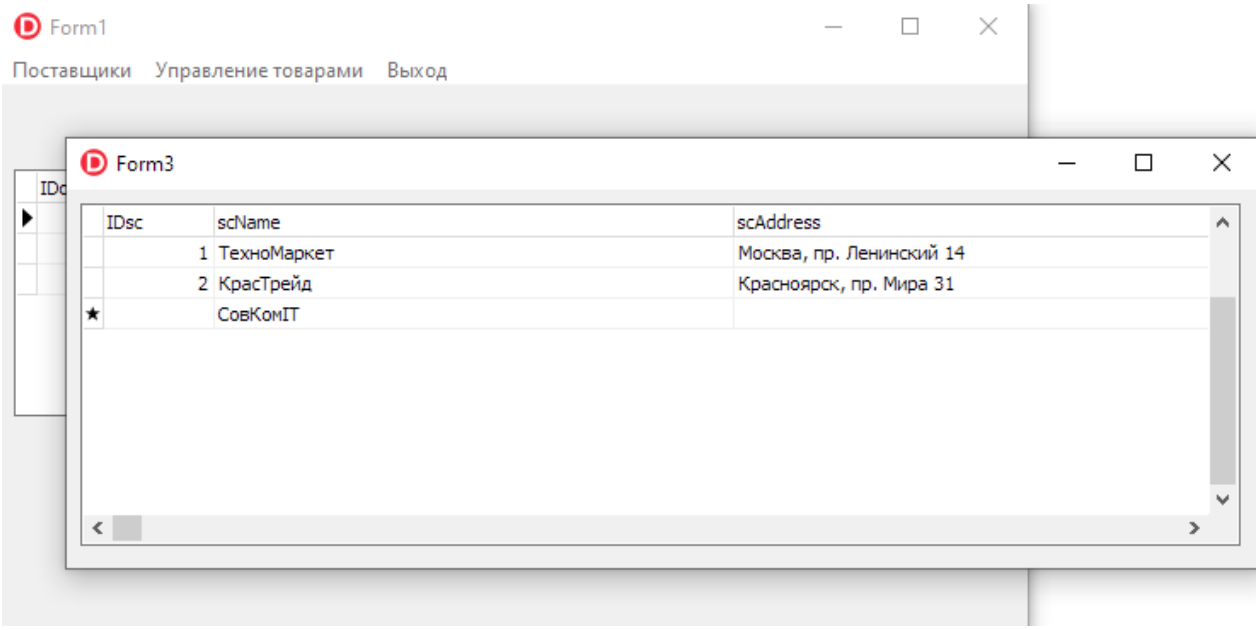


Рис. 2.20 – Редактирование данных через **DBGrid**

Можно запретить добавление и изменение данных через компонент **DBGrid**, установив для таблицы **T_suppliers_catalog** свойство **ReadOnly** в значение **True**. В таком случае придется воспользоваться подходом редактирования, который будет рассмотрен ниже.

2.5. Редактирование данных и синхронизация отображения результатов запроса

На **Form4** разместите компонент **PageControl** который позволяет организовывать просмотр контента в виде вкладок. Чтобы доступная область данного компонента покрывала всю форму, необходимо свойству **Align** задать значение **alClient** (рис 2.21).

Для добавления вкладки необходимо вызвать контекстное меню и выбрать пункт **New Page**. На данной форме будет размещаться каталог товаров и перечень типов товаров. В связи с этим добавим две вкладки (рис 2.22).

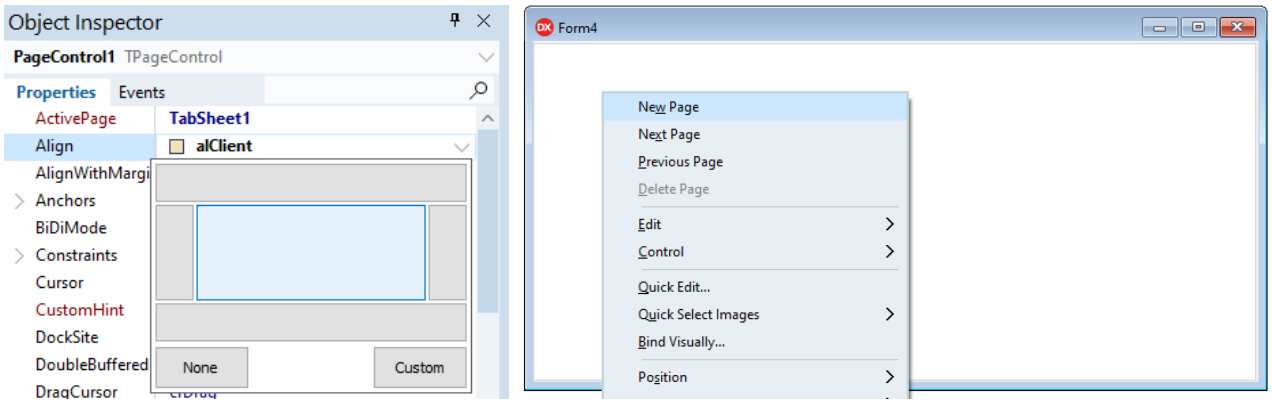


Рис. 2.21 – Настройка свойства Align и контекстное меню для создания вкладки.

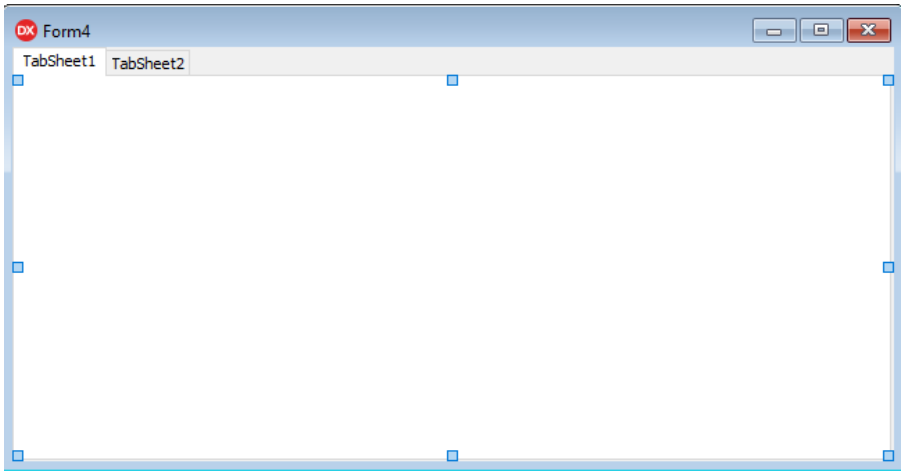


Рис. 2.22 – **PageControl** с двумя созданными вкладками.

Поместим на обоих вкладках **TabSheet** по компоненту **DBGrid** у которых выставите свойству **Align** (Выравнивание) – значение **alClient**. Далее присвойте для (рис 2.23):

– **DBGrid1** свойству **DataSource** значение **DS_Q_v_goods_catalog**

– **DBGrid2** свойству **DataSource** значение **DS_T_type_good**.

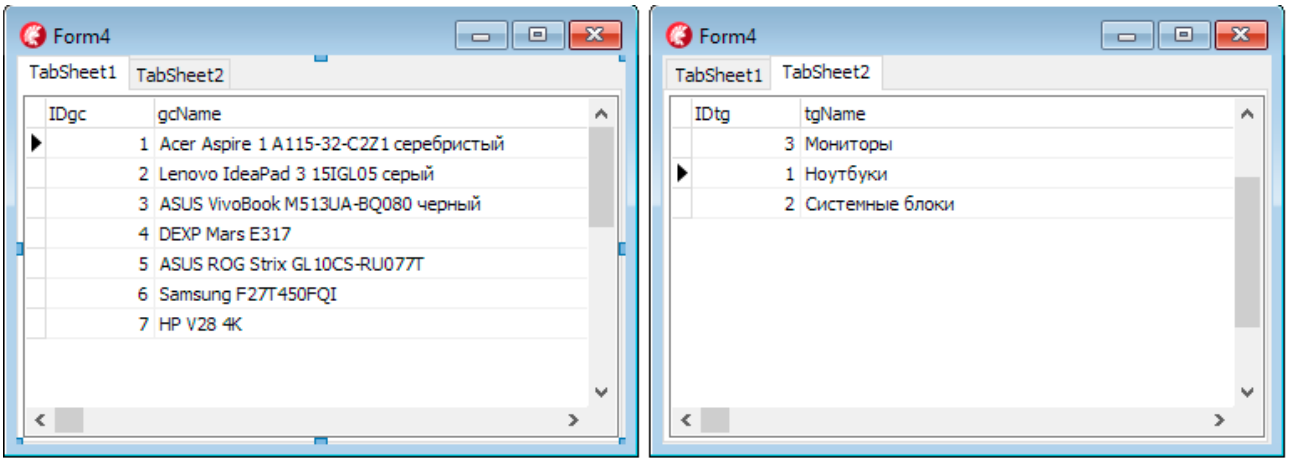


Рис. 2.23 – **TabSheet** с компонентами **DBGrid**.

Нам не требуется отображать все столбцы таблицы пользователю, поэтому настроим их отображение при помощи редактора столбцов компонента **DBGrid.** Для этого щелкнем правой кнопкой мыши на компоненте **DBGrid1** и во всплывающем меню выберем элемент **Columns Editor**. На экране появится окно редактора столбцов компонента. Далее щелкнем на кнопке **Add All Fields**, в результате чего будут добавлены столбцы, каждый из которых соответствует полю **Q_v_goods_catalog** (рис. 2.24).

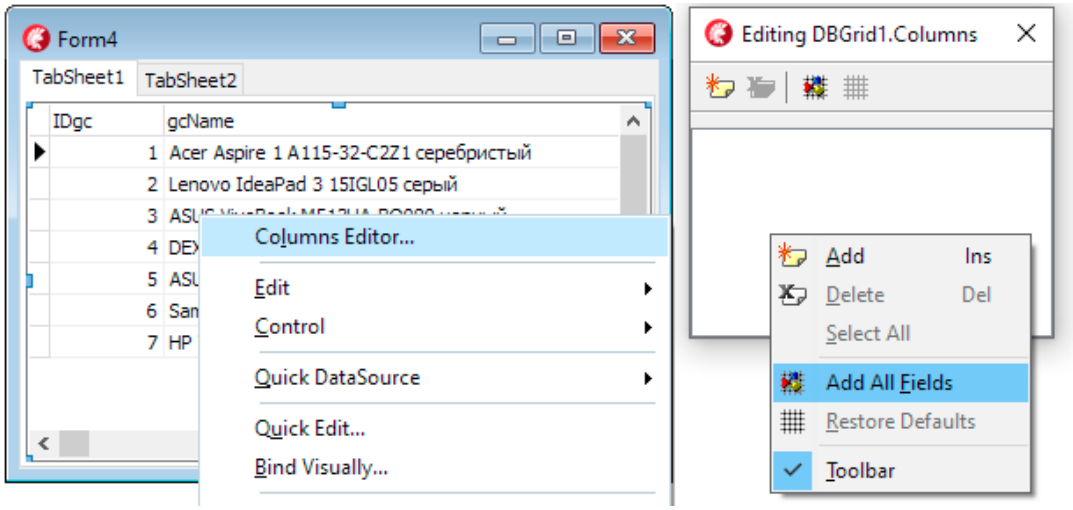


Рис. 2.24 – Добавление полей в **DBGrid**.

Для отображения в таблице нам будет достаточно полей, показывающих **тип товара**, его **название** и **цену**, поэтому оставим только их (рис. 2.25).

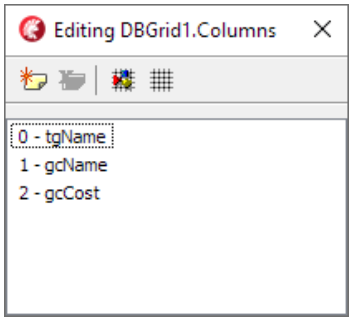


Рис. 2.25 – Удаление лишних столбцов из **DBGrid**.

Далее измените заголовок каждого столбца. Кликните в редакторе столбцов на необходимое поле и найдите его свойство **Title** в инспекторе объектов. Измените свойство **Caption** на необходимый заголовок, чтобы наименования столбцов соответствовали рис. 2.26:

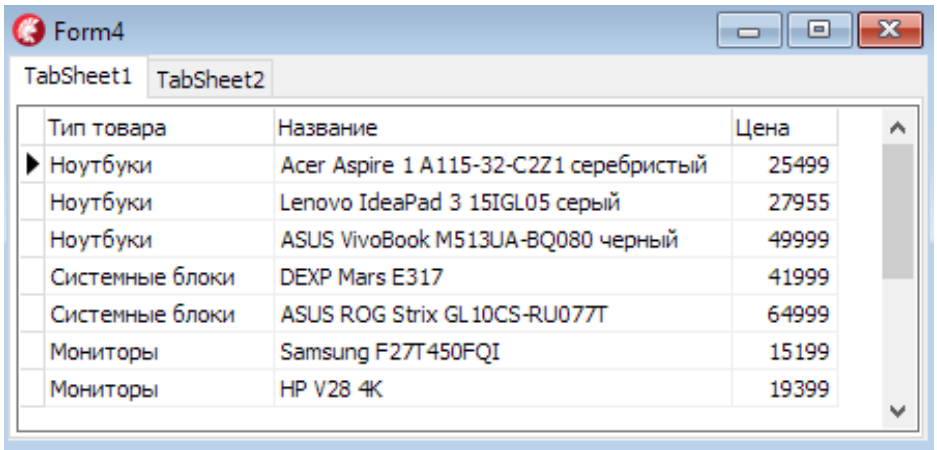


Рис. 2.26 – Изменение отображения столбцов в **DBGrid**.

Для изменения ширины столбца настроим параметр **Width**. Измените соответствующим образом заголовки и для таблицы **type_goods** на **TabSheet2**.

Затем добавьте на **TabSheet1** два компонента: **Label** (Метка) и **DBText** (Текстовое поле из БД). Назначьте им следующие свойства.

Для **Label:**

Свойство	Значение
Align	alBottom
Caption	Описание:

Для **DBText:**

Свойство	Значение
Align	alBottom
DataSource	DS_T_goods_catalog
DataField	gcDescription
AlignWithMargins	True
AutoSize	True
WordWrap	True

Пример отображения компонентов после выполнения настройки приведен на рис. 2.27

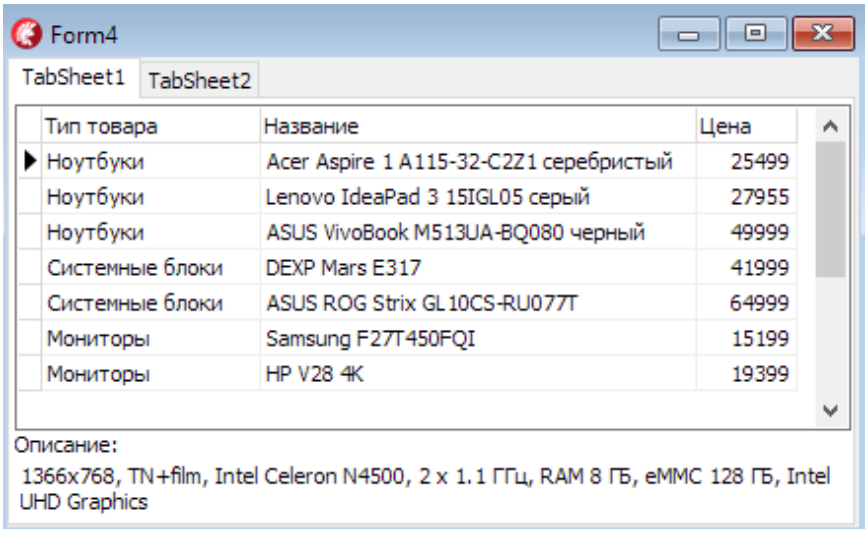


Рис. 2.27 – Добавление компонентов **Label** и **DBText**

Для того чтобы сделать автоматическое размещения одного компонента под другим используется свойство **Align** (Выравнивание) с значением **alBottom** (Снизу). При стандартном значении текст будет стоять достаточно плотно к окружающим компонентам, и для того чтобы избежать этого нужно свойству **AlignWithMargins** (Выравнивание с отступами)

назначить значение **True**, сами же параметры отступов можно отрегулировать раскрыв блок **Margins** (Отступы) и при необходимости скорректировать значения. Автоматическая подстройка размеров компонента задается при помощи свойства **AutoSize** (Авторазмер). В случае длинного текстового пояснения необходимо организовать вывод в несколько строк, для этого свойству **WordWrap** (Перенос слов) задается значение **True**.

Далее поместим на первой вкладке **TabSheet1** два компонента **Panel**. Назначим им следующие свойства:

Для **Panel1:**

Свойство	Значение
Align	alBottom
Name	InsertEditDeletePanel
Visible	True

Для **Panel2:**

Свойство	Значение
Align	alBottom
Name	PanelToInputValues
Visible	False

На **Panel1** (InsertEditDeletePanel) установим три кнопки – компонент **Button** и зададим им свойства **Caption/Name** следующими: **Вставить/InsertButton**, **Изменить/EditButton**, **Удалить/DeleteButton** (рис. 2.28). Для задания отображаемой надписи у большинства компонентов существует свойство **Caption**, для обращения к их свойствам из программного кода используется свойство **Name**.

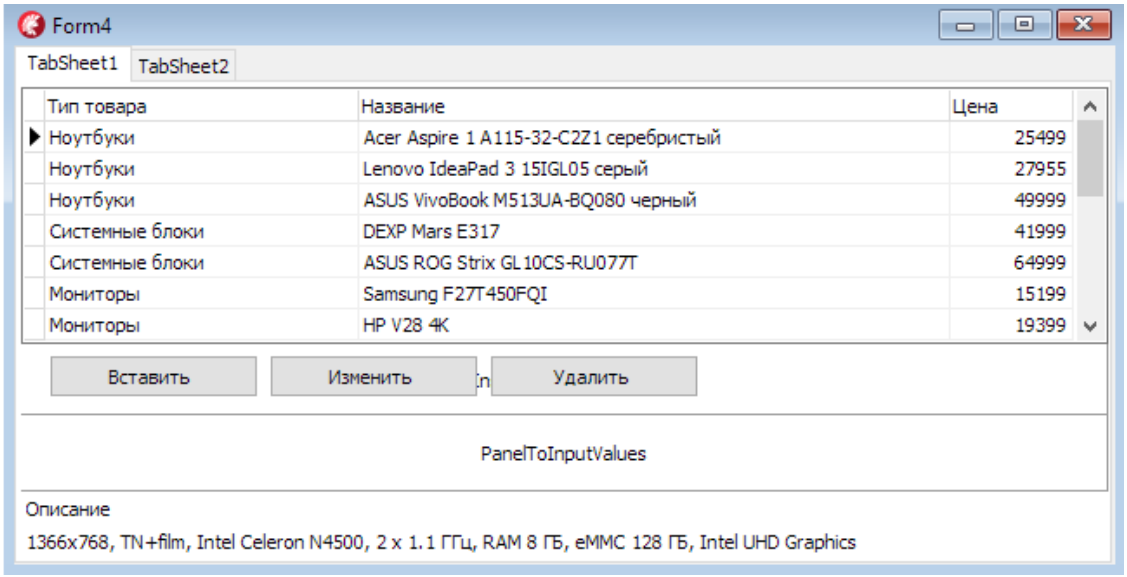


Рис. 2.28 – Добавление компонентов **Panel**

На **Panel2** (PanelToInputValues) расположим один компонент **DBLookupComboBox** и три компонента, **DBEdit**, со страницы палитры компонентов **Data Controls**. Данные компоненты позволят изменять значения соответствующих полей. Для пояснения компонентов редактирования поставим компоненты **Label**. Для сохранения или отказа от сделанных изменений в таблице поместим на панели кнопки **Запомнить/PostButton** и **Отменить/CancelButton** (рис. 2.29).

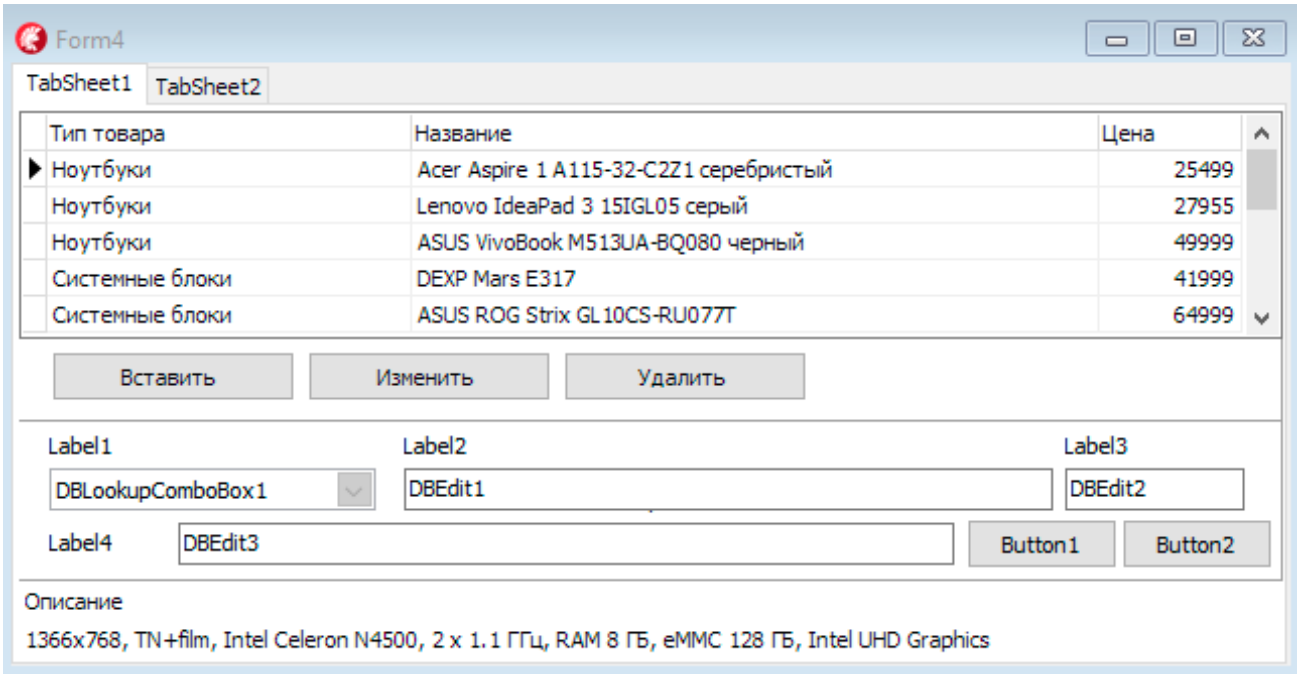


Рис. 2.29 – Размещение компонентов на **PanelToInputValues**

Теперь настроим компоненты, чтобы пользовательский интерфейс формы соответствовал рис. 2.30.

Для **DBLookupComboBox1:**

Свойство	Значение
DataSource	DS_T_goods_catalog
DataField	IDtg
ListSource	DS_T_type_goods
ListField	tgName
KeyField	IDtg

Для **DBEdit1:**

Свойство	Значение
DataSource	DS_T_goods_catalog
DataField	gcName

Для **DBEdit2:**

Свойство	Значение
DataSource	DS_T_goods_catalog
DataField	gcCost

Для **DBEdit3:**

Свойство	Значение
DataSource	DS_T_goods_catalog
DataField	gcDescription

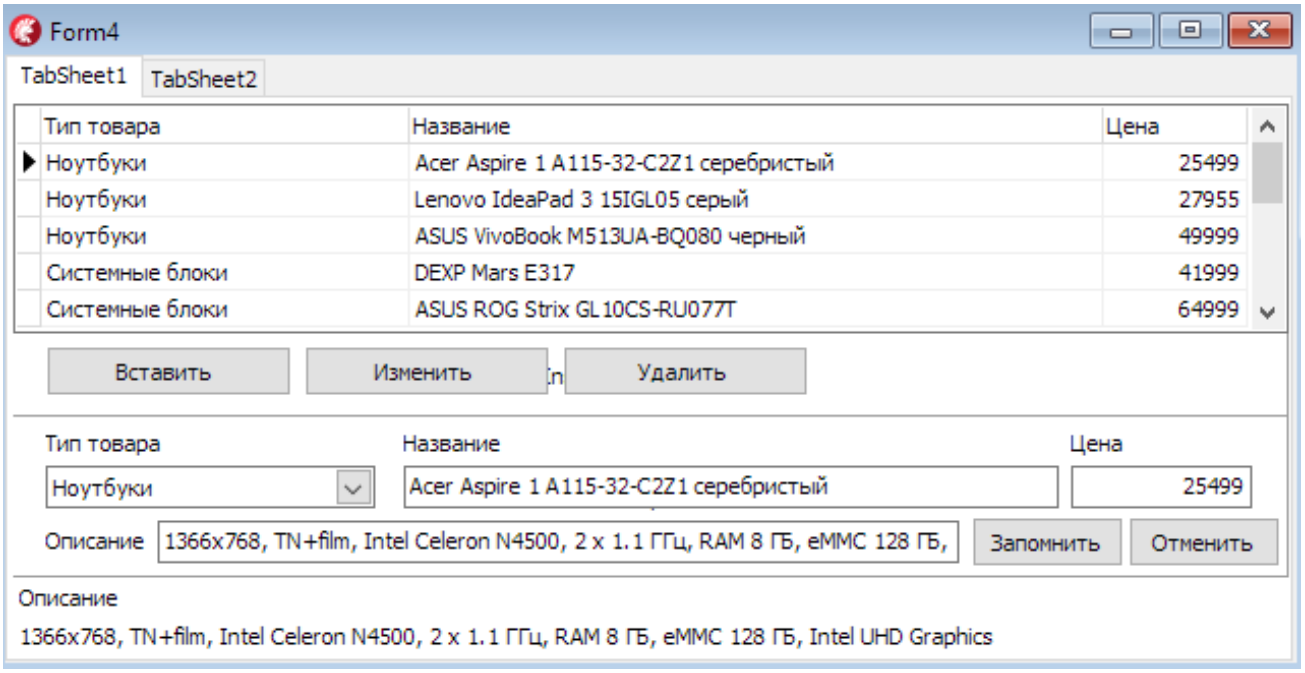


Рис. 2.30 – Окно формы редактирования товаров

Теперь необходимо прописать код кнопок.

Перейдем к определению обработчиков события нажатия на кнопки, за это действие отвечает обработчик **OnClick**. Для его определения необходимо выполнить двойное нажатие левой кнопки мыши на компоненте **Button** или в области события **OnClick** в окне инспектора объектов (пример для кнопки **InsertButton** представлен на рис. 2.31). Шаблон процедуры обработчика генерируется средой разработки автоматически, поэтому необходимо будет прописывать только код тела процедуры (между **begin** и **end**).

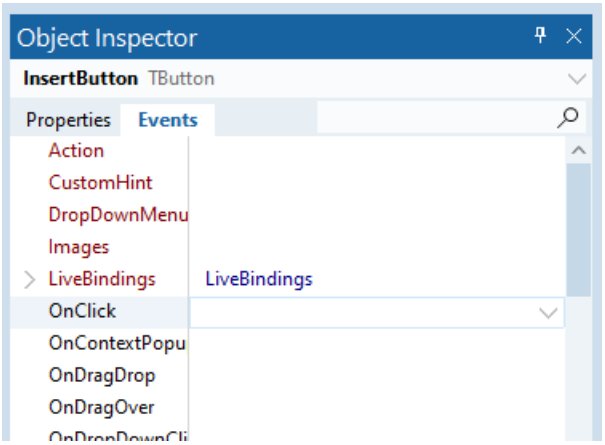


Рис. 2.31 – Настройка события **OnClick** компонента

Компонент **ADOTable** позволяет выполнять добавление информации в таблицу при помощи двух методов **Insert** и **Append**. Оба метода переводят набор данных **ADOTable** (в нашем случае **T_goods_catalog**) в состояние добавления записи **dsInsert**. Их различие заключается в том, что метод **Append** добавляет запись в конец набора данных, в то время как метод **Insert** добавляет ее после текущей записи. Для перехода в состояние **dsInsert** необходимо, чтобы набор данных находился в режиме просмотра **dsBrowse**.

Определим обработчик нажатия кнопки **InsertButton**, т. е. обработчик события **OnClick**:

```
procedure TForm4.InsertButtonClick(Sender: TObject);
      begin
        if DataModule2.T_goods_catalog.State = dsBrowse then
        begin
          // Отключить возможность изменения курсора НД
          // в компоненте DBGrid1
          DBGrid1.Enabled := False;
          // Сделать невидимой панель с кнопками Вставить, Изменить,
          // Удалить
          InsertEditDeletePanel.Visible := False;
          // Сделать видимой панель для ввода значения в запись
          PanelToInputValues.Visible := True;
          // Перевести НД в режим добавления записи, используя
          // метод Append
          DataModule2.T_goods_catalog.Append;
        end;
      end;
```

Определим обработчик нажатия кнопки **EditButton**:

```
procedure TForm4.EditButtonClick(Sender: TObject);
      begin
        if DataModule2.T_goods_catalog.State = dsBrowse then
```

```
      begin
        // Отключить возможность изменения курсора НД
        // в компоненте DBGrid1
        DBGrid1.Enabled := False;
        // Сделать невидимой панель с кнопками Вставить,
        // Изменить,Удалить
        InsertEditDeletePanel.Visible := False;
        // Сделать видимой панель для ввода значения в запись
        PanelToInputValues.Visible := True;
        // Перевести НД в режим редактирования записи, используя
        // метод Edit
        DataModule2.T_goods_catalog.Edit;
      end;
    end;
```

Метод **Edit** переводит набор данных **T_goods_catalog** в состояние редактирования записи **dsEdit**. Редактирование значений полей осуществляется в компонентах **DBLookupComboBox1**, **DBEdit1**, **DBEdit2**, **DBEdit3**. При этом необходимо, чтобы набор данных находился в режиме просмотра **dsBrowse**.

Определим обработчик нажатия кнопки **DeleteButton**:

```
procedure TForm4.DeleteButtonClick(Sender: TObject);
begin
  if DataModule2.T_goods_catalog.State = dsBrowse then
    if MessageDlg('Подтвердите удаление записи', mtConfirmation, [mbYes, mbNo],
      0) = mrYes then
      DataModule2.T_goods_catalog.Delete;
end;
```

Если набор данных **T_goods_catalog** находится в режиме просмотра **dsBrowse**, то при выполнении функции **MesssageDlg** будет вызвано окно диалога. При нажатии кнопки **Yes** произойдет удаление текущей записи в наборе данных **T_goods_catalog**.

Определим обработчик нажатия кнопки **PostButton**:

```
procedure TForm4.PostButtonClick(Sender: TObject);
      begin
        if DataModule2.T_goods_catalog.State in [dsInsert, dsEdit] then
        begin
          DataModule2.T_goods_catalog.Post;
          // Сделать видимой панель с кнопками Вставить, Изменить,
          // Удалить
          InsertEditDeletePanel.Visible := True;
          // Сделать невидимой панель для ввода значения в запись
          PanelToInputValues.Visible := False;
          // Включить возможность изменения курсора НД
          // в компоненте DBGrid1
```

```
        DBGrid1.Enabled := True;
      end;
    end;
```

Если набор данных находится в режиме добавления новой записи или редактирования, то происходит выполнение метода **Post**, который запоминает текущее состояние записи в таблице. После запоминания набор данных переводится в режим просмотра **dsBrowse**.

Определим обработчик нажатия кнопки **CancelButton**:

```
procedure TForm4.CancelButtonClick(Sender: TObject);
    begin
      if DataModule2.T_goods_catalog.State in [dsInsert, dsEdit] then
      begin
        DataModule2.T_goods_catalog.Cancel;
        // Сделать видимой панель с кнопками Вставить, Изменить,
        // Удалить
        InsertEditDeletePanel.Visible := True;
        // Сделать невидимой панель для ввода значения в запись
        PanelToInputValues.Visible := False;
        // Включить возможность изменения курсора НД
        // в компоненте DBGrid1
        DBGrid1.Enabled := True;
      end;
    end;
```

В этом случае нахождение набора данных в режиме добавления новой записи или в режиме редактирования приводит к выполнению метода **Cancel**. Этот метод отменяет запоминание записи в таблице и переводит набор данных в режим просмотра **dsBrowse**.

Установите следующие свойства для компонента **DBGrid1**:

Свойство	Значение
ReadOnly	True
Options (dgRowSelect)	True

Для запрета изменения данных в таблице через компонент **DBGrid1** используется свойство **ReadOnly**. Для выделения целой строки (а не ячейки) в **DBGrid1** используется свойство **dgRowSelect**.

При переходе по записям в наборах данных (**ADOTable**, **ADOQuery** и т. п.), возникают события **BeforeScroll** (перед переходом на другую запись) и **AfterScroll** (после перехода).

В нашем случае необходимо, чтобы после перемещения по записям в наборах данных компонентов **ADOQuery** в наборе данных **ADOTable** связанным с редактируемой таблицей выбиралась соответствующая за-

пись. Для этого необходимо настроить событие **AfterScroll.** В нашем случае будет происходить настройка события у компонента **Q_v_goods_catalog**:

```
procedure TDataModule2.Q_v_goods_catalogAfterScroll(DataSet: TDataSet);
      begin
          // проверка количества отображаемых записей
          if Q_v_goods_catalog.RecordCount > 0 then
            // выполнение синхронизации методом Locate
            T_goods_catalog.Locate('IDgc', Q_v_goods_catalogIDgc.Value, [])
      end;
```

Для того чтобы данный фрагмент кода заработал необходимо в редакторе полей для компонента **Q_v_goods_catalog** добавить все поля (рис. 2.32):

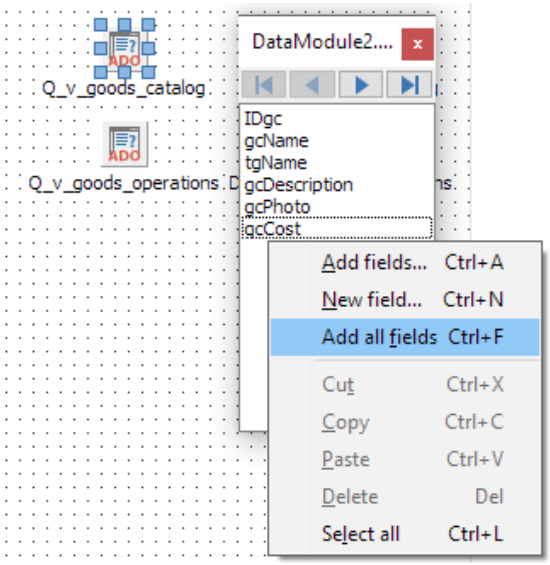


Рис. 2.32 – Добавление полей в **Q_v_goods_catalog**

При срабатывании события **AfterScroll** проверяется, имеются ли отображаемые записи (количество записей больше нуля), после чего в **T_goods_catalog** будет происходить поиск записи по значению **Q_v_goods_catalogIDgc**. Переход на нужную запись осуществляется при помощи метода **Locate**.

Компонент **ADOQuery**, имеет следующую особенность: после выполнения запроса на чтение к базе данных происходит отображение записи НД в неизменном виде независимо от того, изменялось ли после запроса содержимое таблицы БД. После обновления информации в базе данных приходится закрывать и снова открывать компонент **ADOQuery**, т. е. повторно выполнять запрос к БД. В связи с этим нам необходимо добавить обработчики событий **AfterPost** и **AfterDelete** для компонента **ADOTable.** Обновление набора данных в **ADOQuery** возможно путем его закрытия и открытия, которое осуществляется при помощи методов

Close() и **Open()** или же изменения свойства **Active** сначала в значение **False**, а потом в значение **True**. Также обновить данные можно при помощи метода **Requery()**.

В нашем случае для компонента **T_goods_catalog** пропишите следующие события:

```
procedure TDataModule2.T_goods_catalogAfterDelete(DataSet: TDataSet);
	begin
		Q_v_goods_catalog.Requery();
	end;

procedure TDataModule2.T_goods_catalogAfterPost(DataSet: TDataSet);
	var pos:TBookmark;
	begin
		pos := Q_v_goods_catalog.GetBookmark;
		Q_v_goods_catalog.Close();
		Q_v_goods_catalog.Open();
		Q_v_goods_catalog.GotoBookmark(pos);
	end;
```

Для того чтобы позиционирование курсора в наборе данных сохранялось в момент редактирования используется закладка (тип данных **TBookmark**). Тип **TBookmark** является указателем на экземпляр типа «закладка». В НД можно использовать закладки аналогично тому как закладки используются в жизни. При помощи закладок легко пометить место в НД и потом вернуться к нему.

КОНТРОЛЬНЫЕ ВОПРОСЫ

Что из себя представляет *ODBC*?
Чем компонент *DataModule* отличается от *Form*?
Что содержит *ConnectionString*?
Что из себя представляет порт (для *MySQL* - 3306)?
Дайте расшифровку аббревиатуры *SQL*.
Для чего применяется компонент *ADOQuery*?
Чем *Events* отличаются от *Properties*?
Опишите назначение компонента *DataSource.*
Дайте характеристику обычным и модальным формам.
Чем свойство *Caption* отличается от *Name*?
Для чего применяется команда *Use Unit*?
За что отвечает свойство компонентов – *Align*?
Охарактеризуйте свойство *Margins*?
В каких режимах могут находится наборы данных?
В какой момент срабатывает событие *AfterScroll*?
Для чего применяется метод *Locate*?
Что из себя представляет *Bookmark*?