МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ МЕЖДУНАРОДНЫХ ОТНОШЕНИЙ (УНИВЕРСИТЕТ)

Кафедра информатики и математических методов

В.М. ГОРДУНОВСКИЙ, С.А. ГУТНИК, С.Ю. САМОХВАЛОВ

ВВЕДЕНИЕ В СИСТЕМЫ БАЗ ДАННЫХ

УЧЕБНОЕ ПОСОБИЕ

Под общей редакцией
В.В. Григорьева

МОСКВА – 2000

ГОРДУНОВСКИЙ Виктор Максимович, ГУТНИК Сергей Александрович, САМОХВАЛОВ Сергей Юрьевич. Введение в системы баз данных: Учебное пособие/ Под редакцией ГРИГОРЬЕВА Владимира Викторовича.
Изд. МГИМО (Университет), 2000 г., с.

В учебном пособии изложены основные понятия систем баз данных. Рассматриваются принципы организации данных, архитектура баз данных, функции систем управления базами данных и конкретные модели данных. На примерах иллюстрируются основные концепции в построении баз данных: файловые системы последовательного доступа, иерархическая, сетевая и реляционная. Подробно рассматривается реляционная модель данных и основные операции в этой модели. Приведены примеры реляционных операций с использованием языка структурированных запросов Microsoft Query. Продемонстрированы возможности работы с базами данных в Microsoft Excel.

Редактор Т.Ф.Тищенко

Оглавление

Глава 1. Понятие о базе данных

Существует понятие о базе данных, как совокупности данных, хранящихся в упорядоченном виде. Простейший пример базы данных – это каталог книг в библиотеке. Можно найти книгу по алфавитному указателю, по тематическому или специальным индексным указателям. Эти указатели позволяют управлять информацией. Среди традиционных способов упорядоченного хранения информации можно назвать словарь, энциклопедию, записную книжку.

В компьютерах данные хранятся на магнитных носителях. Поэтому, когда речь заходит о понятии компьютерной базы данных, имеют в виду совокупность структурированных данных, хранящихся в памяти компьютера и набор прикладных программ, которые обрабатывают эти данные.

В дальнейшем изложении понятие базы данных будет уточняться, а пока приведем следующее определение базы данных.

- БАЗА ДАННЫХ представляет собой совокупность взаимосвязанных, структурированных, совместно используемых управляемых данных.

Отметим, что совместное использование данных является важной характеристикой базы данных. При совместном использовании данных сотрудники различных отделов организации используют для своих собственных приложений общее множество данных. Его и называют КОРПОРАТИВНОЙ БАЗОЙ ДАННЫХ. Каждый отдел может работать только со своими данными, например: отдел маркетинга торговой компании - со своими, отдел закупок - со своими, бухгалтерия - со своими и т. д., взаимодействие этих отделов будет неэффективным, так как отдел маркетинга не сможет получить доступ к данным отдела закупок, особенно к данным по оценке товаров, что затруднит проведение маркетинговой компании. В свою очередь отдел закупок, не имея доступ к данным отдела маркетинга, не получит информации о том, довольны ли покупатели товаром. Поэтому для улучшения взаимодействия

отделов в организации требуется сведение всех разрозненных данных в единую базу данных. Такой вид объединения данных называется ИНТЕГРАЦИЕЙ ДАННЫХ.

Каждый отдел может иметь базу данных отдела, учитывающую профиль работы этого отдела.

В то же время каждый сотрудник организации может иметь свою ПЕРСОНАЛЬНУЮ БАЗУ ДАННЫХ, содержащую, например, адреса клиентов для рассылки маркетинговых материалов.

Чтобы создать свою персональную базу данных, нужно прежде всего решить, из каких элементов она будет состоять, т. е. задать её структуру. Здесь можно положиться на свой повседневный опыт. Например, если необходимо создать электронную копию своей личной записной книжки, то включим в нее фамилии, имена и отчества своих знакомых, их адреса, телефоны и т. д. Из этих элементов можно составить запись, т. е. объединить их так же, как это делается на бумаге. При объединении элементов в запись между ними устанавливается определенная связь. Наличие структуры таких связей и определяет базу данных.

База данных состоит из элементов данных и связей между ними. В базе данных много различных типов элементов данных, и поэтому необходима специальная схема, позволяющая отобразить связи между типами элементов данных. Эта схема называется МОДЕЛЬЮ ДАННЫХ. Модель данных представляет только *логические связи* между элементами данных и не имеет отношения к физическому расположению данных на магнитных носителях. Подобным образом официальная схема Московского метрополитена не имеет отношения к физическому расположению путей и станций. На ней не показаны реальные изгибы путей и реальные расстояния между станциями. Подобно схеме базы данных на ней просто представлено логическое описание связей между станциями. Эту схему можно рассматривать как модель транспортной системы Московского метрополитена. Метростроевцы могут изменить физические пути,

прокладывая дорогу над Москвой рекой, а не под ней, но при этом логическая схема не изменится.

Описание общей логической структуры базы данных называют ОБЩЕЙ МОДЕЛЬЮ ДАННЫХ. Модель представляет структуру, в которую помещаются значения элементов данных. Подобно табло в аэропорту, на котором высвечивается информация о прибытии и отправлении самолетов, модель данных не меняется, в то время как величины, помещенные в ней, время от времени изменяются.

Введем еще одно определение базы данных с точки зрения информационного моделирования.

- БАЗА ДАННЫХ – это множество взаимосвязанных структурированных данных, которые описываются какой-либо моделью данных.

1.1. Развитие технологии систем баз данных

- Под ДАННЫМИ будем понимать разрозненные факты.

  Под ИНФОРМАЦИЕЙ – организованные и обработанные данные.

В развитии технологии управления данными можно выделить несколько этапов. Вначале данные обрабатывались вручную. Впервые автоматизированная обработка информации появилась в 1890 году в США, когда Холлерит использовал технологию перфокарт и электромеханические машины для сортировки и обработки большого количества записей во время переписи населения Соединенных Штатов. Каждая перфокарта содержала двоичную запись по каждой семье. Машины сводили подсчеты в таблицы по жилым кварталам, территориальным и административным округам и штатам. Этот бизнес по производству оборудования для записи и обработки данных на перфокартах привел к возникновению в 1915 году компании International Business Machines (IBM). К середине 50-х годов у многих компаний имелись целые этажи библиотек данных на перфокартах. На других этажах размещались

системы управления данными в виде рядов электромеханических перфораторов, сортировщиков и табуляторов.

В 50-х годах появились электронные компьютеры с хранимыми программами и оборудование с магнитными лентами, каждая из которых могла хранить столько информации, сколько десятки тысяч перфокарт. Ключевым компонентом этой новой технологии являлось программное обеспечение. Оказалось сравнительно легко приспособить компьютеры для обработки данных. Появились стандартные пакеты программ для таких популярных бизнес-приложений, как общая бухгалтерия, расчет заработной платы, банковская деятельность и ведение библиотек документов.

Программное обеспечение того времени поддерживало МОДЕЛЬ ОБРАБОТКИ ЗАПИСИ на ОСНОВЕ ФАЙЛОВ. Типичные программы последовательно читали несколько входных файлов и создавали в результате новые файлы.

Применение таких систем быстро стало популярным в банковской сфере. Данные по счетам хранились на магнитных лентах, а программы выполняли пакетную обработку последовательных файлов. Раз в день записи по счетам сортировались, после чего они объединялись с хранимым на ленте основным файлом (базой данных) для производства нового основного файла. На базе этого основного файла также производился отчет, который использовался как гроссбух на следующий бизнес-день. Технология пакетной обработки файлов обладала двумя серьезными недостатками. Первый - если в записи имелась ошибка, она не распознавалась до вечерней обработки основного файла и требовалось несколько дней для ее исправления. Второй, и более важный, - нельзя было получить текущее состояние базы данных, поскольку записи обрабатывались один раз в день. Но для таких приложений, как ведение операций на фондовой бирже или резервирование билетов и пр., требуется знание текущей информации. Этим приложениям нужен немедленный доступ к текущим данным.

В 60-х годах появилась технология оперативного доступа к данным, появился термин БАЗА ДАННЫХ и было введено понятие схемы базы данных и оперативного навигационного доступа к данным. Технология магнитных лент не обеспечивала оперативный доступ, так как, для того чтобы прочитать информацию, размещенную в файле на ленте, ее нужно сначала перемотать к тому месту, где расположен этот файл, а потом последовательно прочитать в файле все записи, предшествующие нужной. Оперативный доступ к данным позволяла обеспечить технология хранения баз данных на магнитных дисках, которые предоставляли доступ к любому элементу данных за доли секунды. Программное обеспечение, позволившее обеспечить оперативный доступ к данным, было основано на ИЕРАРХИЧЕСКОЙ МОДЕЛИ БАЗ ДАННЫХ.

Рассмотрим иерархическую модель базы данных на простом примере системы резервирования авиабилетов. В системе резервирования авиабилетов используются следующие типы записей: ГОРОДА, РЕЙСЫ, ПУТЕШЕСТВИЯ, ЗАКАЗЧИКИ (рис.1.1):

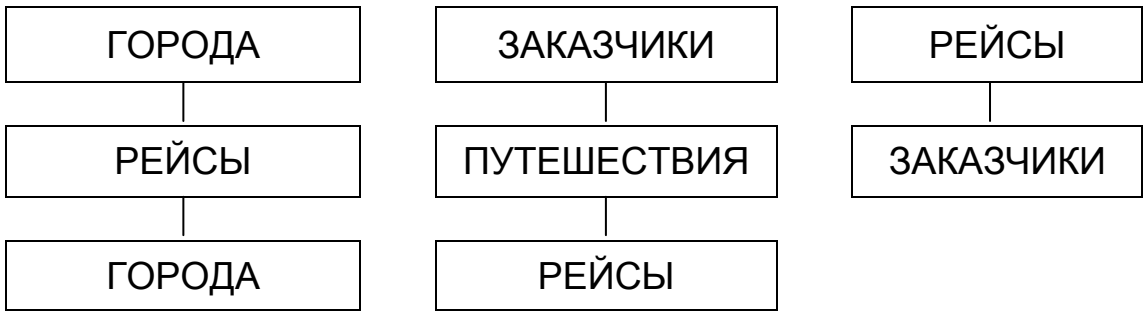


Рис. 1.1. Иерархическая модель

Рассмотрим связи между этими типами записей. ГОРОД-РЕЙСЫ – каждому городу соответствует набор отбывающих рейсов. ЗАКАЗЧИК-ПУТЕШЕСТВИЯ – каждое путешествие состоит из набора рейсов. РЕЙС-ЗАКАЗЧИКИ – каждому рейсу соответствует набор пассажиров. Эта

информация может быть представлена в виде трех иерархий наборов с записями, сгруппированными под другими записями. Каждая из трех иерархий отвечает на отдельный вопрос. Первая иерархия – это планирование рейсов в городе. Вторая иерархия дает представление о рейсах заказчика-пассажира. Третья иерархия говорит, к какому рейсу относится каждый заказчик. Программа резервирования билетов нуждается в этих трех представлениях данных. Но иерархическое представление данных обладает существенным недостатком – избыточностью. Например, при создании нового рейса или обновлении информации о нем необходимо обновить данные во всех трех местах (во всех трех иерархиях).

Для решения этих проблем информацию стали представлять в виде СЕТЕВОЙ МОДЕЛИ ДАННЫХ. Например, в системе резервирования авиабилетов каждую запись стали хранить в одном экземпляре и соотносить с набором других записей посредством связей. Так все РЕЙСЫ, используемые в ПУТЕШЕСТВИИ конкретного ЗАКАЗЧИКА, связываются с этим ПУТЕШЕСТВИЕМ, как показано на рис.1.2.

Рис. 1.2. Сетевая модель

При необходимости между записями могут создаваться новые связи.

Несмотря на успех сетевой модели данных, проектирование и программирование этих баз данных вызывали большие трудности. В 1970 году

Э.Ф. Коддом (США) была предложена РЕЛЯЦИОННАЯ МОДЕЛЬ. В реляционной модели, которая будет подробно рассмотрена ниже, данные представляются в виде таблиц-отношений. Отношение легко представить как множество записей в электронных таблицах. Названия столбцов таблицы представляют собой имена полей записей.

Перечислим современные технологии обработки информации в реляционных базах данных. Первая - задачи оперативной обработки данных, так называемые задачи OLTP (On Line Transaction Processing). Это задачи обработки текущей информации в реальном времени, например системы резервирования авиабилетов или системы обработки банковских счетов.

Другой важной задачей является автоматическое обобщение и аналитическая обработка данных в соответствии с запросами пользователей. В настоящее время в аналитических системах активно используется технология хранилищ данных - Data Warehouse. Технология хранилищ данных представляет собой набор взаимосвязанных технологий, которые позволяют создавать единый информационный ресурс масштаба организации, в рамках которого решаются задачи анализа бизнеса и поддержки принятия решений. Технология хранилищ данных предлагает архитектуру по организации информационных систем, универсально применимую к предприятиям любых масштабов - от малых до масштаба отрасли. Существует ряд критических факторов, определивших появление набора технологий хранилищ данных:

- Сбор разрозненных и разнотипных данных.
- Использование и управление большими и сверхбольшими базами данных и необходимость в нестандартных сложных запросах к базе данных.
- Потребность в непротиворечивой и достоверной информации.
- Применение к корпоративным базам данных методов математического анализа.

- И в конечном счете осознание того, что информация – это ресурс, подлежащий планомерной аккумуляции и переработке с целью получения новых знаний.

В рамках технологии хранилищ данных развиваются или нашли свое применение такие направления современных информационных технологий и методов обработки данных, как OLAP (On Line Analytical Processing) - интерактивная аналитическая обработка, DSS (Decision Support System) - системы поддержки принятия решений, EIS (Executive Informatin System) - информационная система руководителя, Data Mining and Knowledge Discovery in Data Bases - интеллектуальный анализ данных, Data Modeling - моделирование данных, Data Cleaning - очистка данных, ERP (Enterprise Resourse Planning) - планирование ресурсов предприятия и пр.

Реляционные базы данных продолжают совершенствоваться и сегодня, предоставляя пользователям возможность решать все более сложные задачи. Важное новшество – переход организаций к работе с реляционными базами данных по технологии клиент-сервер.

Появление в 1981 году IBM PC сделало персональный компьютер неотъемлемой частью офисного оборудования. Программы обработки текстов, работы с электронными таблицами и многие другие сами по себе оправдывали использование персонального компьютера. Кроме того, было вполне естественно связывать компьютеры в сеть, чтобы пользователи могли общаться по электронной почте и работать с общими ресурсами, такими, как принтеры, диски. Вначале серверы были созданы для управления печатью и доступом к файлам. Это были серверы печати и файловые серверы. Например, в ответ на запрос клиента на доступ к конкретному файлу, файловый сервер пересылал этот файл через сеть на клиентский компьютер. Сегодня же большинство серверов составляют серверы баз данных – программы, которые запускаются на серверной машине и обслуживают доступ клиентов к базе данных. Например, клиент запускает прикладную программу, и ему требуется запросить базу

данных. Для этого он обращается к серверу за нужными ему данными, сервер выполняет запрос и возвращает результат клиенту. Прикладная программа может также посылать данные на сервер с требованием обновить базу данных. Сервер вносит необходимые изменения. Система клиент-сервер – это локальная сеть, состоящая из клиентских компьютеров, которые обслуживает компьютер-сервер. В основе производительности системы клиент-сервер лежит принцип разделения труда. Клиент – это та машина, с которой работает пользователь. Управление клиентом происходит с использования графического пользовательского интерфейса (GUI – Graphical User Interface). На клиентской части производятся вычисления и другая работа, необходимая непосредственно конечному пользователю. Сервер выполняет работу, общую для нескольких клиентов: доступ к базе данных, обновление базы данных и т. д.

В концептуальном плане принцип клиент-сервер – это часть открытой системы, объединяющей все те способы, с использованием которых можно связать и заставить согласованно работать на благо пользователя компьютеры, операционные системы, сетевые протоколы, сетевое оборудование и программное обеспечение. Однако на практике заставить вместе работать разнообразные операционные системы, сетевые протоколы, базы данных и т. д. достаточно сложно. Цель принципа открытых систем – добиться возможности взаимодействия (совместимости), когда две или более различных системы обмениваются информацией, и каждая из них вносит свой вклад в решение общей задачи.

В некотором смысле технология клиент-сервер – наиболее яркое воплощение сочетания распределенной обработки данных с централизованным управлением и доступом к данным.

Другое важное новшество – объектно-ориентированные мультимедийные базы данных, которые хранят более сложные типы данных, - документы, включающие графические, звуковые и видео образы. Эти системы нового

поколения представляют собой базовые средства хранения для приложений Internet и Intranet.

Развитие объектно-ориентированного программирования (ООП) привело к разработке объектно-ориентированных систем управления базами данных, которые позволяют обрабатывать сложные объекты, содержат наследование и другие свойства, что делает возможным прямую реализацию объектно-ориентированных концептуальных моделей. В то же время реляционная технология баз данных расширяется, сочетая возможности управления данными с применением логических правил, обеспечивающих управление более сложной информацией. Такие системы называются БАЗАМИ ЗНАНИЙ (БЗ).

В базах знаний содержится информация об объектах, существующих в реальном мире. Этими объектами могут быть люди, книги в библиотеках, счета или квитанции, географические карты, средства на банковских счетах. Эти объекты являются материальными. Какова бы ни была природа объекта, он должен иметь физическое представление, даже если это только изображение на дисплее компьютера, которое никогда не появится на принтере.

Материальные объекты обладают свойствами с присущим только им поведением, а у объектов ООП имеются свойства и методы. Объекты баз данных обладают свойствами и поведением.

Свойства определяют хранимую объектом информацию. Например, книги имеют унифицированные коды. Эти коды, равно как и название, автор, количество страниц, являются свойствами книги. Это статические свойства книги: они не зависят от того, находится ли книга на полках библиотеки или взята читателем. Информация о банковском счете, такая, как номер счета, имя и адрес клиента, также статическая, хотя клиенты иногда меняют адреса. Имя читателя, имеющего книгу на руках, и бухгалтерские балансы представляют собой динамические свойства: они меняются ежемесячно или ежедневно.

Чтобы отличить один элемент данных от другого, каждый из них должен быть уникальным. Например, банковский идентификационный номер, номер

клиента и номер чека однозначно определяют информацию о вкладе. Сумма на счете или сумма, указанная в чеке, используются для подведения баланса. Данная операция заключается в простом вычитании или сложении, в том, что обычно называют транзакцией.

На подведение баланса путем обработки всех транзакций с момента открытия счета требуется много времени. Однако можно добавить в элемент данных для клиента производное статическое свойство, в котором хранится сумма последнего баланса. Значение данного свойства вычисляется ежемесячно. Для определения текущего баланса (вычисляемого динамического свойства) требуется обработать только те транзакции, которые были произведены после подведения последнего баланса.

Поведение объектов базы данных определяют характеристики транзакций, в которых участвуют элементы данных объектов. Поведение объекта, связанного с банковским счетом, описывается очень просто: при открытии счета, размещении вклада и начислении процентов баланс увеличивается. При выписывании чеков и получении наличных баланс уменьшается. Снятие денег с банковского счета или размещение вкладов представляет собой пример транзакции. Транзакции происходят в ответ на события, например на размещение вклада или получение денежных средств. В объектной модели транзакции реализуются с помощью методов – процедур, которые представляют собой отклик на события, инициируемые пользователем, например открытие формы или нажатие кнопки.

В программах, написанных на языке системы управления базами данных, используются методы баз данных. Объекты данных обладают свойствами и методами. Файлы объектно-реляционных баз данных и таблицы, чье содержимое согласуется с парадигмой объекта, рассматриваются в смысле модели в объектно-ориентированного представлении.

По мере развития коммерческих компьютерных систем произошел переход от ОБРАБОТКИ ДАННЫХ к ОБРАБОТКЕ ИНФОРМАЦИИ. Это изменение

отражает то, что информация представляет не просто набор деловых записей, а организованные и обработанные данные. Постепенно пришло понимание ценности информации и огромного потенциала компьютерных систем в поддержании информационного ресурса и управления им, что и привело к появлению ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ. Такие системы используют уже содержащиеся в компьютере данные, давая ответы на широкий круг управленческих вопросов.

Сегодняшние технологии, такие, как объектно-ориентированные базы данных, технологии клиент-сервер, Web-технологии и системы электронной коммерции E-commerce в Internet, решают сложнейшие задачи и приведут к появлению в будущем более мощных систем.

1.2. Функциональная структура базы данных

Почему организации выгодно хранить данные в базе данных? Наиболее общий ответ состоит в том, что база данных обеспечивает централизованное управление данными всей организации. Это предполагает, что в организации, имеющей систему баз данных, имеется вычислительный центр по управлению данными, есть сотрудник – администратор базы данных, который отвечает за работу с данными. На практике администратором базы данных может быть даже группа специалистов. Отметим, что роль администратора базы данных в организации очень важна. Фактически администратор базы данных рассматривается как часть системы управления базы данных.

Таким образом, система баз данных представляет собой не только совокупность данных, программного обеспечения и оборудование, но и персонал.

Типичной ситуацией нецентрализованного управления данными в организации является случай создания файлов для каждого отдельного отдела, размещаемых на собственных магнитных дисках, в результате чего данные

оказываются разрозненными. Ситуация усложнится, если не предпринять усилий по систематическому управлению данными. Например, у отдела маркетинга могут быть свои файлы данных, у отдела закупок свои, у бухгалтерии свои и т. д., т. е. каждый отдел работает только со своими данными (рис.1.3.).

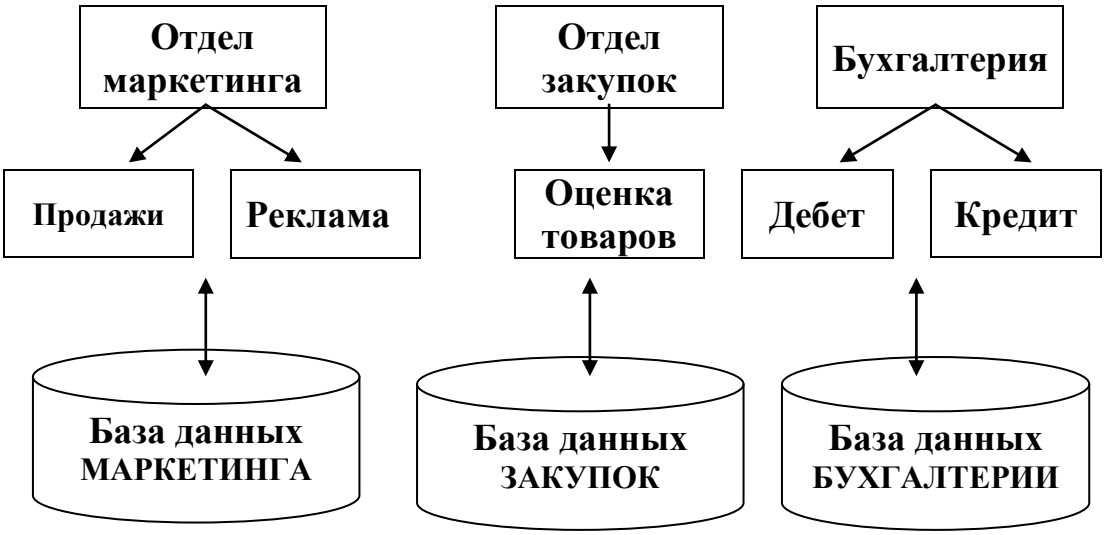


Рис. 1.3. Пример нецентрализованного управления данными

Но база данных определялась как совокупность структурированных, взаимосвязанных, совместно используемых и управляемых данных. Их совместное использование и управление достигаются средствами интеграции данных. Таким образом, определение содержит три критерия эффективности базы данных.

Во-первых, данные должны находиться в совместном использовании. Ранее упоминалось, что данные могут совместно использоваться различными отделами организации.

Во-вторых, данными нужно управлять. Управление обеспечивается системой управления базой данных, средствами которой управляют администраторы баз данных.

В - третьих, данные должны быть таким образом интегрированы, чтобы исключить избыточность и противоречивость.

Можно обеспечить минимально необходимую (например только для обеспечения требуемой производительности системы) избыточность (дублирование) хранимых данных. При использовании несколькими программами одинаковых данных такие данные интегрируют и хранят в единственном экземпляре. Прямым следствием дублирования будет являться рассогласование базы данных. Например, некоторый товар может быть отправлен из магазина или списан с соответствующим изменением в файле данных магазина, а финансовый отдел из-за отсутствия централизованной информационной системы может ничего не знать об этом и периодически запрашивать отчеты об этом товаре от различных подчиненных отделов.

Следствием устранения избыточности данных является устранение возможности возникновения противоречивости одних и тех же данных в различных приложениях. Действительно, поскольку устраняется возможность хранения одного и того же данного в различных записях, устраняется ситуация, когда при фактическом изменении значения данного оно окажется измененным не во всех записях.

Кроме того, технология баз данных обеспечивает возможность организации санкционированного доступа к данным. Интеграция данных приводит к тому, что данные, используемые различными пользователями, могут пересекаться различным образом. В этих условиях особенно важно наличие механизма защиты данных от несанкционированного доступа, т.е. доступ к определенным группам данных должен разрешаться только пользователям с соответствующими полномочиями.

Рассматривая данные как один из ресурсов информационной системы, можно сказать, что база данных управляет этим ресурсом в рамках всей системы. Наличие централизованного управления данными – главная отличительная черта баз данных.

1.2. Основные компоненты базы данных

Главная компонента базы данных – это СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ. СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ (СУБД) – это набор программных средств, которые предоставляют пользователям доступ к информации в БАЗЕ ДАННЫХ (БД). Как следует из названия, СУБД предназначена для того, чтобы обеспечить управление базой данных. Программная часть СУБД выступает в качестве интерфейса между пользователем и БД (рис.1.4). СУБД представляет собой программы, которые обеспечивают всю работу с базой данных: создание, загрузку, запросы и обновление данных. СУБД также контролирует все действия, связанные с управлением, вводом-выводом и памятью БД, на нее также возлагается решение проблем безопасности и совместного использования данных.

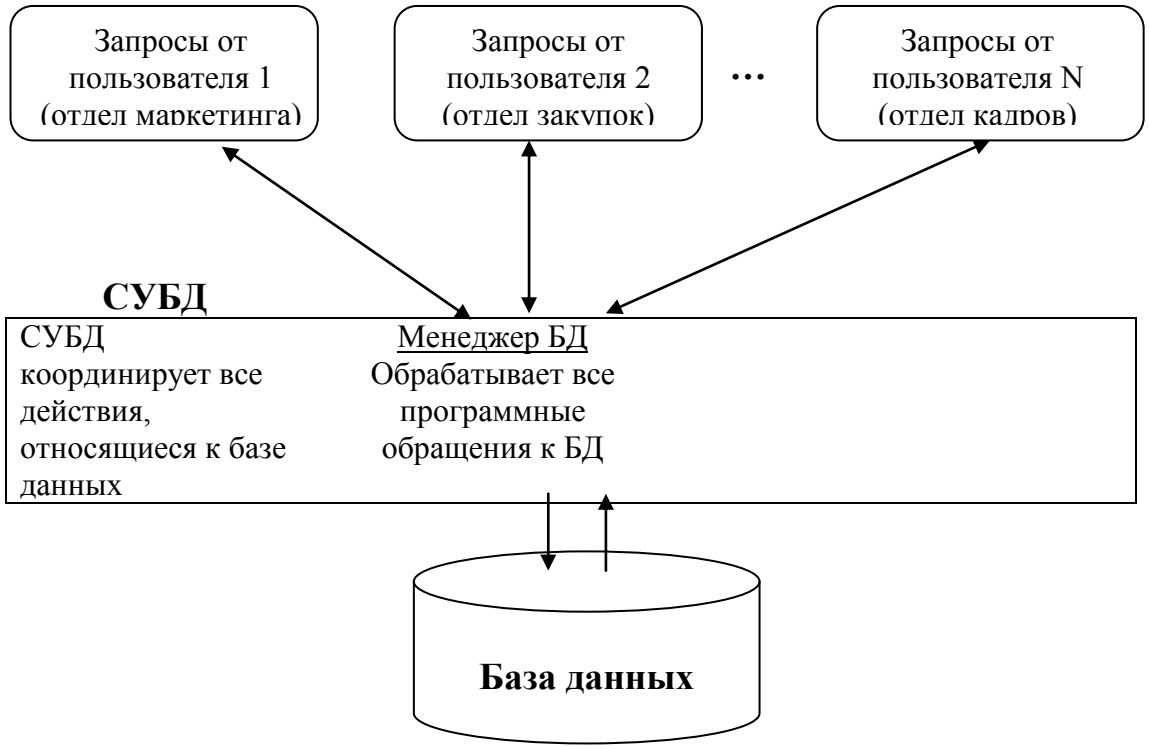


Рис. 1.4. Основные компоненты архитектуры СУБД

Для того чтобы создать БД, нужно сначала определить, какие данные в ней будут храниться и какого они типа, т. е. будут ли это числа, над которыми предстоит выполнять арифметические действия, текстовая информация или какая-то другая, например графическая или видео. Кроме того, следует задать каждому элементу информации конкретную длину, ограничив ее конкретными пределами.

Для описания всех этих свойств СУБД обычно использует свой собственный ЯЗЫК ОПИСАНИЯ ДАННЫХ (ЯОД) и, после того как данные описаны, принимает решение об их размещении в памяти.

После того как закончено описание всех элементов будущей БД, начинается формирование ее структуры, т. е. определяются связи между элементами. Подробно остановимся на этом ниже, а сейчас отметим, что обычно СУБД поддерживает структуру лишь какого-то определенного типа, например, иерархическую, сетевую или реляционную. Все описания заносятся в память компьютера, и система обращается к ним по мере надобности.

Другая задача, решаемая СУБД, – обеспечение ввода данных или, как говорят, загрузка базы данных. На этом этапе СУБД обычно осуществляет контроль правильности вводимой информации. Это может быть, например, проверка данных по типу. Так, если вместо числа будут вводиться буквы, программа откажется пересылать их в память и выдаст на экран монитора соответствующее сообщение. Такие процедуры помогают при загрузке базы данных выявить значительную часть неизбежных при вводе ошибок. Размещением вводимой информации в памяти компьютера также управляет СУБД, используя описание элементов и схемы БД. На этом процесс создания БД можно считать законченным.

Перечислим теперь задачи пользователя БД. Прежде всего это задача обновления БД – замена устаревших данных новыми и добавление свежей информации. Эту задачу актуализации БД также позволяет решать система управления базой данных.

Затем пользователь стремится выбрать из всех хранимых данных только те, которые ему необходимы в данный момент. Свой запрос он обращает к СУБД. Для этого можно использовать либо специальный язык запросов, который близок к естественному языку, либо использовать специальное меню запросов.

Если базой данных одновременно могут пользоваться несколько человек, то СУБД должна позаботиться о том, чтобы они не мешали друг другу. Иначе может возникнуть, например, такая ситуация, когда один пользователь хочет получить из БД какой-то элемент данных, а другой в это же самое время начинает этот элемент менять.

Современные СУБД могут разрешать такие конфликтные ситуации. Кроме того, в СУБД предусмотрены возможности сохранения основной информации при неожиданном отключении питания или машинном сбое. Этот круг задач называется ОБЕСПЕЧЕНИЕМ ЦЕЛОСТНОСТИ БАЗЫ ДАННЫХ.

Данные, извлеченные из БД, обычно подвергаются какой-либо обработке. Одним из часто используемых видов такой обработки – сортировка (числа по возрастанию или убыванию, а символьные строки - в алфавитном порядке). Нередко приходится производить объединение элементов или целых блоков. Например, несколько названий и номеров можно объединить в полный адрес. СУБД обеспечивает и чисто математическую обработку, такую, как подсчет суммы или среднего значения нескольких чисел.

И наконец, полученную информацию обычно нужно оформить таким образом, чтобы она была представлена в доступной и наглядной форме. Это может быть, например, таблица, снабженная заголовками с поясняющими надписями, или график, диаграмма, или гистограмма с цифровыми данными по осям. Такие процедуры выполняет генератор отчетов, также входящий в состав СУБД.

1.4. Модели данных

МОДЕЛЬ ДАННЫХ вводится для обеспечения независимости прикладных программ от данных. В файловых системах при изменении структуры файлов приходилось переписывать прикладные программы, предназначенные для их обработки.

Модель отражает для пользователей информационное содержание БД, но подробности организации физического хранения данных в ней отсутствуют. Каждая модель имеет свою схему, в которой отражена структура ее данных, имена записей, имена и форматы полей. Для работы с данными модели разрабатывается конкретный язык манипулирования данными. Запросы к данным из БД выражаются в прикладных программах пользователей с помощью этих языковых терминов, принятых моделью данных.

При обработке информации человек начинает с формирования понятий об интересующих его фактах, явлениях, предметах и событиях. Для обозначения прообраза понятия любой природы используется термин СУЩНОСТЬ. Сущностью может быть мысленный образ или множество однотипных предметов реального мира (например, множество служащих, домов). Сущность в свою очередь характеризуется своими основными свойствами. Примерами свойств сущности СЛУЖАЩИЙ являются фамилия, имя, отчество, возраст, номер страхового свидетельства государственного пенсионного страхования, зарплата и т. п. В информационном моделировании сфера понятий заменяется сферой информационных представлений. Это приводит к описанию логического представления в терминах структуры данных информационной модели. В информационной модели сущность представляется ТИПОМ ЗАПИСИ. Следует отметить общее использование термина тип записи при информационном представлении сущности, хотя тип записи в различных моделях может играть различную роль. Например, в реляционной модели типу записи соответствует отношение, в иерархической модели –

сегмент или узел, а в сетевой модели записи-владельцы или записи-члены. Свойства, характеризующие сущность, называются АТРИБУТАМИ. Физическая структура базы данных, соответствующая её логическому представлению, описывает данные, хранящиеся в памяти компьютера.

На СУБД возлагается задача реализации отображения (прямого и обратного) модели данных на физическую базу данных. На рис.1.5. и 1.6. показаны два варианта архитектуры СУБД. На рис.1.5. изображена двухуровневая архитектура СУБД ранних поколений. Такая СУБД поддерживает единственную предлагаемую пользователям внешнюю модель данных. Внешний уровень наиболее близок к пользователям, т. е. связан с тем, как отдельные пользователи представляют себе эти данные. Например, отдел маркетинга будет в первую очередь интересовать данные по спросу и оценке товаров, отдел кадров интересуют данные по персоналу и т. д. Отображение логической структуры внешней модели в соответствующие структуры физической базы данных называют внутренней схемой. Внутренний уровень наиболее близок к физической памяти, т. е. связан со способом фактического хранения данных на магнитных дисках.

Рис.1.5. Двухуровневая архитектура СУБД

На рис.1.6. представлена обобщенная трехуровневая архитектура СУБД. В этом варианте одной концептуальной модели соответствует несколько внешних моделей.

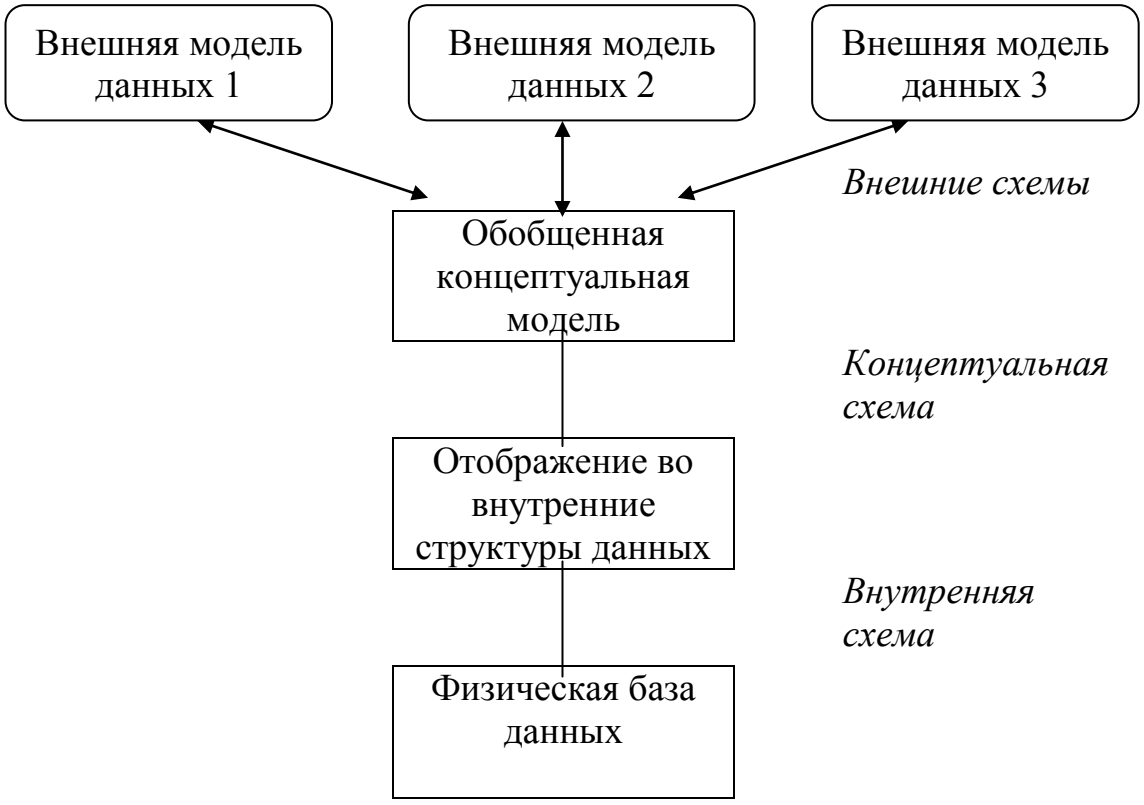


Рис.1.6. Трехуровневая архитектура СУБД

Внешняя модель является информационным содержанием базы данных в том виде, в каком его представляет конкретный пользователь. Например, пользователь из отдела кадров может рассматривать базу данных как совокупность записей об отделах и о служащих (и он может совсем не знать о записях поставщиков и товаров, с которыми имеет дело пользователь из отдела закупок).

КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ, или просто МОДЕЛЬ ДАННЫХ, есть представление полного информационного содержания базы данных в несколько абстрактной форме. Концептуальная модель определяется посредством концептуальной схемы, которая включает определения каждого типа записей.

Например, она может состоять из совокупности экземпляров записей об отделах, плюс совокупность экземпляров записей о служащих, плюс совокупность экземпляров записей о поставщиках, плюс совокупность экземпляров записей о товарах и т. д. Таким образом, концептуальная модель есть представление общего содержания базы данных, а концептуальная схема - определение этого представления.

Итак, для базы данных имеется одна внутренняя схема, описывающая внутреннюю модель данных, одна концептуальная схема, описывающая концептуальную модель данных, и столько внешних схем, сколько требуется, чтобы описать различные внешние модели данных, подлежащие реализации.

Рассмотренный трехуровневый подход к построению баз данных, включающий внешний, концептуальный и внутренний уровни представления данных, в настоящее время получил наибольшее распространение.

При такой архитектуре база данных обладает высокой способностью адаптации к возможным изменениям как в прикладных программах, так и в самих данных: любые изменения внешних схем и внутренней схемы изолированы друг от друга концептуальной схемой и могут выполняться независимо. Концептуальный уровень должен обеспечивать долговременную работу всей системы. Мотивировкой введения внутреннего уровня является обеспечение требований производительности, экономичного использования ресурсов вычислительной системы и относительной независимости системы от используемых технических средств.

Кроме названных трех уровней абстрагирования в базе данных существует еще один, им предшествующий. Модель этого уровня должна выражать информацию о предметной области в виде, независимом от конкретной используемой СУБД. С этой моделью предметной области работает администратор базы данных и пользователи системы.

Переход от одного уровня абстракции в представлении данных к другому составляет в общем случае процесс проектирования базы данных.

Этот процесс представляют последовательностью достаточно простых, обычно итеративных процессов проектирования отображений между промежуточными моделями данных.

1.5. Основные определения моделей данных

Пусть имеются множества A и B. Отношение A **R** B указывает на связь между отдельными элементами этих множеств. Различают рефлексивные отношения A **R** A (связи между элементами одного и того же множества), транзитивные (опосредованные связи) и т. д. На практике используется некоторая интерпретация связей между множествами и кардинальных чисел этих связей (т. е. числа элементов в экземпляре связи). Множества могут соответствовать атрибутам или типам записей. Связи могут быть функциональными, т. е. удовлетворяющими определению математической функции. Кардинальные числа связей используются также для определения типа отображения между парами множеств. Существуют отображения один-к-одному (1:1), один-ко-многим (1: M) и многие-ко-многим (M:N).

Таким образом, СВЯЗЬ – это соответствие или отображение между элементами двух (или более) множеств. Поясним это понятие на ряде примеров. На рис.1.7. приведены некоторые примеры сущностей, их свойств и соответствующего представления с помощью типов записей и атрибутов.

В этих примерах имеются разные типы множеств. Один из них – это множества аналогичных сущностей, такие, как все люди, работающие в организации и называемые СЛУЖАЩИЕ. Аналогично ДОМА, ПОСТАВЩИКИ, ДЕТАЛИ представляют собой множества сходных сущностей. Каждая сущность в свою очередь представляется своими свойствами. В результате каждому множеству сущностей здесь соответствует несколько множеств, содержащих значения соответствующих свойств. Иначе говоря, каждое множество сущностей представляется типом записи, а тип записи характеризуется

соответствующими атрибутами. Экземпляр какого-либо типа записи соответствует единичному представителю сущности, такому как отдельный служащий, дом, поставщик или деталь (так же как в файловой системе: запись файла служащих соответствует отдельному лицу).

Область понятий		**Область информационных моделей**
Сущности	**Свойства**	**Типы записей (список атрибутов)**
Служащие	Фамилия, имя, отчество, номер страхового свидетельства, номер служащего, зарплата	Служащие (ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, НОМЕР_СТРАХОВОГО_ СВИДЕТЕЛЬСТВА, НОМЕР_СЛУЖАЩЕГО, ЗАРПЛАТА)
Дома	Номер участка, тип, стоимость	Дома (НОМЕР_УЧАСТКА, ТИП, СТОИМОСТЬ)
Поставщики	Номер поставщика, имя, город	Поставщик (НОМЕР_ ПОСТАВЩИКА, ИМЯ_ ПОСТАВЩИКА, ГОРОД)
Детали	Номер детали, описание детали, наличие деталей	Детали (НОМЕР_ДЕТАЛИ, ОПИСАНИЕ_ДЕТАЛИ, НАЛИЧИЕ_ДЕТАЛИ)

Рис.1.7. Примеры сущностей и их свойств

Приведем примеры некоторых связей:

а) каждому номеру служащего соответствует единственный номер страхового свидетельства и наоборот;

б) служащие могут иметь дома;

в) некоторые служащие имеют подчиненных;

г) поставщики поставляют (продают) детали;

д) тип дома, в котором живет служащий, можно определить по его стоимости.

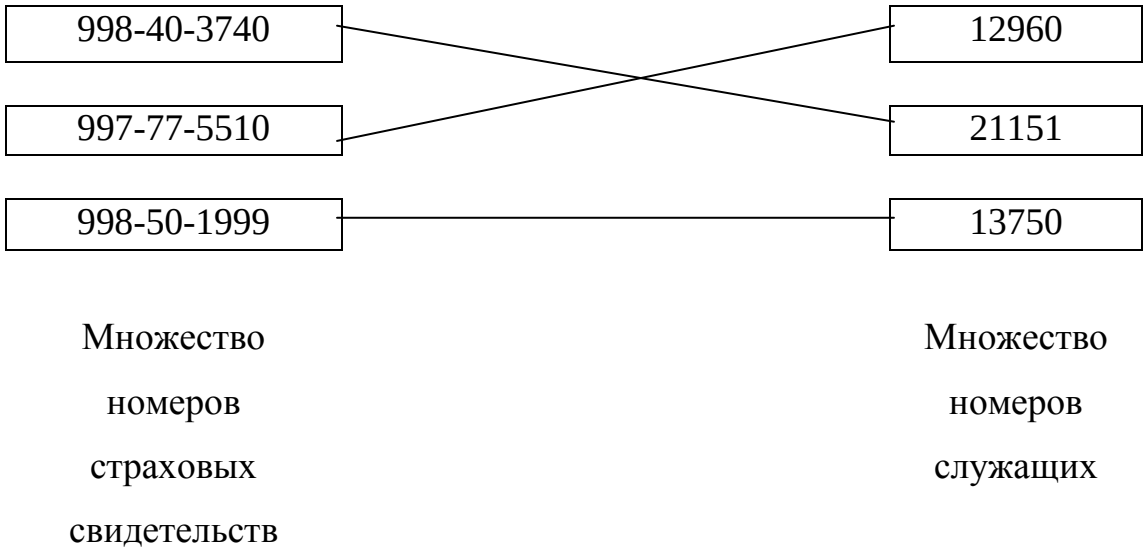


Рис.1.8. Связи между множествами сущностей – соответствие значений

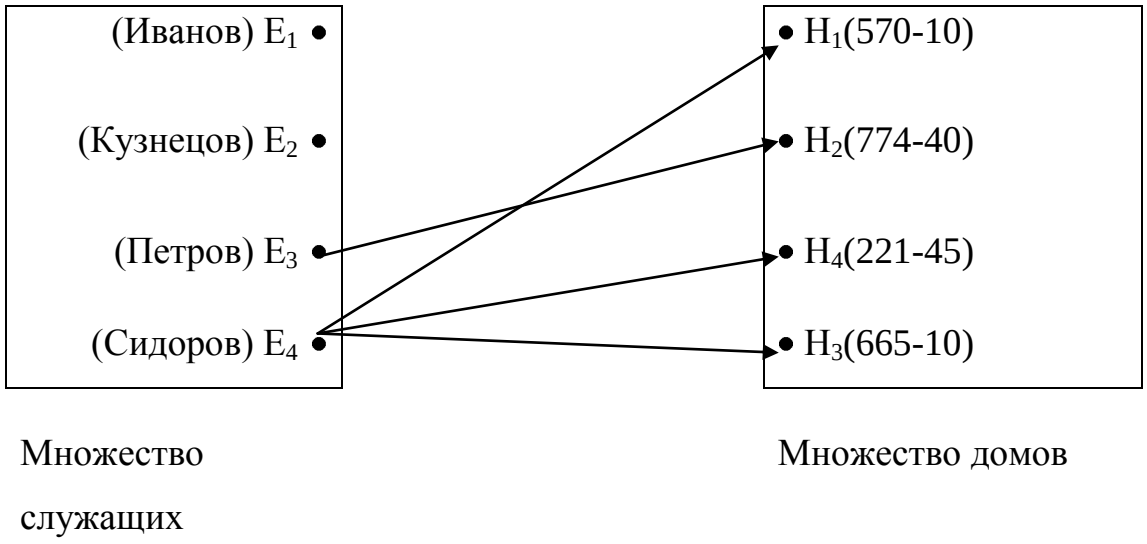
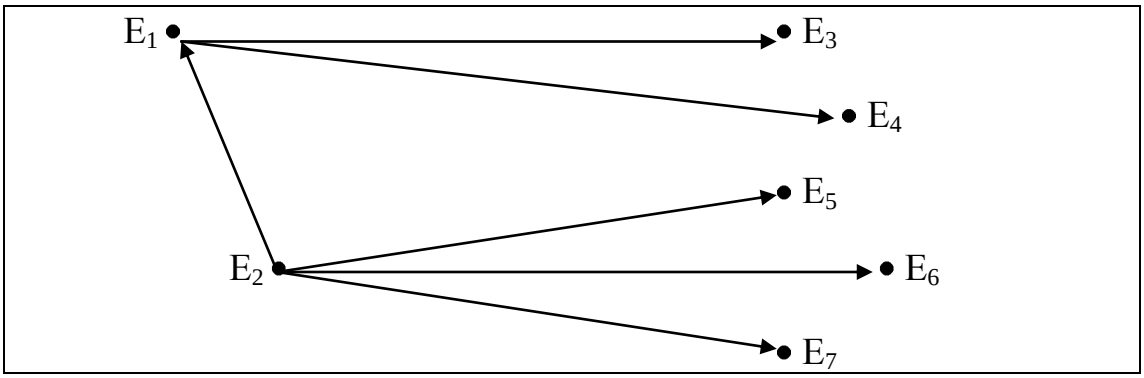
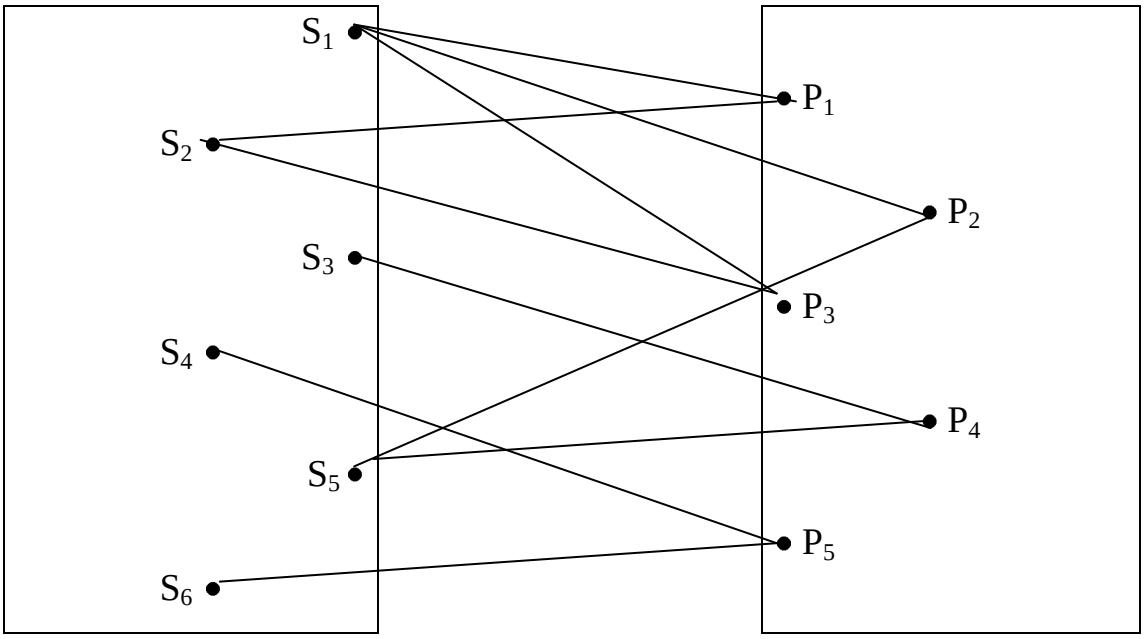


Рис.1.9. Связи между множествами сущностей – связь «владеет»



Множество служащих

Рис.1.10. Связи между множествами сущностей – связь «руководит»



Множество поставщиков

Множество деталей

Рис.1.11. Связи между множествами сущностей – связь «поставляет»

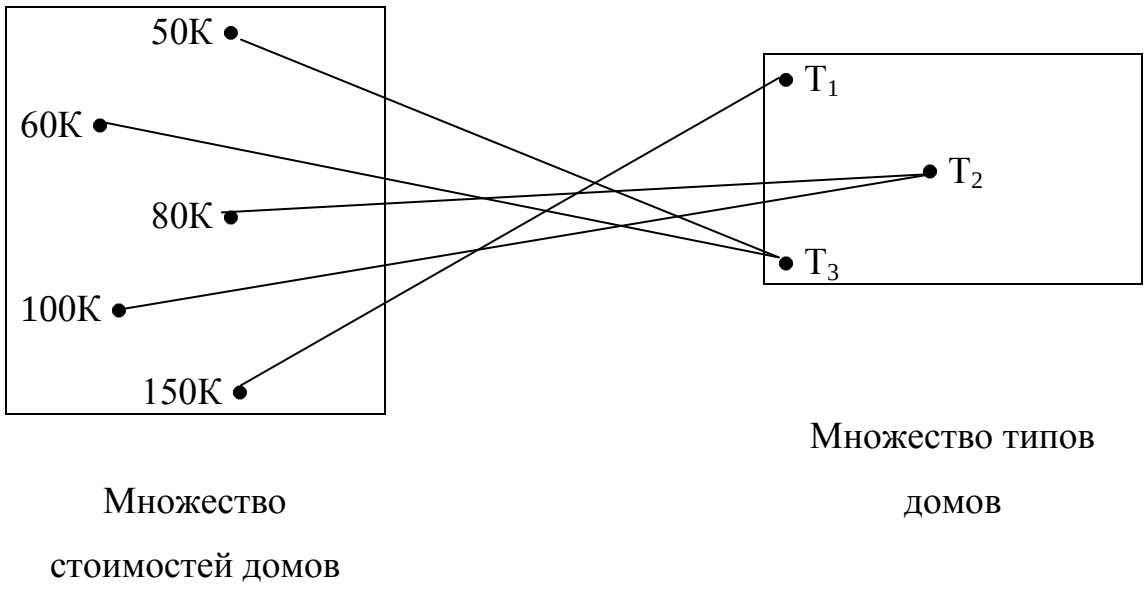


Рис.1.12. Связь между типами домов и их стоимостью

Дальнейшие рассуждения строятся на примере этих связей. На рис.1.8-1.12 показаны различные связи между экземплярами разных множеств в порядке соответствующем пунктам а)-д) приведенного выше списка.

Исследуя типы связей на рис.1.8-1.12, можно сделать следующие замечания:

1. На рис.1.8 и 1.12 представлены связи между множествами атрибутов, называемые также *межатрибутными* связями. Эти связи имеют внутрисущностный тип.
2. На рис.1.9, 1.10, и 1.11 представлены связи между сущностями (*межсущностные* связи). Возможны связи, охватывающие более двух сущностей , как в примере «поставщики поставляют детали для проектов и проекты используют детали». Связи между сущностями называются также «ассоциациями». Особый случай связей между сущностями представлен на рис.1.10, где изображен пример связи «руководит», включающий

единственный тип сущности, т.е. связь между элементами одного и того же множества сущностей.

3. На рис.1.8 приведен пример отображения 1:1 между соответствующими множествами. Это пример взаимно-однозначного отображения.
4. На рис.1.9, показано, что не все служащие владеют домами, а некоторые владельцы могут иметь более одного дома.
5. Из рис.1.10 видно, что некоторые служащие могут руководить другими служащими и что каждый служащий имеет руководителя (у E_2 есть подчиненный E_1, который также является руководителем). Исключение составляет президент компании, руководящий сам собой или руководимый советом директоров.
6. Из рис.1.11 видно, что все поставщики являются «активными», т. е. все они поставляют детали. При этом каждый поставщик может поставлять несколько деталей, а некоторые детали могут поставляться несколькими поставщиками.
7. На рис.1.12 показано, что стоимость дома определяет его тип.

На рис.1.13 приведено изображение межсущностной связи (рис.1.9), использующее диаграмму сущность-связь (Entity/Relationship-E/R).

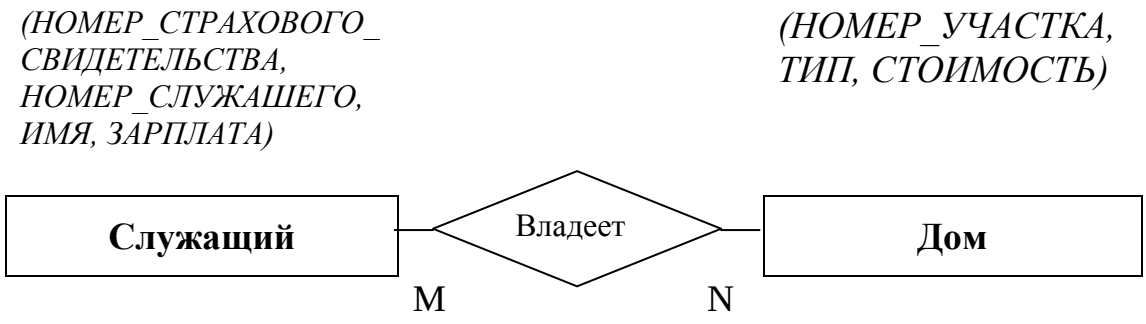


Рис.1.13. Связь «владеет» в терминах модели сущность-связь.

1.6. Предметная область баз данных

ПРЕДМЕТНАЯ ОБЛАСТЬ базы данных– это область применения конкретной базы данных. Различают базы данных, применяемые в управлении предприятиями, транспортом, в медицине, научных исследованиях, образовании и т.д.. Базы данных также различаются по масштабу применения в информационных системах – информационные системы верхнего уровня – уровень министерств, отраслей, крупных корпораций, средних и мелких компаний, базы данных для персонального применения.

Необходимо различать термины ДОКУМЕНТАЛЬНАЯ БАЗА ДАННЫХ - совокупность произвольных текстовых документов, и ФАКТОГРАФИЧЕСКАЯ БАЗА ДАННЫХ - множество сведений, хранящихся в информационной системе и удовлетворяющих фиксированной совокупности форматов. СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ (СУБД) - это средства программного обеспечения, необходимые для использования фактографических баз данных.

Глава 2. Методы проектирования систем баз данных

Рассмотрим пример информационной системы, на котором проиллюстрируем технологии, применяемые в проектировании баз данных.

2.1. Пример базы данных

В качестве примера выберем информационную систему дистрибьюторской компании (ДК), которая продает свыше 1000 видов товаров более 100 производителей во всем мире. У фирмы есть около 10 представительств. На рис.2.1 представлены структура фирмы и связи с ее поставщиками и клиентами.

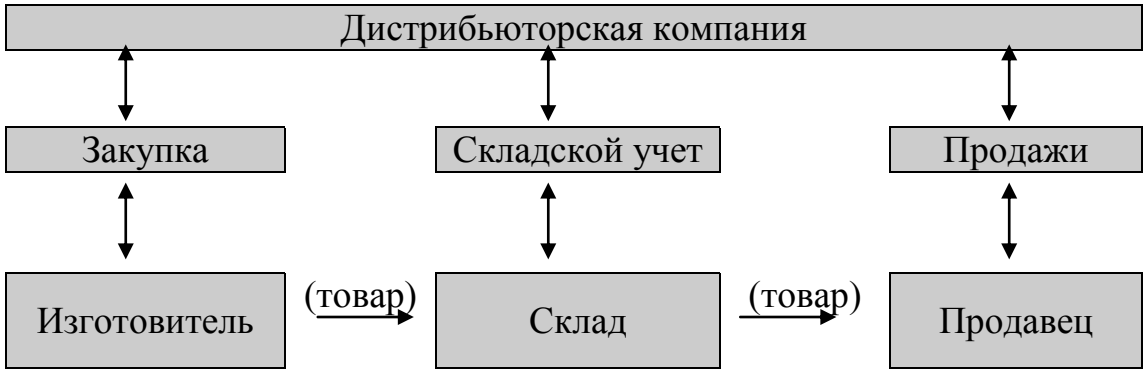


Рис.2.1 Структурная схема товарных потоков

На рис. 2.2 и 2.3 представлены образцы данных файловой системы дистрибьюторской компании. Каждая таблица представляет один файл системы. Каждая строка соответствует одной записи в файле. Имя ключа записи выделено подчеркиванием.

ЗАКАЗЧИК

ИНДЕКС_ ЗАКАЗЧИКА	ИМЯ_ ЗАКАЗЧИКА	АДРЕС	СТРАНА	НАЧАЛЬНЫЙ_ БАЛАНС	ПЛАТЕЖИ_ МЕСЯЦ
100	Кузнецов	А/я 100	Россия	45551	40113
101	Шмидт	А/я 101	Австрия	75314	65200
105	Смит	А/я 105	США	49333	49811
110	Ката	А/я 110	Япония	27400	28414

ТОРГОВЫЙ ПРЕДСТАВИТЕЛЬ

ИНДЕКС_ ТОРГОВОГО_ ПРЕДСТАВИТЕЛЯ	ИМЯ_ ТОРГОВОГО_ ПРЕДСТАВИТЕЛЯ	ИНДЕКС_ МЕНЕДЖЕРА	ОФИС	КОММ_%
10	Доу Джонс	27	Чикаго	10
14	Мацума	44	Токио	11
23	Гутенберг	35	Лейпциг	15
37	Кузнецов	12	Москва	13
39	Акай	44	Токио	10

ТОВАР

ИНДЕКС_ ТОВАРА	ОПИСАНИЕ_ ТОВАРА	ИНДЕКС_ ИЗГОТОВИТЕЛЯ	ЗАКУПОЧНАЯ_ ЦЕНА	ЦЕНА_ ПРОДАЖИ
1035	Принтер	210	110.25	220.00
2241	Монитор	317	220.25	330.25
2518	Клавиатура	253	15.60	21.20

Рис.2.2. Примеры данных из информационной системы дистрибьюторской компании

ПРОДАЖИ

ДАТА	ИМЯ_ ЗАКАЗЧИКА	ИНДЕКС_ ТОРГОВОГО_ ПРЕДСТАВИТЕЛЯ	ИНДЕКС_ ИЗГОТОВИТЕЛЯ	ЗАКУПОЧНАЯ_ ЦЕНА	ЦЕНА_ ПРОДАЖИ
08.02	100	14	2241	200	6650
12.02	101	23	2518	300	6360
12.02	101	23	1035	150	3300
19.02	100	39	2518	200	4240
22.02	101	23	1035	200	4400
25.02	105	10	2241	100	3325
25.02	110	37	2518	150	3180

ИЗГОТОВИТЕЛЬ

ИНДЕКС_ ИЗГОТОВИТЕЛЯ	ИМЯ_ИЗГОТОВИТЕЛЯ	АДРЕС	СТРАНА
210	Принт	Сан-Франциско	США
253	Клавит	Шанхай	Китай
317	Сант	Токио	Япония

Рис.2.3. Примеры данных из информационной системы дистрибьюторской компании

Так, файл ТОВАР содержит три записи. Каждая из этих записей относится к отдельному товару. Элементарные группы данных или поля ТОВАР таковы: ИНДЕКС_ТОВАРА, ОПИСАНИЕ_ТОВАРА, ИНДЕКС_ИЗГОТОВИТЕЛЯ, ЗАКУПОЧНАЯ_ЦЕНА и ЦЕНА_ПРОДАЖИ.

2.2. Файловые системы баз данных последовательного доступа

Первые коммерческие компьютерные системы использовались в основном для ведения бухгалтерских операций в повседневной работе компании. Затраты ручного труда на ведение ведомостей по заработной плате или выписывание счетов были столь велики, что автоматизация таких работ могла бы быстро окупиться. Поскольку эти системы выполняли обычные функции работы с документами, они были названы системами обработки данных. По аналогии с теми операциями, которые прежде выполнялись вручную, компьютерные файлы соответствовали папкам для бумаг, и компьютерный файл содержал ту информацию, которая вполне могла бы лежать в обычной папке.

Файлы допускают лишь ПОСЛЕДОВАТЕЛЬНЫЙ ДОСТУП, т. е. каждая запись в файле может быть прочитана и обработана только после того, как прочитаны все предшествующие ей записи в файле. Именно так обстояло дело в шестидесятые годы, когда хранение информации на диске обходилось относительно дорого и большинство файлов хранилось на ленте, а записи извлекались и обрабатывались последовательно. Обычно с файлами работали в пакетном режиме, т. е. все записи файла обрабатывались за один раз, как правило, в конце рабочего дня.

Например, для создания программы подсчета причитающихся сумм и составления счета для заказчиков из файловой системы, представленной на рис. 2.2. и 2.3., используются файлы ЗАКАЗЧИК и ПРОДАЖИ. Оба файла сначала упорядочиваются по полю ИМЯ ЗАКАЗЧИКА, потом объединяются, и на основе обновленного файла, также упорядоченному по полю ИМЯ ЗАКАЗЧИКА, программа распечатывает счета.

Однако для обработки большого количества данных требуется произвольный доступ к данным, т. е. возможность напрямую обращаться к конкретной записи без предварительной сортировки файла или

последовательного чтения всех записей. Задача произвольного доступа была решена с появлением файлов произвольного доступа. Файлы произвольного доступа позволяют извлекать записи в произвольном порядке. Эти файлы дают возможность определить одно или несколько полей для точного задания того, какую запись требуется извлечь. Такие, заранее определенные поля данных называются ключом.

- Ключ - это поля данных, которые однозначно определяют запись в файле.

Однако файлы с произвольным доступом решили проблему лишь частично. Полным решением проблемы доступа к данным явилось создание системы управления базами данных.

2.3. Иерархические системы

Информационные системы, использующие базы данных, позволили преодолеть ограничения файловых систем, избавиться от проблем избыточности и слабого контроля данных.

Рассмотрим пример на рис. 2.2. и 2.3. Если прочитана первая запись о продажах в файле ПРОДАЖИ и требуется узнать имя и адрес клиента, с которым была заключена эта сделка, можно просто воспользоваться идентификатором клиента (100), чтобы посмотреть соответствующую запись в файле ЗАКАЗЧИК. Таким образом будет выяснено, что заказ был сделан Кузнецовым.

Теперь предположим, что требуется обратное. Вместо того чтобы выяснять, с каким клиентом заключена сделка, хотелось бы *найти все продажи данному клиенту*. Начнем с записи Кузнецов в файле ЗАКАЗЧИК, а затем найдем все продажи этой компании путем *перебора* всех продаж. В файловой системе последовательного доступа не возможно напрямую (без перебора) получить ответ на вопрос. Именно для подобных прикладных задач и были созданы системы управления базами данных.

ЗАКАЗЧИК

ИНДЕКС_ ЗАКАЗЧИКА	ИМЯ_ ЗАКАЗЧИКА	АДРЕС	СТРАНА	НАЧАЛЬНЫЙ_ БАЛАНС	ПЛАТЕЖИ_ МЕСЯЦ
100	Кузнецов	А/я 100	Россия	45551	40113
101	Шмидт	А/я 101	Австрия	75314	65200
105	Смит	А/я 105	США	49333	49811
110	Ката	А/я 110	Япония	27400	28414

СЧЕТ

СЧЕТ_#	ДАТА	ИМЯ_ ЗАКАЗЧИКА	ИНДЕКС_ТОРГОВОГО_ ПРЕДСТАВИТЕЛЯ
1012	10.02	100	39
1015	14.02	110	37
1020	20.02	100	14

СТРОКА СЧЕТА

СЧЕТ_#	ДАТА	ИНДЕКС_ ТОВАРА	КОЛИЧЕСТВО	ОБЩАЯ_ЦЕНА_ ПРОДАЖИ
1012	01	1035	100	2200
1012	02	2241	200	6650
1012	03	2518	300	6360
1015	01	2241	150	3300
1015	02	2518	200	4240
1020	01	2241	100	3325
1020	02	2518	150	3180

Рис. 2.4. Файлы, имеющие иерархическую структуру

Первая информационная система, использующая базы данных, появивилась в середине 60-х годов и была основана на ИЕРАРХИЧЕСКОЙ МОДЕЛИ. Это означает, что отношения между данными имели иерархическую структуру. Для того чтобы пояснить это, слегка изменим базу данных, приведенную на рис. 2.2 и 2.3. Вместо продаж, записанных в виде одной строки, у нас будут счета-фактуры, которые в свою очередь состоят из нескольких строк. Каждая строка обозначает продажу одного товара. На рис.2.4 представлен пример. Теперь вместо файла ПРОДАЖИ есть файлы СЧЕТ и СТРОКА СЧЕТА.

- ИЕРАРХИЧЕСКАЯ МОДЕЛЬ – это модель данных, в которой связи между данными имеют вид иерархий.

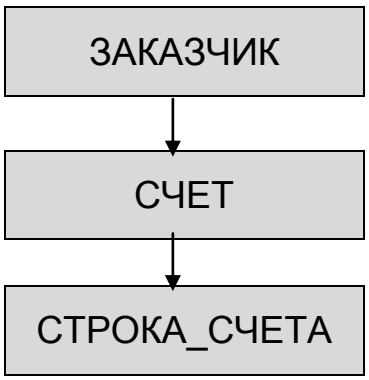


Рис.2.5. Иерархическая модель отношений между файлами ЗАКАЗЧИК, СЧЕТ и СТРОКА_СЧЕТА

На рис.2.5 показано, как выглядит иерархия отношений между клиентами, счетами и строками счетов. Клиенту «подчинены» счета, которым в свою очередь «подчинены» строки. В иерархической базе данных эти три файла будут связаны между собой физическими указателями или полями данных, добавленными к отдельным записям. УКАЗАТЕЛЬ – это физический адрес, означающий, где запись находится на диске. Каждая запись о клиенте будет содержать указатель первой записи счета этого клиента. В свою очередь записи счетов будут содержать указатели на другие записи счетов и на записи строк

счетов. Таким образом, система легко сможет извлечь все *записи счетов* и *строк счетов*, относящихся к данному *клиенту*.

Предположим, что необходимо добавить в иерархическую базу данных информацию о клиентах. Например, если клиентами являются торговые компании, может понадобиться список магазинов каждой компании. В этом случае необходимо расширить диаграмму, приведенную на рис. 2.5, придав ей вид, представленный на рис.2.6. Файл ЗАКАЗЧИК по-прежнему находится над файлом СЧЕТ, который находится над файлом СТРОКА_СЧЕТА. Но в то же время с файлом ЗАКАЗЧИК связан файл МАГАЗИН, а с ним - файл ПРЕДСТАВИТЕЛЬ. Под представителем подразумевается закупщик, которому продаются товары для конкретного магазина. Из этой диаграмме видно, что заказчик является вершиной иерархии. На диаграммах 2.5 и 2.6 показаны те разновидности связей между файлами, которые могут быть легко реализованы в иерархической модели.

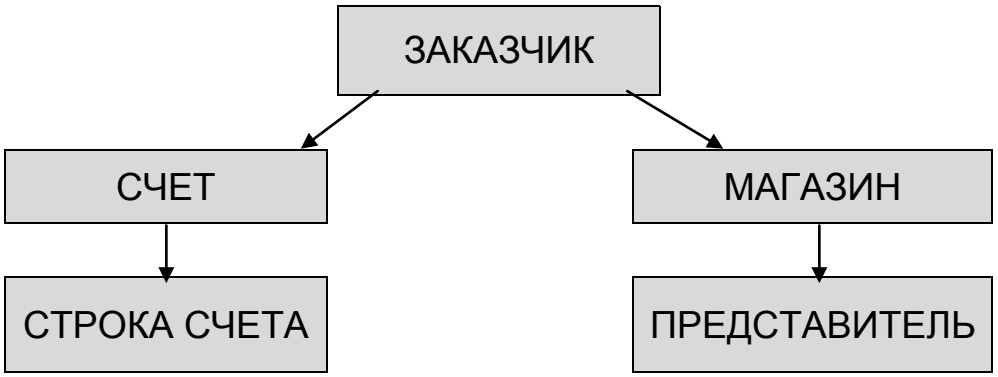


Рис.2.6. Иерархическая модель отношений между файлами ЗАКАЗЧИК, СЧЕТ и МАГАЗИН

Введем теперь формальное определение иерархической структуры. Основным типом логической структуры, поддерживаемой иерархическими СУБД, является иерархия или древовидная структура. Иерархическая структура

определяется согласно дереву определений. В этом дереве типы записей являются узлами, а дуги представляют связи, порожденные между узлами дерева различных уровней.

Если разность уровней двух связанных узлов равна единице, то связь является непосредственной (без промежуточных узлов). Иерархическая структура реализует отображение: 1:N: между порожденным и исходным типами записей. Это отображение полностью функционально, так как дерево не может содержать порожденный узел без исходного узла (за исключением корня). Следовательно, отображения 1:1, 1:N могут непосредственно представляться иерархическими структурами, однако для представления отображения типа M:N необходимо дублирование деревьев.

Другими словами, древовидная структура, или дерево, - это связный неориентированный граф, который не содержит циклов, т. е. петель из замкнутых путей. Обычно при работе с деревом выделяют какую-то конкретную вершину, определяют ее как корень дерева и рассматривают особо: в эту вершину не заходит ни одно ребро. В этом случае дерево становится ориентированным. Ориентация на корневом дереве определяется либо от корня, либо к корню.

На рис.2.7 приведен пример схемы иерархической базы данных по составлению экзаменационных ведомостей.

Наглядным примером иерархической структуры является файловая система Windows.

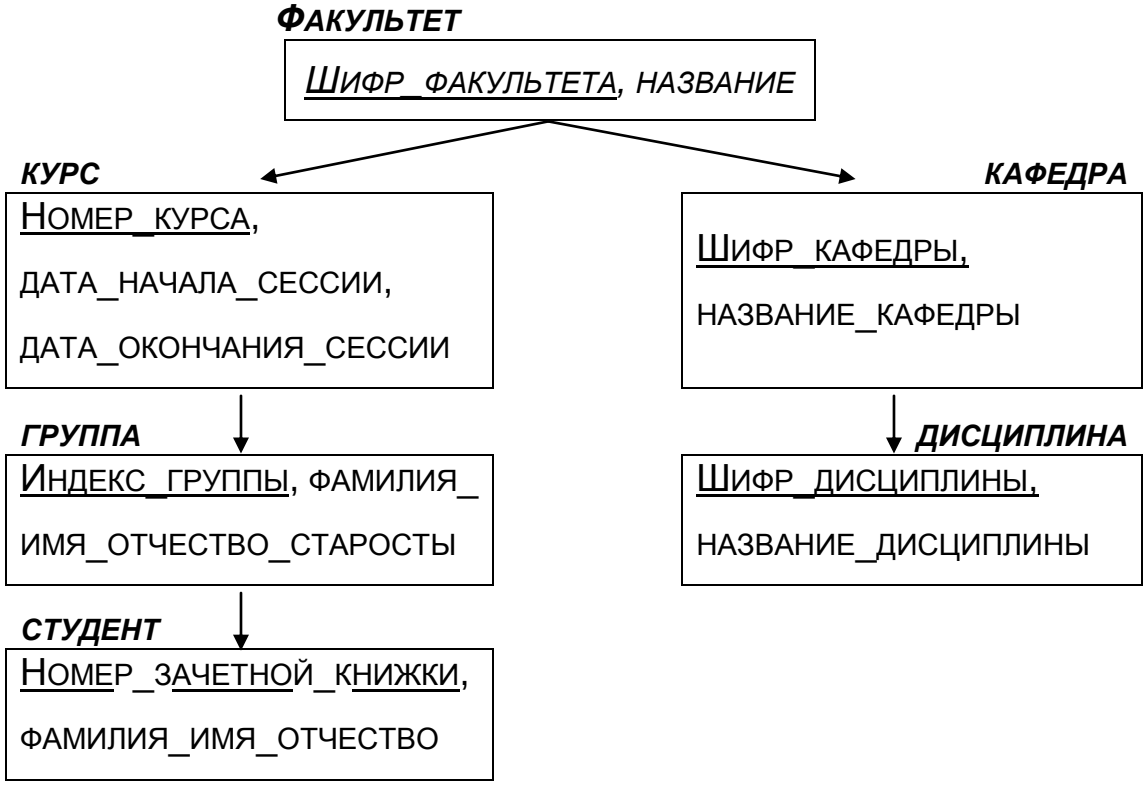


Рис.2.7. Пример схемы иерархической базы данных

2.4. Сетевые системы

У иерархической модели есть некоторые существенные ограничения, поскольку не все отношения можно представить в виде иерархии. Вернемся к нашему основному примеру и сделаем следующий шаг. Очевидно, что нас могут интересовать связи не только между клиентами и счетами, но и между торговыми представителями и счетами. То есть мы хотим иметь список всех счетов на продажи, произведенные определенным торговым представителем, чтобы подсчитать сумму причитающихся ему комиссионных. Новые связи представлены на рис. 2.8. Однако эта диаграмма не является иерархической. В иерархии у каждого ПОТОМКА может быть только один ПРЕДОК.

- ПОТОМОК – ПОДЧИНЕННАЯ ЗАПИСЬ В ИЕРАРХИИ. ПРЕДОК – ПОДЧИНЯЮЩАЯ ЗАПИСЬ В ИЕРАРХИИ.

На рис.2.6 СЧЕТ - потомок, ЗАКАЗЧИК – его предок. Однако на рис.2.8 у файла СЧЕТ имеется два предка – файл ТОРГОВЫЙ ПРЕДСТАВИТЕЛЬ и файл ЗАКАЗЧИК. Такого рода диаграммы называются СЕТЕВЫМИ. В связи с необходимостью обрабатывать СЕТЕВЫЕ отношения в конце 60-х годов появились СЕТЕВЫЕ СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ. Как и в иерархических, в сетевых системах баз данных для связывания файлов использовались физические указатели.

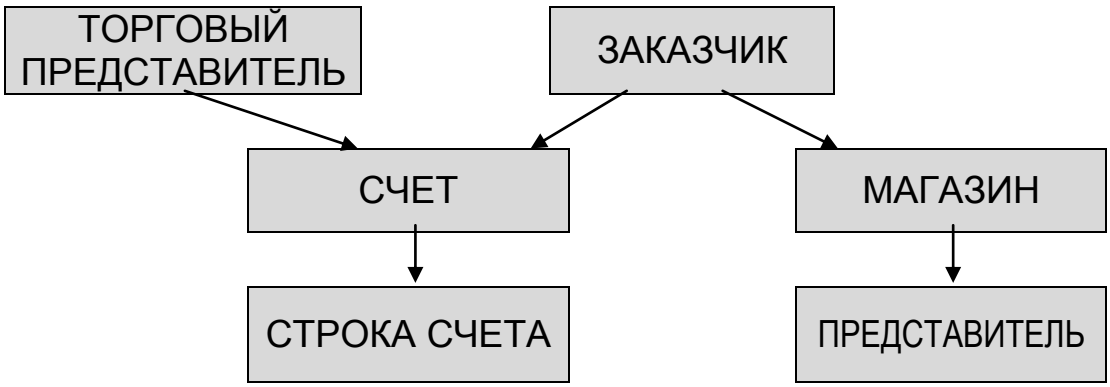


Рис.2.8. Сетевая модель отношений между файлами ТОРГОВЫЙ ПРЕДСТАВИТЕЛЬ, ЗАКАЗЧИК и СЧЕТ

СЕТЕВЫЕ СУБД используют сетевую модель данных. С точки зрения теории графов сетевой модели соответствует произвольный граф (возможно, имеющий циклы и петли). В узлах графа помещаются типы записей, а ребра интерпретируются как связи между типами записей.

Наглядным примером сетевой системы является структура Web-страниц и ссылок всемирной сети Internet.

2.5. Реляционные системы управления базами данных

Использование физических указателей было одновременно и сильной и слабой стороной иерархических и сетевых систем управления базами данных. Сильной - поскольку они позволяли извлекать данные, связанные с определенными отношениями. Слабой - поскольку эти отношения должны быть определены до запуска системы. Извлечь данные на основе других отношений было сложно или вообще невозможно. Как только пользователи освоили информационные системы, использующие базы данных, и разобрались с их возможностями по работе с данными, такие ограничения стали неприемлемыми.

Как уже было упомянуто, в 1970 году Е.Ф.Кодд (США) опубликовал революционную по содержанию статью «Реляционная модель данных для разделяемых баз данных», которая всерьез поколебала устоявшиеся представления о базах данных. Он выдвинул идею, что данные нужно связывать в соответствии с их внутренними логическими взаимоотношениями, а не с физическими указателями. Таким образом пользователи могут комбинировать данные из разных источников, если логическая информация, необходимая для такого комбинирования, присутствует в исходных данных. Это открыло новые возможности для информационно-управляющих систем, поскольку запросы к базам данных теперь не были ограничены физическими указателями.

Для того чтобы понять, какие недостатки присущи иерархическим и сетевым системам, рассмотрим рис.2.9. На нем показано, что файлы ЗАКАЗЧИК, СЧЕТ и СТРОКА СЧЕТА связаны физическими указателями. Файлы ИЗГОТОВИТЕЛЬ и ТОВАР тоже связаны. Линия между СТРОКА СЧЕТА и ТОВАР обозначает, что между ними существует *логическая связь*, поскольку каждая строка счета относится к конкретному товару. Однако предположим, что файл ТОВАР не привязан к файлу СТРОКА СЧЕТА физическим указателем. Как тогда составить следующий отчет? *Для каждого клиента перечислить изготовителей приобретенных им товаров*.

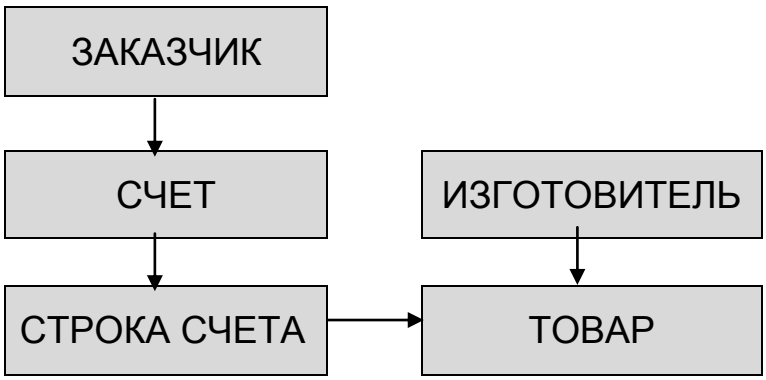


Рис.2.9. Логическая связь, не поддерживаемая физическими указателями

Для составления такого отчета требуется двигаться от файла «Заказчик» через файл «Счет» и файл «Строка счета» к файлу «Товар» и затем к файлу «Изготовитель». Но поскольку между файлами «Строка счета» и «Товар» нет физической связи, то обычными средствами базы данных такой путь проделать невозможно. Для того чтобы все-таки получить такую информацию, придется пользоваться трудоемкими способами работы с файлами. А информационные системы, использующие базы данных, которые поддерживают извлечение данных на основе *логических связей*, легко ответят на такой вопрос.

Кодд предложил простую модель данных, согласно которой все данные сведены в таблицы, состоящие из строк и столбцов. Эти таблицы получили название РЕЛЯЦИЙ, или ОТНОШЕНИЙ, а модель стала называться соответственно РЕЛЯЦИОННОЙ. Было также предложено пользоваться для работы с данными в таблице двумя языками: реляционной алгеброй и реляционным исчислением. Оба эти языка обеспечивают работу с данными на основе логических характеристик, а не физических указателей, которыми пользовались в иерархических и сетевых моделях.

Рассматривая данные с концептуальной, а не физической точки зрения, была предложена еще одна интересная идея. В реляционных системах баз

данных целые файлы данных могут обрабатываться одной командой, тогда как в традиционных системах за один раз обрабатывается только одна запись. Данный подход чрезвычайно повысил эффективность программирования в базах данных.

Логический подход к данным сделал также возможным создание языков запросов, более доступных для пользователей, не являющихся специалистами по компьютерам.

Результатом развития реляционной теории баз данных стало создание во второй половине 70-х годов реляционных систем, которые поддерживали такие языки, как Structured Query Language (SQL) - язык структурированных запросов, Query Language (Quel) - язык запросов и Query-by-Example (QBE) - запросы по образцу.

В настоящее время реляционные базы данных рассматриваются как стандарт для современных коммерческих систем работы с данными. На рынке информационных систем прослеживается общая тенденция перехода на реляционные системы. Но файловые системы, иерархические и сетевые базы данных все еще многочисленны, и сегодня во многих случаях именно их применение является наиболее выгодным.

В таблице 2.1 приведены сравнительные характеристики различных способов доступа к данным.

Таблица 2.1.

Сравнительная характеристика способов доступа к данным.

Способ доступа к данным	Характеристика
Файлы последовательного доступа	Записи должны обрабатываться в последовательном порядке
Файлы произвольного доступа	Поддерживают прямой доступ к конкретной записи. Сложно обращаться к нескольким записям, связанным с одной
Иерархическая база данных	Поддерживает доступ к нескольким записям, связанным с одной. Отношения между данными ограничиваются иерархическими отношениями. Зависит от предопределенных физических указателей
Сетевая база данных	Поддерживает иерархические и неиерархические отношения между данными. Зависит от предопределенных физических указателей
Реляционная база данных	Поддерживает все логические отношения между данными. Логический доступ к данным, не зависящий от физической реализации

Глава 3. Реляционная модель данных

Перейдем к подробному рассмотрению реляционной модели данных.

3.1. Понятие отношения

Прежде чем перейти к понятию отношения приведем некоторые сведения из алгебры множеств.

Пусть S_1 и S_2 некоторые множества, тогда объединением $S_1 \cup S_2$ называется множество, состоящее из элементов, принадлежащих либо S_1, либо S_2, пересечением $S_1 \cap S_2$ называется множество, состоящее из элементов, принадлежащих S_1 и S_2 одновременно, разностью $S_1 \backslash S_2$ называется множество, состоящее из тех элементов, принадлежащих S_1, которые не принадлежат S_2.

- ДЕКАРТОВЫМ ПРОИЗВЕДЕНИЕМ множеств S_1, S_2,…,S_n называется множество $S_1 \times S_2 \times \ldots \times S_n$, представляющее собой всевозможные упорядоченные наборы $(s_1,\ldots,s_n)$, где $s_i \in S_i$ для $i=1,2,\ldots,n$.

В основе реляционной модели данных лежит понятие ОТНОШЕНИЯ.

- В терминах теории множеств ОТНОШЕНИЕ можно определить следующим образом: пусть D_1, D_2,…, D_n - это множества, которые будем называть доменами, и $D=D_1 \times D_2 \times \ldots \times D_n$ – декартово произведение доменов. Отношением **R** называется любое подмножество множества D.

Из определения следует, что элементами отношения являются упорядоченные строки длины n, которые называются КОРТЕЖАМИ. Число n называется СТЕПЕНЬЮ, или АРНОСТЬЮ, отношения. Таким образом, отношение можно представить в виде таблицы, строки которой представляют собой кортежи, а столбцы состоят из элементов соответствующего домена. Порядок строк в отношении (таблице) не имеет значения, а порядок столбцов фиксирован.

Отношение используется для представления различных объектов. Рассмотрим некоторый класс объектов, то есть множество (набор) подобных

объектов. Например, совокупность всех студентов института, множество компаний, имеющих лицензию на торговлю цветными металлами и т. д.

- Введем понятие АТРИБУТА. АТРИБУТОМ называется информационное отображение отдельного свойства некоторого объекта. Классу объектов S, состоящему из m элементов $\{s_1,s_2,\ldots,s_m\}$, ставится в соответствие некоторое конечное множество атрибутов $A_1, A_2, \ldots, A_n$.

Например, для класса объектов "студент" (совокупность всех студентов института) атрибутами могут быть: фамилия студента, год его рождения, год поступления в институт и т.д. Каждый атрибут имеет конкретное значение в зависимости от описываемого объекта заданного класса. Каждому объекту s_i (i=1,2…m) можно поставить в соответствие набор значений его атрибутов $\{a_{i1}, a_{i2},\ldots, a_{in}\}$, где a_{ik}–значение атрибута A_k для объекта s_i.

Таким образом, класс объектов S представлен с помощью некоторого отношения **R**, при этом роль домена D_k играет множество возможных значений атрибута A_k. В этом отношении будет m кортежей. Столбцам отношений (атрибутам) принято давать имена. При этом столбец будет определяться по его имени, а не по порядковому номеру, и в этом случае отпадает необходимость в фиксированном порядке столбцов.

СХЕМОЙ ОТНОШЕНИЯ называется список имен атрибутов этого отношения и обозначается как **R**($B_1, B_2, \ldots, B_n$.), где **R** это имя отношения, $B_1, B_2, \ldots, B_n$ список имен атрибутов.

- РЕЛЯЦИОННАЯ БАЗА ДАННЫХ - это набор из нескольких отношений.

Приведем пример отношения «Студент».

Фамилия_ Имя_Отчество	**Год_ рождения**	**Год_ поступления**	**Факультет**
Иванов И.И.	1980	1999	Экономический
Петров П.П.	1981	1999	Политологии
Семенова К.Н.	1979	1998	Юридический
Сидорова С.С.	1980	1998	Юридический

В этом примере первая строка таблицы представляет собой имена столбцов отношения. Степень отношения равна четырем. Строки таблицы, кроме первой, – это КОРТЕЖИ отношения. Число кортежей равно четырем.

Подведем итог. Классы объектов можно представить в виде ОТНОШЕНИЯ. ОТНОШЕНИЯ в свою очередь представляются в виде ТАБЛИЦ. КОРТЕЖИ ОТНОШЕНИЯ – это строки таблицы, которые также называют ЗАПИСЯМИ, столбцы таблицы представляют собой значения АТРИБУТОВ, берущихся из соответствующего домена. Первая строка таблицы – это имена атрибутов.

Операции над отношениями

Так как отношения представляют собой множества, то для отношений можно определить стандартные операции объединения, пересечения, разности и декартова произведения. Однако в случае отношений они имеют свои особенности. Операции объединения, пересечения и разности определяются над отношениями с одинаковой схемой.

Пусть $\mathbf{R_1}$ и $\mathbf{R_2}$ два отношения с одинаковой схемой.

- ОБЪЕДИНЕНИЕМ ОТНОШЕНИЙ $\mathbf{R_1}$ и $\mathbf{R_2}$ называется отношение $\mathbf{R=R_1 \cup R_2}$, которое содержит строки, принадлежащие либо отношению $\mathbf{R_1}$, либо $\mathbf{R_2}$.
- ПЕРЕСЕЧЕНИЕМ ОТНОШЕНИЙ $\mathbf{R_1}$ и $\mathbf{R_2}$ называется отношение $\mathbf{R=R_1 \cap R_2}$, которое содержит строки, принадлежащие отношению $\mathbf{R_1}$ и $\mathbf{R_2}$ одновременно.
- РАЗНОСТЬЮ ОТНОШЕНИЙ $\mathbf{R_1}$ и $\mathbf{R_2}$ называется отношение $\mathbf{R=R_1 - R_2}$, которое содержит те строки, принадлежащие отношению $\mathbf{R_1}$, но которые не принадлежат $\mathbf{R_2}$.
- ДЕКАРТОВЫМ ПРОИЗВЕДЕНИЕМ ОТНОШЕНИЙ $\mathbf{R_1}$ и $\mathbf{R_2}$ (в данном случае $\mathbf{R_1}$ и $\mathbf{R_2}$ имеют разные схемы отношений и списки имен отношений $\mathbf{R_1}$ и $\mathbf{R_2}$ не пересекаются) называется отношение $\mathbf{R=R_1 \times R_2}$, которое содержит всевозможные строки, получающиеся в результате слияния строк отношения $\mathbf{R_1}$ и $\mathbf{R_2}$.

Приведем примеры.

Ниже приведены три отношения **R_1, R_2, R_3**.

R_1

И1	**И2**
A	1
B	2

R_2

И1	**И2**
a	2
b	2

R_3

И3	**И4**
c	1
c	2

Объединение, пересечение и разность отношений будут иметь вид:

$R_1 \cup R_2$

И1	**И2**
A	1
A	2
B	2

$R_1 \cap R_2$

И1	**И2**
b	2

$R=R_1-R_2$

И1	**И2**
a	1

Декартово произведение отношений **$R_1 \times R_3$** будет равно:

$R_1 \times R_3$

И1	**И2**	**И3**	**И4**
a	1	c	1
a	1	c	2
b	2	c	1
b	2	c	2

Существуют специальные операции реляционной алгебры, такие, как выборка, взятие проекции, соединение, деление и присвоение.

- Операция ВЫБОРКА производит выбор строк данного отношения **R** по определенному условию, называемому условием выборки. В результате получается отношение, все строки которого удовлетворяют условию выборки.
- Операция ПРОЕКЦИЯ производит выбор указанных столбцов в указанном порядке из отношения **R**. Получившееся в результате новое отношение не должно иметь одинаковых строк.
- Операция СОЕДИНЕНИЯ формирует из двух отношений $\mathbf{R_1}$ и $\mathbf{R_2}$ новое отношение по определенным правилам.

Существует несколько операций СОЕДИНЕНИЯ, наиболее важной из которых является естественное соединение, определение которого и дадим. Предполагаем, что список имен отношений $\mathbf{R_1}$ и $\mathbf{R_2}$ имеет пересечение B_1, B_2, …, B_k. Новое отношение получается путем слияния таких строк отношений $\mathbf{R_1}$ и $\mathbf{R_2}$, в которых значения атрибутов в столбцах с именами B_1, B_2, …, B_k совпадают. При этом копии столбцов B_1, B_2, …, B_k должны быть удалены.

- Операция ДЕЛЕНИЯ отношения $\mathbf{R_1}$ на $\mathbf{R_2}$ возможна, если список имен отношения $\mathbf{R_2}$ является подмножеством списка имен отношения $\mathbf{R_1}$. Тогда в результате деления получается отношение со списком имен, которые входят в $\mathbf{R_1}$ и не входят в $\mathbf{R_2}$, а строка входит в новое отношение только в том случае, если она входит в $\mathbf{R_1}$ с каждой строкой $\mathbf{R_2}$.

Операция ПРИСВОЕНИЯ дает имя отношению.

Нормализация отношений

В процессе пректирования базы данных необходимо определить число отношений, входящих в базу данных и состав атрибутов, входящих в каждое отношение. При этом необходимо учитывать требования:

1. Все необходимые данные должны храниться в базе данных.

2. Необходимо по возможности исключить избыточность информации.

3. Число отношений в базе данных должно быть минимально.

4. Обновление и удаление данных не должно приводить к двусмысленности или потере информации.

Одним из способов преобразования отношений с целью удовлетворения перечисленным требованиям является НОРМАЛИЗАЦИЯ ОТНОШЕНИЙ.

- Процесс приведения отношений (реляционных таблиц) к стандартному виду называется НОРМАЛИЗАЦИЕЙ.

Первая нормальная форма.

- Отношение находится в первой нормальной форме, если все значения атрибутов являются атомарными.

Вторая нормальная форма.

Введем понятие функциональной зависимости. Пусть X и Y атрибуты отношения **R**, тогда говорят, что атрибут Y функционально зависит от атрибута X, если в каждый момент времени каждому значению атрибута X соответствует единственное значение атрибута Y. Другими словами, если два кортежа в отношении имеют одно и тоже значение атрибута X, то они имеют одно и тоже значение атрибута Y. Это обозначается: $X \rightarrow Y$. Определение функциональной зависимости сохраняет силу в случае, когда X и Y состоят из нескольких атрибутов. Пусть некоторое множество атрибутов K из отношения **R** обладает следующими свойстами:

Любой атрибут, не принадлежащий K, функционально зависит от K.

Любое собственное подмножество K не обладает свойством 1.

Такой набор атрибутов K называется ключом отношения **R,** и этот набор однозначно определяет строку отношения. Можно также сказать, что ключ - это минимальный набор атрибутов, однозначно определяющий каждую строку отношения. Ключ, содержащий более одного атрибута, называется составным.

- Отношение находится во второй нормальной форме, если зависимость неключевых атрибутов от ключа является функционально полной, т. е. неключевой атрибут не может зависить от части ключа. Таким образом, вторая нормальная форма может нарушаться только, если ключ составной.

Третья нормальная форма.

Отношение находится в третьей нормальной форме, если в нем нет транзитивных зависимостей. Транзитивная зависимость - это такая зависимость, при которой неключевой атрибут зависит от одного или более неключевых атрибутов.

Иногда используют усиленный вариант третьей нормальной формы, так называемая нормальная форма Бойса-Кодда.

Отношение находится в нормальной форме Бойса-Кодда, если для каждой функциональной зависимости $X \rightarrow Y$ X является ключом.

Если отношение находится в нормальной форме Бойса-Кодда, то оно также находится и в третьей нормальной форме.

3.2. Теоретико-множественные операции управления данными

Объединение таблиц

Приведем конкретные примеры из теории множеств. Известно, что объединением множеств A и B является множество C, которое состоит из элементов, присутствующих хотя бы в одном из множеств A и B. Например, результатом объединения множеств A={1;3;5;6;7} и B={2;3;4;5} является множество $C = A \cup B$={1;2;3;4;5;6;7}. Множество C состоит из элементов 1, 2, 4, 6, 7, которые по одному разу встречаются в A и B, а также элементы 3 и 5, которые присутствуют и в A, и в B одновременно. Количество элементов множества C равно 7. Элементы, присутствующие одновременно и в A и в B, учитываются в C только один раз.

Применяя теорию множеств к реляционным базам данных, будем рассматривать таблицы как множество записей, другими словами: таблица-множество, запись-элемент множество. При этом сами реляционные таблицы, к которым применяются теоретико-множественные операции, должны иметь одинаковое количество атрибутов, а сами атрибуты соответственно - одинаковые названия.

Практическую реализацию операций над множествами будем проводить в среде MS EXCEL с использованием программы MS QUERY. Программа MS QUERY интегрирована в пакет программ MS OFFICE и доступна из любой компоненты пакета MS OFFICE. При обращении к данным в EXCEL-формате MS QUERY формирует окно запроса и позволяет организовать доступ к отдельным записям данных и произвести из них выборку требуемой информации с помещением в ячейки активной в данный момент таблицы.

В программе MS EXCEL вызов MS QUERY осуществляется в меню Данные-Внешние данные-Создать запрос. При вызове вначале появляется диалоговое окно Выбор источника данных, где необходимо указать формат базы

данных. В нашем случае это будут таблицы EXCEL. Следующий этап – диалоговое окно Выбор книги. Здесь необходимо указать файл, который содержит таблицы рассматриваемой базы данных.

Рассмотрим операцию объединения на конкретном примере таблиц 3.1 и 3.2.

Таблица 3.1.

Россия_экспорт: исходная

	Наименование	**Страна**
1	Нефть	Германия
2	Газ	Франция
3	Металлопрокат	США
4	Древесина	Финляндия

Таблица 3.2.

Белоруссия_экспорт: исходная

	Наименование	**Страна**
1	Цветной металл	Германия
2	Металлопрокат	США
3	Древесина	Италия

Эти таблицы имеют по два атрибута с одинаковыми именами, следовательно, к ним может быть применима теоретико-множественная операция объединения таблиц, например для того, чтобы определить структуру экспорта союза двух стран. Следует заметить, что две записи считаются одинаковыми, если у них попарно равны между собой значения соответствующих полей.

Результирующая таблица должна содержать первую, вторую, и четвёртую записи из таблицы 3.1. Эти записи содержатся только в таблице 3.1. Затем первую и третью записи из таблицы 3.2, которые не совпадают ни с одной записью из таблицы 3.1. И, наконец, третью запись из таблицы 3.1 или вторую запись из 3.2, так как они одинаковы. Таким образом, три записи из таблицы 3.1, две из 3.2 и одна общая для двух таблиц составят общее число (6) записей в результирующей таблице.

Таблица 3.3.

Результат теоретико-множественного объединения таблиц

	Наименование	Страна
1	Нефть	Германия
2	Цветной металл	Германия
3	Древесина	Италия
4	Металлопрокат	США
5	Древесина	Финляндия
6	Газ	Франция

Для создания запроса активизируем на панели инструментов QUERY кнопку SQL и в появившееся диалоговое окно записываем текст запроса.

В запросе будет использоваться оператор UNION (объединение), перед и после которого с помощью инструкций SELECT и предложений FROM, задаются исходные таблицы (рис.3.1). В инструкции SELECT можно не перечислять все атрибуты, которые должны быть включены в результирующую таблицу, а заменить их звёздочкой (*).

```
SELECT          Россия_экспорт.*
FROM            Россия_экспорт
UNION
SELECT          Белоруссия_экспорт.*
FROM            Белоруссия_экспорт
ORDER BY        Россия_экспорт.Страна
```

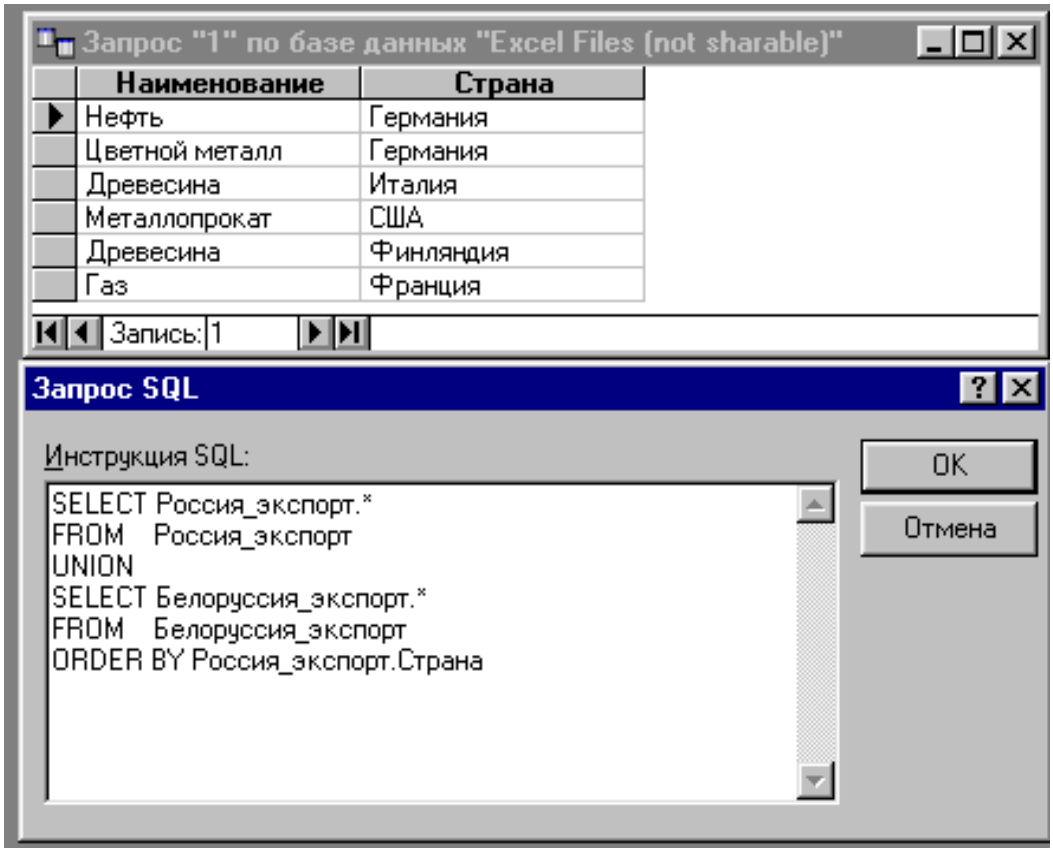


Рис 3.1. Пример запроса с использованием оператора UNION

Если вместо UNION указать UNION ALL, то результирующая таблица будет содержать оба экземпляра записи, общие для обеих исходных таблиц. Предложение ORDER BY сортирует в порядке возрастания по алфавиту записи результирующей таблицы по полю *Россия_экспорт.Страна*. Можно задать сортировку по убыванию, если после имени поля *Россия_экспорт.Страна* указать порядок сортировки DESC.

Теперь предположим, что атомарным (атомарно-неделимым) элементом данных является часть строки-поле. Например, нужно определить, в какие страны экспортирует свою продукцию союз двух стран. То есть необходимо вычислить объединение двух таблиц не по записям, а по полям атрибута *Страна*, который есть как в одной, так и другой таблице. В результате выполнения запроса

```
SELECT          Россия_экспорт.Страна
FROM            Россия_экспорт
UNION
SELECT          Белоруссия_экспорт.Страна
FROM            Белоруссия_экспорт
```

получим объединение реляционных таблиц по атрибуту *Страна*:

Страна
Германия
Италия
США
Финляндия
Франция

Пересечение таблиц

Пересечением множеств A и B является множество C, которое состоит из элементов, присутствующих в обоих множествах A и B. Результатом объединения множеств A={1;3;5;6;7} и B={2;3;4;5} является множество $C = A \cap B = \{3;5\}$.

Рассмотрим операцию пересечения на примере таблиц 3.1 и 3.2. Условия применимости теоретико-множественных операций соблюдены, так как эти таблицы имеют по два атрибута с одинаковыми названиями. Предположим, нужно узнать одинаковые для обеих стран экспортные товары и соответственно одинаковые для этих товаров страны-импортеры. Результатом выполнения такого запроса будет одна запись:

Металлопрокат	США

Эта запись содержится как в таблице 3.1, так и в таблице 3.2. Предложение WHERE инструкции SELECT запроса задает два условия отбора записей с одинаковыми полями в исходных таблицах, соединенных логическим оператором AND (рис. 3.2.):

```
SELECT      Россия_экспорт.*
FROM        Россия_экспорт, Белоруссия_экспорт
WHERE       Россия_экспорт Наименование =
            Белоруссия_экспорт.Наименование
AND         Россия_экспорт.Страна =
            Белоруссия_экспорт.Страна
```

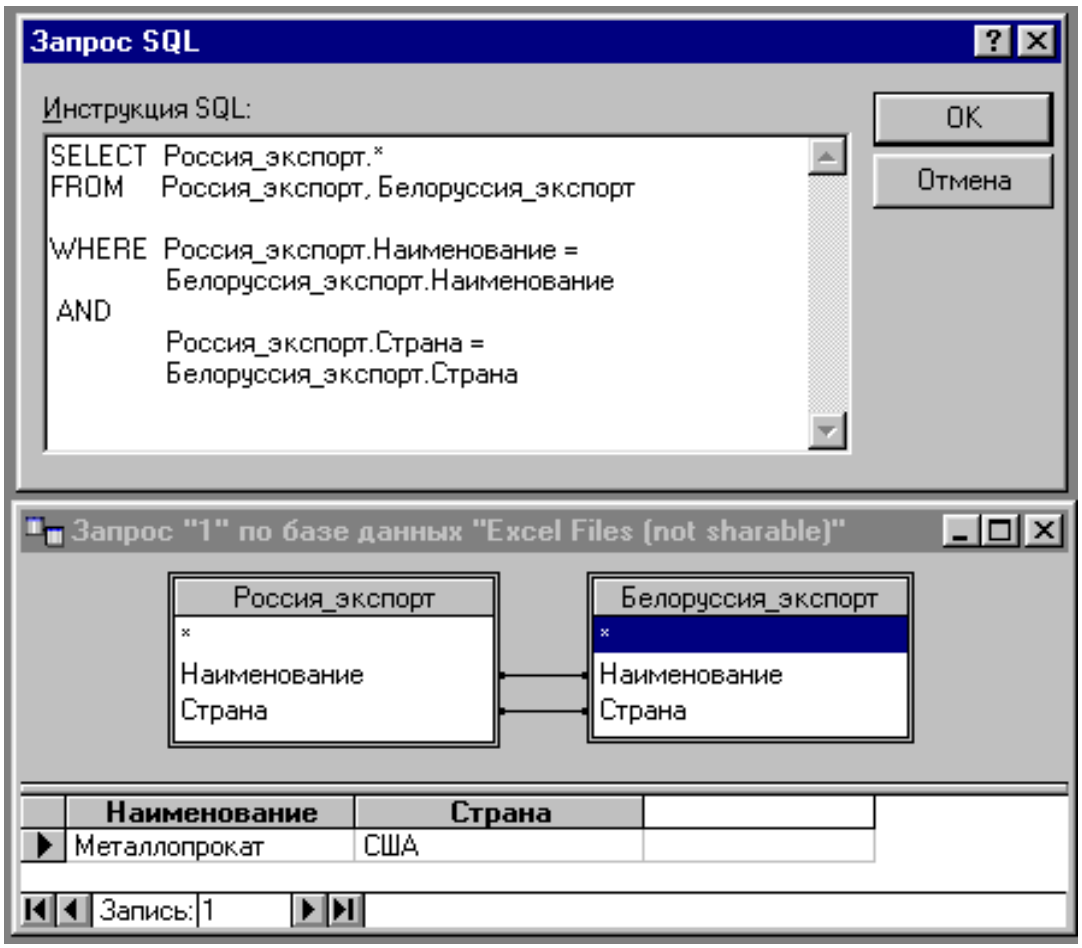


Рис 3.2. Пример выполнения операции пересечения таблиц

Аналогично выполняется запрос на пересечение по одному совместному для двух таблиц атрибуту. Например, найти страны-импортеры, общие для двух стран

```
SELECT      Россия_экспорт.Страна
FROM        Россия_экспорт, Белоруссия_экспорт
WHERE       Россия_экспорт.Страна =
            Белоруссия_экспорт.Страна
```

Результатом является: США.

Разность таблиц

Разностью множеств A и B является множество C, которое состоит из элементов множества A, которых нет в B. Разностью множеств A={1;3;5;6;7} и B={2;3;4;5} является множество C=A-B={1;6;7}. Для практической реализации разность удобнее представить в виде: $C = A - (A \cap B)$. Найдем разность двух таблиц 3.1 и 3.2, которые удовлетворяют условию применимости теоретико-множественных операций. Разность этих таблиц определяет уникальную структуру экспорта России в союзе двух стран. Результирующая таблица должна содержать первую, вторую и четвертую записи из таблицы 3.1. Третья запись таблицы 3.1 отсутствует, так как она совпадает со второй записью таблицы 3.2. Выполнение запроса осуществляется в два этапа:

1. Находим пересечение таблиц *Россия_экспорт* и *Белоруссия_экспорт*. Результирующей таблице-пересечению присваиваем новое имя и сохраняем.
2. Находим разность таблицы *Россия_экспорт* и таблицы-пересечения.

Запрос первого этапа на создание пересечения в отличие от аналогичного запроса из предыдущего раздела содержит оператор INTO, за которым следует название новой таблицы *Пересечение_таблиц.* Инструкция

SELECT … INTO (запрос на создание таблицы) создает новую таблицу, пользуясь данными, выбранными из одной или нескольких таблиц.

```
SELECT      Россия_экспорт.*
INTO        Пересечение_таблиц
FROM        Россия_экспорт, Белоруссия_экспорт
WHERE       Россия_экспорт Наименование =
            Белоруссия_экспорт.Наименование
AND         Россия_экспорт.Страна =
            Белоруссия_экспорт.Страна
```

Запрос второго этапа имеет следующий вид:

```
SELECT      Россия_экспорт.*
FROM        Россия_экспорт, Пересечение_таблиц
WHERE       (Россия_экспорт Наименование <>
            Белоруссия_экспорт.Наименование)
OR          (Россия_экспорт.Страна <>
            Белоруссия_экспорт.Страна)
```

В результате получаем следующую таблицу:

Наименование	**Страна**
Нефть	Германия
Газ	Франция
Древесина	Финляндия

Процедура соединения записей реляционных таблиц

Внутреннее соединение

При внутреннем соединении из обеих таблиц выбираются только те записи, которые имеют одинаковые значения в объединяющих полях. Они отображаются как одна запись в таблице результатов запроса. Если записи не совпадают, то они не выводятся в таблице результатов.

Таблица 3.4.

Клиенты: исходная

Наименование	Код_клиента
Белый ветер	23
Формоза	45
Партия	12

Таблица 3.5.

Заказы: исходная

Код_клиента	Код_ товара	Наименование
23	56	Принтер
31	98	Клавиатура
12	37	Мышь

В этих таблицах объединяющими полями являются поля *Код_клиента*. Записи таблицы 3.4, у которой значение полей *Код_клиента* равны соответствующим значениям полей *Код_клиента* таблицы 3.5, соединяются в одну запись и включаются в таблицу результатов запроса.

Клиенты, заказы: результат внутреннего соединения таблиц 3.4 и 3.5

Наименование	Код_клиента	Код_клиента	Код_товара	Наименование
Белый ветер	23	23	56	Принтер
Партия	12	12	37	Мышь

В этой таблице показаны все атрибуты, которые содержатся в исходных таблицах. Так как соединение таблиц проведено по атрибутам *Код_клиента*, то

в результате получилось два атрибута *Код_клиента* с одинаковыми значениями. Чтобы избежать дублирования информации, один из них не выводится в результирующую таблицу (выделен пунктиром).

Окно запроса представлено на рис. 3.3.

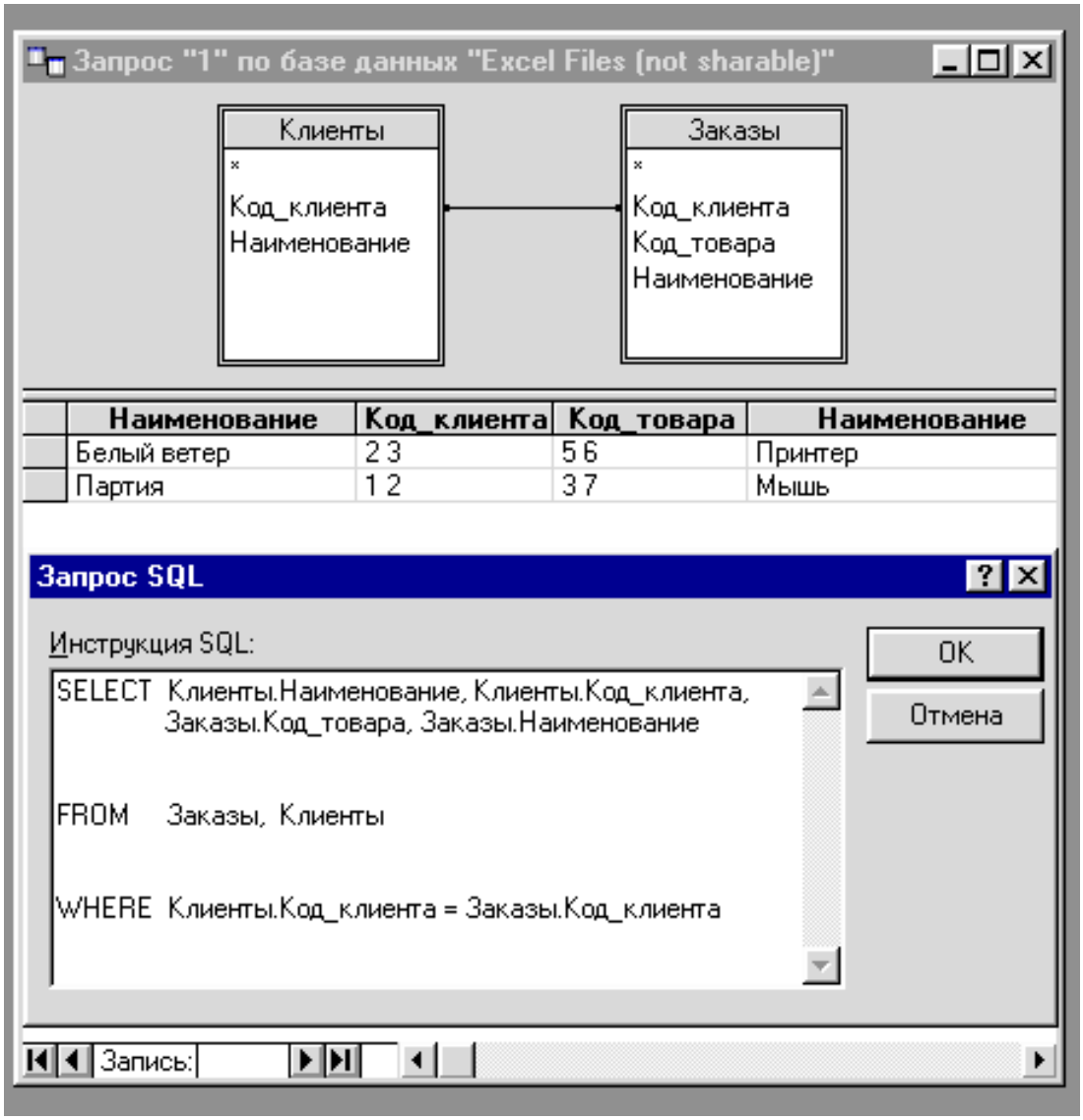


Рис.3.3. Внутреннее соединение таблиц

Можно при создании запроса воспользоваться операцией INNER JOIN (внутреннее соединение) ON по условию: значения полей *Код_клиента* таблицы *Клиенты* равны значениям полей *Код_клиента* таблицы *Заказы*.

```
SELECT      Клиенты.Наименование, Клиенты.Код_клиента,
            Заказы.Код_товара, Заказы.Наименование
FROM        Клиенты
INNER JOIN  Заказы
ON          Клиенты.Код_клиента = Заказы.Код_клиента
```

Результат выполнения этого запроса адекватен предыдущему.

Процедура соединения записей реляционных таблиц.

Внешнее соединение

Рассмотрим внешнее соединение таблиц. При создании внешнего соединения выбираются все записи из одной таблицы независимо от того, есть ли совпадающие значения в сравниваемых полях другой таблицы. При совпадении значений сравниваемых полей двух таблиц соответствующие записи соединяются и отображаются как одна запись в таблице результатов запроса. Если нет пригодной для соединения записи в одной из таблиц, то в таблице результатов будут выведены пустые поля.

Для создания внешнего соединения воспользуемся таблицами *Клиенты* и *Заказы*. Как и в случае внутреннего соединения, таблиц объединяющими полями соединения являются поля *Код_клиента*. Также должно быть указано, какая из таблиц первая или вторая (левая или правая) является ведущей, то есть назначена для вывода всех своих записей. Сначала предположим, что это первая по порядку таблица. Записи первой таблицы, у которой значение полей *Код_клиента* равны соответствующим значениям полей *Код_клиента* второй таблицы, соединяются в одну запись и включаются в таблицу результатов запроса.

Кроме этих записей в таблицу результатов будут включены и те записи из первой таблицы, для которых не нашлось соответствия во второй. Вместо них присоединяются пустые поля.

Клиенты, заказы: результат внешнего соединения таблиц 3.4 и 3.5.

Наименование	Код_клиента	Код_клиента	Код_товара	Наименование
Белый ветер	23	23	56	Принтер
Формоза	45			
Партия	12	12	37	Мышь

Примечание. Слева расположена ведущая таблица

Здесь, во второй записи левой таблицы для поля *Код_клиента,* не нашлось равного значения поля *Код_клиента* таблицы *Заказы,* и поэтому к ней присоединились пустые поля.

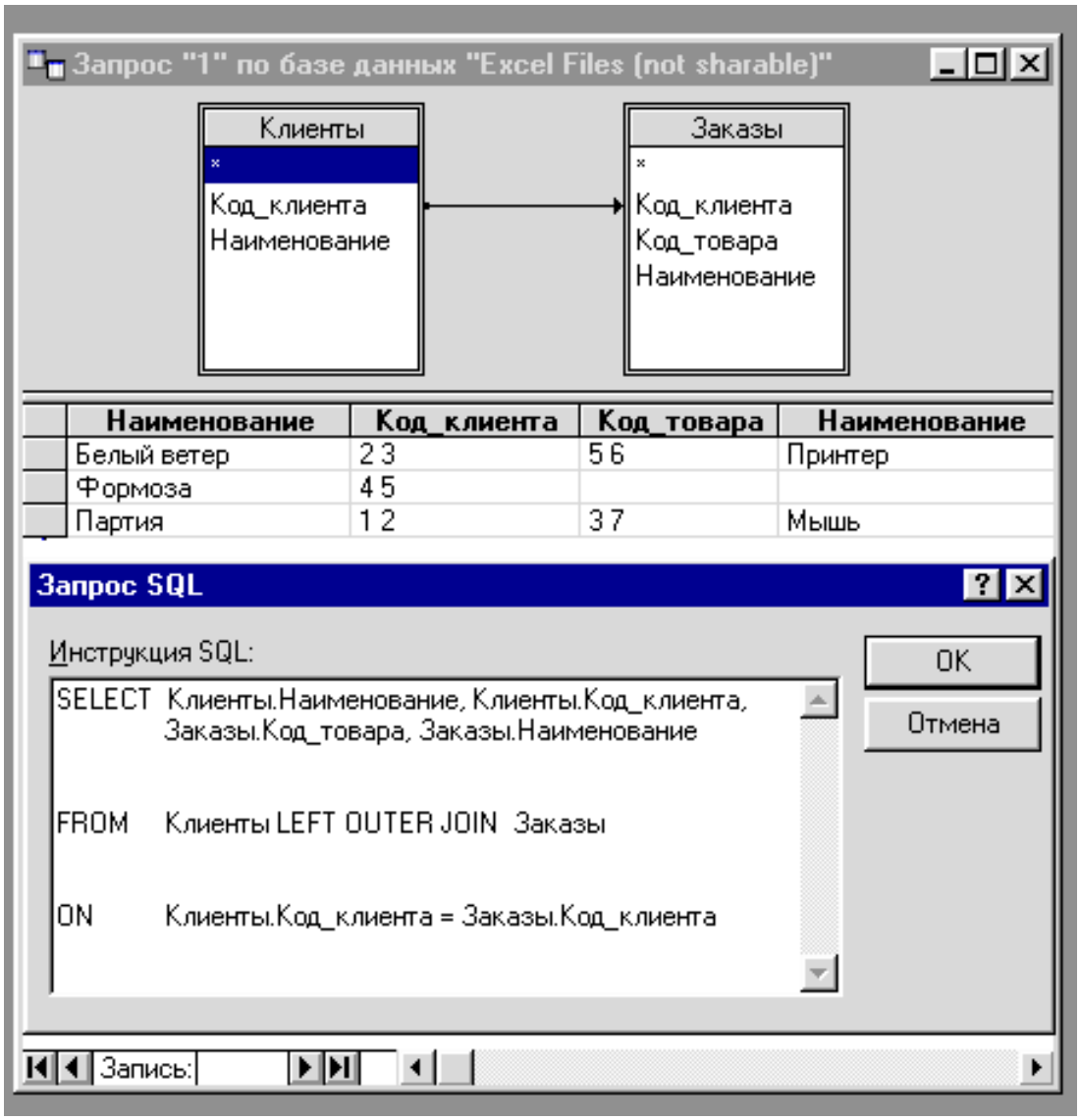


Рис. 3.4. Внешнее соединение таблиц

Для реализации запроса воспользуемся операцией LEFT OUTER JOIN (внешнее соединение с ведущей таблицей, расположенной слева) ON по условию: значения полей *Код_клиента* таблицы *Клиенты* равны значениям полей *Код_клиента* таблицы *Заказы*. Атрибут, обозначенный пунктиром, не включен в список атрибутов, предназначенных для вывода в результирующую таблицу (*Заказы.Код_клиента*).

```
SELECT      Клиенты.Наименование, Клиенты.Код_клиента,
            Заказы.Код_товара, Заказы.Наименование
FROM        Клиенты
LEFT OUTER  Заказы
JOIN
ON          Клиенты.Код_клиента = Заказы.Код_клиента
```

Окно запроса представлено на рис. 3.4. Внешнее соединение на рисунке изображено стрелкой, слева направо соединяющей атрибуты *Клиенты.Код_клиента* и *Заказы.Код_клиента.* По этим атрибутам происходит отбор записей, предназначенных для соединения в запросе.

Внешнее соединение может осуществляться другим способом. Если указать в запросе, что ведущей таблицей является правая таблица, то получим следующий результат.

Клиенты и заказы: Результат внешнего соединения таблиц 3.4 и 3.5.

Наименование	Код_клиента	Код_клиента	Код_товара	Наименование
Белый ветер	23	23	56	Принтер
		31	98	Клавиатура
Партия	12	12	37	Мышь

Прмечание. Справа расположена ведущая таблица.

В данном запросе вторая запись правой таблицы осталась без пары из левой таблицы, и поэтому к ней присоединились пустые поля. Все записи из правой таблицы вошли в результирующую таблицу.

Запрос реализуется операцией RIGHT OUTER JOIN (внешнее соединение с ведущей таблицей, расположенной справа) ON по условию: значения полей *Код_клиента* таблицы *Клиенты* равны значениям полей *Код_клиента* таблицы *Заказы*. Атрибут, обозначенный пунктиром, не включен в список атрибутов, предназначенных для вывода в результирующую таблицу (*Клиенты.Код_клиента*).

```
SELECT          Клиенты.Наименование, Клиенты.Код_клиента,
                Заказы.Код_товара, Заказы.Наименование
FROM            Клиенты
RIGHT OUTER     Заказы
JOIN
ON              Клиенты.Код_клиента = Заказы.Код_клиента
```

Внешнее соединение на рисунке 3.5 изображено стрелкой, справа налево соединяющей атрибуты *Клиенты.Код_клиента* и *Заказы.Код_клиента.* По этим атрибутам происходит отбор записей, предназначенных для соединения в запросе.

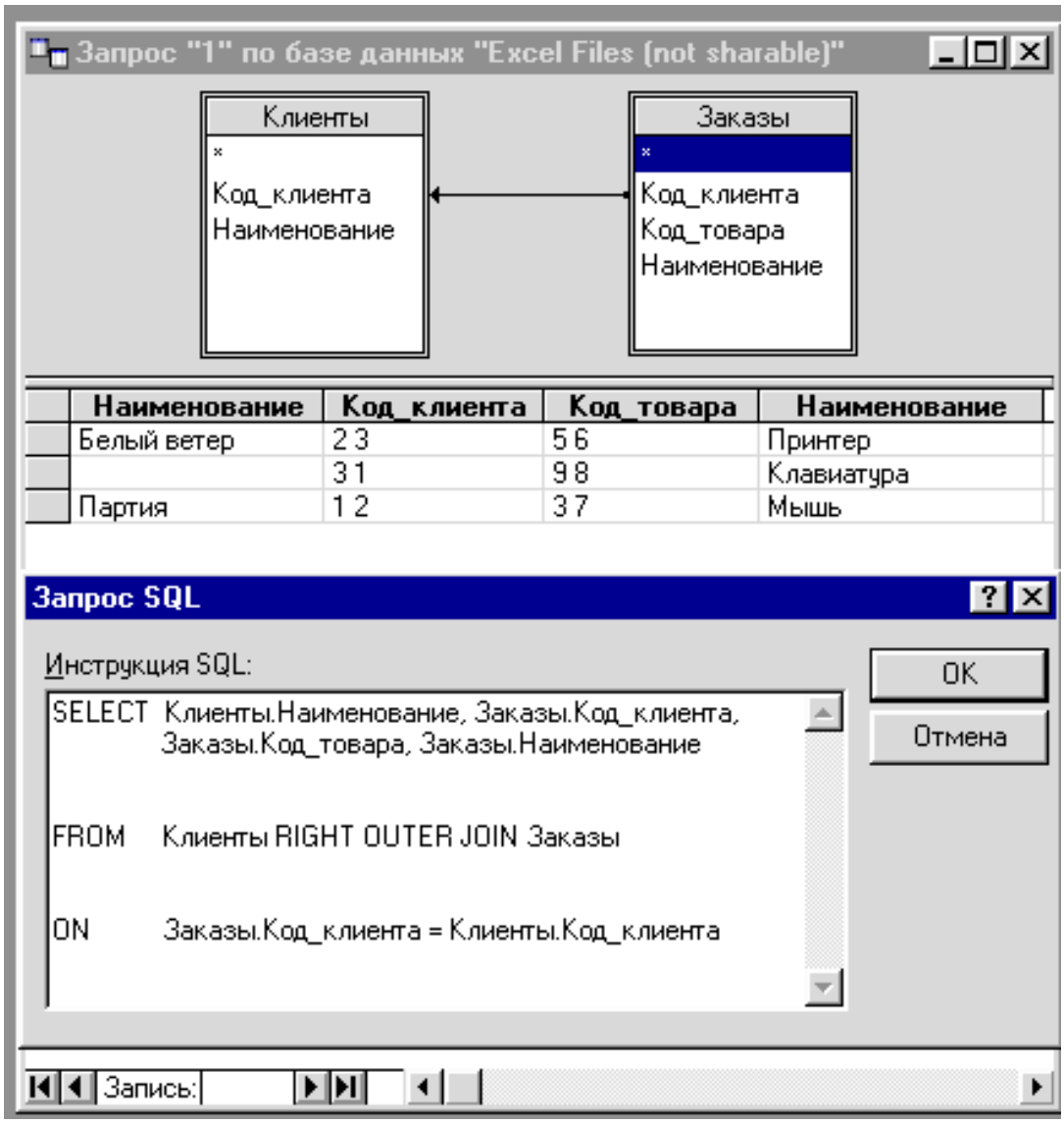


Рис. 3.5. Пример внешнего соединения

Декартово произведение реляционных таблиц

Рассмотрим две исходные таблицы. Таблица 3.6 - список студентов учебной группы факультета международных отношений, таблица 3.7 - расписание экзаменов.

Таблица 3.6.

Группа_МО1-04

Студент	*Зач_книжка*
Ачикян П.	115-98
Байбулатов Р.	211-98
Кострош Я.	110-98
Кравченко В.	132-98
Логинова Ю.	121-98
Федотов И.	119-98

Таблица 3.7.

Расписание_Экзаменов

Дисциплина	*Дата*	*Оценка*	*Подпись*
Информатика	04.01. 2000		
История	13.01. 2000		

К началу экзаменационной сессии необходимо приготовить экзаменационные ведомости для каждой группы и для каждого экзамена. Применим операцию декартова произведения для реализации данного запроса. При этом результирующая таблица будет содержать записи, которые являются результатом слияния каждой записи таблицы 3.6 с каждой записью таблицы 3.7. Таблицы - сомножители декартова произведения могут иметь разное количество атрибутов, но результирующая таблица содержит все атрибуты обеих исходных таблиц - сомножителей. Количество записей в результирующей таблице равно произведению количеств записей первой и второй исходных таблиц.

Запрос будем формировать с помощью инструкции SELECT, которая указывает, какие атрибуты нужно показать в результирующей таблице. Предложение FROM содержит имена таблиц, являющихся источниками данных. Если в предложении FROM содержится более одной таблицы, то они будут преобразованы в одну таблицу по правилу декартового (прямого) произведения.

Для решения нашей задачи в предложении SELECT перечислим все столбцы первой и второй таблиц, так как нам нужно, чтобы они присутствовали в результирующей таблице. В предложении FROM укажем названия двух исходных таблиц, из которых будет сформирована одна результирующая таблица по правилу декартового произведения.

```
SELECT    Группа_МО1_04.Студент,
          Группа_МО1_04.Зач_книжка,
          Расписание_Экзаменов.Дисциплина,
          Расписание_Экзаменов.Дата,
          Расписание_Экзаменов.Оценка,
          Расписание_Экзаменов.Подпись
FROM      Группа_МО1_04, Расписание_Экзаменов
```

После выполнения такого запроса из двух исходных таблиц, перечисленных в предложении FROM, получим следующую таблицу.

Студент	Зач_книжка	Дисциплина	Дата	Оценка	Подпись
Ачикян П.	115-98	Информатика	04.01.2000		
Ачикян П.	115-98	История	13.01.2000		
Байбулатов Р.	211-98	Информатика	04.01.2000		
Байбулатов Р.	211-98	История	13.01.2000		
Кострош Я.	110-98	Информатика	04.01.2000		
Кострош Я.	110-98	История	13.01.2000		
Кравченко В.	132-98	Информатика	04.01.2000		
Кравченко В.	132-98	История	13.01.2000		
Логинова Ю.	121-98	Информатика	04.01.2000		
Логинова Ю.	121-98	История	13.01.2000		
Федотов И.	119-98	Информатика	04.01.2000		
Федотов И.	119-98	История	13.01.2000		

Получена таблица, которая содержит 12 записей, то есть шесть записей из таблицы 3.6, умноженные на две записи из таблицы 3.7. Заметим, что к фамилии каждого студента присоединилась информация о дисциплине, дате и пустые поля для простановки оценки и подпись преподавателя. Однако требуется отдельная ведомость на каждый экзамен. Для этого из общей таблицы необходимо отобрать только те записи, которые относятся к конкретному экзамену. Предположим, что готовится ведомость для экзамена по информатике. Для указания критерия отбора записей из общей таблицы, относящихся только к информатике, воспользуемся предложением WHERE, в котором укажем, что необходимо отобрать записи, в которых значение поля *Расписание_Экзаменов.Дисциплина* должно быть равно значению «Информатика». Дополним запрос предложением WHERE, как показано ниже, с указанием критерия отбора записей. Чтобы не перечислять все выходные атрибуты, в инструкции SELECT можно указать имя таблицы, к которой они относятся и вместо названия атрибута поставить символ *, что означает вывод всех атрибутов данной таблицы. Пользоваться ведомостью удобнее, когда записи отсортированы по фамилиям студентов в алфавитном порядке. Для этого используем предложение ORDER BY в конце инструкции SELECT с указанием поля, по которому будет производиться сортировка, как показано на рис.3.6.

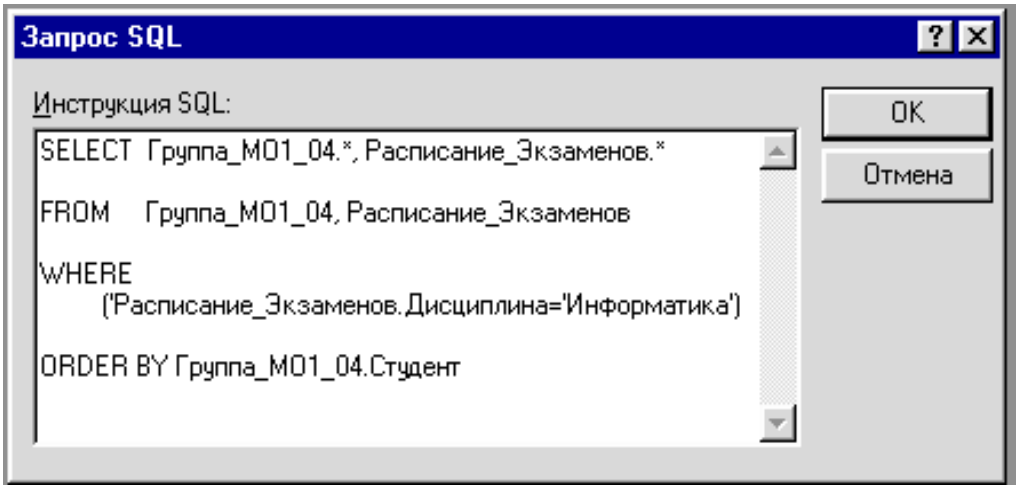


Рис.3.6. Пример создания ведомости для экзамена

```
SELECT      Группа_МО1_04.*,
            Расписание_Экзаменов.*
FROM        Группа_МО1_04, Расписание_Экзаменов
WHERE       (Расписание_Экзаменов.Дисциплина = 'Информатика')
ORDER BY    Группа_МО1_04.Студент
```

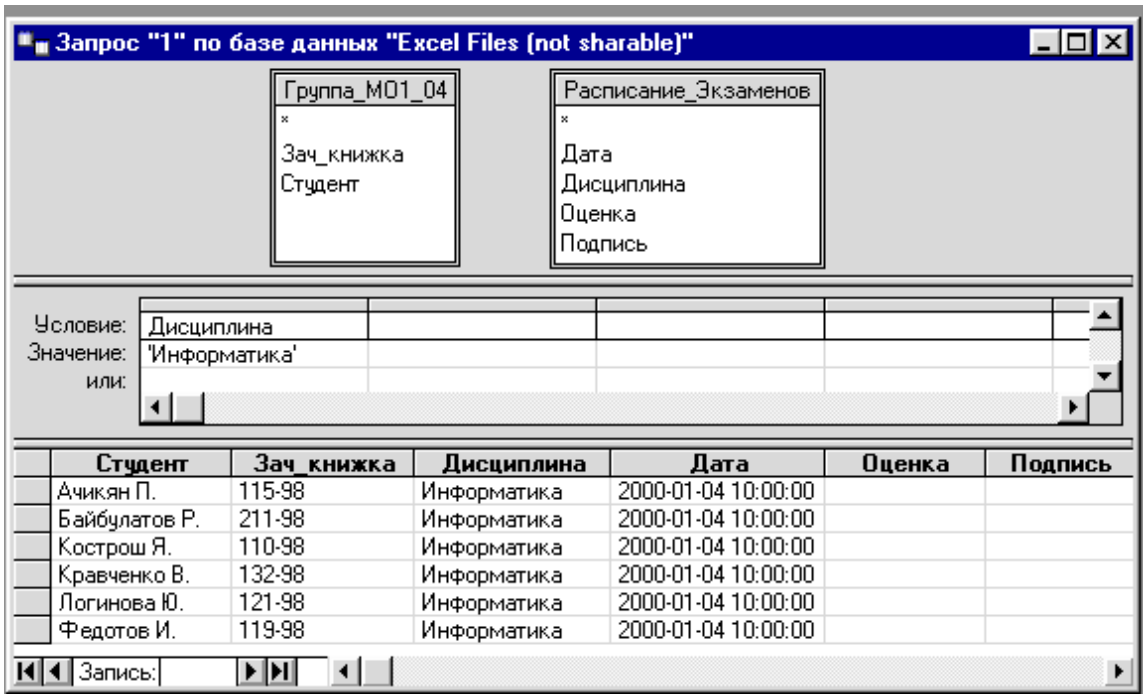


Рис.3.7. Окно «Запроса SQL»

На рисунке 3.7 показано, как текст запроса ввести в окно запроса SQL. Таким образом получена экзаменационная ведомость из двух исходных таблиц: расписание экзаменов и список студентов учебной группы. При формулировании запроса использовалось декартово произведение двух таблиц, отбор записей по условию и сортировка по алфавиту. Совокупность этих предложений является текстом инструкции SELECT.

Операция присвоения имени таблице данных

В предыдущем примере получена выборка, которая может быть использована в качестве экзаменационной ведомости. Чтобы сохранить полученную таблицу, ей необходимо присвоить имя, в данном случае *Экз_ведом-информатика*, и указать адрес файла для её размещения. Затем можно оформить документ, снабдив его необходимыми реквизитами, такими как название документа, фамилии экзаменаторов, фамилия, должность и место для подписи ответственного лица. Готовый документ распечатывается. По результатам опроса студентов проставляются оценки, экзаменаторы подписываются. Оценки вводятся в ранее сохранённую таблицу *Экз_ведом-информатика.* Эта таблица, дополненная новыми данными, в дальнейшем может быть использована совместно с другими таблицами базы данных в управлении учебным процессом.

Для присвоения имени таблице воспользуемся инструкцией "SELECT…INTO" (запрос на создание таблицы). Эта инструкция создаёт новую таблицу, пользуясь значениями, выбранными из одной или нескольких таблиц. Запросы на создание таблицы особенно полезны в тех случаях, когда создаётся резервная копия таблицы либо в конце отчётного периода создаётся таблица с итогами. Следует обратить особое внимание на то, что в EXCEL новая таблица создаётся в сохранённом файле, который содержит исходные таблицы запроса. Например, файл Faculty, размещённый на диске D, содержит исходные таблицы *Группа_МО1_04* и *Расписание_Экзаменов* для создания по запросу новой таблицы *Экз_ведом-информатика*, которая будет сохранена в том же файле Faculty. Последовательность действий в EXCEL следующая:

1. Не открывая файл Faculty, выполняем команду *Данные-Внешние Данные-Создать Запрос*.
2. Появляется диалоговое окно *Выбор источника данных,* выделяем строку Excel Files, как показано на рис. 3.8.

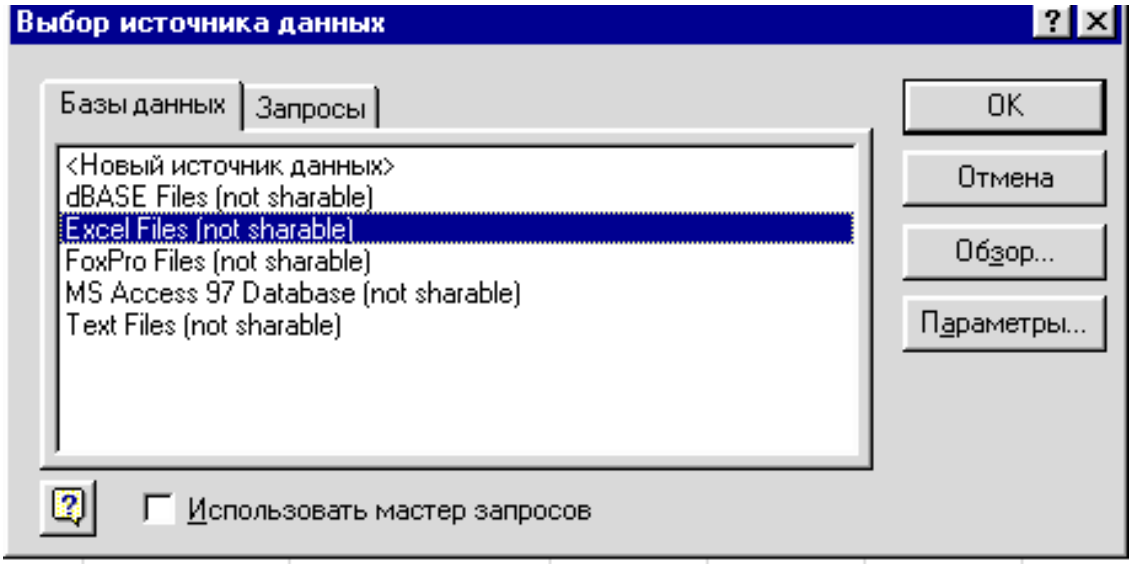


Рис. 3.8. Окно «Выбор источника данных»

3. В появившемся окне *Выбор файла* необходимо пометить файл Faculty на диске D (рис.3.9).

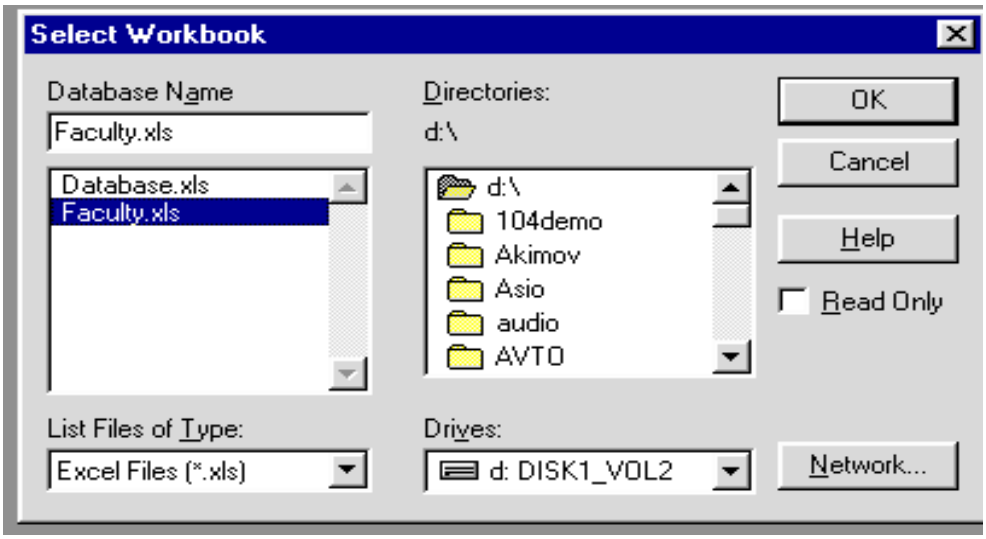


Рис. 3.9. Окно «Выбор файла рабочей книги»

4. Далее появляется окно, в котором нужно выбрать из списка таблиц файла Faculty.xls исходные таблицы данных: *Группа_МО1_04* и *Расписание_Экзаменов.* После нажатия кнопки SQL на панели инструментов вводим запрос SQL, как показано на рис.3.10.

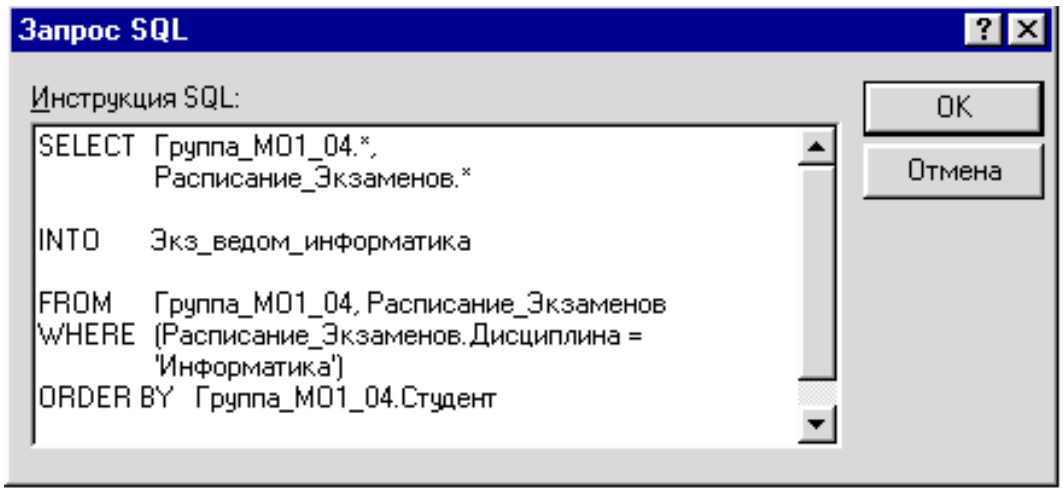


Рис.3.10. Окно «SQL запроса»

Имя новой таблицы *Экз_ведом_информатика* задаётся после предиката INTO инструкции SELECT.

Операция деления реляционных таблиц данных.

На входе операции деления имеются две таблицы, например A и B. Пусть таблица A, называемая делимым, содержит атрибуты (A_1, A_2, …, A_N). Таблица B-делитель содержит подмножество атрибутов таблицы A: A_1, A_2, …, A_K (k<n). Результирующая таблица - частное от деления C определена на атрибутах таблицы A, которых нет в B: A_{K+1}, A_{K+2}, … , A_N. Запись включается в результирующую таблицу C только тогда, если её декартово произведение с B содержится в делимом-таблице A.

Операцию деления таблиц рассмотрим, используя данные предыдущего раздела. После сессии по результатам сдачи экзаменов образовалась следующая сводная таблица с оценками, которая использовалась в предыдущем разделе для получения экзаменационной ведомости:

Студент	Зач_книжка	Дисциплина	Дата	Оценка
Ачикян П.	115-98	Информатика	04.01.2000	хорошо
Ачикян П.	115-98	История	13.01.2000	отлично
Байбулатов Р.	211-98	Информатика	04.01.2000	отлично
Байбулатов Р.	211-98	История	13.01.2000	хорошо
Кострош Я.	110-98	Информатика	04.01.2000	хорошо
Кострош Я.	110-98	История	13.01.2000	хорошо
Кравченко В.	132-98	Информатика	04.01.2000	отлично
Кравченко В.	132-98	История	13.01.2000	отлично
Логинова Ю.	121-98	Информатика	04.01.2000	хорошо
Логинова Ю.	121-98	История	13.01.2000	отлично
Федотов И.	119-98	Информатика	04.01.2000	отлично
Федотов И.	119-98	История	13.01.2000	хорошо

Эта таблица будет делимым. Таблицу делимое обозначим А. Она определена на множестве атрибутов, состоящим из $n = 5$ элементов: *Студент, Зач_книжка, Дисциплина, Дата, Оценка.* Необходимо узнать фамилии студентов, которые получили по информатике оценку «отлично», а по истории – «хорошо». Создадим таблицу В - делитель:

Дисциплина	Оценка
Информатика	отлично
История	хорошо

Эта таблица содержит $k = 2$ атрибута, которые являются подмножеством множества атрибутов таблицы А (k<n): *Дисциплина, Оценка.* В результате деления таблицы А на таблицу В должна получится таблица С, которая содержит список студентов, получивших по двум указанным в таблице В предметам соответствующие оценки. Таблица С - частное от деления содержит два атрибута: *Студент, Зач_книжка.* Эти атрибуты, с одной стороны, являются подмножеством множества атрибутов таблицы А, с другой - отличаются от атрибутов таблицы В.

Глава 4. База данных в Excel

4.1. Создание базы данных в Excel

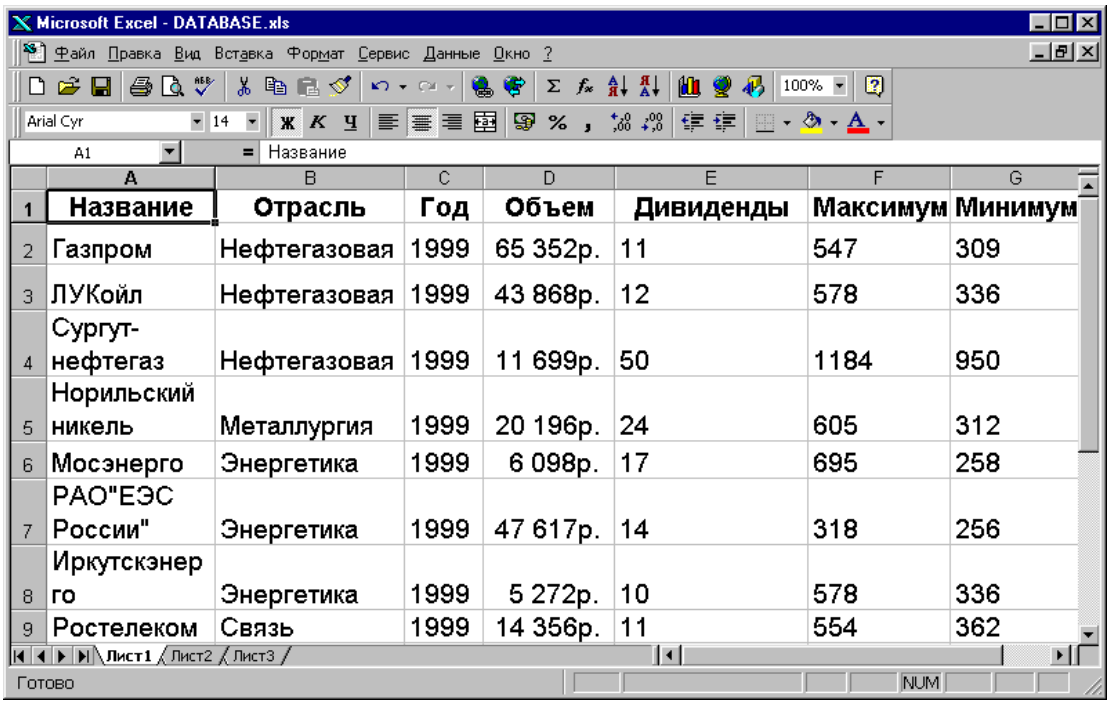
Используя возможности Excel, построим базу данных, предназначенную для хранения данных о курсах акций различных акционерных обществ. Углубленный анализ этих данных поможет проводить эффективную торговлю ценными бумагами.

Информация из базы данных с записями курсов акций может быть извлечена и обработана посредством функций обработки базы данных.

База данных в Excel, так же как и другие базы данных, представляет информацию, которая упорядочена определенным образом в рамках заданной структуры. Структура базы и форматы данных устанавливаются перед вводом информации.

База данных в Excel представляет собой таблицу, строки которой являются ЗАПИСЯМИ, а отдельные элементы записи, содержащиеся в каждой ячейке таблицы, – ПОЛЯМИ ЗАПИСИ. Названия столбцов таблицы представляют собой ИМЕНА ПОЛЕЙ.

В Excel для базы данных отводится область обычной таблицы. Так как таблица не может иметь более чем 16384 строк и 256 колонок, Excel-база данных не может включать в себя более 16383 записей (одна строка выделяется под заголовок имен полей), причем в каждой их них не может быть более 256 полей. Для небольших приложений этого вполне хватает.



	A	B	C	D	E	F	G
1	**Название**	**Отрасль**	**Год**	**Объем**	**Дивиденды**	**Максимум**	**Минимум**
2	Газпром	Нефтегазовая	1999	65 352р.	11	547	309
3	ЛУКойл	Нефтегазовая	1999	43 868р.	12	578	336
4	Сургут-нефтегаз	Нефтегазовая	1999	11 699р.	50	1184	950
5	Норильский никель	Металлургия	1999	20 196р.	24	605	312
6	Мосэнерго	Энергетика	1999	6 098р.	17	695	258
7	РАО"ЕЭС России"	Энергетика	1999	47 617р.	14	318	256
8	Иркутскэнерго	Энергетика	1999	5 272р.	10	578	336
9	Ростелеком	Связь	1999	14 356р.	11	554	362

Рис.4.1. Пример базы данных в Excel

Построим базу данных, в которой каждая запись содержит информацию о некоторой компании и состоит из семи полей: НАЗВАНИЕ, ОТРАСЛЬ, ГОД, ОБЪЕМ, РОСТ, МАКСИМУМ, МИНИМУМ (рис.4.1). Этот файл Shares.xls находится на сервере на диске Z, в папке DATABASE.

4.2. Сортировка данных

Для выполнения сортировки базы данных по определенному признаку необходимо задать диапазон данных, используемых для сортировки, и с помощью имен полей определить критерии сортировки. Для каждого столбца таблицы базы данных можно задать свой способ сортировки. Имя поля, по которому проводится сортировка, называется КЛЮЧОМ СОРТИРОВКИ. В Excel существует возможность провести упорядочивание записей, используя до трех ключевых полей для одного процесса сортировки в убывающей или возрастающей последовательности.

Для того чтобы выполнить сортировку базы данных, нужно выделить любую ячейку этого диапазона данных и выбрать команду Сортировка из меню

Данные. Вся база данных, включающая выделенную ячейку, автоматически подлежит сортировке.

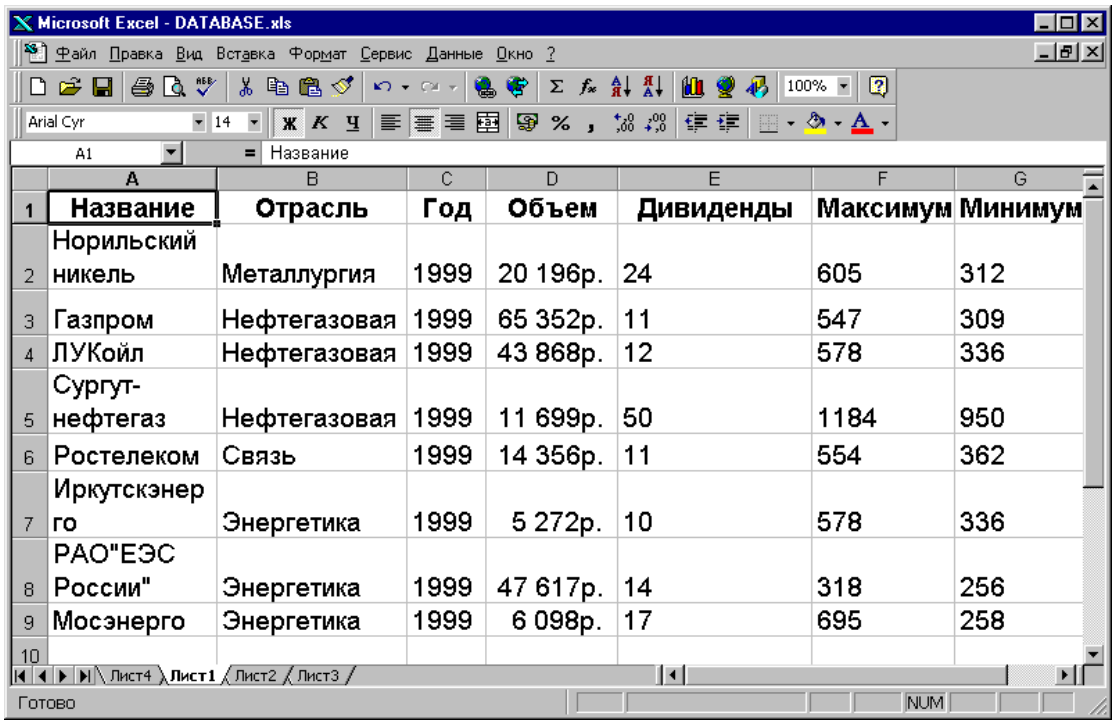
Пример. Требуется отсортировать записи базы данных (рис. 4.1) по курсам акций в алфавитном порядке названий компаний.

Выделяя ячейку **Название** (можно выбрать и любую другую ячейку базы данных), в меню Данные нужно выбрать команду Сортировка. В группе Сортировать по следует щелкнуть на стрелке «вниз» и в списке заголовков столбцов выделить строку **Название,** и включить опцию По возрастанию.

В результате выполнения команды ОК записи базы будут расположены в алфавитном порядке названия компаний.

Заметим, что при сортировке строки перемещаются на новые позиции и им присваиваются новые порядковые номера.

Иногда требуется использовать более одного критерия сортировки. Например, если требуется отсортировать компании в алфавитном порядке отраслей, а внутри отрасли расположить в порядке возрастания дивидендов, используются два ключа сортировки **Отрасль** и **Дивиденды** (рис.4.2).



	A	B	C	D	E	F	G
1	**Название**	**Отрасль**	**Год**	**Объем**	**Дивиденды**	**Максимум**	**Минимум**
2	Норильский никель	Металлургия	1999	20 196р.	24	605	312
3	Газпром	Нефтегазовая	1999	65 352р.	11	547	309
4	ЛУКойл	Нефтегазовая	1999	43 868р.	12	578	336
5	Сургут-нефтегаз	Нефтегазовая	1999	11 699р.	50	1184	950
6	Ростелеком	Связь	1999	14 356р.	11	554	362
7	Иркутскэнерго	Энергетика	1999	5 272р.	10	578	336
8	РАО"ЕЭС России"	Энергетика	1999	47 617р.	14	318	256
9	Мосэнерго	Энергетика	1999	6 098р.	17	695	258
10							

Рис.4.2. Пример сортировки по двум ключам

4.3. Фильтрация данных. Автофильтр

Для поиска нужной информации в Excel предусмотрены также функции фильтрации записей в меню Данные Фильтр Автофильтр и Расширенный Фильтр.

С помощью имен полей записей можно автоматически отфильтровать и вывести на экран только нужные данные. Режим автоматической фильтрации можно включить, выделив некоторую ячейку списка записей, а затем выбрать команду Автофильтр. Когда выделена одна из ячеек записи на рабочем листе, Excel выполняет фильтрацию всего списка в целом. Далее с помощью имен полей следует определить условия, по которым будет выполняться фильтрация базы данных. Получив необходимые данные, можно отключить Автофильтр, выбрав эту же команду.

Пример. Выбрать наиболее прибыльные компании за 1999 год. В меню Данные выбрать команду Фильтр-Автофильтр. В заголовках каждого столбца данных появятся кнопки фильтра со стрелочками вниз. Далее нужно щелкнуть на стрелке фильтра в ячейке с заголовком **Отрасль**. В появившемся окне списка критериев следует выбрать нужный критерий (ключ) фильтрации. Критерии фильтрации находятся в списке в алфавитном порядке. В дополнение к ним имеется еще пять ключей: Все, Первые10, Условие, Пустые, Непустые. Эти ключи позволяют найти в списке десять записей, ведущих по выбранному показателю: записи, удовлетворяющие заданному пользователем условию; записи, не содержащие никакой информации в данном поле; записи, ячейки выделенного поля которых не пусты. Например, в качестве ключа фильтрации в списке поля **Отрасль** выделяется слово «Нефтегазовая».

В базе данных останутся названия компаний только нефтегазовой отрасли. Все остальные записи таблицы не удалены, а лишь временно скрыты. Порядковые номера строк в таблице теперь не являются последовательными. Они выделены на листе синим цветом. Чтобы снова вывести на экран полный список, нужно либо щелкнуть на любой кнопке со стрелкой фильтра и в списке

критериев фильтрации выделить строку Все, либо в меню Данные выбрать команду Фильтр-Показать все.

Составим теперь список трех наиболее доходных компаний. В поле **Дивиденды** нужно установить критерий фильтрации Первые 10. Появится диалоговое окно Наложение условия по списку. Необходимо установить параметры окна в группе вывести так: «3», «наибольших», «элементов списка» и щелкнуть на кнопке ОК.

На рабочем листе останется список компаний, значения в поле **Дивиденды** которых составляют три наибольших в полном списке компаний (рис.4.3).

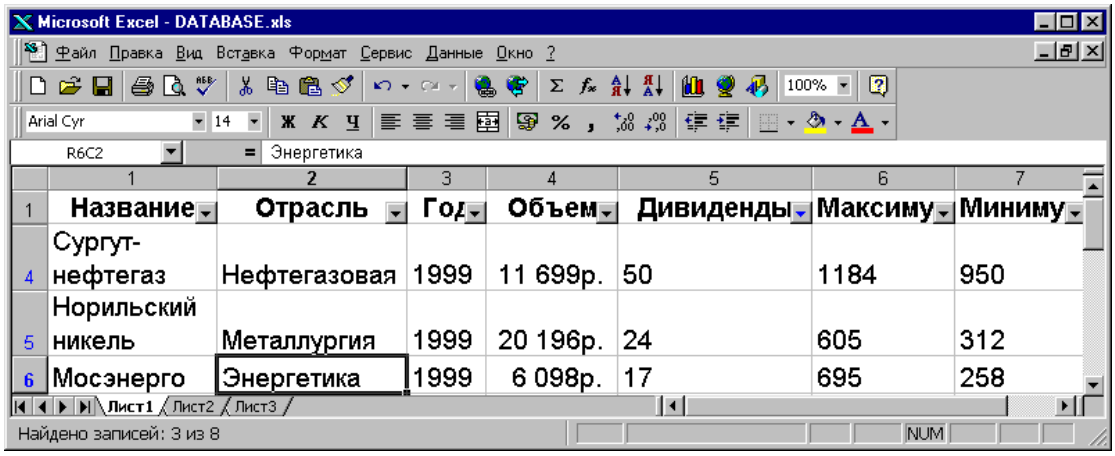


Рис.4.3. Пример фильтрации данных в Excel

Восстановление полного списка необходимо сделать, выбрав команду Фильтр-Показать все в меню Данные.

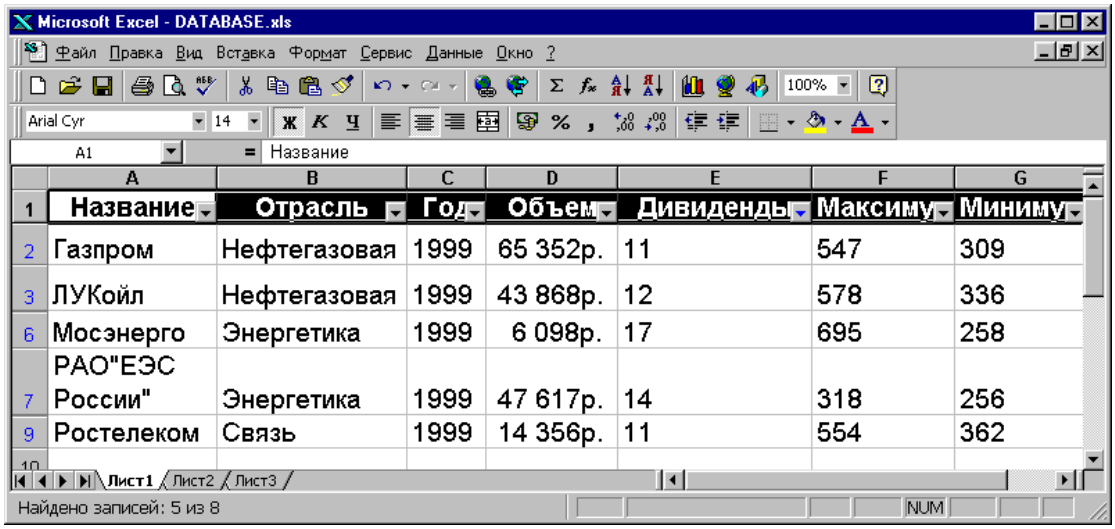
4.4. Фильтрация данных. Поиск по критерию

Часто требуется найти информацию, удовлетворяющую критериям, не входящим в список Автофильтра. Например, если нужно найти список компаний дивиденды акций которых за 1999 год находятся в диапазоне от 10 до 20 рублей за акцию, то в списке ключей фильтрации в поле **Дивиденды** и в

появившемся диалоговом окне «Пользовательский автофильтр» требуется набрать условие > 10 и <20. Для выбора условия можно использовать операторы, такие как равно (=), больше чем (>) или меньше чем (<), а также операторы нестрогих неравенств.

Для выполнения данного примера нужно щелкнуть на стрелку фильтра поля **Название**. В списке ключей выбрать Условие, в результате чего откроется диалоговое окно «Пользовательский автофильтр». В группе Название в верхнем поле операторов нужно выбрать знак «>» и в соседнее поле ввести число 10. Далее следует включить опцию логической функции И (логическое умножение) и, щелкая на стрелке нижнего поля операторов, выбрать знак «<», а в нижнем поле критериев нужно ввести число 20.

После выполнения команды ОК получатся отфильтрованные записи базы данных, удовлетворяющие заданному критерию фильтрации (рис. 4.4.)



	A	B	C	D	E	F	G
1	Название	Отрасль	Год	Объем	Дивиденды	Максиму	Миниму
2	Газпром	Нефтегазовая	1999	65 352р.	11	547	309
3	ЛУКойл	Нефтегазовая	1999	43 868р.	12	578	336
6	Мосэнерго	Энергетика	1999	6 098р.	17	695	258
7	РАО"ЕЭС России"	Энергетика	1999	47 617р.	14	318	256
9	Ростелеком	Связь	1999	14 356р.	11	554	362

Рис 4.4. Фильтрация данных по критерию

4.5. Фильтрация данных. Расширенный фильтр

Данные, отфильтрованные с использованием функции Автофильтр, помещаются в таблице на месте исходных данных. Часто требуется иметь несколько результатов поиска по критерию одновременно, например для того,

чтобы продолжить работу над ними, используя формулы, функции или просто распечатать их. В этом случае используются возможности Расширенного фильтра. Также Расширенный фильтр применяется для извлечения из базы данных информации, пригодной для дальнейшего использования другими средствами Excel.

Для использования возможностей «Расширенного фильтра» сначала необходимо определить область критериев (условий) внутри обрабатываемого рабочего листа Excel и в ней сформулировать необходимые критерии поиска. Область критериев должна быть установлена таким образом, чтобы она не мешала расширению базы данных. Как правило, ее создают перед или рядом с областью базы данных. Она должна иметь размер по крайней мере в две строки и одну колонку. Первая строка области критериев должна содержать имя критерия. Сюда, как правило, переносятся все имена полей базы данных или часть из них. В строках под именем поля формулируется критерий поиска. В зависимости от сложности запроса критерии могут занимать и несколько строк, например в случае объявления условий ИЛИ (логического сложения).

В области критериев каждая строка представляет собой критерий поиска. Полный критерий поиска, заданный всей областью критериев, состоит из объединенных с помощью логической операции ИЛИ условий, заданными отдельными строками критериев. Различные условия, заданные в одних и тех же повторяющихся полях, объединяются с помощью логической операции И.

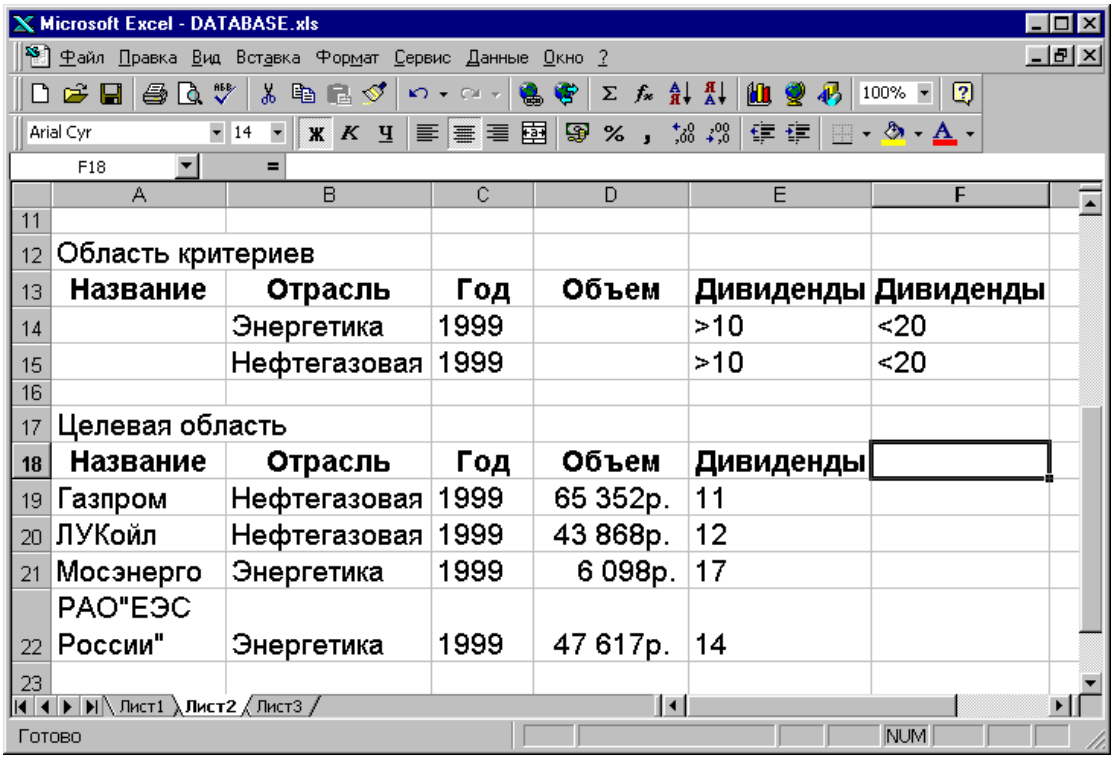
Например, если требуется найти в базе данных все предприятия нефтегазовой или металлургической отрасли, то в области критериев в столбце Отрасль в первой строке должно стоять «Нефтегазовая», а в следующей - «Металлургия». Для извлечения из базы данных информации о компаниях нефтегазовой отрасли, у которых за 1999 дивиденды превышали значение 15, сформируем критерий поиска, показанный на рис.4.5.

Прежде чем приступать к поиску в соответствии с сформулированными критериями, на листе следует предусмотреть еще одну область, в которой Excel сможет скопировать найденные данные. Она называется ЦЕЛЕВОЙ ОБЛАСТЬЮ. Ее также следует располагать так, чтобы избегать конфликтов с частями таблицы, выделенными под базу и критерии. В первую строку целевой области следует ввести имена полей, содержимое которых требуется определить в найденных записях. Если же требуется найти записи целиком, то в целевую область следует скопировать из области базы данных строку-заголовок со всеми именами полей. Если при задании целевой области будет задано кроме строки заголовка еще несколько строк (например пять), то тем самым при поиске в базе будут найдены только первые пять записей, удовлетворяющих критериям поиска. В случае, если будет найдено более пяти записей, то система покажет только первые пять и выдаст сообщение о том, что существуют и другие записи, удовлетворяющие критериям поиска.

Итак, для работы с Расширенным фильтром требуется: задать диапазон базы данных, создать область критериев и сформулировать сами критерии, выбрать целевую область.

Пример. Требуется найти в базе данных записи компаний, принадлежащих отраслям Нефтегазовая и Энергетика дивиденды, которых за 1999 год находились в диапазоне от 10 до 20 рублей за одну акцию. Необходимо отметить, что в диалоговом окне «Расширенный фильтр» в разделе Обработка следует устанавливать опцию Скопировать результат в другое место. В Исходный диапазон задается область базы данных, в Диапазон условий – область критериев и целевая область задается в разделе Поместить результат в диапазон. В диалоговом окне «Расширенного фильтра» имеется опция Только уникальные записи, которая позволяет устранить повторную выборку из базы одинаковых записей. Если в область критериев, ниже названий полей будет входить пустая строка, это будет означать, что в ней отсутствует условия выбора по полям записи базы данных. С точки зрения условий поиска пустая строка задает

критерий, которому удовлетворяют все записи базы данных. Пустая строка будет объединяться логическим условием ИЛИ с предыдущими условиями, в результате чего в целевую область будут извлечены все записи базы данных. Результаты поиска приведены на рис. 4.5.



	A	B	C	D	E	F
11						
12	Область критериев					
13	**Название**	**Отрасль**	**Год**	**Объем**	**Дивиденды**	**Дивиденды**
14		Энергетика	1999		>10	<20
15		Нефтегазовая	1999		>10	<20
16						
17	Целевая область					
18	**Название**	**Отрасль**	**Год**	**Объем**	**Дивиденды**	
19	Газпром	Нефтегазовая	1999	65 352р.	11	
20	ЛУКойл	Нефтегазовая	1999	43 868р.	12	
21	Мосэнерго	Энергетика	1999	6 098р.	17	
22	РАО"ЕЭС России"	Энергетика	1999	47 617р.	14	
23						

Рис.4.5. Найденные записи данных

4.6. Обработка информации с помощью формы данных

При заполнении базы построчно, записи поле за полем, вводятся в таблицу. Однако существует другой путь, который полезен тогда, когда разработчик организует базу данных для других пользователей. Речь идет о форме данных (шаблоне ввода) как средстве поддержания диалога в процессе работы с базой данных. Форма данных используется при заполнении базы, выполнении поиска в базе и внесении в нее изменений.

Для ввода данных, изменения, удаления или поиска в базе с помощью формы данных необходимо выполнить команду работы с формой в меню Данные-Форма.

Форма данных предоставляет возможность поиска информации. Например, если необходимо отобрать предприятия, которые платили более 11 рублей за каждую акцию, следует в диалоговой части Формы данных в поле ввода Год ввести в качестве критерия поиска 1999, а в поле Дивиденды ввести > 11. Excel последовательно покажет записи, удовлетворяющие заданным критериям (рис.4.6).

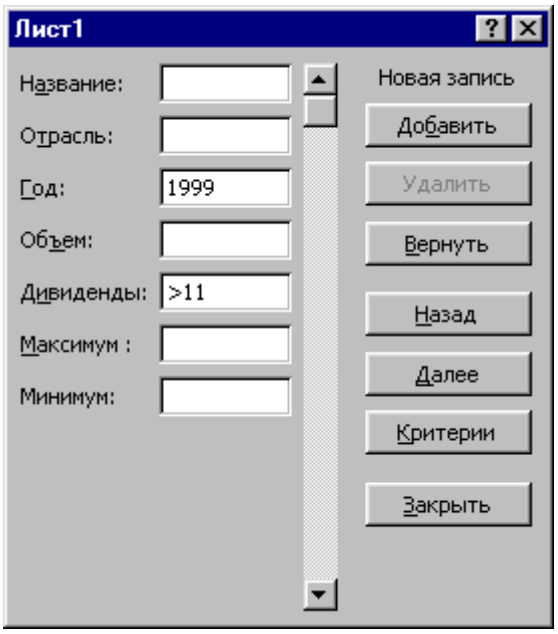


Рис.4.6. Форма данных

4.7. Функции для работы с базой данных

Рассмотрим некоторые функции Excel, предназначенные для обработки баз данных: ДМАКС(), ДМИН(), БДСУММ(), БСЧЁТ(), БСЧЁТА(), ДСРЗНАЧ(), ДСТАНДОТКЛ().

Библиотека Excel содержит набор встроенных функций в разделе Мастер функций – Работа с базой данных, позволяющих получить информацию из базы данных или произвести в ней необходимые вычисления.

Перечислим описания и форматы использования функций базы данных:

ДМАКС(база_данных;поле;критерий) – возвращает максимальное значение среди выделенных фрагментов базы данных;

ДМИН(база_данных;поле;критерий) – возвращает минимальное значение среди выделенных фрагментов базы данных;

БДСУММ(база_данных;поле;критерий) – суммирует числа в поле столбца записей базы данных, удовлетворяющих условию;

БСЧЁТ(база_данных;поле;критерий) – подсчитывает количество числовых ячеек в выборке из заданной базы данных по заданному критерию;

БСЧЁТА(база_данных;поле;критерий) – подсчитывает количество непустых ячеек в выборке из заданной базы данных по заданному критерию;

ДСРЗНАЧ(база_данных;поле;критерий) – возвращает среднее значение выбранных фрагментов базы данных;

ДСТАНДОТКЛ(база_данных;поле;критерий) – оценивает стандартное отклонение по выборке из выделенной части базы данных. Например, стандартное отклонение для базы данных курсов акций компаний будет равно среднему отклонению от среднего значения курса и определяет размах колебаний курсов акций.

Пример. Используя перечисленные функции базы данных, можно вычислить следующие показатели для курсов акций в базе данных – Высший, Низший, Средний, Отклонение (рис.4.7). Области базы удобно присвоить имя

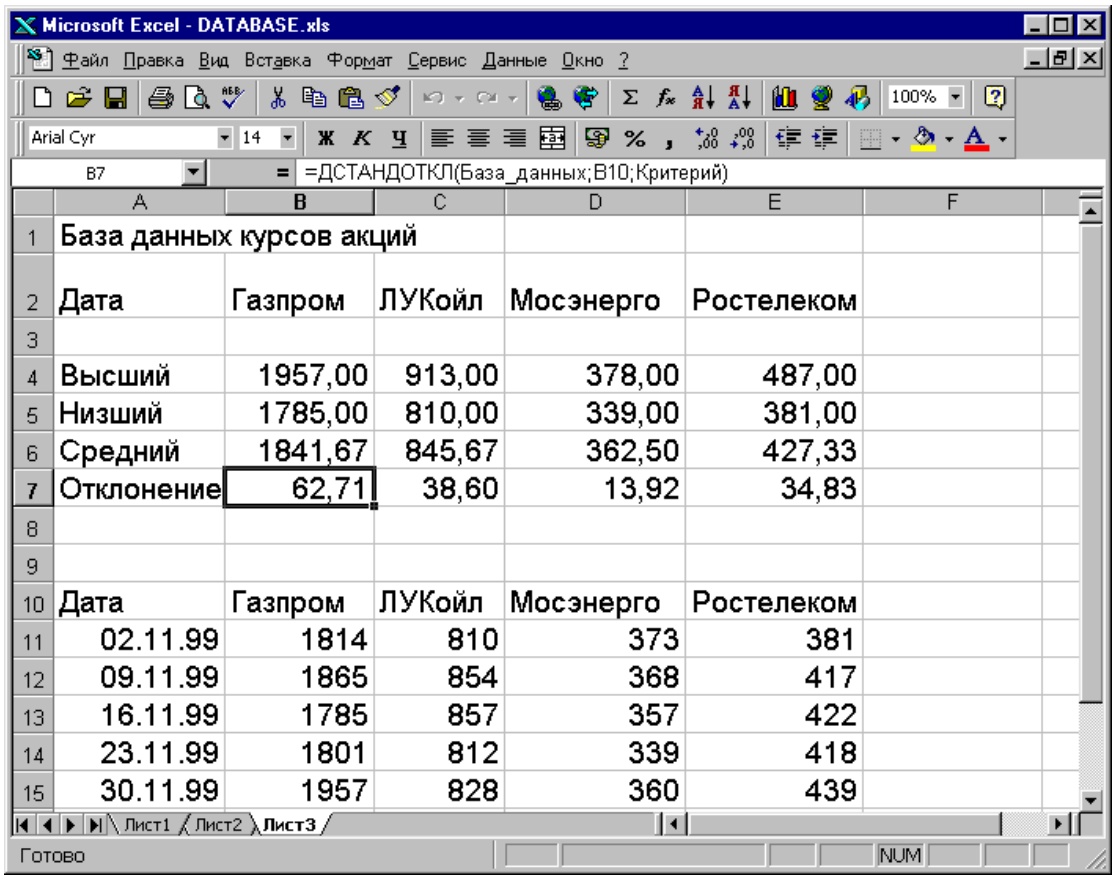
База_данных, а области критериев, содержащий имена полей базы и условия, – *Критерий*. Тогда, например, функция, вычисляющая стандартное отклонение для курса акций Газпром за определенный период времени, будет иметь вид:

ДСТАНДОТКЛ(База_данных;B10;Критерий).

Функция БСЧЁТ(база_данных;поле;критерий) позволяет подсчитать, сколько раз в базе данных в столбце поле встречаются величины, удовлетворяющие критерию поиска. Например, из базы данных курса акций, используя функцию БСЧЁТ(), можно узнать сколько недель курс акций Газпром был выше значения 1810. Функция имеет вид:

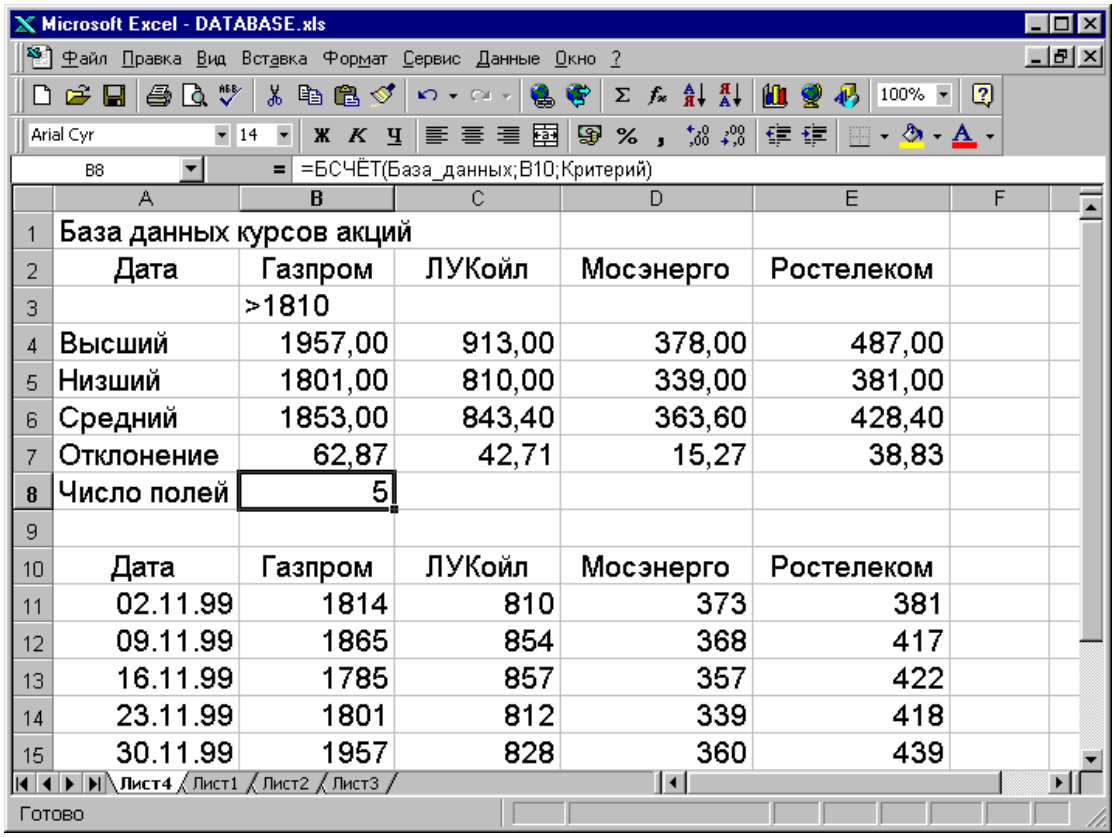
БСЧЁТ(База_данных;B10;Критерий).

Результат вычислений представлен на рис.4.8.



	A	B	C	D	E	F
1	База данных курсов акций					
2	Дата	Газпром	ЛУКойл	Мосэнерго	Ростелеком	
3						
4	Высший	1957,00	913,00	378,00	487,00	
5	Низший	1785,00	810,00	339,00	381,00	
6	Средний	1841,67	845,67	362,50	427,33	
7	Отклонение	62,71	38,60	13,92	34,83	
8						
9						
10	Дата	Газпром	ЛУКойл	Мосэнерго	Ростелеком	
11	02.11.99	1814	810	373	381	
12	09.11.99	1865	854	368	417	
13	16.11.99	1785	857	357	422	
14	23.11.99	1801	812	339	418	
15	30.11.99	1957	828	360	439	

Рис. 4.7. Вычисление стандартного отклонения



	A	B	C	D	E	F
1	База данных курсов акций					
2	Дата	Газпром	ЛУКойл	Мосэнерго	Ростелеком	
3		>1810				
4	Высший	1957,00	913,00	378,00	487,00	
5	Низший	1801,00	810,00	339,00	381,00	
6	Средний	1853,00	843,40	363,60	428,40	
7	Отклонение	62,87	42,71	15,27	38,83	
8	Число полей	5				
9						
10	Дата	Газпром	ЛУКойл	Мосэнерго	Ростелеком	
11	02.11.99	1814	810	373	381	
12	09.11.99	1865	854	368	417	
13	16.11.99	1785	857	357	422	
14	23.11.99	1801	812	339	418	
15	30.11.99	1957	828	360	439	

Рис. 4.8. Вычисление количества записей

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Дейт К. Введение в системы баз данных. Москва, «Наука», 1980.
2. Ульман Дж. Основы систем баз данных Москва, «Мир», 1980.
3. Мартин Дж. Организация баз данных в вычислительных системах. Москва, «Мир», 1980.
4. Хансен Г.,Хансен Д. Базы данных: разработка и управление. Москва, «Бином», 1999.
5. Microsoft Excel для Windows 95. Шаг за шагом. Практ.пособ., Москва, «ЭКОМ»,1997.