Государственное бюджетное профессиональное образовательное учреждение
«Байконурский электрорадиотехнический техникум имени М.И. Неделина»
(ГБ ПОУ «БЭРТТ»)

Методические рекомендации
по выполнению лабораторных работ

по междисциплинарному курсу
«Инструментальные средства разработки программного обеспечения»

для специальности 09.02.03 «Программирование в компьютерных системах»

г. Байконур
2016 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Данные методические указания по выполнению лабораторных работ составлены в соответствии с ФГОС по специальности СПО 09.02.03 «Программирование в компьютерных системах» (базовый уровень) по МДК «Инструментальные средства разработки программного обеспечения».

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения МДК должен:

уметь:

- владеть основными методологиями процессов разработки программного обеспечения;
- выполнять интеграцию модулей в программную систему;
- выполнять отладку программного продукта с использованием специализированных программных продуктов;
- осуществлять разработку тестовых наборов и тестовых сценариев.

знать:

- основные подходы к интегрированию программных модулей;
- основные методы и средства эффективной разработки;
- принципы построения, структуры и приемы работы с инструментальными средствами, поддерживающими создание программного обеспечения.

Перечень лабораторных работ

1	Проектирование многооконных приложений. Создание и тестирование dll-библиотек средствами Delphi
2	Разработка приложения для работы с базами данных в Delphi. ADO-технологии
3	Разработка приложения для работы с базами данных в Delphi. ADO-технологии
4	Разработка приложения для работы с базами данных в Delphi. Создание запросов
5	Разработка приложения для работы с базами данных в Delphi. Создание связанных таблиц
6	Создание и настройка справочной системы приложения. Создание установочного пакета
7	Построение функциональной модели
8	Разработка UML диаграмм
9	Создание сценариев использования
10	Разработка прототипа автоматизированной системы с использованием ForeUI
11	Основы работы с CMS Joomla
12	Работа с компонентами CMS Joomla
13	Работа с модулями CMS Joomla
14	Разработка интернет-магазина средствами CMS Joomla
15	Разработка интернет-магазина CMS Joomla
16	Разработка интернет-магазина CMS Joomla
17	Изучение приложения Adobe Dreamweaver

Правила выполнения лабораторных работ

Студент перед выполнением лабораторной работы должен строго выполнить весь объем домашней подготовки, указанный в описании соответствующей лабораторной работы. Выполнению каждой работы может предшествовать проверка готовности студента, которая проводится преподавателем.

Отчет о лабораторной работе должен содержать исчерпывающие систематизированные сведения о выполненной работе. Оформление отчета о лабораторной работе должно быть предусмотрено заданием или планом выполнения работы. Отчет составляется исполнителем работы и защищается в индивидуальном порядке в сроки, установленные преподавателем.

Общими требованиями к отчету являются:

- четкость и логическая последовательность изложения материала;
- убедительность аргументации;
- краткость и точность формулировок, исключающих возможность неоднозначного толкования;
- конкретность изложения результатов работы;
- обоснованность выводов.

Лабораторная работа №1
Тема: «Проектирование многооконных приложений.
Создание и тестирование dll-библиотек средствами Delphi»

Цель лабораторной работы:

- получение практических навыков создания многооконных приложений;
- получение практических навыков создания dll-библиотек;
- развитие познавательного интереса.

Предварительная подготовка: изучить лекционный материал междисциплинарного курса.

Количество часов: 2 часа.

Постановка задачи

Задание № 1. Разработать приложение Delphi, позволяющее создавать многооконное приложение.

Задание № 2. Разработать приложение Delphi, позволяющее создать dll-библиотеку.

Методика выполнения работы

Windows используются два вида приложений - это однодокументные и многодокументные.

Многодокументное приложение может использовать в своей работе несколько документов. Примером может быть Microsoft Word и Excel.

В многодокументном окне главная форма может содержать несколько дочерних окон. Свойство FormStyle определяет какой будет форма (главной или дочерней). Главная форма может быть только одна и свойство FormStyle должно быть fsMDIForm, а для дочерних форм - fsMDIChild.

В главной форме лучше не использовать такие средства управления как кнопки, таблицы строк и надписи, т.к. они будут мешать при просмотре дочерних форм. В интерфейс для главной формы лучше включать: меню, строку состояния, панель инструментов. Остальная незанятая область будет использована для дочерних форм.

Если при запуске приложения не требуется создание экземпляра дочернего окна, то его создание лучше организовать динамически, во время создания приложения.

На первом этапе создаётся приложение состоящее из 2-х форм. Первой присваивается, используя Инспектор Объектов, заголовок "Главная форма", имя - MainForm, FormStyle - fsMDIForm. Для второй "Дочерняя форма", ChaildForm, fsMDIChild - соответственно. На первую форму помещается объект меню и задаются такие пункты как на рисунке 1.

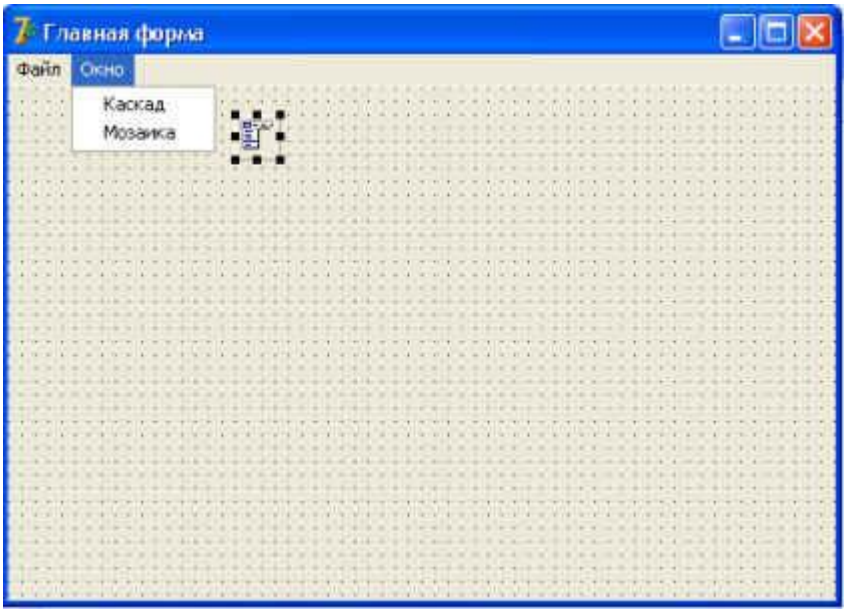


Рисунок 1

На вторую форму помещаются компонент SaveDialog, кнопку Button, Memo (рисунок 2):

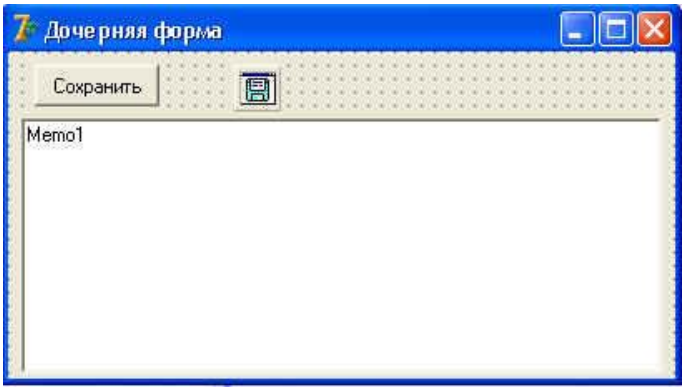


Рисунок 2

Для того чтобы второе окно создавалось динамически, в процессе работы приложения, нужно через меню Delphi Project -> Options открыть вкладку Forms. В левой части окна, под надписью Auto-create forms перечислены формы, создаваемые автома-тически, в правой части формы, под надписью Available forms, будут располагаться формы, которые необходимо создавать вручную (рисунок 3).

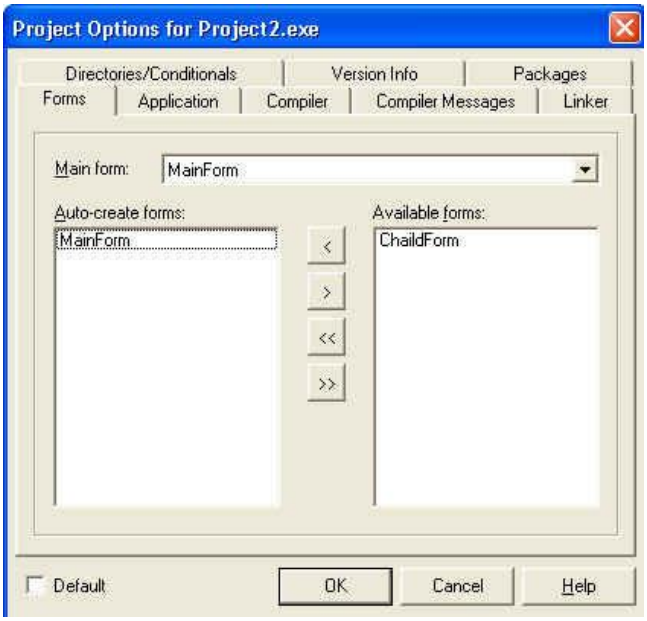


Рисунок 3

Ниже следующий код помещается в обработчик меню "Файл" главной формы:

```
procedure TMainForm.N2Click(Sender: TObject);
begin
  ChaildForm:=TChaildForm.Create(Application);
end;
```

Запустив приложение и нажав на "Файл" создаётся вторая формв, при повторном нажатии и далее будут создаваться дочерние формы (рисунок 4).

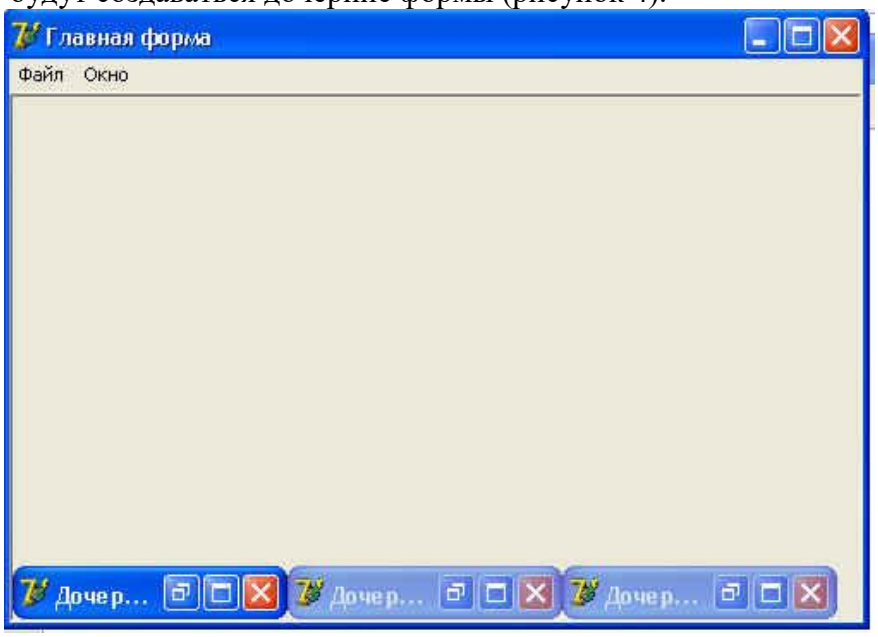


Рисунок 4

Для закрытия дочерней формы необходимо прописать в обработчике onClose формы свойство Action := caFree;

Свойство Action может принимать значение caNone - ничего не делать, caMinimize - свернуть окно (это значение используется по умолчанию), caFree закрыть окно.

На дочерней форме расположен компонент, в который необходимо печатать текст и чтобы случайно не потерять набранное нужно делать проверку. Обработчик закрытия формы будет следующим:

```
procedure TChaildForm.FormClose(Sender: TObject; var Action: TCloseAction);
```

```
begin
  if Memo1.Modified then
   begin
    if MessageDlg('Текст не сохранен! Закрыть окно?',
      mtConfirmation, [mbYes, mbNo], 0) = mrYes
       then Action := caFree ;
  end
   else Action := caFree;
end;
```

Для сохранения содержимого редактора пишется следующий обработчик кнопки:

```
procedure TChaildForm.BitBtn1Click(Sender: TObject);
begin
   if SaveDialog1.Execute then
     Memo1.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

Дочерние окна, при таком создании, могут отличаться по размеру от первоначально установленных. Если это критично, то можно задать высоту и ширину такой какая необходима в обработчике создания формы (рисунок 5):

```
procedure TChaildForm.FormCreate(Sender: TObject);
begin
   Height:= 231;
   Width := 410; end;
```

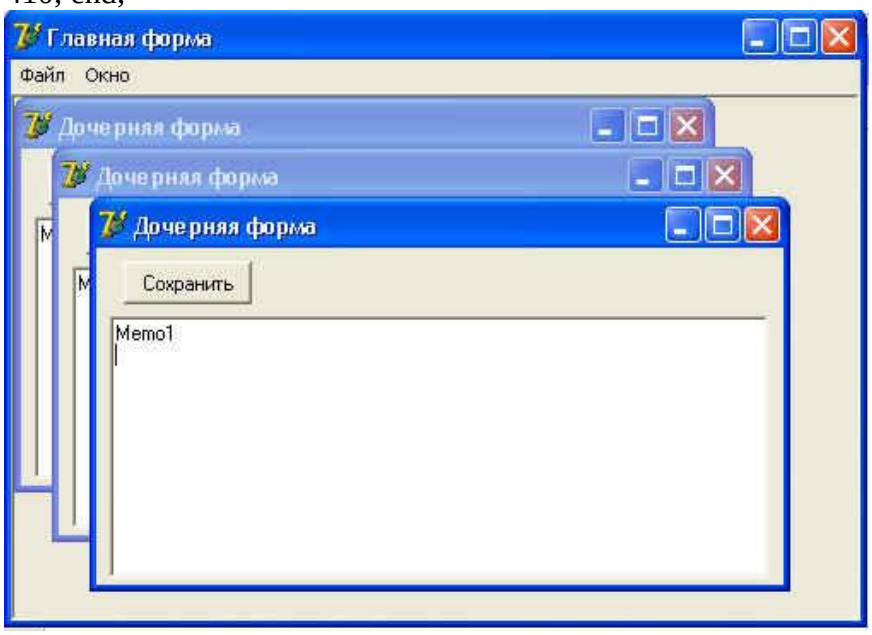


Рисунок 5

Дочерние формы по умолчанию размещаются каскадом. Для пункта меню главной формы "Каскад" напишем обработчик:

```
procedure TMainForm.N4Click(Sender: TObject);
begin
    Cascade;
end;
```

Дочерние формы могут располагаться мозаикой. Для этого свойство формы будет Tile. При мозаичном расположении мы можем дочерние окна разметить по всей ширине

или по всей высоте клиентской области. Это определяет свойство TitleMode (tbHorizontal - вся ширина, tbVertical - вся высота) (рисунок 6).

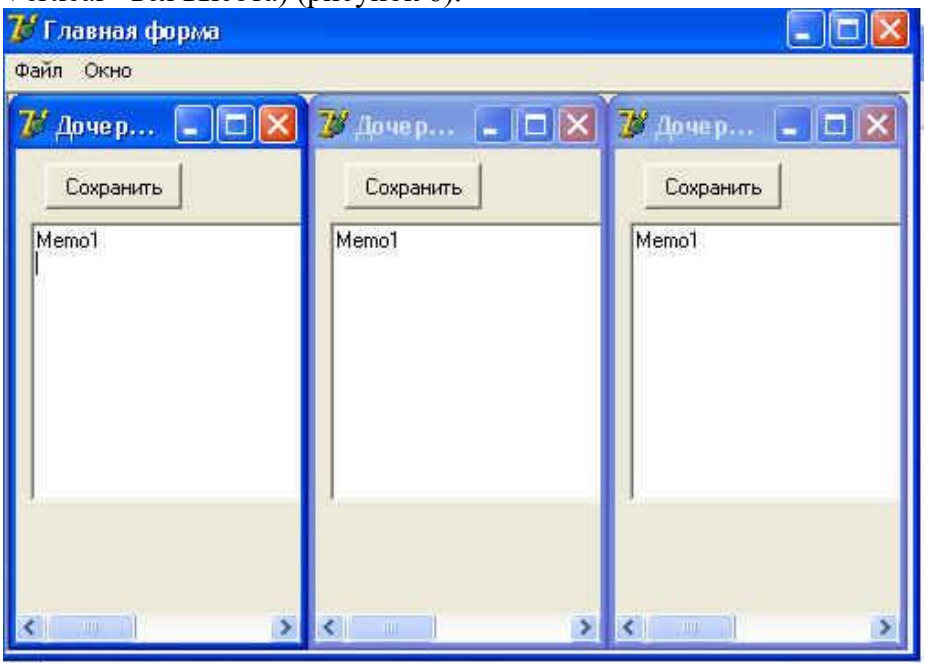


Рисунок 6

Для того чтобы дочерние формы располагались мозаикой и располагались во всю длину клиентской области пункт меню главной формы "Мозаика" будет таким:

```
procedure TMainForm.N5Click(Sender: TObject);
begin
    Tile;
    TileMode := tbVertical;
end;
```

Пример создания динамической библиотеки Windows

Чтобы создать новую библиотеку DLL в Delphi, выбирается команда меню File > New > Other. В панели Items Categories окна New Items выбирается узел Delphi Projects, после чего необходимо выбрать элемент Dynamic-link Library в правой панели окна (рисунок 7).

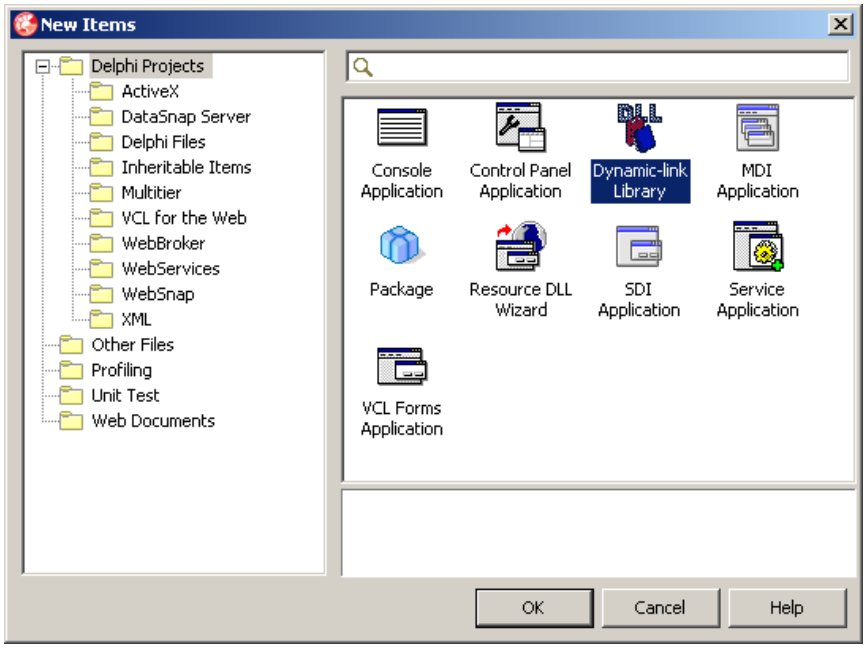


Рисунок 7

Мастер DLL Wizard создает главный файл исходного кода библиотеки DLL, который выглядит практически так же, как и исходный код, сгенерированный для обычного приложения. Единственное отличие состоит в том. что этот файл начинается с зарезервированного слова library, а не program.

Отчет должен содержать:

1. Название, цель и задание лабораторной работы;
2. Алгоритм работы программы;
3. Результаты работы программы;
4. Ответы на контрольные вопросы.

Контрольные вопросы:

1. Дать определение понятию «Жизненный цикл»
2. Каким международным стандартом регламентируется состав процессов жизненного цикла
3. Перечислить этапы жизненного цикла программного продукта
4. Перечислить модели жизненного цикла программного обеспечения. Описать каждый из них.
5. Что такое «спецификация» и что она описывает?
6. Перечислить простые и составные управляющие структуры

Лабораторная работа № 2, 3
Тема: «Разработка приложения для работы с базами данных в Delphi. ADO-технологии»

Цель лабораторной работы:

ознакомится с компонентами ADOConnection, ADOTable и DataSource для связи БД с Delphi;

развитие познавательного интереса;

изучение новых алгоритмов решения задач;

формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Количество часов: 4 часа.

Постановка задачи

Создать клиентское приложение для работы с БД «Товар». Приложение имеет главную форму (окно), две рабочие формы (окна) и модуля данных. Главное окно содержит меню вызова таблиц БД (рисунок А). Выбор пункта меню вызывает соответствующее окно для работы с тремя таблицами. Таблицы можно редактировать и добавлять новыми записями.

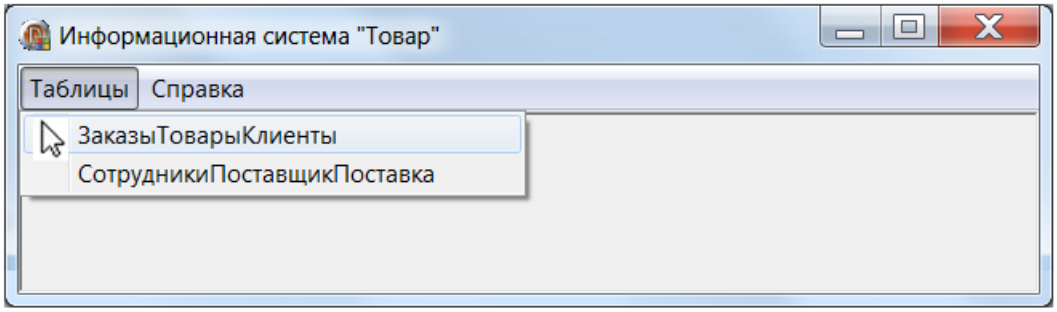


Рисунок А – Начальное окно главной формы приложения

В модуле данных размещаются компоненты связи приложения с БД.

Порядок выполнения работы:

1) создаем главную форму приложения. Размещаем на ней компонент Меню для выбора нужных таблиц;

2) связываем БД «Товар» с нашим приложением;

3) создаем две подчиненные формы. На первой будут таблицы «Заказы», «Товары» и «Клиенты». На второй – «Сотрудники», «Поставщик» и «Поставка».

Необходимые компоненты

Data Module – специальная форма, предназначенная исключительно для размещения на ней невизуальных компонент для доступа к данным.

ADOConnection – используется для подключения приложения к БД (закладка dbGO или в поздний версиях ADO).

ADOTable (таблица ADO)– связывается с конкретной таблицей БД (закладка dbGO или в поздний версиях ADO);

DataSource (источник данных) – компонент, используется как связка данных из таблиц, с отображающими и управляющими компонентами Delphi (закладка Data Access);

DBGrid – таблица, позволяющая вывести содержимое таблицы БД на пользовательскую форму (закладка Data Controls);

DBNavigator – кнопочная панель, которая обеспечивает перемещение указателя текущей записи таблицы, активизацию режима редактирования, добавление и удаление записей (закладка Data Controls).

Методика выполнения работы

Создание структуры базы данных «Товар»

Описание предметной области. Магазину компьютерной техники необходимо автоматизировать процессы продажи и поставки товаров от поставщиков. Для достижения этой цели постоянно требуется работать с информацией об имеющихся товарах, заказах, поставщиках и о поставляемой ими номенклатуре товаров, клиентах и сотрудниках. Подобные сведения содержатся в накладных, бланках заказов, чеках. С точки зрения пользователя БД должна обрабатывать эти документы. Более детальный анализ предметной области привел к структуре БД «Товар», изображенной на рисунке Б.

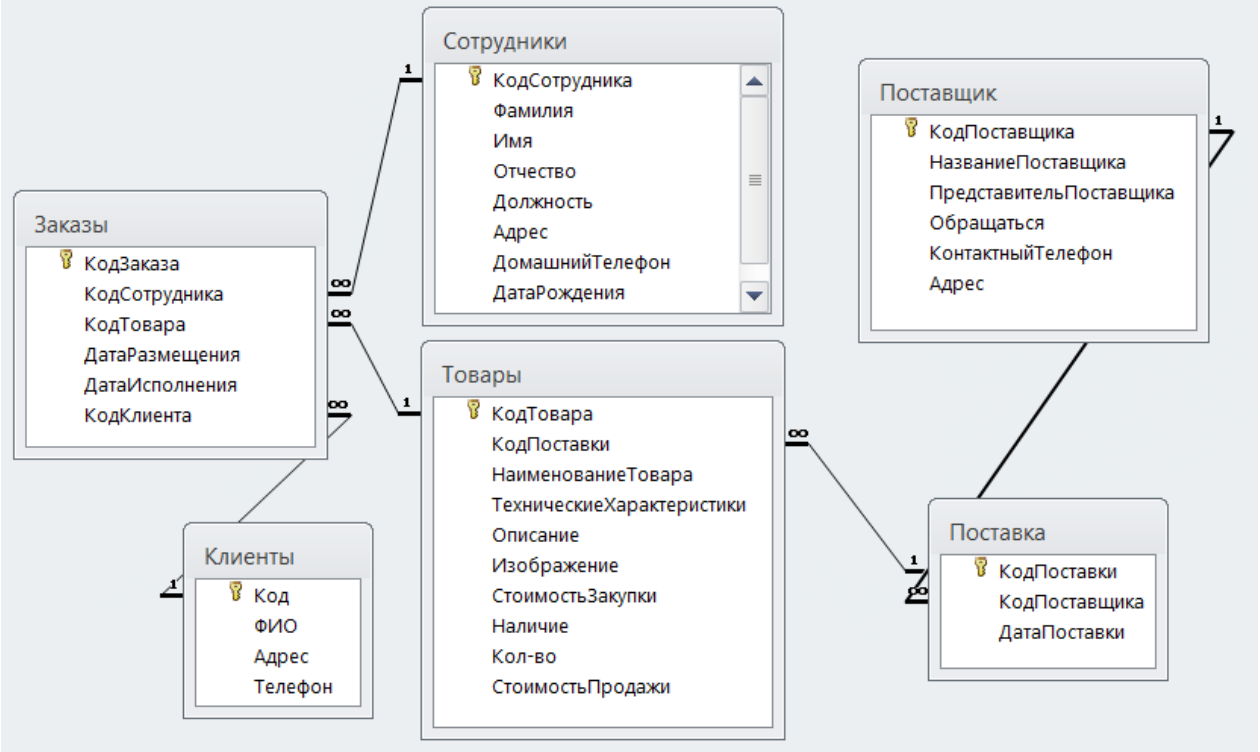


Рисунок Б – Структура БД «Товар»

Создадим БД Товар.accdb в СУБД Microsoft Access 2010.

Таблица «Заказы»

Имя Поля	Тип данных	Свойство поля
КодЗаказа	Числовой	Ключевое поле. Размер поля –Целое. Подпись – Номер п/п.
КодСотрудника	Числовой	Размер поля – Целое. Значение по умолчанию – 0. Обязательное поле – Да. Индексированное поле –Да (Допускаются совпадения).
КодТовара	Текстовый	Размер поля – 50. Подпись – Код товара. Индексированное поле –Да (Допускаются совпадения). Сжатие Юникод – Нет.
ДатаРазмещения	Дата/время	Краткий формат даты. Маска ввода: 00.00.0000;0;_. Подпись – Дата подачи заявки. Индексированное поле – Да (Допускаются совпадения).
ДатаИсполнения	Дата/время	Краткий формат даты. Маска ввода: 00.00.0000;0;_. Подпись – Дата доставки.
КодКлиента	Числовой	Размер поля – Целое. Значение по умолчанию – 0.

Таблица «Клиенты»

Имя Поля	Тип данных	Свойство поля
Код	Числовой	Ключевое поле. Размер поля –Целое. Значение по умолчанию – 0.
ФИО	Текстовый	Размер поля – 50.
Адрес	Текстовый	Размер поля – 50.
Телефон	Текстовый	Размер поля – 15. Сжатие Юникод – Нет.

Таблица «Поставка»

Имя Поля	Тип данных	Свойство поля
КодПоставки	Числовой	Ключевое поле. Размер поля –Целое. Значение по умолчанию – 0.
КодПоставщика	Числовой	Ключевое поле. Размер поля –Целое. Значение по умолчанию – 0.
ДатаПоставки	Дата/время	Краткий формат даты. Маска ввода: 00.00.0000;0;_.

Таблица «Поставщик»

Имя Поля	Тип данных	Свойство поля
КодПоставщика	Числовой	Ключевое поле. Размер поля –Целое. Значение по умолчанию – 0.
НазваниеПоставщика	Текстовый	Размер поля – 50.
ПредставительПоставщика	Текстовый	Размер поля – 50.
Обращаться	Текстовый	Размер поля – 50. Подпись – ФИО.
КонтактныйТелефон	Текстовый	Размер поля – 15. Сжатие Юникод – Нет.
Адрес	Текстовый	Размер поля – 50.

Таблица «Сотрудники»

Имя Поля	Тип данных	Свойство поля
КодСотрудника	Числовой	Ключевое поле. Размер поля –Целое. Подпись – Код сотрудника.
Фамилия	Текстовый	Размер поля – 50. Пустые строки – Нет. Индексированное поле – Да (Допускаются совпадения). Сжатие Юникод – Нет.
Имя	Текстовый	Размер поля – 50. Пустые строки – Нет. Сжатие Юникод – Нет.
Отчество	Текстовый	Размер поля – 30. Пустые строки – Нет. Сжатие Юникод – Нет.
Должность	Текстовый	Размер поля – 50. Пустые строки – Нет. Сжатие Юникод – Нет.
Адрес	Текстовый	Пустые строки – Нет. Сжатие Юникод – Нет.
ДомашнийТелефон	Текстовый	Размер поля – 30. Подпись – Домашний телефон. Пустые строки – Нет. Сжатие Юникод – Нет.
ДатаРождения	Дата/время	Краткий формат даты. Маска ввода: 99.99.00;0. Подпись – Дата рождения.
Заметки	Поле МЕМО	Пустые строки – Нет. Сжатие Юникод – Нет.

Таблица «Товары»

Имя Поля	Тип данных	Свойство поля
КодТовара	Текстовый	Ключевое поле. Размер поля – 50.
КодПоставки	Числовой	Размер поля – Целое. Значение по умолчанию – 0. Индексированное поле – Да (Допускаются совпадения).
НаименованиеТовара	Текстовый	Размер поля – 50. Подпись – Наименование товара. Индексированное поле – Да (Совпадения не допускаются).
ТехническиеХарактеристики	Текстовый	
Описание	Поле MEMO	
Изображение	Текстовый	Размер поля – 50.
СтоимостьЗакупки	Денежный	
Наличие	Логический	Формат поля – Да/Нет.
Кол-во	Числовой	Размер поля – Целое. Значение по умолчанию – 0.
СтоимостьПродажи	Денежный	

1 Создание нового приложения

1.1 Запускаем Delphi. Выбираем File -> New-> VCL Forms Application. Сохраняем модуль Unit1 и проект приложения под именем Tovar.dpr в своей папке. В эту же папку помещаем файл БД Товар.accdb.

1.2 Сделаем нашу форму главной MDI формой (многодокументный интерфейс). Для этого в инспекторе объектов в свойствах Form1 свойство FormStyle установим в fsMDIForm. Форма стала стартовой.

1.3 В свойстве Caption главной формы пишем «Информационная система Товар».

1.4 Добавим на форму компонент TMainMenu из вкладки Standard. Щелкнем дважды мышкой на компоненте MainMenu1 и увидим окно для создания меню.

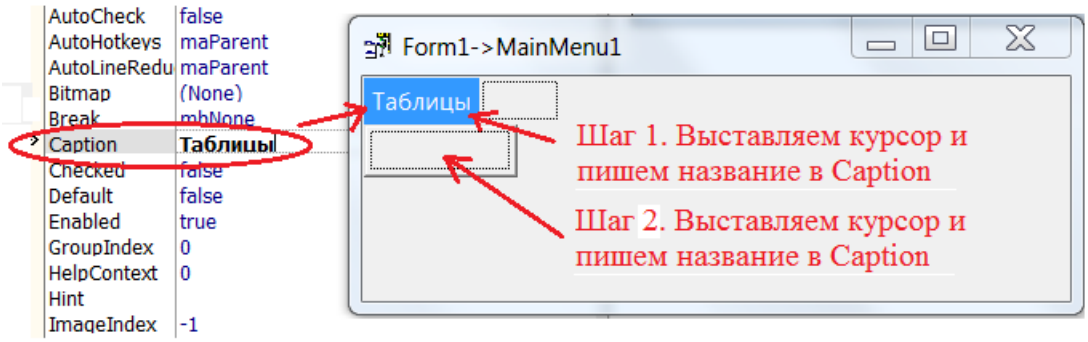


Рисунок 2 – Настройка компоненты MainMenu1

Выставляем курсор на «прямоугольник» и в появившемся инспекторе свойств пишем Таблицы (рисунок 2), нажимаем Enter. Перемещаем курсор на «прямоугольник» ниже «Таблицы». В появившемся инспекторе свойств пишем «ЗаказыТоварыКлиенты» и нажимаем Enter, а потом – «СотрудникиПоставщикПоставка». Перемещаем курсор на «прямоугольник» правее «Таблицы» и пишем «Справка».

1.4 Компилируем и запускаем программу (F9).

2 Привязка БД «Товар.accdb» к приложению

2.1 Командой File / New / Data Module (в категории Delphi Files) добавьте в проект новый модуль данных. Дайте ему имя dm (для краткости) и сохраните его как DatMod.pas в папку приложения.

2.2 Положите на модуль dm компонент ADOConnection (в закладке dbGo). Компонент ADOConnection1 позволит привязать БД Товар.accdb к приложению.

Настраивается компонент ADOConnection. Дважды щелкните на этом компоненте. В открывшемся окне установите источник связи, выбрав Use Connection String (рисунок 3).

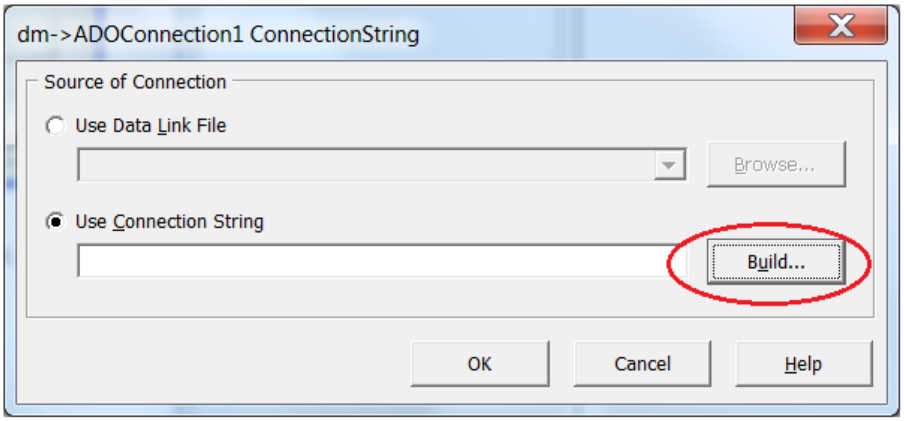


Рисунок 3 – Окно модуля данных

Нажмите кнопку Build..., перейдите на закладку «Поставщик данных» и выберите Microsoft Office 12.0 Access Database Engine Provider (рисунок 4). Нажмите кнопку Далее.

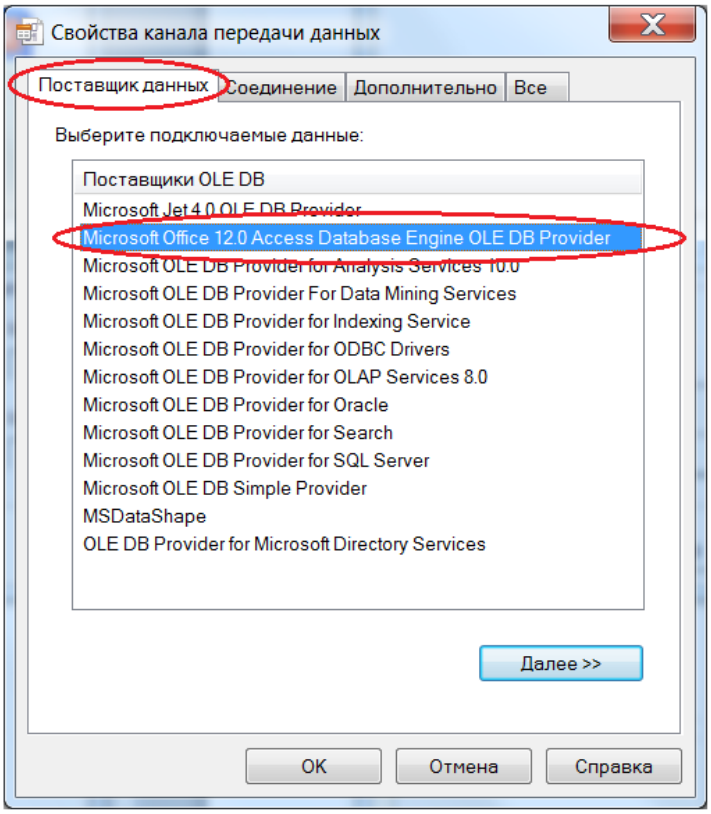


Рисунок 4 – Окно свойств связи с данными

Перейдите за закладку Соединение, введите в строку Источник данных имя нашей БД, как показано на рисунке 5. Если для доступа к БД необходим пароль и идентификатор пользователя, то их надо указать (по умолчанию к БД, созданной в MS Access, доступ есть

у пользователя Admin, но пароль не нужен). После этого нажмите кнопку Проверить подключение, чтобы проверить подключается ли БД Товар.accdb. Если будет дан ответ «Проверка подключения выполнена», то щелкая по кнопке ОК закрыть все окна.

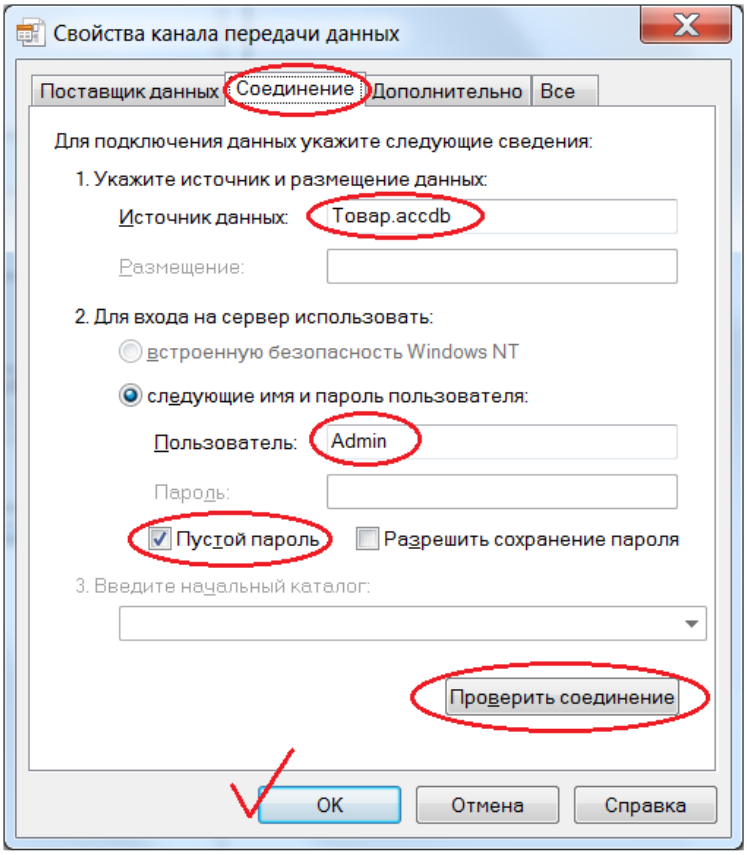


Рисунок 5 – Подключение БД Товар.accdb к приложению

2.3 Установим свойства ADOConnection в Инспекторе объектов:

- LoginPrompt = False – запрос имени и пароля пользователя отключен.
- Mode = cmShareDenyNone.
- Connected = True – подключение к базе активировано.

3 Связывание таблиц БД с приложением

3.1 Положите на модуль данных dm компоненты ADOTable (закладка dbGo) и DataSource (закладка DataAccess).

3.2 Для удобства работы дайте им имена TabZakaz и dsZakaz, соответственно. Для этого в инспекторе объектов ADOTable1 в свойство Name записываем TabZakaz, а в инспекторе объектов DataSource1 в свойство Name записываем dsZakaz.

3.3 Подключим таблицу TabZakaz к компоненту ADOConnection1 и к одноименной таблице «Заказы» нашей БД (рисунок 6).

В инспекторе объектов таблицы TabZakaz устанавливаем свойства:

- Connection = ADOConnection1.
- TableName = Заказы.
- Active = True (! это свойство устанавливать в последнюю очередь).

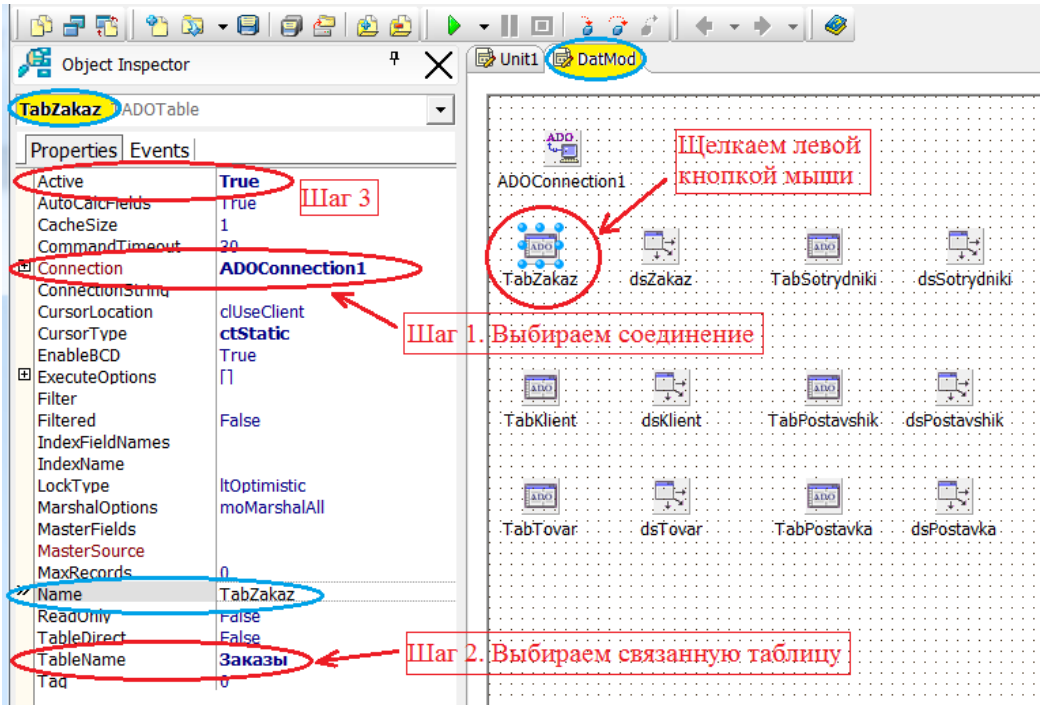


Рисунок 6 – Установка свойств компонента TabZakaz (ADOTable1)

3.4 В инспекторе объектов компонента dsZakaz (бывшая DataSource1) установите свойство DataSet = TabZakaz (рисунок 7).

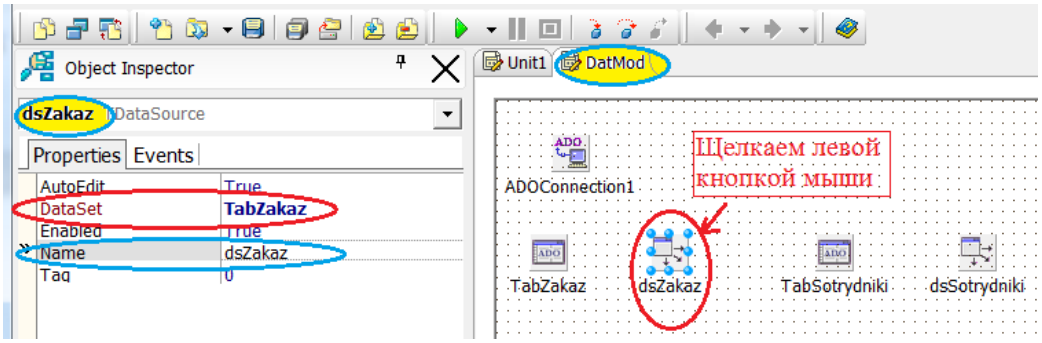


Рисунок 7 – Установка свойств компонента dsZakaz (DataSource1)

После определения свойств исчезнут красные знаки вопроса слева от компонентов в окне Stucture, что говорит о готовности компонента к работе.

3.5 Двойным щелчком на компоненте TabZakaz откроем Окно редактора полей, щелкнем в окне правой кнопкой мыши и в контекстном меню выберем команду Add all fields. Окно редактора заполнится списком всех полей таблицы Заказы. Если щелкнуть на любом поле в окне редактора полей, то в окне инспектора объектов станут доступными свойства объекта-поля.

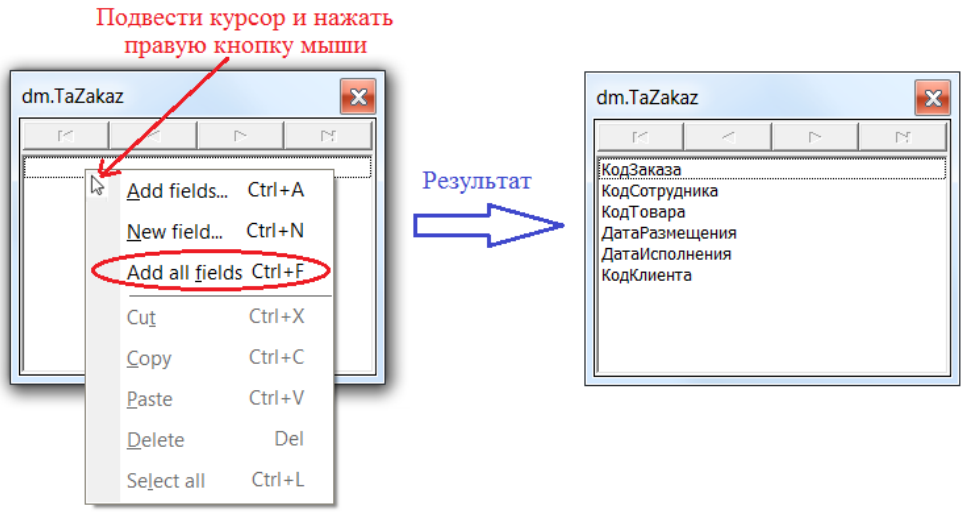


Рисунок 8 – Добавление полей в компонент TabZakaz

3.6 Поступая аналогичным образом, установите в модуль dm соответствующие компоненты для таблиц «Товары» (компоненты TabTovar, dsTovar), «Клиенты» (компоненты TabKlient, dsKlient), «Сотрудники» (компоненты TabSotrydniki, dsSotrydniki), «Поставщик» (компоненты TabPostavshik, dsPostavshik), «Поставка» (компоненты TabPostavka, dsPostavka).

3.7 Сохраняем проект – File -> Save All.

4 Разработка рабочих форм для работы с таблицами БД

Постановка задачи. При выборе пункта меню Таблицы (рисунок 2), должно появиться новое окно, в котором отображается тройка выбранных таблиц. Также необходим механизм навигации и редактирования таблиц.

4.1 Создаем новую форму File->New->Form-Delphi. Сохраняем модуль с именем Unit2 (предлагается по умолчанию).

4.2 В инспекторе объектов для Unit2 устанавливаем свойство:

Caption = «Заказы» «Товары» «Клиенты»;

Visible = False // При запуске приложения – окно невидимо.

4.3 Знакомим Form1 и Form2.

4.3.1 Делаем активной Form1 (щелкнем по Unit1, а затем по форме). Далее File->Use Unit.. (Использовать модуль), где в контекстном меню выбираем Unit2 и жмем OK (рисунок 9). Теперь форма Form1 знает о существовании Form2, в коде Unit1 в разделе implementation прописалась строка `uses Unit1;` (проверьте это нажав F12).

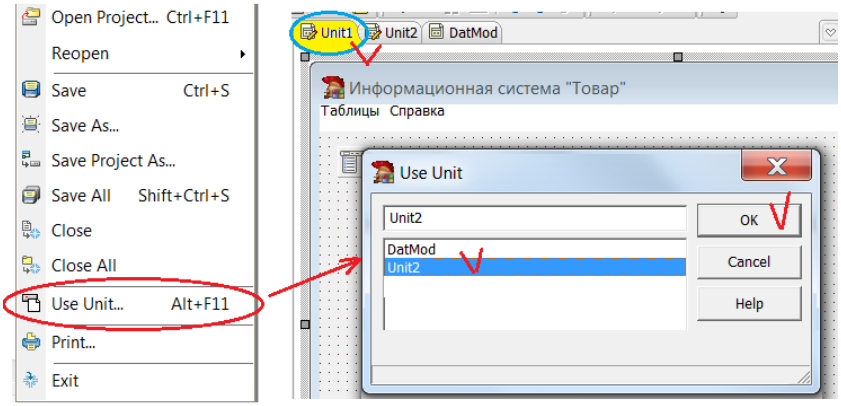


Рисунок 9 – Знакомство форм Form1 с Form2

4.3.2 Делаем активной Form2. Далее, File->Use Unit.. и в контекстном меню выбираем Unit1 и жмем OK. Форма Form2 знает о Form1 (проверьте).

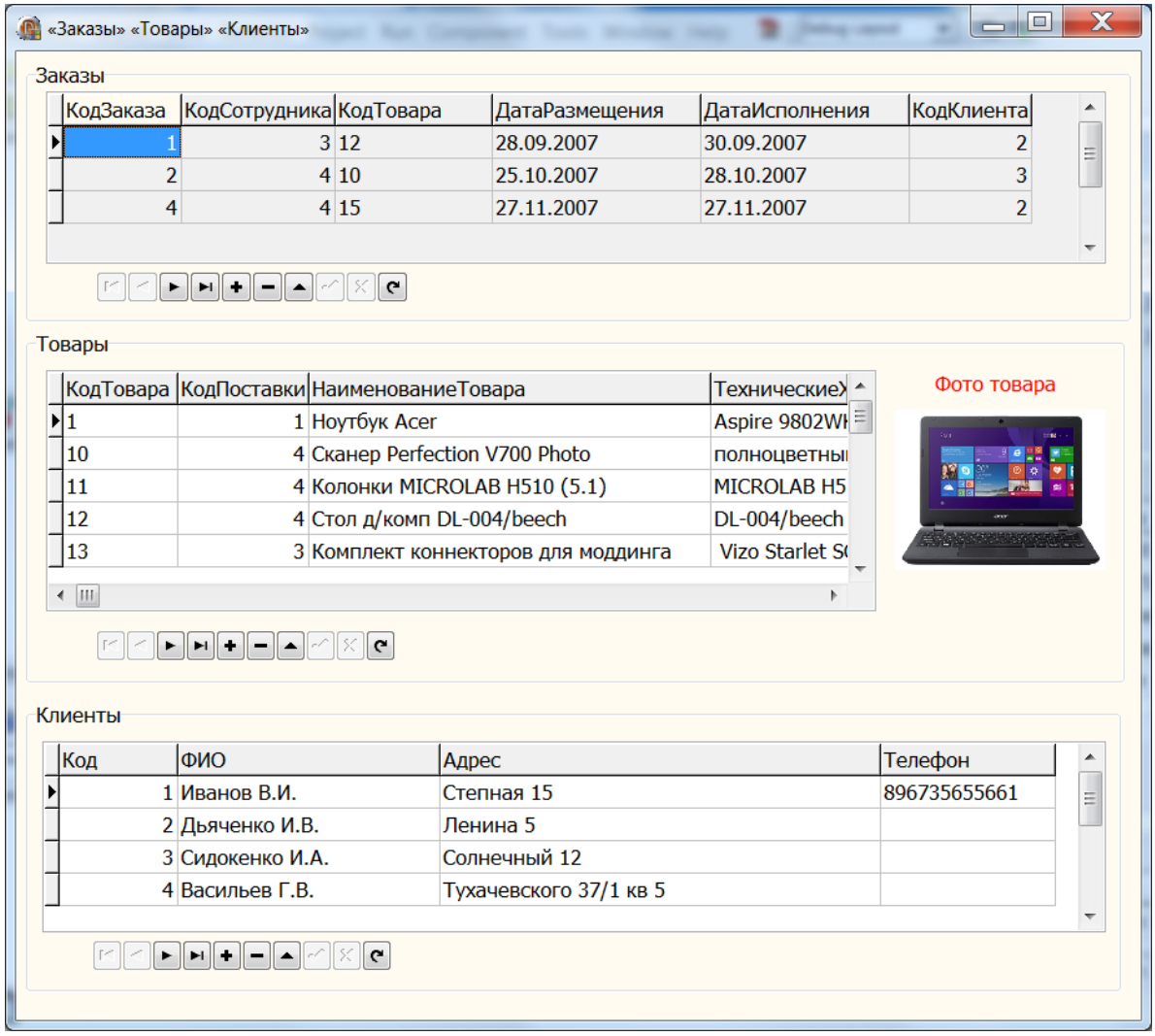


Рисунок 10 – Вид окна для пункта ЗаказыТоварыКлиенты

4.4 Переходим на Form1. Щелкаем дважды мышкой по компоненту MainMenu1, а затем дважды по пункту ЗаказыТоварыКлиенты. Пишем код.

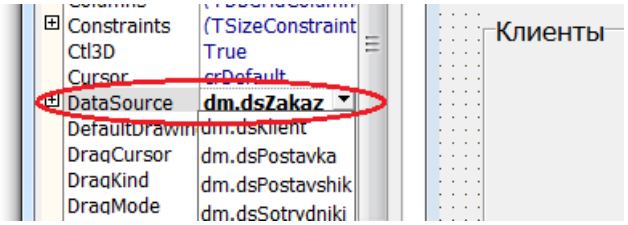
```
procedure TForm1.N2Click(Sender: TObject);
begin
    Form2.Show;    // Метод Show выводит на экран окно Form2
end;
```

4.5 Оформляем Form2. Окончательный вид окна показан на рисунке 10.

4.5.1 Знакомим Form1 и Form2 с модулем DatMod (см. 4.3.1 и 4.3.2).

4.5.1 Положим на Form2 три компонента GroupBox (закладка Standard) и используя свойства Caption дайте им заголовки «Заказы» (для GroupBox1), «Товары» (для GroupBox2) и «Клиенты» (для GroupBox3).

4.5.2 Положите в GroupBox1 сетку DbGrid1 (закладка Data Controls). Для DbGrid1 свяжите свойство DataSource с источником данных dsZakaz. В Окне редактора полей (двойной клик по DbGrid1) выберите команду Add all fields.



Проделайте ту же операцию для «Товары» и «Клиенты».

4.6 Сохраняем (File -> Save All) и компилируем (F9) проект.

4.7 Положите в GroupBox1 под сетку DbGrid1 компонент DBNavigator (закладка Data Controls). В инспекторе DBNavigator свойство DataSource = dsZakaz.

Проделайте ту же операцию для «Товары» и «Клиенты».

4.8 Вывод картинки. В бокс GroupBox2 (Товары) положите компонент Image (вкладка Additional) для визуализации изображений товаров, а также компонент Label (вкладка Standart) для поясняющего текста.

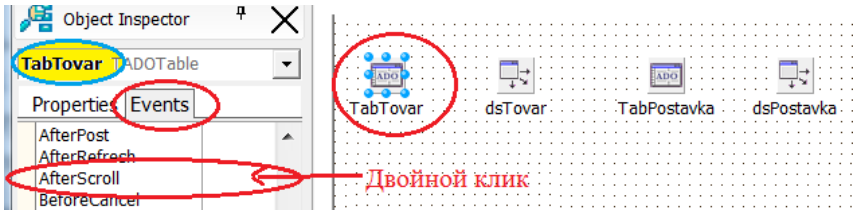
4.9 Пишем код отображения фото. Переходим в модуль dm (DatMod).

Заходим в Код (F12). Чтобы приложение понимало формат файлов *.jpg, добавьте в список USES (в блоке implementation) имя модуля JPEG, т.е.

```
implementation
uses Unit2, jpeg;
```

. Для формата BMP – ничего делать не надо.

Возвращаемся к форме dm (F12). Выделяем компонент TabTovar, в Инспекторе переходим во вкладку Events и выбираем событие AfterScroll (двойной клик).

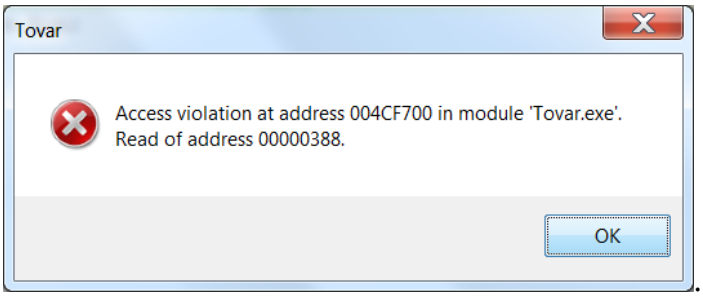


В окне кода для модуля dm (DatMod) пишем код

```
// Событие возникает при переходе к другой записи таблицы TabTovar (следующей,
// предыдущей, первой или последней), т.е. при скроллинге
procedure Tdm.TabTovarAfterScroll(DataSet: TDataSet);
var
  PicPath: string; // Для нахождения пути к фото
  PicName: string; // Для файла картинки
begin
 PicPath:=ExtractFilePath(ParamStr(0))+'foto\'; // Путь к папке foto
 if TabTovar.FieldValues['Изображение']<>'' then  // Поле 'Изображение' не пусто
  PicName:=TabTovar.FieldByName('Изображение').AsString  //Из поля 'Изображение'
  else  PicName:='fotonet.bmp';   // Если поле 'Изображение' пусто
 try  // Если не будет ошибки при загрузке фото (имя задано не правильно)
   Form2.Image1.Picture.LoadFromFile(PicPath+PicName); // Загружаем картинку
   Form2.Image1.Visible:=True;   // делаем Image1 видимой
 except   // Если ошибка при загрузке фото
   Form2.Image1.Picture.LoadFromFile(PicPath+'fotonet.bmp'); //Выводим Нет фото
   Form2.Image1.Visible:=True;
 end;  // Конец обработки исключительной ситуации try...except
end;   // Конец функции скроллирования таблицы
```

Краткое пояснение. Событие AfterScrool происходит всякий раз после перехода к другой записи таблицы. Процедура Tdm.TabTovarAfterScroll анализирует содержимое поля 'Изображение' таблицы TabTovar и, если оно не пустое, выводит картинку из папки foto. В переменной PicPath сохранится полный путь к папке с приложением, а в переменной PicName – содержимое ячейки поля 'Изображение' в записи с курсором. Функция ExtractFilePath(ParamStr(0)) возвращает путь к папке, где запущено приложение, возвращает тип string. Методы TabTovar.FieldValues['Изображение'] и TabTovar.FieldByName('Изображение').AsString – два способа извлечь содержимое из ячейки текущей записи поля 'Изображение'. Метод LoadFromFile(PicPath+PicName) загружает картинку из файла.

4.10 Сохраняем (File -> Save All) и компилируем (F9) проект. Если при этом может выскочить сообщение о том, что в приложении есть обращение на несуществующий объект:



Ничего страшного. Просто мы сначала создали модуль dm (DatMod), а потом Form2 (Unit2), и это прописалось в модуле Tovar.pas. Поэтому при запуске приложения создается модуль dm, при этом процедура Tdm.TabTovarAfterScroll ссылается на форму Form2, которой еще нет. Чтобы обойти эту ситуацию откройте файл Tovar.dpr и поменяйте местами соответствующие строчки:

```
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(Tdm, dm);
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

Замена

```
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(Tdm, dm);
  Application.Run;
end.
```

4.11 Сохраняем (File -> Save All) и компилируем (F9) проект.

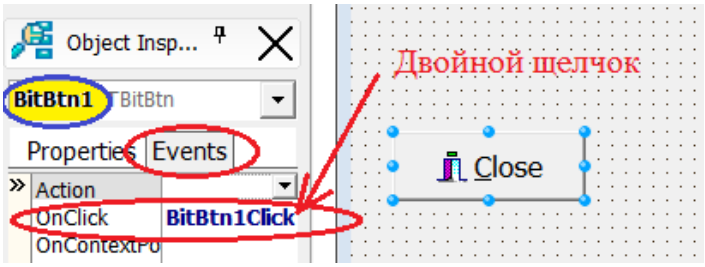
Задание. Создайте форму Form3 и проведите разработку окна для работы с пунктом меню «СотрудникиПоставщикПоставка». Не забудьте в выбранных таблицах (модуль dm) установить свойство Active = True.

5 Закрытие главной формы

Чтобы внесенные пользователем изменения в таблицы были сохранены в БД «Товар» в Access, таблицы в Delphi по окончании работы необходимо закрыть. Для этого используем метод Close. Для закрытия главной формы (и приложения в целом) используем метод Free – закрывает форму и освобождает память. Отметим, что если главную форму закрыть, то рабочие формы закрываются автоматически.

5.1 Переходим на Form1. Устанавливаем кнопку BitBtn1 (вкладка Additional). В инспекторе BitBtn1 устанавливаем свойство Kind = bkClose.

5.2 В инспекторе BitBtn1 заходим в Events и вызываем обработчик события OnClick (Двойным щелчком).



В открывшемся окне Code пишем команду на закрытие Form1.

```
// Нажатие на кнопку Close (BitBtn1)
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 Form1.Free; // Закрыть Form1. Освободить память.
end;
```

5.3 В инспекторе Form1 заходим в Events и вызываем обработчик события OnCloseQuery (двойным щелчком).

В открывшемся окне Code пишем код обработчика события OnCloseQuery (запрос на закрытие).

```
// Обработчик события закрытия TForm1. Включается при закрытии формы
// любым из способов (кнопка Close, команда системного меню или Alt+F4 )
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
CanClose := MessageDlg('Закрыть приложение?', mtConfirmation,
        [mbYes, mbNo], 0) = mrYes;  // Подтверждение закрытия приложения
if CanClose=True then  begin
 dm.TabZakaz.Close; // Закрыть таблицу TabZakaz
 dm.TabKlient.Close;  // Закрыть таблицу TabKlient
 dm.TabTovar.Close;
 dm.TabSotrydniki.Close;    // Закрытие таблиц
 dm.TabPostavshik.Close;
 dm.TabPostavka.Close;
 Form2.Free;    // Закрыть Form2. Освободить память.
 Form3.Free;    // Закрыть Form3. Освободить память.
 end;
end;
```

Отчет должен содержать:

1. Анализ предметной области.
2. Схему данных.
3. Алгоритм работы программы.
4. Результаты работы программного продукта.

Контрольные вопросы:

1. Дайте определение понятиям «Сущность», «Атрибут сущности», «Внешний ключ», «Первичный ключ».
2. Виды связей между сущностями.
3. Какие функции выполняет администратор БД и пользователь БД?
4. Что такое схема данных

Лабораторная работа № 4

Тема: «Разработка приложения для работы с базами данных в Delphi. Создание запросов»

Цель лабораторной работы:

- научиться создавать SQL-запросы к базам данных в среде программирования Delphi, используя компонент ADOQuery;
- развитие познавательного интереса;
- изучение новых алгоритмов решения задач;
- формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Количество часов: 2 часа.

Постановка задачи

Организуйте выбор информации из БД с помощью заданного и клиентского SQL-запросов, по аналогии с разработкой Form5 проекта «Информационная система Товар». Создайте пять SQL-запросов для своей базы данных.

Окончательный вид окна просмотра SQL-запроса, приведен на рисунке 1. Окно содержит три именованные рабочие области.

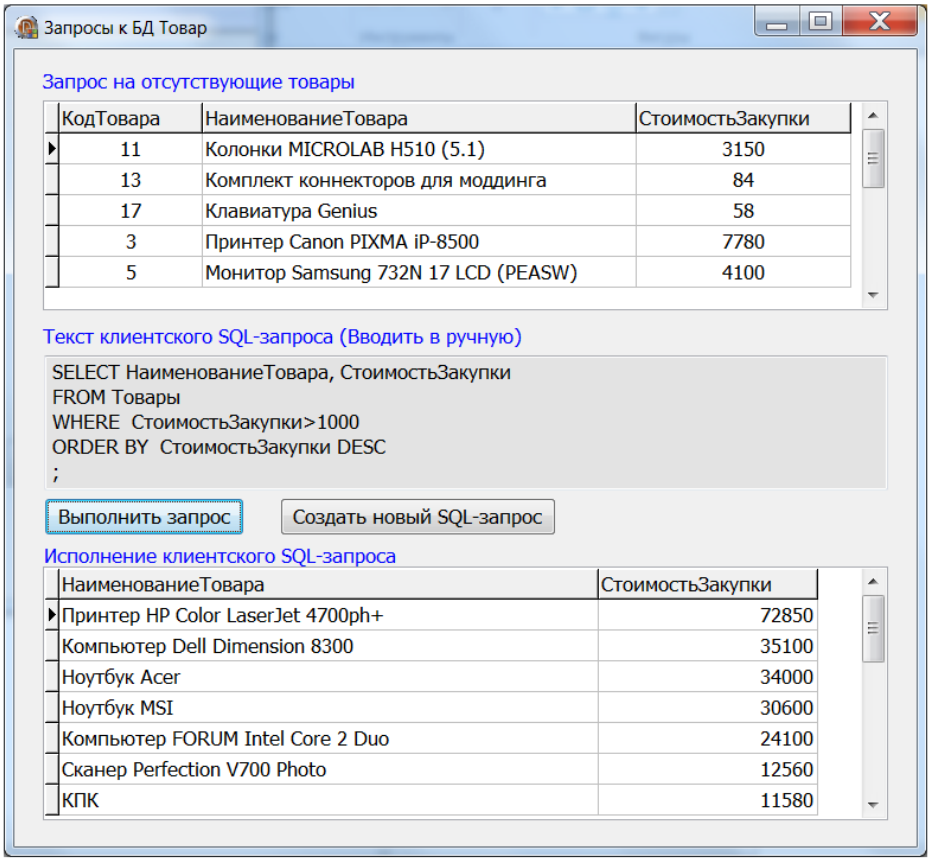


Рисунок 1

В первой области (сетка DBGrid1) выводится готовый SQL-запрос на отсутствующие товары. Код этого запроса прописан в программе.

Во вторую область (компонент Memo) в ручном режиме вводится клиентский SQL-запрос. Нажимается кнопка «Выполнить запрос», после чего в третьей области (сетка DBGrid2) появляются записи удовлетворяющие условию введенного клиентского запроса.

Методика выполнения работы

1 Создание нового окна «Запросы к БД»

1.1 Создаем Form5 (File -> New-> Form-Delphi), сохраняем как Unit5.

1.2 Устанавливаем свойство Form5.Caption = Запросы к БД.

1.3 Знакомим Form5 с Form1 и DatMod (с помощью File->Use Unit..).

1.4 Переходим на Form1. В меню приложения добавляем новую закладку Запросы. Для этого дважды кликаем на компонент MainMenu1 и в окне создания меню выбираем нужные места (выставляя курсор) и в соответствующие свойстве Caption пишем заголовки пунктов (см. рисунок 2).

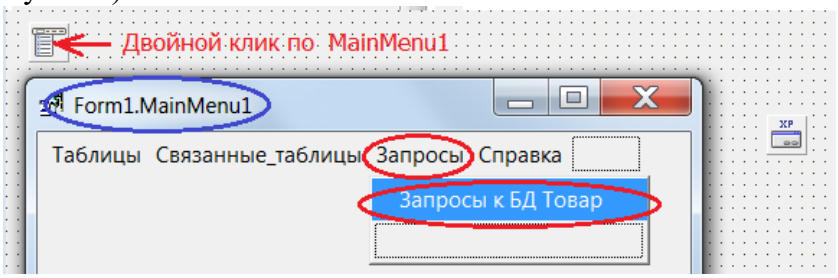


Рисунок 2 – Настройка компонента MainMenu1

1.5 Дважды кликаем по пункту меню Запросы к БД Товар из вкладки Запросы (см. рисунок 2). В обработчике событий пишем код:

```
procedure TForm1.N7Click(Sender: TObject);
begin
  Form5.Show;     // Метод Show выводит на экран окно Form5
end;
```

1.6 Переходим на Form5. Положите на Form5 компонент Memo (закладка Standard), две кнопки Button (закладка Standard) с надписями Выполнить запрос и Создать новый SQL-запрос, две сетки DbGrid (закладка Data Controls) и три метки Label (закладка Standard) с соответствующими надписями синего цвета. Расположите компоненты как на рисунке 1.

1.7 Сохраняем проект (File -> Save All) и компилируем (F9).

2 Создание запроса «Отсутствующие товары»

2.1 На модуль dm положите компонент ADOQuery ((закладка dbGo) и источник DataSource (закладка DataAccess), которому дайте имя dsQuery1.

2.2 Свяжите ADOQuery1 с ADOConnection1, как показано на рисунке 3.

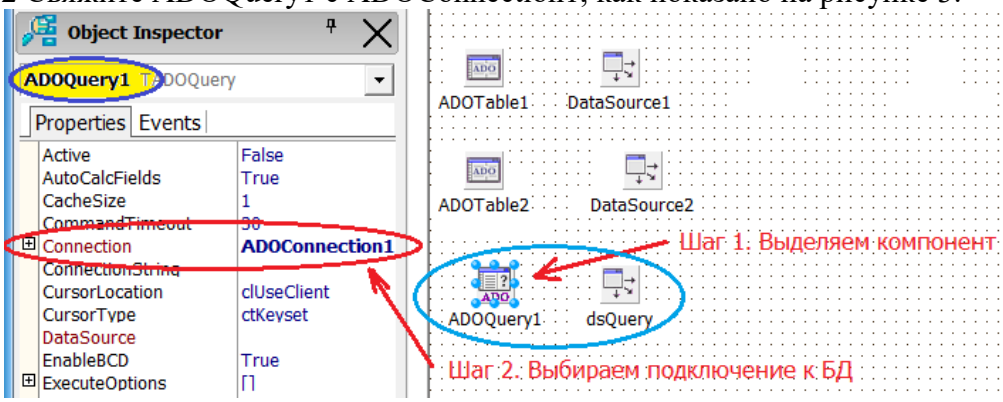
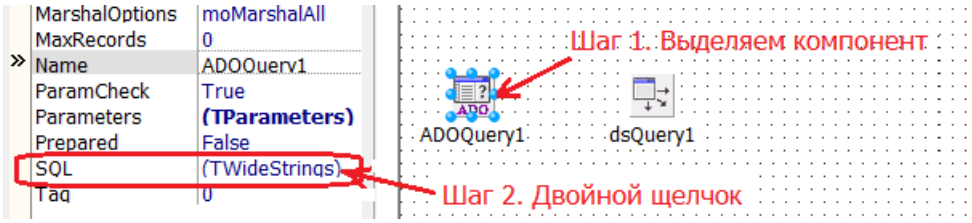


Рисунок 3 – Установка свойств ADOQuery1

2.3 В инспекторе dsQuery1 установите свойство DataSet = ADOQuery1.

2.4 Переходим на Form5 и выделяем DbGrid1. В инспекторе DbGrid1 установите свойство DataSource = dsQuery1.

2.5 Переходим на dm. Выделяем компонент ADOQuery1 и кликаем в свойстве SQL по строке TWideStrings.



Появляется окно редактора SQL-запросов. Записываем туда код для отсутствующего товара (см. рисунок 4). Нажимаем OK.

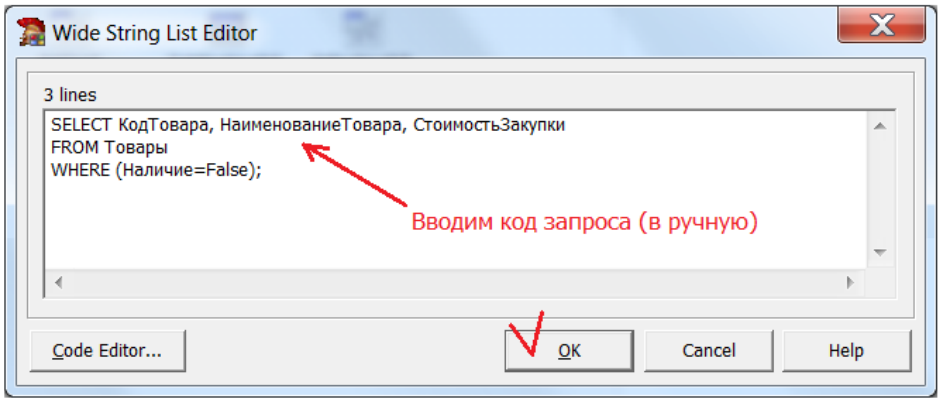


Рисунок 4 – Окно редактора SQL-запросов с запросом

2.6 В Инспекторе ADOQuery1 устанавливаем свойство: Active = True. Это свойство устанавливается в последнюю очередь, в противном случае, возможно возникновение ошибки «ADOQuery1: Missing SQL property», если свойство ADOQuery1.SQL пусто (рисунок 5).

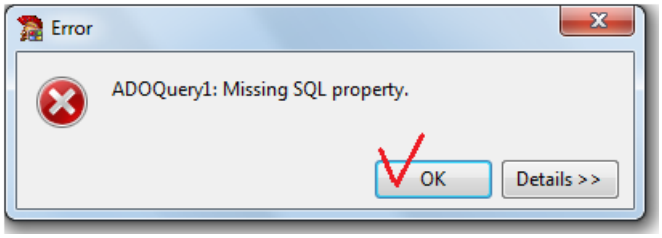


Рисунок 5 – Ошибка отсутствия запроса в свойстве

2.7 Сохраняем проект (File -> Save All) и компилируем (F9).

3 Создание клиентского запроса

3.1 На модуль dm положите компонент ADOQuery ((закладка dbGo) и источник DataSource (закладка DataAccess), которому дайте имя dsQuery2.

3.2 Свяжите ADOQuery2 с ADOConnection1, как показано на рисунке 6.

3.3 В Инспекторе dsQuery2 установите свойство DataSet = ADOQuery2.

3.4 Переходим на Form5 и выделяем DbGrid2. В инспекторе DbGrid2 установите свойство DataSource = dsQuery2.

3.5 По щелчку на Button1 (Выполнить запрос) SQL-запрос из Memo1 следует передать в свойство ADOQuery2.SQL где он автоматически выполняется. Создаем обработчик этого события (OnClick):

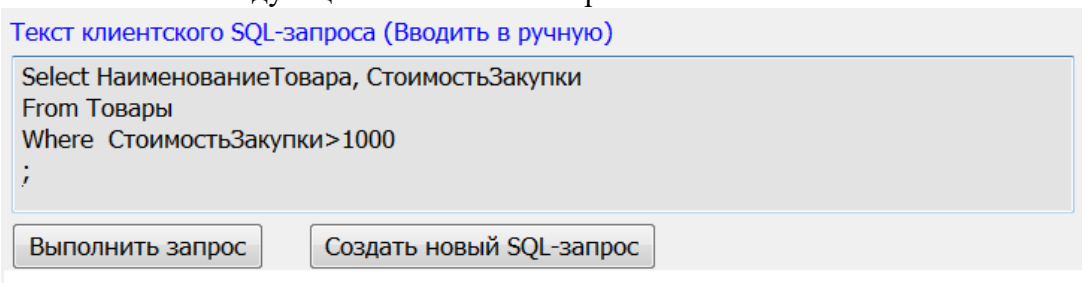
```
// Кнопка "Выполнить запрос"
procedure TForm5.Button1Click(Sender: TObject);
begin
 try  // Если при обработке не произойдет ошибки
   // закрыть ADOQuery2 перед его изменением
   dm.ADOQuery2.Active:= false;
   // Очистка полей свойства SQL компонента ADOQuery2
   dm.ADOQuery2.SQL.Clear;
   //Передать запрос из Memo1 в свойство ADOQuery2.SQL
   dm.ADOQuery2.SQL.Text:=Memo1.Lines.Text;
   // Активизировать ADOQuery2. SQL-запрос выполняется.
   dm.ADOQuery2.Active:= true;
 except // в случае ошибки (например, запрос не верен)
   on e:Exception do  // Универсальная обработка ошибок
    ShowMessage('Не удалось выполнить запрос.');
 end;
end;
```

Пояснение. В Memo1 вводится SQL запрос, после чего нажимается кнопка «Выполнить запрос». В случае какой-либо ошибки, например, выражение не соответствует правилам SQL-запроса, на экран выводится типичное сообщение типа «OK» с надписью «Не удалось выполнить запрос».

3.6 Сохраняем проект (File -> Save All) и компилируем (F9).

4 Наводим «красоту» на форме Form5

4.1 Переходим на Form5. Сделаем так, чтобы при первом открытии этой формы, в Memo1 был заложен следующий клиентский запрос:



В Инспекторе компонента Form5 вызываем событие OnCreate (или двойной щелчок по Form5). Отметим, что событие OnCreate генерируется только один раз при создании формы.

В открывшемся окне Code вводим код:

```
// Событие OnCreate генерируется только при создании формы
procedure TForm5.FormCreate(Sender: TObject);
begin
 Memo1.Lines.Clear; // Ощищаем строки Memo
 // Заполняем строк Memo1 начиная с первой
 Memo1.Lines.Add('Select НаименованиеТовара, СтоимостьЗакупки');
 Memo1.Lines.Add('From Товары');   //Во вторую строку пишем From
 Memo1.Lines.Add('Where  СтоимостьЗакупки>1000'); //Третья строка
 Memo1.Lines.Add(';'); // В третья строку ";"
end;
```

4.2 Сохраняем проект (File -> Save All) и компилируем (F9).

4.3 Переходим на Form5. В Инспекторе Button2 устанавливаем свойство: Caption = Создать новый SQL-запрос.

4.4 Создайте обработчик события OnClick кнопки Button2:

```
// Кнопка Создание нового SQL-запроса
procedure TForm5.Button2Click(Sender: TObject);
var
  p:TPoint; //Переменная для координат x,y курсора в Memo
begin
 // Создаем заготовку для нового SQL-запроса
 Memo1.Lines.Clear; // Ощищаем строки Memo
 Memo1.Lines.Add('Select    '); // В первую строку пишем Select
 Memo1.Lines.Add('From   ');   //Во вторую строку пишем From
 Memo1.Lines.Add(';'); // В третья строку ";"
 // Установка курсора в позицию 0,8
 p.X:=8;  p.Y:=0;  //  координаты для курсора в Memo
 Memo1.CaretPos:=p; // Выбор позиции 8,0
 Memo1.SetFocus;   // Установка курсора. Заготовка готова.
 // Очищаем сетку DBGrid2 от старого запроса
 dm.ADOQuery2.SQL.Text:='';
end;
```

4.5 На компоненте Memo1 устанавливаем свойство ScrollBars=ssBoth (добавляем горизонтальную и вертикальную полосы прокрутки).

4.6 Сохраняем проект (File -> Save All) и компилируем (F9).

Пример 1. Вывести записи всех полей таблицы Клиенты.

Текст клиентского SQL-запроса (Вводить в ручную)

```
Select *
From Клиенты
;
```

Исполнение клиентского SQL-запроса

Код	ФИО	Адрес	Телефон
1	Иванов В.И.	Степная 15	896735655661
2	Дьяченко И.В.	Ленина 5	896611111111
3	Сидокенко И.А.	Солнечный 12	896522222222
4	Васильев Г.В.	Тухачевского 37/1 кв 5	
5	Илларионов М.П.	Кулакова 295/1 кв 15	

Пояснение: вместо перечисления имен всех полей можно использовать символ «*» (звездочка).

Пример 2. Вывести записи всех полей таблицы Товары, упорядоченных по возрастанию стоимости закупки.

Текст клиентского SQL-запроса (Вводить в ручную)

```
Select * From Товары
Order by СтоимостьЗакупки ;
```

В случае требования упорядочивания по убыванию стоимости закупки:

Текст клиентского SQL-запроса (Вводить в ручную)

```
Select * From Товары
Order by СтоимостьЗакупки DESC;
```

Пример 3. Вывести записи полей КодТовара, НаименованиеТовара, СтоимостьЗакупки таблицы Товары, где стоимость закупки товаров колеблется от 1000 до 4000 у.е. Произвести упорядочение по возрастанию стоимости закупки.

Текст клиентского SQL-запроса (Вводить в ручную)

```
Select  КодТовара, НаименованиеТовара, СтоимостьЗакупки
From  Товары
Where  (СтоимостьЗакупки>1000) AND (СтоимостьЗакупки<4000)
Order by СтоимостьЗакупки DESC;
```

Выполнить запрос | Создать новый SQL-запрос

Исполнение клиентского SQL-запроса

КодТовара	НаименованиеТовара	СтоимостьЗакупки
11	Колонки MICROLAB H510 (5.1)	3150
9	Модем D-Link	1990
12	Стол д/комп DL-004/beech	1980
8	Модем Acorp	1450

Отчет должен содержать:

1. Название, цель и задание лабораторной работы;
2. Алгоритм работы программы;
3. Результаты работы программы;
4. Текст SQL запросов
5. Ответы на контрольные вопросы.

Контрольные вопросы:

1. К какой категории языковых средств относится оператор SELECT?
2. Какие специальные символы используются в операторе SELECT для выполнения групповых операций?
3. Зачем в операторе SELECT использовать обращение к полям по составному имени?

Лабораторная работа № 5

Тема: «Разработка приложения для работы с базами данных в Delphi. Создание связанных таблиц»

Цель лабораторной работы:

- ознакомится с возможностью связывания двух наборов данных (НД) с помощью свойств MasterSource/MasterFields, с возможностью расширения таблиц до НД за счет добавочных Lookup-полей, организация спрятанных (невидимых пользователю) полей.;
- развитие познавательного интереса;
- изучение новых алгоритмов решения задач;
- формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Количество часов: 2 часа.

Постановка задачи

Окончательный вид окна просмотра товаров и заказов, приведен на рисунке 1. На нем изображены два набора данных.

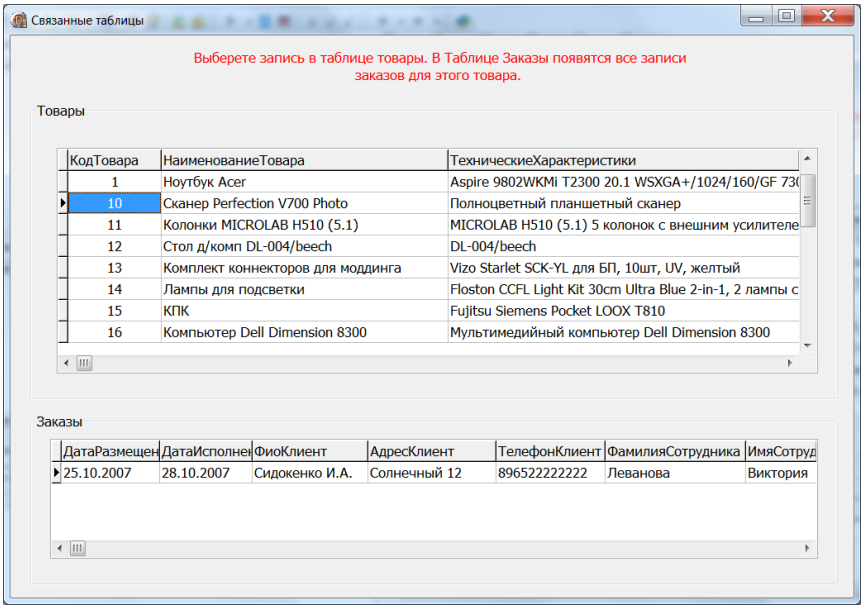


Рисунок 1 – Вид окна «Связанные таблицы»

Первый набор – это таблица «Товары», у которой невидимыми являются поля КодПоставки, Описание и Изображение. Второй набор данных «Заказы» – это подчиненная таблица «Заказы» с добавленными полями ФИО, Адрес и Телефон из таблицы «Клиенты» и полями Фамилия, Имя из таблицы «Сотрудники». В этом НД поля КодСотрудника и КодТовара из «Таблицы «Заказы» являются невидимыми. При выборе записи главной таблицы «Товары» во второй таблице (НД Заказы) выводятся только те данные, которые соответствуют выбранной записи из первой таблицы.

Усовершенствуйте проект своего варианта, введя в него просмотр связанных наборов данных, по аналогии с разработкой Form4 проекта «Информационная система Товар».

Методика выполнения работы

Создаем новую рабочую форму приложения (Form4). В меню главной формы (Form1) создаем закладку вызова Form4.

Создаем две новые таблицы ADOTable1 и ADOTable2, связываем их с таблицами «Товары» и «Заказы».

Производим связывание таблиц ADOTable1 и ADOTable2 с помощью свойств MasterSource/MasterFields и выводим их на экран формы Form4 с помощью сеток DBGrid1 и DBGrid2.

Расширяем таблицу ADOTable2 до набора данных «Заказы», путем формального (не физического) дополнения новых полей из таблиц «Клиенты» и «Сотрудники». Делаем невидимыми кодовые поля (их пользователю видеть не обязательно).

Запрещаем редактирование таблиц на Form4.

1 Создание нового окна «Связанные таблицы»

1.1 Создаем Form4 (File -> New-> Form-Delphi), сохраняем как Unit4.

1.2 Устанавливаем свойство Form4.Caption = Связанные таблицы.

1.3 Знакомим Form4 с Form1 и DatMod (с помощью File->Use Unit..).

1.4 Переходим на Form1. В меню приложения добавляем новую закладку Связанные таблицы. Для этого дважды кликаем на компонент MainMenu1 и в окне создания меню выбираем нужные места (выставляя курсор) и в соответствующие свойстве Caption пишем нужные заголовки пунктов (см. рисунок 2).

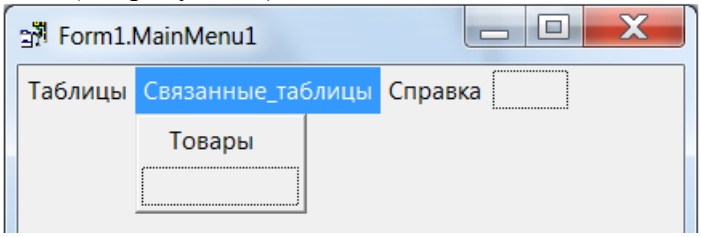


Рисунок 2 – Настройка компонента MainMenu1

1.5 Дважды кликаем по пункту меню Товары из вкладки Связанные_таблицы (см. рисунок 2). В обработчике событий пишем код:

```
procedure TForm1.N5Click(Sender: TObject);
begin
Form4.Show;     // Метод Show выводит на экран окно Form4
end;
```

1.6 Сохраняем проект (File -> Save All) и компилируем (F9).

2 Установка отношений между таблицами БД

Таблица TabZakaz (Заказы) является подчиненной по отношению к главной таблице TabTovar (Товары), т.к. в БД между соответствующими таблицами есть связь «один (от главной таблице) ко многим (к подчиненной таблице)» (см. рисунок 3).

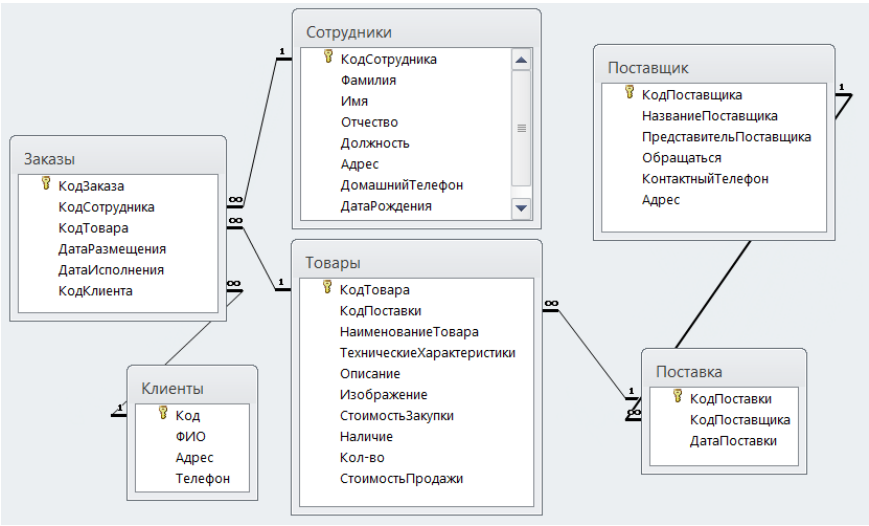


Рисунок 3 – Структура БД «Товар»

Сразу оговоримся, что если мы сделаем связывание таблиц TabZakaz и TabTovar с помощью свойств MasterSource/MasterFields, то это повлечёт изменение вывода этих таблиц на Form2. А это не входит в наши планы, т.к. Form2 предназначена для

редактирования таблиц целиком. Поэтому создаем новые таблицы ADOTable1 и ADOTable2, которые дублируют таблицы Товары и Заказы из БД.

2.1 Переходим на модуль данных dm. Положите на dm два компонента ADOTable (закладка dbGo) и два DataSource (закладка DataAccess).

2.2 Подключим таблицу ADOTable1 к компоненту ADOConnection1 и к главной таблице «Товары» нашей БД «Товар».

В инспекторе объектов таблицы ADOTable1 устанавливаем свойства:

— Connection = ADOConnection1.
— TableName = Товары.
— Active = True (! это свойство устанавливать в последнюю очередь).

2.3 В инспекторе объектов компонента DataSource1 установите свойство DataSet = ADOTable1.

2.4 Проведя аналогичные действия (п.2.2 и п.2.3) подключите таблицу ADOTable2 и DataSource2 к компоненту ADOConnection1 и к подчиненной таблице «Заказы» нашей БД.

2.5 Двойным щелчком на компоненте ADOTable1 откроем Окно редактора полей, щелкнем в окне правой кнопкой мыши и в контекстном меню выберем команду Add all fields.

2.6 Переходим на Form4 и оформляем вывод таблиц на дисплей. Возьмите два компонента GroupBox (закладка Standard), используя свойства Caption дайте им заголовки «Товары» (для GroupBox1) и «Заказы» (для GroupBox2).

2.7 Положите в GroupBox1 сетку DbGrid1 (закладка Data Controls). Для DbGrid1 свяжите свойство DataSource с источником данных DataSource1. Если данные не отобразились, то в Окне редактора полей (двойной клик по DbGrid1) выберите команду Add all fields.

2.8 Переходим на модуль данных dm. В Инспекторе объектов подчиненной таблицы ADOTable2 устанавливаем свойство MasterSource = DataSource1 (источник главной таблицы).

2.9 Обращаемся к свойству MasterFillds. В появившемся окне Редактора полей (см. рисунок 4) выберите ключевое поле КодТовара главной таблицы (окно Master Filds) и поле КодТовара подчиненной таблицы (окно Detail Filds). Нажмите кнопку Add, а затем «OK». Связь установлена.

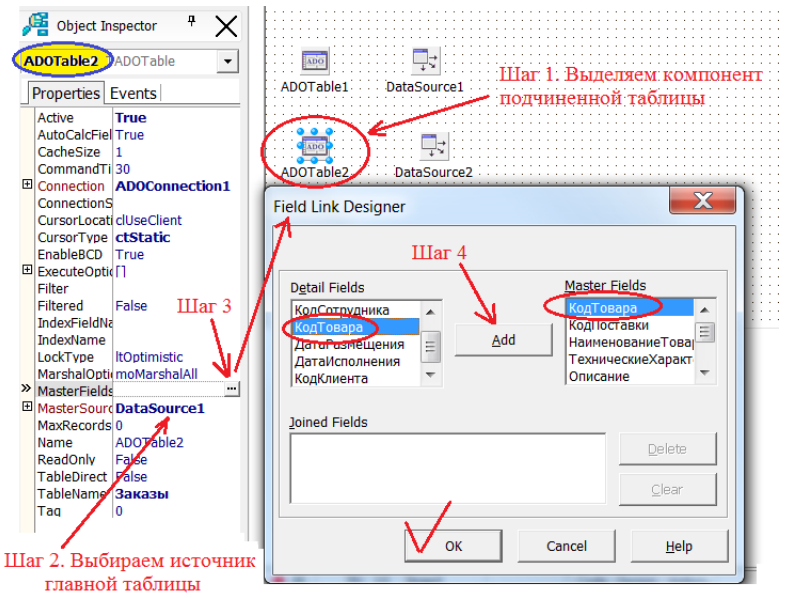


Рисунок 4 – Установка связи между таблицами ADOTable1 и ADOTable2

2.10 Сохраняем проект (File -> Save All), компилируем и запускаем проект (F9 или Run). Окно формы показано на рисунке 5. Обратите внимание на эффект от связывания таблиц. Сравните с выводом этих данных в окне «Заказы» «Товары» «Клиенты» (Form2).

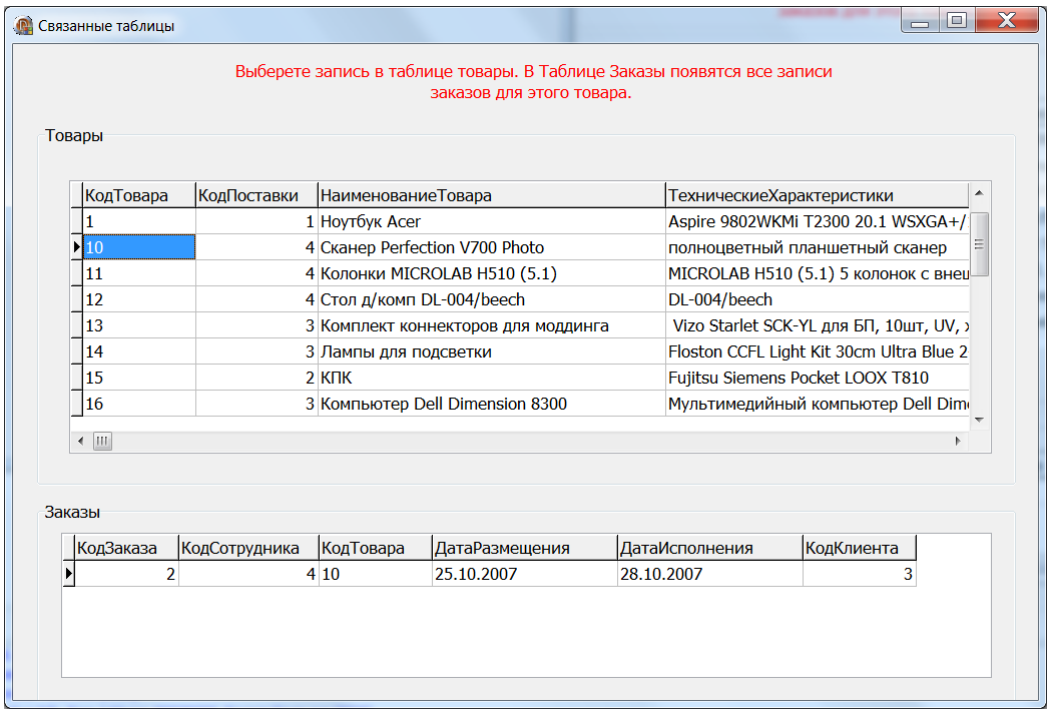


Рисунок 5 – Окно закладки «Связанные таблицы»

3 Добавление подстановочных полей НД «Заказы»

3.1 Организуем Lookup-поле «ФИО_Клиента» в таблице ADOTable2, связанное с полем ФИО таблицы TabKlient.

3.1.1 Переходим на модуль dm. Выделяем ADOTable2.

3.1.2 В Инспекторе объектов таблицы ADOTable2 устанавливаем свойство Active = False – перед созданием новых полей необходимо закрыть компонент.

3.1.3 Двойным щелчком на компоненте ADOTable2 откройте окно редактора полей и щелчком правой кнопки мыши вызовете контекстное Меню. Выберете команду New Field, чтобы открыть окно конструктора нового поля (рисунок 6).

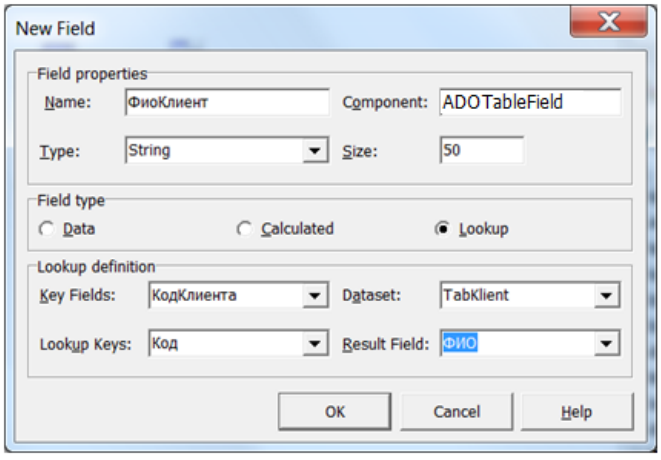


Рисунок 6 – Окно конструктора нового поля «ФиоКлиент»

В окне конструктора нового поля заполняем:

Элементы группы Field properties:

— Name = ФиоКлиент – имя нового поля.
— Component = ADOTable2Field (устанавливается автоматически).
— Type = String – тип нового поля (как в БД).
— Size = 50 (как размер поля ФИО в таблице Клиенты).

Элементы группы Field type:

— Устанавливаем переключатель в Lookup.

Элементы группы Lookup definition:

— Key fields = КодКлиента – внешний ключ в нашем наборе данных (таблице Заказы).
— DataSet = TabKlient – дополнительный НД (таблица Клиенты).
— Lookup keys = Код – ключевое поле дополнительного НД.
— Result field = ФИО – поле дополнительного НД, которое будет отображаться в поле ФиоКлиент нашего НД (таблице Заказы).

3.1.4 В Инспекторе объектов таблицы ADOTable2 установите свойство Active = Try.

3.1.5 Сохраните проект (File->Save All), скомпилируйте и запустите (F9).

3.2 Организуйте Lookup-поле «АдресКлиент» в таблице ADOTable2, связанное с полем Адрес таблицы TabKlient (см. рисунок 7).

3.3 Организуйте Lookup-поле «ТелефонКлиент» в таблице ADOTable2, связанное с полем Телефон таблицы TabKlient (тип – string, размер 15).

3.4 Организуйте Lookup-поле «ФамилияСотрудника» в таблице ADOTable2, связанное с полем Фамилия таблицы TabSotrydniki (тип – string, размер 50).

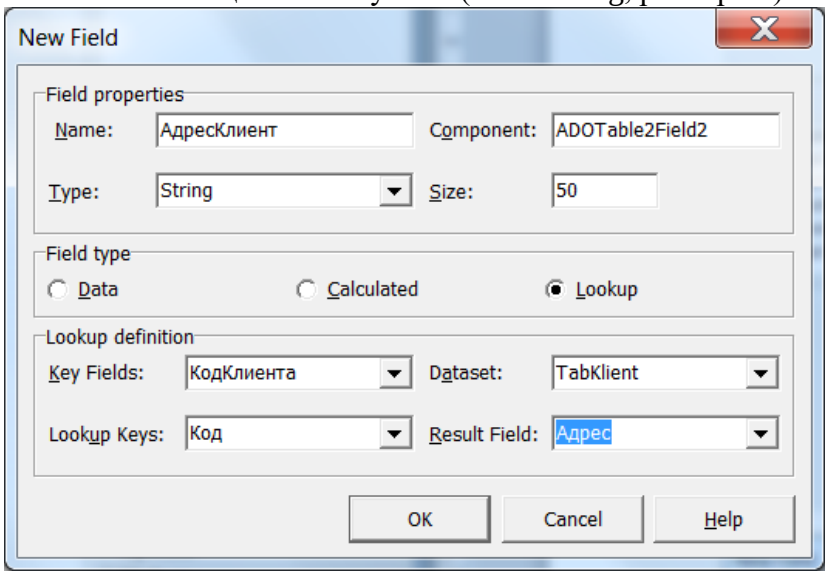


Рисунок 7 – Окно конструктора нового поля «АдресКлиент»

3.5 Организуйте Lookup-поле «ИмяСотрудника» в таблице ADOTable2, связанное с полем Имя таблицы TabSotrydniki (тип – string, размер 50).

4 Сделаем невидимыми кодовые поля таблиц на Form4, т.к. их не обязательно видеть пользователю. Установим запрет на редактирование.4.1 Переходим на Form4. Дважды щелкните мышью на сетке DBGrid1, вызовется редактор столбцов (рисунок 8) и, если необходимо, выберете команду Add All Fields. Далее редактируем свойства каждого столбца в отдельности.

4.2 Редактируем поле КодТовара. В инспекторе объектов для DBGrid1.Columns[0] устанавливаем свойства: Alignment=taCenter (по центру), Visible=Try (изображать на сетке).

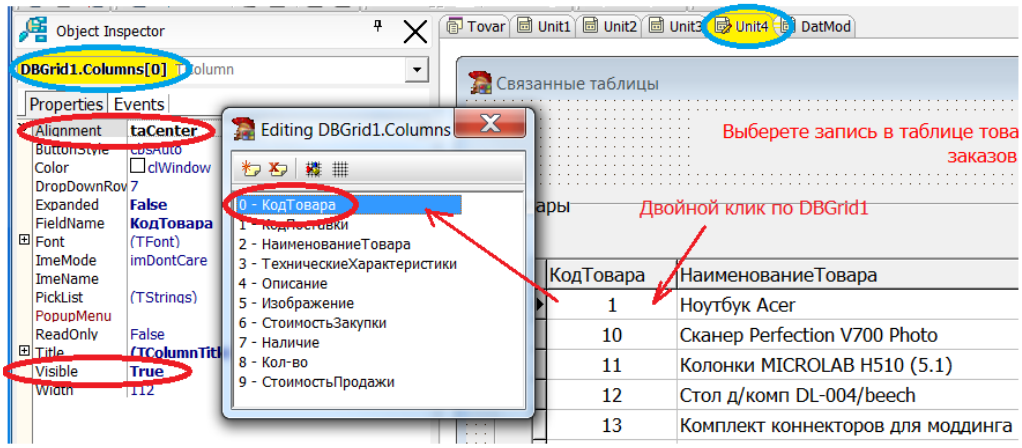


Рисунок 8 – Настройка столбцов сетки DBGrid1

4.2 В редакторе столбцов DBGrid1 последовательно выберайте столбцы КодПоставки, Описание и Изображение, и выставляйте свойство: Visible=False (не изображать на сетке).

4.3 Для НД «Заказы» (сетка DBGrid2) сделайте невидимыми поля КодЗаказа, КодСотрудника, КодТовара, КодКлиента.

4.4 Сохраните проект (File->Save All), скомпилируйте и запустите (F9).

4.5 Сделаем запрет на редактирование сеток DBGrid1 и DBGrid2. Для этого в Инспекторе раскрываем вкладку Option (нажимаем +) и устанавливаем свойство dgEditing=False.

4.6 Сохраните проект (File->Save All), скомпилируйте и запустите (F9).

5. Проблема. При изменении записей в таблицах «Заказы», «Товары», «Клиенты» на форме Form2, не происходит обновления в таблицах связанных наборов данных «Товары» и «Заказы» на форме Form4.

5.1 Обновление НД «Товары» на форме Form4.

5.1.1 Переходим на Form2. Заходим в Инспектор объектов DBNavigator2. Вызываем обработчик события OnClick (двойным кликом).

5.1.2 В открывшемся окне Code пишем код обновления таблицы ADOTable1, которая связана с НД «Товары» на форме Form4.

```
// Обновление записи при нажатие кнопки Навигатора2 "Обновить"
procedure TForm2.DBNavigator2Click(Sender: TObject; Button: TNavigateBtn);
begin
if Button = nbRefresh then
  begin
    dm.ADOTable1.Requery; // Обновить таблицу ADOTable1
  end;
end;
```

Пояснение. Если на DBNavigator2 нажата кнопка  nbRefresh, то дополнительно (свойство Requery) обновляется таблица ADOTable1.

5.2 Обновление НД «Заказы» на форме Form4, который связан с таблицами «Заказы», «Клиенты» и «Сотрудники».

5.2.1 Изменение таблицы «Заказы» на форме Form2.

```
// Обновление записи при нажатие кнопки Навигатора 1 "Обновить"
procedure TForm2.DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
begin
if Button = nbRefresh then
  begin
    dm.ADOTable2.Requery; // Обновить таблицу ADOTable2
  end;
end;
```

5.2.2 Изменение таблицы «Клиенты» на форме Form2.

```
// Обновление записи при нажатие кнопки Навигатора 3 "Обновить"
procedure TForm2.DBNavigator3Click(Sender: TObject; Button: TNavigateBtn);
begin
if Button = nbRefresh then
  begin
    dm.ADOTable2.Requery; // Обновить таблицу ADOTable2
  end;
end;

end.
```

5.2.3 Аналогично, произведите изменение таблицы «Сотрудники» на форме Form3.
5.3 Сохраните проект (File->Save All), скомпилируйте и запустите (F9).

Отчет должен содержать:
1. Название, цель и задание лабораторной работы;
2. Алгоритм работы программы;
3. Результаты работы программы;
4. Ответы на контрольные вопросы.

Контрольные вопросы:
1. Порядок установки отношений между таблицами БД.
3. Какие образом добавляются подстановочные поля?