МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.Алексеева

Кафедра «Информатика и системы управления»

БАЗЫ ДАННЫХ

СЕМАНТИЧЕСКИЕ МОДЕЛИ ДАННЫХ

Методические указания к лабораторной работе №2 для студентов специальности 230102, 230201

Нижний Новгород 2010

Составитель Балашова Т.И.

ББК

БАЗЫ ДАННЫХ. СЕМАНТИЧЕСКИЕ МОДЕЛИ ДАННЫХ: учебно-методические указания к лабораторной работе №2 для студентов специальности 230102, 230201/ НГТУ; сост.: Балашова Т.И. Н.Новгород 2010. 20с.

В учебно-методические материалах изложены основные современные подходы к проектированию баз данных, основанных на использовании семантических моделей данных, приведен пример разработки базы данных с использованием Borland Delphi 7

Подписано к печати . Формат 60x84 1/16. Бумага офсетная.
Печать офсетная. Усл.печ.л. Уч.-изд. л. .Тираж экз. Заказ .

Нижегородский государственный технический университет
им. Р.Е. Алексеева
Типография НГТУ им. Р.Е. Алексеева
Адрес университета и полиграфического предприятия:
603950, Нижний Новгород, ул. Минина, 24.

Цель работы: изучение современного подхода к проектированию баз данных, основанного на использовании семантических моделей данных.

Краткие сведения из теории

1. Общие сведения

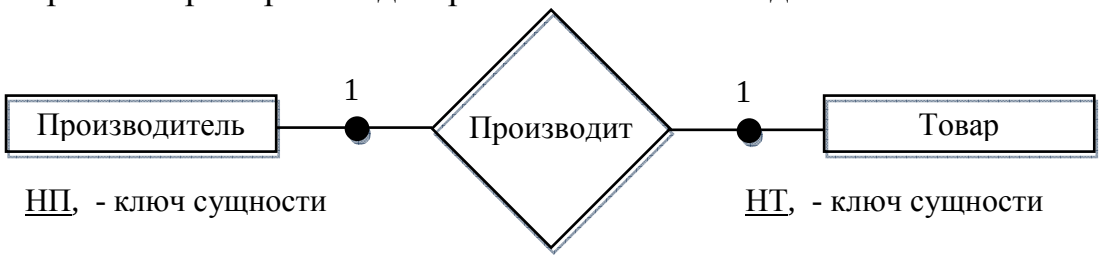
1.1. Семантическое моделирование

В реальном проектировании структуры базы данных применяется метод, так называемого, ***семантического моделирования***, который представляет собой моделирование структуры данных, опираясь на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты ***диаграмм сущность – связь (Entry – Relationship (ER))***.

Первый вариант модели «сущность – связь» был предложен в 1976 г. Питером Пин – Шэн Ченом. В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.). Кроме того, различные программные средства, реализующие одну и ту же модель, могут отличаться своими возможностями. По сути, все варианты диаграмм «сущность – связь» исходят из одной идеи – рисунок всегда нагляднее текстового описания. Все подобные диаграммы используют графическое изображение сущностей предметной области их свойств (атрибутов), а так же взаимосвязей между ними.

1.2. Пример построения ER – диаграммы:

Простой пример ER – диаграммы может выглядеть так:



Сущности изображаются прямоугольниками, связи в виде ромбов, ниже каждой сущности указывают атрибут или набор атрибутов, являющийся ключом сущности.

1.3. Проектирование базы данных с помощью метода «сущность – связь»

Проектирование базы данных с помощью метода «сущность-связь» можно разбить на несколько шагов:

• описание предметной области. Формулировка автоматизируемых бизнес – процессов, определение их участников, а также их действий, информацию о которых нужно фиксировать в БД.

• построение инфологической модели базы данных. Производится анализ и структурирование данных: определение сущностей, оптимизация их количества и связей между ними с использованием правил нормализации. На данном этапе проектирования база данных не привязывается к какой–либо конкретной СУБД.

• построение диаграммы ER-типа, включающей все сущности и все связи, обнаруженные в результате анализа инфологической модели предметной области.

• построение набора предварительных отношений и указание предполагаемого ключа для каждого отношения.

• подготовка списка всех атрибутов и распределение этих атрибутов по полученным отношениям. Необходимо определить для каждого отношения функциональные зависимости и проверить, находятся ли эти отношения в НФБК. Если хотя бы одно отношение не находится в НФБК или некоторые атрибуты не могут логично включиться ни в одно отношение, то необходимо пересмотреть диаграммы ER – типа.

2. Рассмотрим реальную ситуацию построения ER – диаграммы:

Предметная область: сеть продуктовых магазинов.

Необходимо создать БД для сети продуктовых магазинов, в которой будут храниться данные о поступлениях товаров в магазины и их продажах.

Выделим следующие сущности:

- «магазин»;
- «товар»;
- «производитель»;
- «тип товара»;
- «экземпляр товара».

Рассмотрим, какие атрибуты будут иметь эти сущности:
Сущность «Магазин»:

- код магазина (первичный ключ этой таблицы);
- название;
- адрес;

- телефон.

Сущность «Производитель»:

- код производителя (первичный ключ этой таблицы);
- название;
- адрес;
- телефон;

Сущность «Товар»:

- код товара (первичный ключ этой таблицы);
- название товара («Коровка», «Юбилейное»).

Сущность «Тип товара»

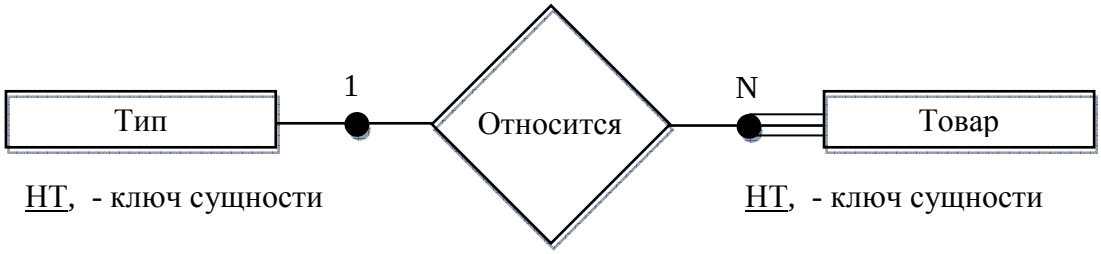
- код типа товара (первичный ключ этой таблицы);
- наименование (молоко, сметана, конфеты и т.д.).

Сущность «Экземпляр товара;:

- код экземпляра товара (первичный ключ данной таблицы);
- количество;
- дата поступления товара;
- флаг (поступил данный товар, или продан).

Товар производится производителем – отношение «производится». Обязательно вхождение каждого производителя и каждого товара в отношение (каждый производитель обязательно что-то производит и каждый товар кем-то производится). При этом у одного производителя множество товаров и один товар могут производить несколько производителей. Связь N:N.

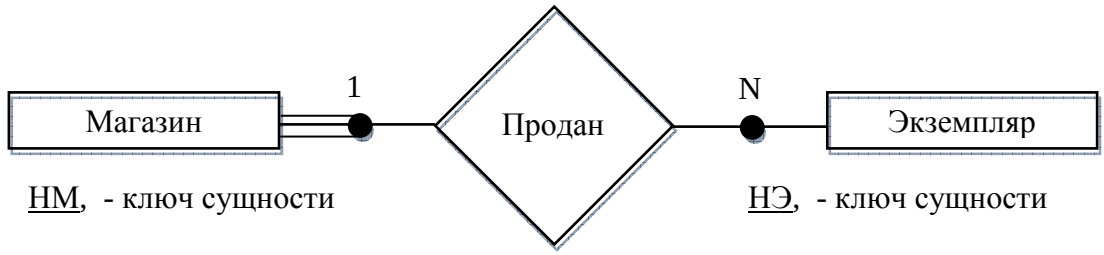
Товар относится к типу – отношение «относится». Обязательно вхождение каждого товара в отношение (товар обязательно имеет тип). Вхождение каждого типа товара в отношение не обязательно, так как товаров данного типа может не быть. При этом может быть множество товаров одного типа и каждый товар принадлежит только к одному типу. Связь 1:N.



Экземпляр товара находится в магазине или продан – отношение «продан». Обязательно вхождение каждого магазина в отношение, так как в магазине обязательно есть товары. Данный экземпляр может быть только в

одном магазине, при этом в магазине может быть множество экземпляров товаров. Отношение 1:N.

Так как для проданных и поступивших товаров все атрибуты совпадают, то они относятся к одной сущности и различаются только по атрибуту «Флаг».



На основании данной диаграммы создадим БД в Borland Delphi (Borland Software Corporation).

3. Проектирование базы данных в среде Borland Delphi

Delphi – Что Это?

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую «быструю разработку», среди которых можно выделить *Borland Delphi* и *Microsoft Visual Basic*. В основе систем быстрой разработки (*RAD* – систем, *Rapid Application Development* – среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть рутинной работы, оставляя программисту работу по конструированию диалоговых окон и функций обработки событий. Производительность программиста при использовании RAD – систем значительно повышается.

3.1. Модель базы данных в Delphi

В *Delphi* каждая таблица физически хранится в отдельном файле. Однако отождествлять базу данных и таблицу нельзя, так как довольно часто поля одной записи распределены по нескольким таблицам и, следовательно, находятся в разных файлах.

3.2. Псевдоним базы данных

Разрабатывая программу работы с базой данных, программист не может знать, на каком диске и в каком каталоге будут находиться файлы базы данных во время ее использования. Например, пользователь может поместить базу данных в один из каталогов дисков C:, D: или на сетевой

диск. Поэтому возникает проблема передачи в программу информации о месте нахождения файлов базы данных.

В *Delphi* проблема передачи в программу информации о месте нахождения файлов базы данных решается путем использования псевдонима базы данных. Псевдоним *(Alias)* — это короткое имя, поставленное в соответствие реальному, полному имени каталога базы данных. Например, псевдонимом каталога *F:\Data\ может быть имя shopsdotnet.* Программа работы с базой данных для доступа к данным использует не реальное имя, а псевдоним.

Для доступа к информации программа, обеспечивающая работу с базой данных, подключает библиотеку *Borland Database Engine (BDE),* которая, в свою очередь, использует конфигурационный файл, содержащий информацию обо всех зарегистрированных в системе псевдонимах.

Псевдоним базы данных может быть создан (зарегистрирован) при помощи утилиты *BDE Administrator*. Эта же утилита позволяет изменить каталог, связанный с псевдонимом.

3.3. Создание базы данных

База данных — это набор файлов (таблиц), в которых находится информация. Как правило, база данных состоит из нескольких таблиц, которые размещают в одном каталоге. Каталог для новой базы данных создается обычным образом, например, при помощи Проводника. Таблицу можно создать, воспользовавшись входящей в состав Delphi утилитой *Borland Database Desktop* или организовав SQL – запрос к серверу базы данных.

Так как для доступа к файлам (таблицам) базы данных библиотека BDE использует не имя каталога, в котором находятся файлы, а его псевдоним, то перед тем как приступить к созданию таблиц новой базы данных, необходимо создать псевдоним для этой базы данных.

Таким образом, процесс создания базы данных может быть представлен как последовательность следующих шагов:

1. Создание каталога.
2. Создание псевдонима.
3. Создание таблиц.

3.4. Создание каталога

Каталог (папка) для файлов базы данных создается обычным образом, например, при помощи Проводника. Обычно файлы локальной базы данных

помещают в отдельном подкаталоге каталога программы работы с базой данных.

***Примечание**: для дальнейшей работы с рассматриваемой в качестве примера базой данных «Сеть продуктовых магазинов» следует в каталоге «Проекты» создать каталог «Сеть продуктовых магазинов» и в нем — подкаталог «Data».*

3.5. Создание псевдонима

Псевдоним базы данных создается при помощи входящей в *Delphi* утилиты *BDE Administrator*, которая запускается из *Windows* выбором из меню *Все Программы → Borland Delphi 7 → BDE Administrator***.**

Вид диалогового окна *BDE Administrator* после запуска приведен на рис.1

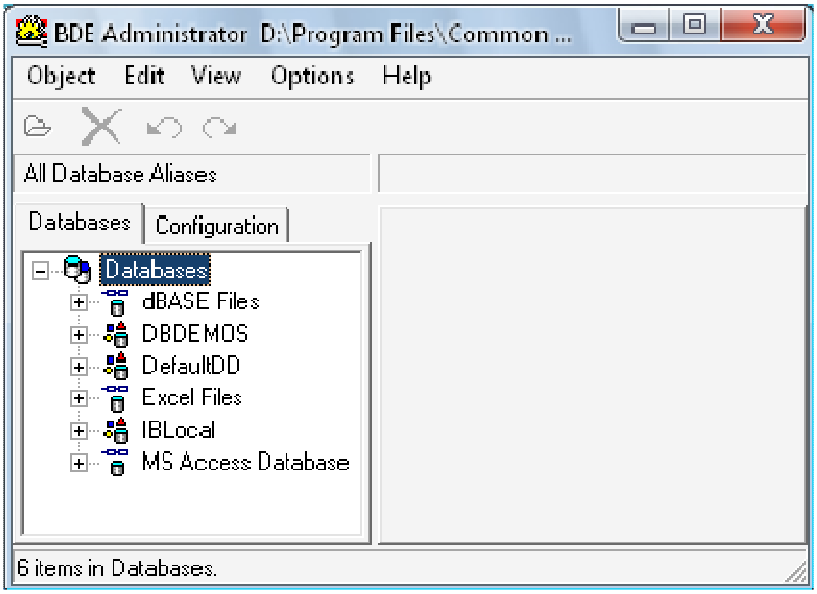


Рис. 1 Окно BDE Administrator

В левой части окна, на вкладке Databases, перечислены псевдонимы, зарегистрированные на данном компьютере. Для того чтобы создать новый псевдоним, необходимо из меню Object выбрать команду New. Затем в открывшемся диалоговом окне New → Database Alias (Новый псевдоним базы данных) из списка Database Driver Name, в котором перечислены зарегистрированные в системе драйверы доступа к базам данных, нужно выбрать драйвер для создаваемой базы данных (рис. 17.3), т. е. фактически выбрать тип создаваемой базы данных.

При создании псевдонима по умолчанию предлагается драйвер STANDARD (default driver), который обеспечивает доступ к таблицам в формате Paradox.

Путь к файлам базы данных можно ввести на вкладке Definition в поле Path с клавиатуры или воспользоваться стандартным диалоговым окном Select Directory (Выбор каталога), которое открывается щелчком на кнопке с тремя точками, находящейся в конце поля Path.

Для работы будем использовать псевдоним базы данных STANDART и укажем путь, как изображено на рис. 2.

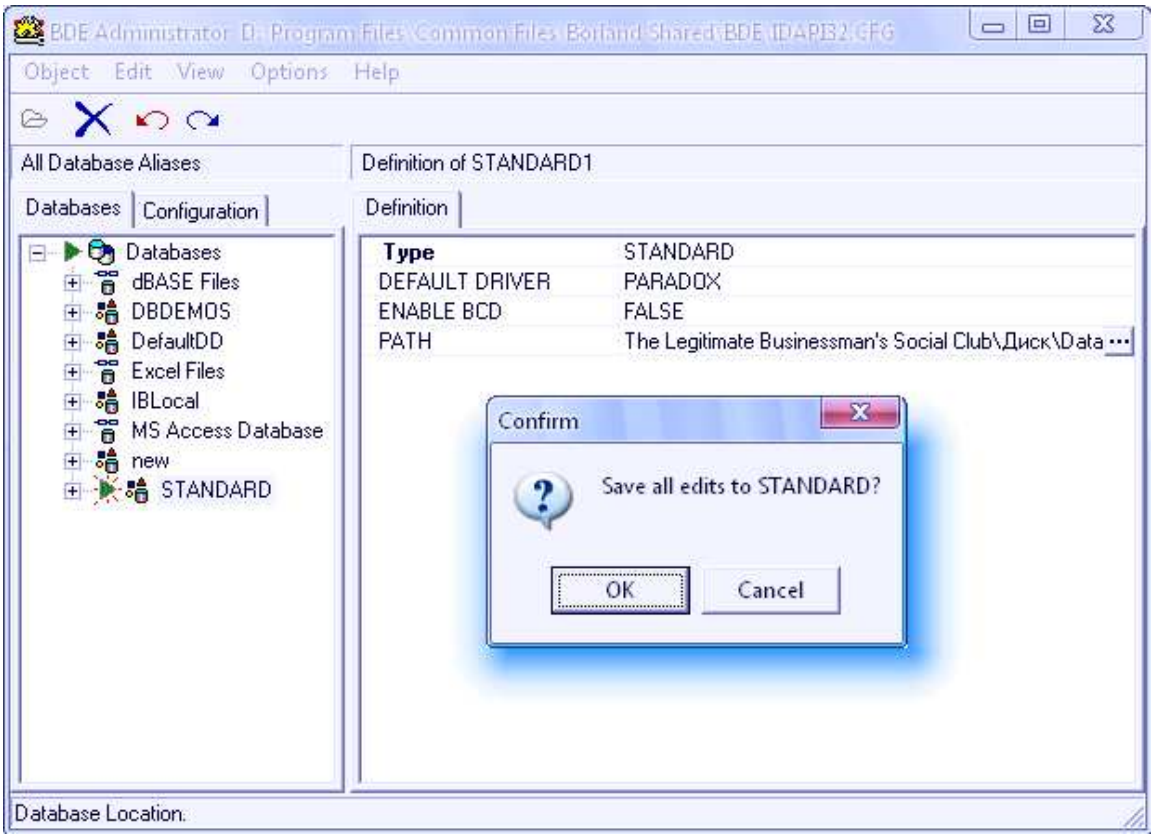


Рис. 2 Окно BDE Administrator Сохранение настроек

3.6. Создание таблиц

Важным моментом при создании базы данных является распределение информации между полями записи. Очевидно, что информация может быть распределена между полями различным образом.

Если предполагается, что во время использования базы данных будет выполняться выборка информации по некоторому критерию, то информацию, обеспечивающую возможность этой выборки, следует поместить в отдельное поле.

После того как определена структура записей базы данных, можно приступить непосредственно к созданию таблицы. Таблицы создаются при помощи входящей в состав Delphi утилиты Database Desktop.

Утилита Database Desktop позволяет выполнять все необходимые при работе с базами данных действия. Она обеспечивает создание, просмотр и модификацию таблиц баз данных различных форматов (Paradox, dBASE, Microsoft Access). Кроме того, утилита позволяет выполнять выборку информации путем создания запросов.

Для того чтобы создать новую таблицу, нужно выбором из меню Tools команды Database Desktop запустить Database Desktop. Затем в появившемся окне утилиты Database Desktop надо из меню File выбрать команду New и в появившемся списке выбрать тип создаваемого файла — Table. Затем в открывшемся диалоговом окне Create Table следует выбрать тип создаваемой таблицы (значением по умолчанию является тип Paradox 7).

В результате открывается диалоговое окно Create Paradox 7 Table, в котором можно определить структуру записей таблицы.

Для каждого поля таблицы необходимо задать имя, тип и, если нужно, размер поля. Имя поля используется для доступа к данным. В качестве имени поля, которое вводится в колонку Field Name, можно использовать последовательность из букв латинского алфавита и цифр длиной не более 25 символов.

Тип поля определяет тип данных, которые могут быть помещены в поле. Тип задается вводом в колонку Type символьной константы. Типы полей и соответствующие им константы приведены в приложении А.

Константа, определяющая тип поля, может быть введена с клавиатуры или путем выбора типа поля из списка, который появляется при щелчке правой кнопкой мыши в колонке Type или при нажатии клавиши < Пробел >.

Одно или несколько полей можно пометить как ключевые. Ключевое поле определяет логический порядок следования записей в таблице при ее выводе, т.е. записи будут упорядочены в соответствии алфавитным порядком или по возрастанию. Если ключевое поле задано не будет, то записи будут выводиться в соответствии с тем порядком, в котором они были добавлены. Следует заметить, что при определении ключа, необходимо проследить за тем, чтобы ключ был уникальным для каждой записи, т.е., например, при указании ключевым полем фамилии возможен ее повтор для однофамильцев, так же, как и для имен. Поэтому в качестве ключевого поля необходимо указать поле или группу полей, значение которых не повторяется, например, серию и номер паспорта.

Для того чтобы пометить поле как ключевое, необходимо выполнить двойной щелчок в колонке Key. Следует обратить внимание на то, что ключевые поля должны быть сгруппированы в верхней части таблицы.

Если данные, для хранения которых предназначено поле, должны обязательно присутствовать в записи, то следует установить флажок Required Field. Например, очевидно, что поле Fam (Фамилия) обязательно должно быть заполнено, в то время как поле Telephone (Телефон) может оставаться пустым.

Если значение, записываемое в поле, должно находиться в определенном диапазоне, то вводом значений в поля Minimum value (Минимальное значение) и Maximum value (Максимальное значение) можно задать границы диапазона. Поле Default value позволяет задать значение по умолчанию, которое будет автоматически записываться в поле при добавлении к таблице новой записи.

Поле Picture позволяет задать шаблон, используя который можно контролировать правильность вводимой в поле информации. Шаблон представляет собой последовательность обычных и специальных символов. Специальные символы перечислены в табл. 2.

Во время ввода информации в позицию поля, которой соответствует специальный символ, будут появляться только символы, допустимые для данного символа шаблона. Например, если в позиции шаблона стоит символ #, то в соответствующую этому символу позицию можно ввести только цифру. Если в позиции шаблона стоит обычный символ, то во время ввода информации в данной позиции будет автоматически появляться указанный символ.

Например, пусть поле Telephone типа A (строка символов) предназначено для хранения номера телефона, и программа, работающая с базой данных, предполагает, что номер телефона должен быть представлен в обычном виде, т. е. в виде последовательности сгруппированных, разделенных дефисами цифр. В этом случае в поле Picture следует записать шаблон: #-###-###-##-##. При вводе информации в поле Telephone будут появляться только цифры (нажатия клавиш с другими символами игнорируются), причем после ввода первой, четвертой, седьмой и девятой цифр в поле будут автоматически добавлены дефисы.

Табл. 2.

*	Цифра, любая буква (прописная или строчная), любой символ
&	Любой символ (если введена буква, то она автоматически преобразуется в прописную)
@	Символ, следующий за символом «точка с запятой», интерпретируется как обычный символ, а не символ шаблона
*.	Любое количество повторяющихся, определяемых следующим за «звездочкой» символом шаблона

Некоторые элементы данных поля могут быть необязательными, например, код города для номера телефона. Элементы шаблона,

обеспечивающие ввод необязательных данных, заключают в квадратные скобки. Например, шаблон [(###)]###-##-## позволяет вводить в поле номер телефона как с заключенным в скобки кодом города, так и без кода.

Шаблоны позволяют не только контролировать правильность вводимых в поле данных путем блокирования ввода неверных символов, но и обеспечивают автоматизацию ввода данных. Это достигается путем указания в шаблоне в квадратных или фигурных скобках списка допустимых значений содержимого поля.

Например, если для поля Address задать шаблон {Санкт-Петербург, Москва, Воронеж}*@ или [Санкт-Петербург, Москва, Воронеж]*@, то во время ввода данных в это поле название соответствующего города будет появляться сразу после ввода одной из букв: с, м или в. Отличие фигурных скобок от квадратных и, следовательно, этих шаблонов друг от друга состоит в том, что в первом шаблоне содержимое поля обязательно должно начинаться с названия одного из перечисленных в списке городов, а во втором – город может называться по-другому, однако его название придется вводить полностью.

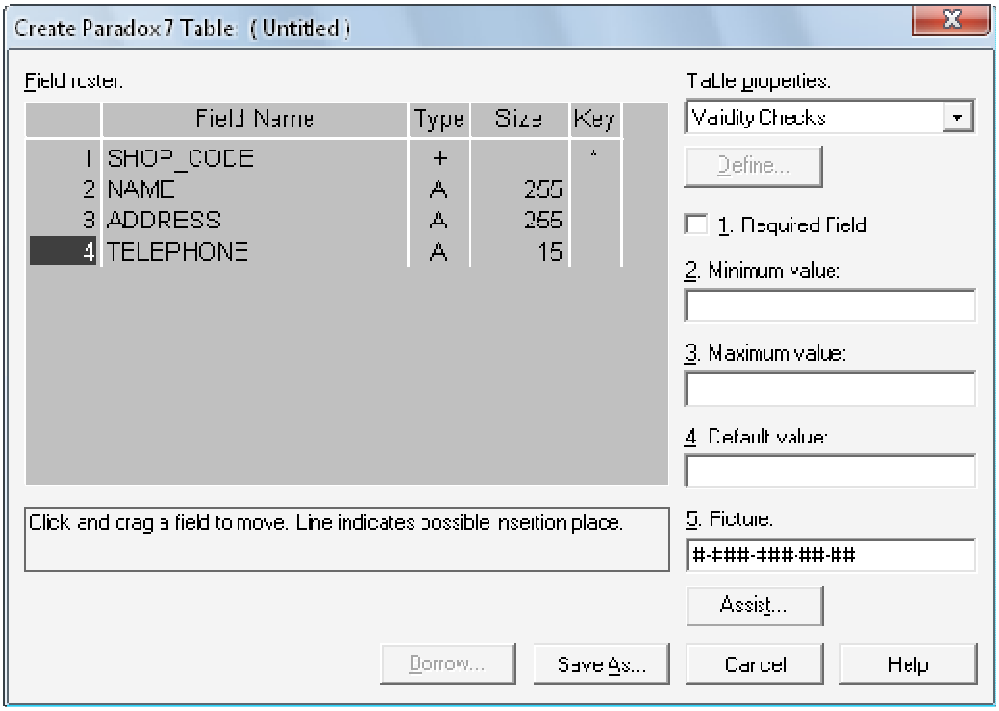


Рис. 3. Окно Database Desktop Create Paradox 7 Table

После того как будет определена структура таблицы, таблицу следует сохранить. Для этого необходимо нажать кнопку Save As. В результате открывается окно Save Table As. В этом окне из списка Alias нужно выбрать псевдоним базы данных, частью которой является созданная таблица, а в поле Имя файла ввести имя файла, в котором нужно сохранить созданную таблицу.

Если перед тем как нажать кнопку «Сохранить», установить флажок Display table, то в результате нажатия кнопки «Сохранить» открывается диалоговое окно Table, в котором можно ввести данные в только что созданную таблицу.

Если таблица базы данных недоступна, то, для того чтобы ввести данные в таблицу, таблицу нужно открыть. Для этого надо из меню File выбрать команду Open | Table, затем в появившемся диалоговом окне Open table в списке Alias выбрать псевдоним нужной базы данных и таблицу. Следует обратить внимание, что таблица будет открыта в режиме просмотра, в котором изменить содержимое таблицы нельзя. Для того чтобы в таблицу можно было вводить данные, нужно активизировать режим редактирования таблицы, для чего необходимо из меню Table выбрать команду Edit Data.

Данные в поля записи вводятся с клавиатуры обычным образом. Для перехода к следующему полю нужно нажать клавишу <Enter>. Если поле является последним полем последней записи, то в результате нажатия клавиши <Enter> в таблицу будет добавлена еще одна запись.

Если во время заполнения таблицы необходимо внести изменения в какое-то уже заполненное поле, то следует выбрать это поле, воспользовавшись клавишами перемещения курсора, нажать клавишу <F2> и внести нужные изменения.

Если при вводе данных в таблицу буквы русского алфавита отображаются неверно, то надо изменить шрифт, используемый для отображения данных. Для этого необходимо в меню Edit выбрать команду Preferences и в появившемся диалоговом окне, во вкладке General, щелкнуть на кнопке Change. В результате этих действий откроется диалоговое окно Change Font, в котором нужно выбрать русифицированный шрифт. Следует обратить внимание, что в Windows 2000 (Windows XP) используются шрифты типа Open Type, в то время как программа Database Desktop ориентирована на работу со шрифтами TrueType. Поэтому в списке шрифтов нужно выбрать русифицированный шрифт именно TrueType. После этого надо завершить работу с Database Desktop, так как внесенные в конфигурацию изменения будут действительны только после перезапуска утилиты.

Созданные в Database Desktop таблицы:

Article (Товар)

	Field Name	Type	Size	Key
1	ID	+		*
2	NAME	A	50	
3	ARTICLE_TYPE_ID	I		

Article_copy (экземпляр товара)

	Field Name	Type	Size	Key
1	ID	+		*
2	ARTICLE_ID	I		
3	SHOP_ID	I		
4	QUANTITY	I		
5	INCOME	L		
6	INCOME_DATE	D		

Article_type (Тип товара)

	Field Name	Type	Size	Key
1	ID	+		*
2	NAME	A	50	

Shops (Магазин)

	Field Name	Type	Size	Key
1	ID	+		*
2	NAME	A	50	
3	ADDRESS	A	100	
4	TEL_NUMBER	A	15	

Vendor (Производитель)

	Field Name	Type	Size	Key
1	ID	+		*
2	NAME	A	50	
3	ASSRESS	A	100	
4	TEL_NUMBER	A	15	

4. Программа управления базой данных

Программа управления базой данных представляет собой программную оболочку (интерфейс) взаимодействия пользователя и базы данных. Она обеспечивает работу с записями БД, выборку записей по заданному критерию, поиск записей, а также формирование отчетов.

Прежде чем взяться за организацию интерфейса БД, сделаем ряд шагов, необходимых для работы создаваемого нами интерфейса с таблицами БД.

1. Создание модуля данных

Для размещения компонентов доступа к данным в приложении баз данных желательно использовать специальную «форму» — модуль данных. В модуле данных можно размещать только невизуальные компоненты. Модуль данных доступен разработчику, как и любой другой модуль проекта,

на этапе разработки. Пользователь приложения не может увидеть модуль данных во время выполнения. Преимуществом размещения компонентов доступа к данным в модуле данных является то, что изменение значения любого свойства проявится сразу же во всех обычных модулях, к которым подключен этот модуль данных. Кроме этого, все обработчики событий этих компонентов, т. е. вся логика работы с данными приложения, собраны в одном месте, что тоже весьма удобно. Модуль данных можно создать, воспользовавшись меню File → New → Data Module.

Создав модуль данных, добавим в него необходимые компоненты доступа к данным: DataSource (DB) и Table (DB Tables). Для каждой таблицы необходимо по одному DataSource (DB) и Table (DB Tables). Найти их можно на вкладках Data Accesss (для Data Source) и BDE (для Table). После добавления нужного количества компонентов, присвоим им имена совпадающие с именами сущностей, определенных ранее.

***Примечание**: так как одно и тоже имя для двух и более компонентов недопустимо, то имя компонента Table можно сделать, как и у Data Source, добавив к нему «_table».*

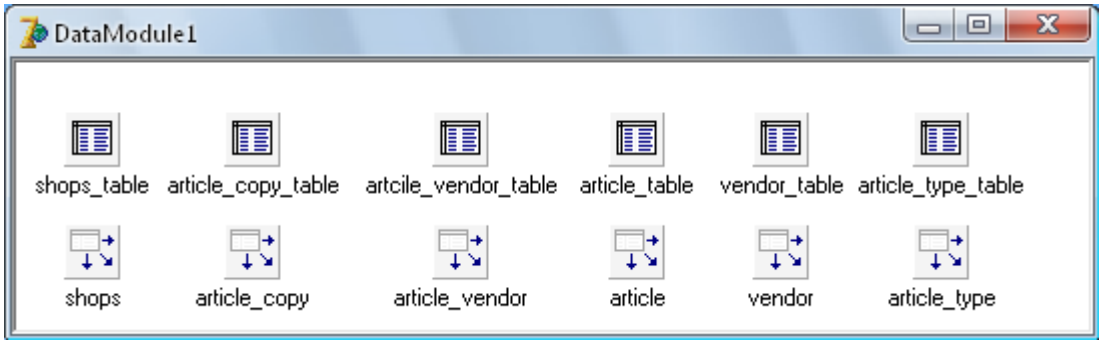


Рис. 4 Окно Модуля Данных

2. Настройка свойств компонентов

После того как имена заданы, свяжем компоненты с соответствующими их именам таблицами БД. Для этого выделим компонент Table, и в Инспекторе объектов - Object Inspector (Инспектор объектов – окно, расположенное в левой части и содержащее свойства выделенного объекта), установим свойства DataBaseName на имя псевдонима БД, а свойство TableName – на имя соответствующей таблицы. Для компонента DataSource установим значение свойства DataSet как имя соответствующего компонента Table. Например, для компонента DataSource с именем shops свойство DataSet будет выглядеть как shops_table.

***Примечание**: под словом соответствующий(ая), понимается имя компонента, заданного согласно описанного выше метода. Имена*

компонентов могут быть произвольными. Похожие имена, отражающие связываемые с компонентами таблицы, используются для наглядности и удобства проектирования.

3. Определение связей между таблицами

Свойства *MasterSource* и *MasterFields* – используются для установления связи «один – ко – многим», которые устанавливаются для подчиненной таблицы. Набор данных главной таблицы не требует никаких дополнительных настроек, и заданная связь будет работать только при перемещениях по записям главной таблицы.

Свойство *Mastersource* определяет компонент *DataSource*, который связан с главной таблицей.

Затем при помощи свойства *MasterFieids* необходимо установить отношения между полями главной и подчиненной таблицы. В нем содержится имя индексированного поля, по которому устанавливается связь. Если таких полей несколько, их имена разделяются точкой с запятой. При этом не все поля, входящие в индекс, обязаны участвовать в создании отношения.

Для задания свойства *MasterFields* можно использовать Редактор связей полей (*Field Link Designer*), который вызывается щелчком на кнопке в поле редактирования этого свойства в Инспекторе объектов (рис. 5).

***Примечание**: при установке свойств компонентов доступа к данным необходимо сначала настроить компонент Table, а затем свойства соответствующего Data Source, так как в Data Source компонент Table будет играть роль набора данных. Одной из возможных ошибок, возникающих при установке связей, могла бы быть некорректная настройка свойств компонентов доступа.*

Здесь в разворачивающемся списке *Available Indexes* выбирается требуемый индекс для подчиненной таблицы. После этого в списке *Detail Fields* появляются имена всех полей, входящих в этот индекс. В списке *Master Fields* отображаются все поля главной таблицы.

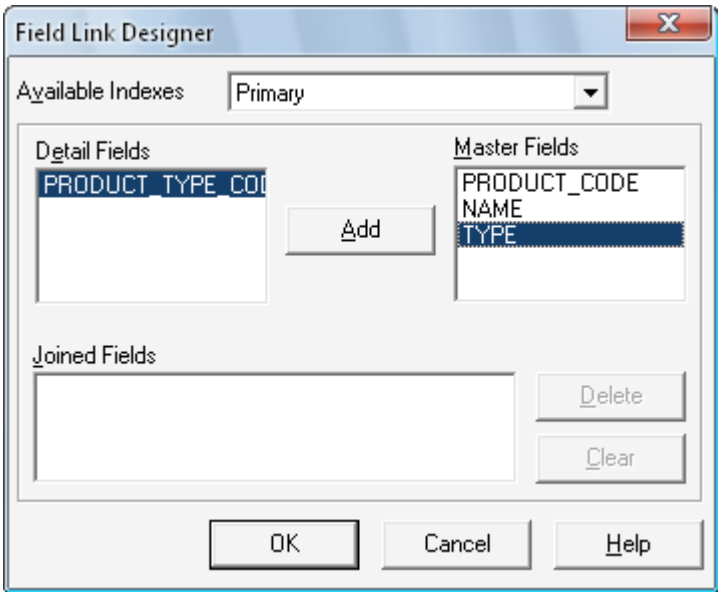


Рис. 5 Окно Field Link Designer

Теперь требуется создать связи между полями. Для этого в левом списке выбирается поле подчиненной таблицы, а затем соответствующее ему поле главной таблицы в правом списке. После этого активизируется кнопка *Add*, щелчок на которой создает отношение по двум полям главной и подчиненной таблиц. Созданная связь отображается в списке *Joined Fields*.

После создания связей между полями отношение «один – ко – многим» считается установленным. Теперь достаточно открыть оба набора данных, чтобы увидеть работу отношения.

Отношение «многие – ко – многим» отличается тем, что подчиненная таблица еще раз связывается в качестве главной с другой подчиненной таблицей аналогичной последовательностью действий, как и в отношении «один – ко – многим».

В качестве примера установки связей рассмотрим упоминавшуюся ранее схему данных сети продуктовых магазинов. В соответствии с определенными ранее сущностями и их атрибутами, схема данных будет иметь следующий вид (рис. 6).

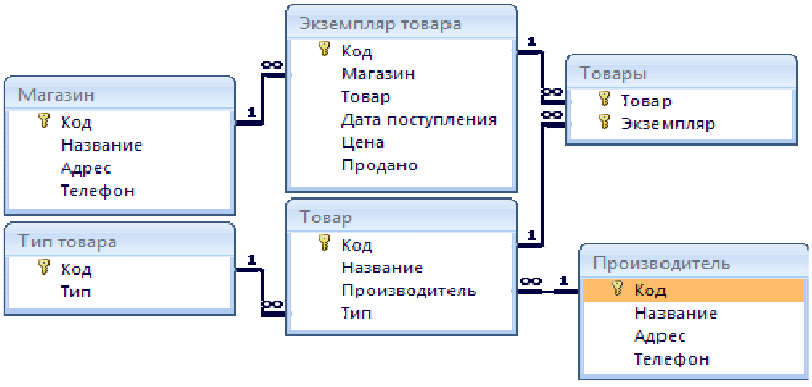


Рис. 6 Схема данных

После добавления необходимых компонентов в Data Module, листинг данного элемента будет выглядеть:

```
unit DataModule;

interface

uses
  SysUtils, Classes, DB, DBTables;

type
  TDataModule1 = class(TDataModule)
    shops: TDataSource;
    article_copy: TDataSource;
    article_vendor: TDataSource;
    article: TDataSource;
    vendor: TDataSource;
    article_type: TDataSource;
    shops_table: TTable;
    article_copy_table: TTable;
    artcile_vendor_table: TTable;
    article_table: TTable;
    vendor_table: TTable;
    article_type_table: TTable;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  DataModule1: TDataModule1;

implementation

{$R *.dfm}

end.
```

Примечание: *представленная в качестве примера схема данных, продемонстрирована в приложении Data Module на компакт диске. Для корректной работы проекта требуется создать псевдоним STANDART и указать путь к папке Data.*

Задание:

На основе результатов проработки предметной области построить ER – диаграмму и дать её краткое описание. По полученной ER – диаграмме построить схему данных. Создать таблицы БД при помощи BDE Administrator и Data Base Desktop. Определить отношения между полученными таблицами.

Контрольные вопросы:

1. Семантическое моделирование.
2. Типы связей.
3. Построить ER – диаграммы связей 1:1, 1:N, N:1, N:M.
4. Этапы проектирования БД с помощью метода «Сущность – Связь».
5. Модель БД в Delphi.
6. Псевдоним БД.
7. Типы данных в БД Delphi.
8. Что такое шаблон ввода? Каково принципиальное отличие шаблонов {Строка1, Строка2, .. , СтрокаN}*@ и [Строка1, Строка2, .. , СтрокаN]*@.
9. Что такое модуль данных?
10. Как задать отношение «один – ко – многим» / «многие – ко – многим» между двумя таблицами?
11. Какое обязательное требование должно выполняться, что бы было возможным определение отношения между таблицами, при условии что компоненты доступа к данным настроены верно?

Приложение

Типы данных Paradox

Alpha	**A**	Строка символов. Максимальная длина строки определяется характеристикой Size, значения которой находятся в диапазоне 1—255
Number	**N**	Число из диапазона 10^{-307}— 10^{308} с 15-ю значащими цифрами
Money	**$**	Число в денежном формате. Цифры числа делятся на группы при помощи разделителя групп разрядов. Также выводится знак денежной единицы
Short	**S**	Целое число из диапазона -32767—32767
Long Integer	**I**	Целое число из диапазона -2 147 483 648-2 147 483 647
Date	**D**	Дата
Time	**T**	Время с полуночи, выраженное в миллисекундах
Time stamp	**@**	Время и дата
Memo	**M**	Строка символов произвольной длины. Поле типа Memo используется для хранения текстовой информации, которая не может быть сохранена в поле типа Alpha. Размер поля (1—240) определяет, сколько символов хранится в таблице. Остальные символы хранятся в файле, имя которого совпадает с именем файла таблицы, а расширение файла — mb
Formatted Memo	**F**	Строка символов произвольной длины (как у типа Memo). Имеется возможность указать тип и размер шрифта, способ оформления и цвет символов
Graphic	**G**	Графика
Logical	**L**	Логическое значение «ИСТИНА» (True) или «ЛОЖЬ» (False)
Auto-increment	**+**	Целое число. При добавлении к таблице очередной записи в поле записывается число на единицу большее, чем находится в соответствующем поле последней добавленной записи
Bytes	**Y**	Двоичные данные. Поле этого типа используется для хранения данных, которые не могут быть интерпретированы Database Desktop
Binary	**B**	Двоичные данные. Поле этого типа используется для хранения данных, которые не могут быть интерпретированы Database Desktop. Как и данные типа Memo, эти данные не находятся в файле таблицы. Поля типа Binary, как правило, содержат audio-данные

Примечание: *типы данных, приведенные в табл. 1, характерны для Paradox 7 и будут существенно отличаться при использовании других типов, например Microsoft Access.*