

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

А. П. ЛАЦЕНКО, Т. В. КИШКУРНО

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ И СУБД ACCESS 2007

*Рекомендовано
учебно-методическим объединением
учреждений высшего образования
Республики Беларусь по экономическому образованию
в качестве лабораторного практикума для студентов
учреждений высшего образования по специальностям
1-25 01 07 «Экономика и управление на предприятии»,
1-25 01 08 «Бухгалтерский учет», 1-26 02 02 «Менеджмент»,
1-26 02 03 «Маркетинг» по дисциплине
«Компьютерные информационные технологии»*

Минск 2011

УДК 004.65(076.5)
ББК 32.973я73
Л32

Рецензенты:

кафедра методов оптимального управления БГУ
(доктор физико-математических наук, профессор,
заведующий кафедрой *А. И. Калинин*);
кандидат технических наук, доцент, заведующий кафедрой
информационных технологий БГЭУ *М. Н. Садовская*

Все права на данное издание защищены. Воспроизведение всей книги или ее части не может быть осуществлено без разрешения учреждения образования «Белорусский государственный технологический университет».

Лашенко, А. П.

Л32 Проектирование баз данных и СУБД Access 2007 : лабораторный практикум для студентов специальностей 1-25 01 07 «Экономика и управление на предприятии», 1-25 01 08 «Бухгалтерский учет», 1-26 02 02 «Менеджмент», 1-26 02 03 «Маркетинг» по дисциплине «Компьютерные информационные технологии» / А. П. Лашенко, Т. В. Кишкурно. — Минск : БГТУ, 2011. — 120 с.
ISBN 978-985-530-116-6.

Лабораторный практикум содержит основные теоретические положения, методические указания для проведения лабораторных работ и подготовки студентов к экзамену по дисциплине «Компьютерные информационные технологии». Пособие посвящено базовым вопросам компьютерной обработки информации и включает три самостоятельных раздела. Все разделы сопровождаются и поддерживаются конкретными практическими примерами, которые облегчают усвоение материала студентами.

**УДК 004.65(076.5)
ББК 32.973я73**

ISBN 978-985-530-116-6 © УО «Белорусский государственный технологический университет», 2011
© Лашенко А. П., Кишкурно Т. В., 2011

ВВЕДЕНИЕ

Развитие средств вычислительной и коммуникационной техники обеспечило возможности для создания и широкого использования систем обработки данных разнообразного назначения. Разрабатываются и внедряются автоматизированные информационные системы для обслуживания различных сфер деятельности.

Одной из важных предпосылок создания таких систем стала возможность их оснащения «памятью» для накопления, хранения и систематизации больших объемов данных о процессах функционирования и свойствах реальных объектов, разнообразных нормативов и данных справочного характера. Благодаря появлению хорошо организованной памяти с удобным доступом резко сократилась трудоемкость, уменьшились сроки подготовки исходных данных и анализа результатов решения крупных вычислительных задач.

Другой существенной предпосылкой нужно признать разработку подходов к информационным объектам, а также создание программных и технических средств конструирования систем, предназначенных для коллективного пользования. Наряду с другими системами, обобществляемыми в такой среде, социальным ресурсом становятся и хранимые в памяти системы данные и информация.

В этой связи разработаны специальные методы и механизмы управления такими совместно используемыми ресурсами данных, которые стали называться базами данных (БД). Исследования и разработки, связанные с проектированием, созданием и эксплуатацией БД, а также необходимых для этих целей языковых и программных инструментальных средств, привели к появлению в начале 60-х гг. XX в. новой самостоятельной ветви информатики, быстро получившей широкое признание.

Практическое применение ее методологических принципов потребовало создания новых подходов и инструментария для эффективной реализации прикладных систем, основанных на концепциях БД. Особую актуальность приобрело создание ориентированных на социальную пользовательскую среду программных средств управления данными, которые обеспечили бы надежное хранение больших объемов данных сложной структуры во внешней памяти вычислительных средств и эффективный доступ к ним

в процессе решения экономических и управленческих задач. Такие программные комплексы, называемые системами управления БД (СУБД), должны выполнять довольно сложный комплекс функций, связанных с централизованным управлением данными в БД в интересах всей совокупности ее пользователей. При этом в качестве пользователей могут выступать, в частности, и различные программные модули систем обработки данных. СУБД служит, по существу, посредником между пользователями и БД.

Наряду с разработкой научных основ сформировалась и получила массовое распространение практическая технология БД со всеми ее ключевыми компонентами. Создана методология проектирования и эксплуатации систем БД, имеющая серьезный теоретический базис, а также развитые инструментальные средства для разработчиков таких систем и персонала администратора БД для удовлетворения разнообразных по характеру потребностей и различных по уровню квалификации категорий пользователей.

Для управления экономическими объектами разрабатываются и внедряются автоматизированные информационные системы. Их ядром являются БД, в которых хранятся данные, адекватно отображающие реальные процессы, события, явления, объекты и которые служат для удовлетворения информационных потребностей пользователей. Будущим специалистам экономического профиля придется работать с БД в среде определенных автоматизированных информационных систем, поэтому они должны владеть технологиями БД. Радикально изменились и сферы применения, и круг пользователей технологий БД. Если раньше эти технологии были доступны лишь крупным вычислительным центрам, то с появлением персональных компьютеров они нашли массовое применение наряду с технологиями обработки текстов, электронными таблицами и коммуникациями. Новые сферы применения связаны с системами поддержки принятия решений, автоматизированным проектированием, разработкой систем программного обеспечения, национальными программами создания электронных библиотек. Роль БД в качестве экономических активов непрерывно возрастает, они шире используются во всех сферах бизнеса и экономической деятельности, поэтому для успешной работы с БД и СУБД в современных условиях необходимы знания о концепциях моделирования данных, принципах организации БД, методах их проектирования и программных средствах для работы с ними. Это и определило содержание данного лабораторного практикума.

1. ТЕОРИЯ БАЗ ДАННЫХ

1.1. Общие сведения о базах данных

Первые компьютеры (англ. *computer* — вычислитель), как это ясно из названия, были ориентированы только для решения вычислительных задач (например, в ядерной физике, механике, баллистике). Особенностью этих задач было то, что они имели небольшой объем исходных данных, которые сравнительно редко менялись, и поэтому их можно было хранить внутри программы.

При попытке использовать компьютер для решения экономических и управленческих задач возникла следующая проблема: такие задачи имели большой объем исходных данных, и эти данные часто менялись. Следовательно, хранение данных вместе с программой было нецелесообразным. Кроме того, в различных программах встречались очень похожие фрагменты кода, выполняющие некоторые стандартные действия: открыть-закрыть файл, найти на внешнем машинном носителе информации (магнитной ленте) нужную запись, отсортировать массив данных, добавить-удалить-изменить (данные в файле) и т. д. Поэтому в середине 50-х гг. XX в. была разработана концепция БД. Основные положения этой концепции следующие:

- централизованное хранение информации;
- хранение данных независимо от программы их обработки;
- возможность использования одних и тех же данных для решения различных задач;
- специальная организация данных для оптимизации времени обращения к ним.

Тогда же и появилось первое упоминание о БД.

БД — это набор сведений (о реальных объектах, процессах, событиях или явлениях), относящихся к некоторой предметной области, организованный по определенным правилам, предусматривающими общие принципы описания, хранения и манипулирования данными, и представленный в виде, пригодном для обработки автоматическими средствами при возможном участии человека.

Одинаковые фрагменты кода программ, встречающиеся в самых разных задачах, организовали в виде библиотеки подпрограмм.

Такую библиотеку подпрограмм называли СУБД. Ее основные функции: определение данных (описание структуры БД), их обработка и управление ими. В настоящее время существуют различные СУБД – MS SQL Server, MySQL, Interbase, Oracle, DB2, Paradox, FoxPro и множество других, менее известных.

В большинстве случаев БД используются для создания автоматизированных информационных систем.

Информационная система – это программно-аппаратный комплекс, предназначенный для сбора, хранения, обработки и передачи информации.

БД является ядром любой информационной системы и позволяет хранить информацию. Для сбора, передачи и представления информации в удобном для пользователя виде используются элементы интерфейса (например, экранные формы или печатные отчеты). Если обработка информации выполняется по достаточно сложному алгоритму и стандартных операций СУБД (таких как поиск, удаление, добавление, сортировка записей) недостаточно, то используются специально разработанные модули обработки информации, дополняющие и расширяющие возможности стандартных СУБД. Таким образом, автоматизированная информационная система – это БД плюс модуль интерфейса плюс дополнительные программы обработки.

При создании БД или информационных систем можно выделить ряд этапов.

1. Постановка задачи. На этом этапе формулируются цели и задачи создаваемой информационной системы.

2. Анализ предметной области. На этом этапе описываются информационные объекты с указанием их характеристик. В результате строится концептуальная информационная модель предметной области.

3. Нормализация отношений в информационной модели. На этом этапе анализируются полученные на втором шаге объекты и устраняются некоторые информационные аномалии (нарушения). В результате получается нормализованная информационная модель предметной области.

4. Создание физической структуры данных. На этом этапе описывается нормализованная информационная модель с учетом требований конкретной СУБД, определяются имена полей и типы данных.

5. Разработка интерфейса. На этом этапе проектируются экранные формы и отчеты для ввода и представления информации.

6. Разработка дополнительных модулей обработки информации. При необходимости создаются дополнительные процедуры или запросы для обработки и поиска информации, хранящейся в БД.

7. Тестирование и отладка информационной системы. На этом этапе происходит актуализация БД (ее заполнение реальной информацией), отладка дополнительных модулей обработки информации.

8. Внедрение. На этом этапе разрабатывается документация по использованию спроектированной информационной системы, обучение персонала, устранение ошибок.

9. Эксплуатация.

Первые три этапа относятся к так называемому «бумажному» проектированию, т. е. выполняются без использования компьютера, хотя существуют специальные программные средства (case-технологии, не относящиеся к СУБД, например, ERwin или Rational Rose), которые позволяют автоматизировать эти операции. От того, насколько правильно и тщательно выполнены первые три шага, зависит успех всей разработки рассматриваемой БД.

Чтобы система полностью удовлетворяла запросам пользователей, необходимо очень внимательно отнестись к процессу проектирования БД. Плохо спроектированная БД будет порождать ошибки, способные привести к принятию неправильных решений, которые повлекут за собой самые серьезные последствия для данной организации. С другой стороны, хорошо спроектированная БД позволит создать систему, поставляющую корректную информацию, которая может успешно использоваться для принятия правильных и эффективных решений.

Последующие этапы выполняются уже с использованием конкретной СУБД, а чтобы научиться использовать БД в соответствующей предметной области, нужно последовательно пройти все эти этапы.

1.2. Категории баз данных

В БД имеется два различных уровня описания и представления данных: физический и логический.

На физическом уровне принята следующая терминология.

1. *Поле* — наименьшая единица памяти, обрабатываемая СУБД.
2. *Физическая запись* — упорядоченная совокупность фиксированного количества полей. Две физические записи однотипны, если совпадают по составу полей.

3. *Файл* — совокупность однотипных записей.

4. *Блок* — размер памяти, передаваемой из внешнего запоминающего устройства в оперативно-запоминающее устройство и обратно за одну операцию чтения-записи. И хотя термин избыточен, его значение весьма существенно, от его величины зависит скорость поиска.

5. *Индексный файл* — структурированная совокупность записей, на которой реализуется какой-либо метод доступа к данным; вводится для увеличения скорости поиска данных и для реализации ограничений целостности.

На логическом уровне принята следующая терминология.

1. *Атрибут (элемент данных)* — наименьшая поименованная единица информации с определенным типом, идентифицируемая СУБД. Обычно соответствует полю на физическом уровне.

2. *Логическая запись* — фиксированная совокупность элементов данных. Две логические записи однотипны, если состоят из одинаковых совокупностей элементов данных.

3. *Отношение* — совокупность всех однотипных логических записей. Обычно (для простых СУБД) соответствует файлу.

4. *Схема БД* — совокупность отношений с установленными связями и ограничениями целостности.

Пример логического уровня представлен на рис. 1.1.



Рис. 1.1. Пример логического уровня

На рис. 1.1 рассматриваются три отношения, соответствующие трем классам объектов: сотрудник, оборудование и рабочее место. Связи представлены стрелками, смысл которых будет пояснен в дальнейшем. Накладываются следующие ограничения целостности:

1) не может быть двух сотрудников с одним и тем же табельным номером;

2) не может быть так, чтобы один и тот же инвентарный номер соответствовал различному оборудованию.

Эти поля являются первичными ключами. Запись из отношения «Сотрудники» нельзя удалить, если с ней связана запись из отношения «Рабочее место». То же самое справедливо и для отношения «Оборудование».

Пример физического уровня представлен на рис. 1.2.



Рис. 1.2. Пример физического уровня

В обязательном порядке должны быть проиндексированы ключевые поля записей:

- индексные файлы для табельного номера сотрудника в первом файле;
- индексные файлы для инвентарного номера сотрудника во втором файле;
- индексные файлы для табельного и инвентарного номеров в третьем файле.

Делается это потому, что в СУБД нет другого механизма реализации ограничений целостности и связи. Кстати, не надо связывать отношения по неключевым полям или полям с неопределяемым типом.

1.3. Требования к базе данных

1.3.1. Неизбыточность и непротиворечивость данных

Если каждое приложение работает со своей системой файлов, а не с единой БД, то в рамках одной прикладной области неизбежно дублирование данных. Следствием этого будет противоречивость: в одном приложении информация была изменена, а в другом — нет. Например, в отделе кадров сотрудника уволили, а в бухгалтерии он еще числится и получает зарплату; причина этого в том, что единственная связь между отделами — это бумажная документация, а бумаги, как и вещи в целом, имеют свойство исчезать. И виноваты отнюдь не сотрудники отдела кадров или бухгалтерии — ошибку допустил программист. БД избавлены от этого недостатка.

В БД допустима так называемая контролируемая избыточность, но за согласованием избыточных данных следит СУБД, а не приложение или пользователь.

Пример. Индексные файлы, дублирующие значение первичного ключа, используются почти во всех СУБД. Более сложный пример — дублирование информации БД на удаленном сервере (тиражирование). При этом программист не должен писать команды на каждый сервер среды.

1.3.2. Защита данных от программных и аппаратных сбоев

Все виды защиты данных должны обеспечиваться СУБД. Сбои бывают двух видов: логические и физические.

Логический сбой. Пусть оператор выполняет попытку дополнения информации об объекте, которая уже содержится в базе. СУБД должна предотвратить операцию дополнения. От проектировщиков требуется определить уникальный первичный ключ и сообщить об этом СУБД. Ситуация сбоя зовется ошибкой I рода. Пусть оператор выполняет удаление информации об объекте, на которую ссылается другой объект. СУБД должна предотвратить удаление. От проектировщика требуется в ограничениях целостности ссылочных данных задать требуемый вид ограничений. В случае ошибки либо сообщать пользователю об исправлении, либо производить каскадное удаление (что сложнее). Однако вариант должен быть максимально простым. Ситуация называется ошибкой II рода.

Физический сбой. Во время работы СУБД возникает аварийная ситуация, причиной которой может быть как ошибка в СУБД или операционной системы, так и сбой оборудования и т. д. При этом

СУБД может не успеть выполнить операции по преобразованию структуры БД, и многие данные могут быть потеряны. Защитить данные от физических сбоев можно несколькими способами:

- обеспечить локальность модифицирующих воздействий — в структурах основных файлов не должно быть элементов связности, например, когда записи в основном файле БД связаны в цепочку и искажение указателя приведет к потере ее хвоста. Сейчас СУБД не структурируют основные файлы подобным образом, однако иногда это делается во вспомогательных файлах, например, индексных.

- В корпоративных БД часто создается и ведется рабочий файл журнала, куда перед выполнением очередной команды заносится информация, достаточная для завершения операции после повторного старта БД. Это приводит к снижению скорости, но иногда сохранность данных важнее.

1.3.3. Мобильность прикладного программного обеспечения

Определение 1. Прикладной программой в БД зовется программа пользователя, взаимодействующая с БД посредством СУБД.

Определение 2. Прикладная программа мобильна, если ее исходный код не зависит от операционной системы и аппаратуры.

Прикладная программа должна быть мобильной (в рамках одной СУБД) и, кроме того, не должна зависеть от места и способа хранения данных. СУБД же создаются для разных платформ и различных операционных систем.

В настоящее время используется трехуровневая модель организации БД, предложенная в 1975 г. комитетом по стандартизации ANSI (American National Standards Institute) (рис. 1.3).

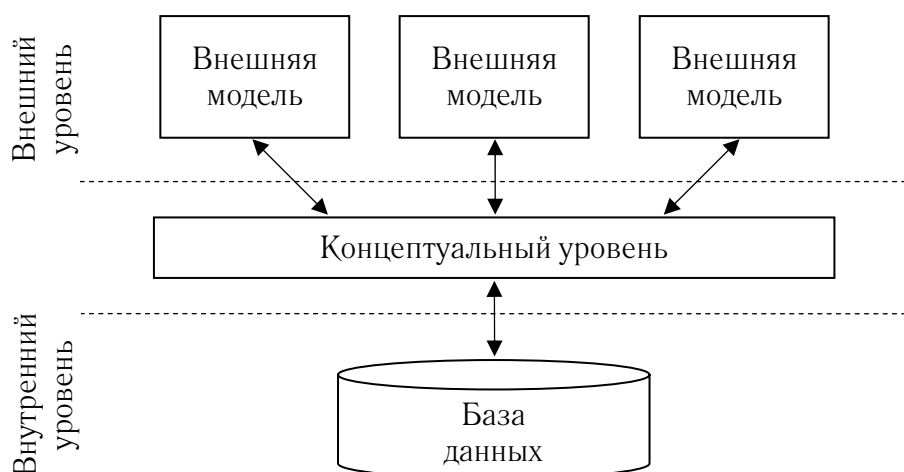


Рис. 1.3. Трехуровневая модель организации БД

Одна и та же БД имеет различные уровни описания.

Внешний уровень — это представление о БД отдельных пользователей и прикладных программ. Каждый пользователь, каждая прикладная программа видят и обрабатывают только те данные предметной области, которые им необходимы. Например, прикладная программа, используемая отделом кадров, обрабатывает сведения о сотрудниках, их адресе, стаже работы и не оперирует данными о заработной плате.

На *концептуальном уровне* БД представляется обобщенно — объединяются данные, используемые различными пользователями и прикладными программами. Концептуальный уровень фактически определяет обобщенную модель предметной области и не содержит никаких сведений о методах хранения данных.

Внутренний уровень поддерживает представление БД в памяти компьютера.

Первоначально исследования в области БД были направлены на разработку способов структуризации данных, получивших название «модели данных». Модель данных — совокупность принципов организации БД. Модели данных различаются принципами определения, манипулирования и хранения данных в базе. Но наиболее важным является способ организации связей между данными в БД.

1.3.4. Секретность данных

Традиционно в БД авторизация доступа выполняется за счет аппарата внешних схем: при входе пользователь вводит имя группы и пароль. В описании схемы присутствуют ограничения на доступ к данным (в виде SQL-команд). Также применяется шифрование на физическом уровне. Например, в СУБД Clarion пароль является ключом шифра.

1.4. Представление и описание информации

Информация не существует без материальных носителей. Она также не существует без восприятия себя. Существует точка зрения, что информация не является ни материальной, ни идеальной; она, фактически, третья составляющая мироздания, мост между материей и сознанием.

Окружающая действительность воспринимается как множество объектов и отношений между ними. Объекты бывают материальными и нематериальными; отношения чаще всего нематериальны.

Отношения между объектами сами могут быть интерпретированы как объекты. Таким образом, информация сводится к объектам, материальным и нематериальным.

Различие между объектами достигается за счет различия их свойств. Набор свойств объектов зависит от прикладной области, т. е. от точки зрения. С одной точки зрения объекты могут быть неразличимы, с другой – резко различаться.

Определение 3. Существенные для данной прикладной области свойства объектов называются атрибутами.

1.4.1. Плоские (двойные) файлы

Определение 4. Объекты прикладной области однотипны, если характеризуются одинаковым набором атрибутов (имеют одинаковую семантику).

Определение 5. Табличное представление информации называется плоским файлом, если выполнены следующие условия:

1) таблица имеет наименование и заголовок в виде наименований атрибутов;

2) содержимым одной строки таблицы является информация об одном объекте данного класса (и, следовательно, в таблице не может быть совпадающих строк);

3) содержимое одного столбца зовется доменом – областью определения атрибута (содержимое столбца всегда однотипно).

Пример 1:

Студенты

| Номер студенческого билета | ФИО студента | Номер группы | Факультет |
|----------------------------|--------------|--------------|----------------|
| 1345678 | Иванов И. И. | М-903 | Математический |
| 7654321 | Петров П. П. | Ф-801 | Физический |

Информация о классе «Студент» представлена в виде плоского (двойного) файла.

В этом примере рассмотрен класс материальных объектов, однако атрибуты его нематериальны. Можно для атрибута «Факультет» взять материальное представление: кабинет декана или секретаря, но это будет уже другой класс объектов, и ему соответствуют уже материальные объекты.

Пример 2:

Супруги

| ФИО мужа | ФИО жены |
|--------------|---------------|
| Иванов И. И. | Иванова М. И. |
| Петров П. П. | Петрова П. П. |

В этом примере класс объектов нематериален, т. к. задает отношение между объектами, а атрибуты — материальны. ФИО, впрочем, тоже нематериально, но за ним стоит вполне конкретный человек. Плоский файл является описанием и представлением информации. Описание представлено именем плоского файла и заголовком таблицы, представление (или реализация) — его содержимым.

1.4.2. Ключи

Определение 6. Для выбранного класса объектов атрибут является поисковым ключом, если одно его значение служит для идентификации нескольких объектов данного класса.

В первом примере любой атрибут является поисковым ключом.

Определение 7. Для выбранного класса объектов атрибут является первичным ключом, если одно его значение служит для идентификации одного объекта данного класса.

В первом примере первичным ключом является номер студенческого билета.

Определение 8. Поисковый ключ в выбранном классе объектов называется сцепленным, если состоит более чем из одного атрибута. Сцепленные ключи обозначаются конкатенацией (+).

В первом примере сцепленным ключом может быть «ФИО студента» + «Факультет». Для реализации поисковых и первичных ключей на физическом уровне используются индексные файлы. Ключевые атрибуты (первичный и поисковый) — основа для установления целостности в БД.

1.5. Модели данных

Первоначально исследования в области БД были направлены на разработку способов структуризации данных, получивших название «модели данных». Модель данных — совокупность принципов организации БД. Они различаются принципами определения,

манипулирования и хранения данных в базе. Но наиболее важным является способ организации связей между данными в БД. Классическими являются иерархическая, сетевая, реляционная модели данных. Кроме того, при разработке БД в последнее время активно используются такие модели, как постреляционная, объектно-ориентированная, объектно-реляционная и многомерная.

1.5.1. Иерархическая модель

В иерархической модели связи между данными можно представить с помощью корневого дерева. На рис. 1.4 показан пример организации данных по иерархической модели.

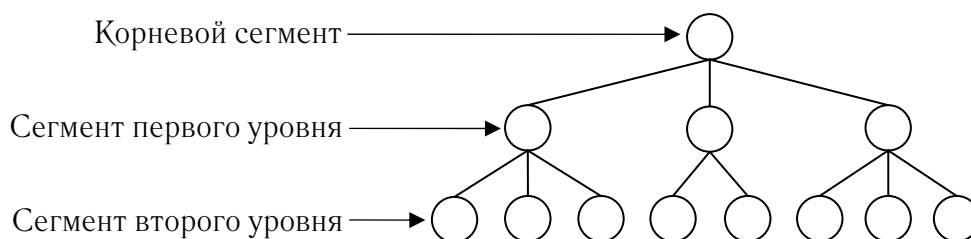


Рис. 1.4. Схематическое изображение иерархической БД

Данные в такой модели расположены на разных иерархических уровнях и называются сегментами. Самый высокий сегмент — корневой. Сегменты, расположенные на более низком уровне, называются сегментами-потомками; на более уровне — сегментами-предками. Каждый сегмент может иметь только одного предка на более высоком уровне и одного или несколько потомков на более низком уровне. Доступ к определенному сегменту осуществляется по цепочке — от сегмента-предка к сегменту-потомку, начиная слева. Иерархическая модель используется для представления организационных структур, по своей природе являющихся иерархическими (например, крупных предприятий, воинских подразделений), или сложных механизмов, состоящих из более простых узлов, которые, в свою очередь, можно подвергнуть декомпозиции. Организовать более сложные связи в такой модели невозможно. Например, если исполнителю необходимо участвовать в нескольких проектах, то потребуются создание дополнительной БД.

Недостатком иерархической модели является ее громоздкость для обработки данных со сложными логическими связями. К достоинствам данной модели относится эффективное использование памяти компьютера при хранении данных.

1.5.2. Сетевая модель

Сетевая модель является развитием иерархической. В отличие от нее, в сетевой модели потомок может иметь любое количество предков. Сегменты, называемые наборами записей, связываются между собой по принципу не только «сверху вниз», но и «по горизонтали» с помощью наборов связей. Пример организации данных по сетевой модели приведен на рис. 1.5.

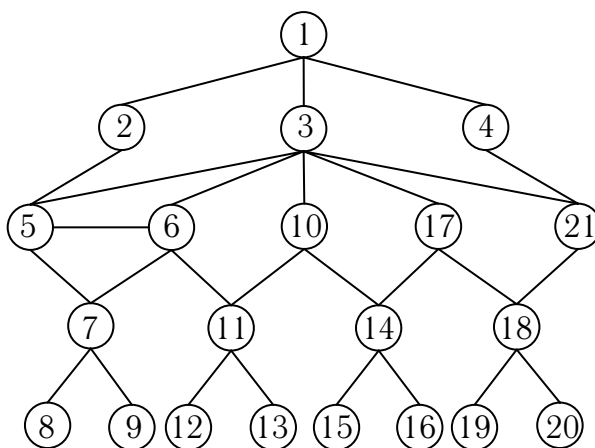


Рис. 1.5. Схематическое изображение сетевой БД

Доступ к данным осуществляется не только «сверху вниз», но и по горизонтальным наборам связей.

К достоинствам сетевой модели данных относятся возможность образования произвольных связей и быстрый доступ к данным. Недостатками сетевой модели являются сложность ее понимания для обычного пользователя и большие объемы памяти компьютера на хранение данных.

1.5.3. Реляционная модель

Реляционная модель данных была предложена американским математиком Э. Ф. Коддом, который в 1970 г. впервые сформулировал ее основные понятия. В настоящее время это самая распространенная модель данных.

В основе реляционной модели данных лежит понятие «отношение» (от англ. *relation*). Отношение отображает некоторый объект, который характеризуется набором атрибутов, а каждый атрибут — набором допустимых значений, называемым доменом.

Создатель реляционной модели Э. Ф. Кодд использовал термин «отношение» как синоним слова «таблица».

Столбцы этой таблицы соответствуют атрибутам, а строки называются кортежами. Количество кортежей в отношении называется мощностью отношения.

Таким образом, реляционная модель данных основана на математическом понятии отношения и представлении отношений в форме таблиц. Таблица в реляционной модели данных (реляционная таблица) обладает следующими свойствами:

1) каждое значение атрибута, содержащееся на пересечении строки и столбца, должно быть атомарным, т. е. не расчленяться на несколько значений;

2) значения в столбце должны быть однородными;

3) каждая строка уникальна, т. е. в таблице не существует двух полностью совпадающих строк;

4) каждый столбец имеет уникальное имя;

5) последовательность столбцов в таблице несущественна;

6) последовательность строк в таблице несущественна.

В качестве примера реляционной таблицы приведем таблицу «Клиенты».

Клиенты

| Код клиента | Клиент | Адрес |
|-------------|--------|-----------------------------|
| АА | БГТУ | Минск, ул. Свердлова, 13а |
| АБ | Сименс | Мюнхен, ул. Лейбница, 8 |
| АС | БГУИР | Минск, ул. Бровки, 6 |
| АД | БГЭУ | Минск, пр. Партизанский, 26 |

В таблице реляционной БД (БД, построенной по реляционной модели) столбцы называют полями, а строки — записями. Одно или несколько полей, значения которых в каждой записи таблицы однозначно ее идентифицируют, называют ключевым полем. В таблице «Клиенты» таковым может быть поле «Код клиента» или поле «Клиент». В реляционной БД между таблицами устанавливаются связи, которые делают их более информативными, чем они являются по отдельности. Связь устанавливается посредством связи ключевых полей, содержащих общую информацию для обеих таблиц.

Пусть таблица R_1 связывается с таблицей R_2 . Тогда таблица R_1 называется основной, а таблица R_2 — подчиненной. Ключевое поле основной таблицы называется первичным ключом, а подчиненной — внешним ключом. Одна запись основной таблицы может быть

связана с одной или несколькими записями подчиненной таблицы. При этом значения первичного ключа уникальны, а внешнего могут повторяться.

В общем виде реляционная модель данных представляет собой множество взаимосвязанных таблиц. Графическое изображение связи между таблицами называется схемой данных.

Важным достоинством реляционной модели данных является то, что на основе уже имеющихся схем отношений можно получать новые схемы, которые ранее в файле БД не были представлены. Таким способом соответствующая СУБД позволяет получать ответы на незапланированные прикладными программами информационные запросы.

Необходимо отметить, что реляционные БД не лишены и определенных недостатков: медленность работы для больших БД, жесткость структуры данных и некоторые другие. Однако для персональных компьютеров и не очень больших БД в настоящее время используются исключительно реляционные модели БД.

1.6. Компоненты описания схемы данных

Представителем логического описания информационной модели предметной области является модель «сущность-связь» (Entity-Relationship, ER) Чена (ER-диаграммы). Моделирование структуры данных в ней базируется на использовании графических средств — ER-диаграмм (диаграмм «сущность-связь»). В наглядном виде они представляют связи между сущностями. Основные понятия ER-диаграммы: сущность, атрибут, связь.

Определение 9. Элементом данных (атрибутом) называется атомарное (неделимое) данное с определенным типом и наименованием.

Пример. В прикладной области отдела кадров «ФИО сотрудника» не является атрибутом, т. к. нарушена атомарность и требуется вместо него использовать отдельно «Фамилия сотрудника», «Имя сотрудника» и «Отчество сотрудника». Однако в другой прикладной области, например, железнодорожной кассе, такой атрибут допустим, поскольку основным показателем являются паспортные данные.

Примеры неправильных атрибутов.

- «Цех». Этот атрибут не обладает однозначной семантической интерпретацией. Возможно, разработчик имел в виду номер цеха, его наименование, продукцию или что-то еще.

- «Средняя зарплата отдела». При заполнении значения данного атрибута пользователь должен сам подсчитать его значение. В БД должна храниться зарплата каждого сотрудника в отдельности, а подсчет максимальной, минимальной и средней зарплат должна выполнять прикладная программа.

- «Возраст сотрудника». Значение атрибута является изменяемой величиной. Вместо нее лучше хранить дату рождения сотрудника. Причина в том, что в БД должны использоваться элементы данных, изменение значений которых инициируется некоторыми событиями. Например, элемент данных «Номер курса» для студента тоже меняется со временем, но это происходит только на основании приказа о переводе.

- «Стаж работы». Тоже изменяемая величина, и атрибут необходимо заменить на пару атрибутов «Стаж работы до приема» и «Дата приема на работу».

Определение 10. Между элементами на схеме выполняется установление связей, причем связи отражают количественное соотношение между значениями элементов данных и не имеют никакого содержания. Если установленной на схеме связи удастся присвоить наименование, то потеряется атрибут с содержанием, соответствующим наименованию.

Связи всегда направлены и имеют следующие типы:

1. **1:1**, или связь «один-к-одному», — одному значению первого элемента данных соответствует одно значение второго элемента. Пример данной связи приведен на рис. 1.6: «Номер читательского билета» — «Номер зачетки».

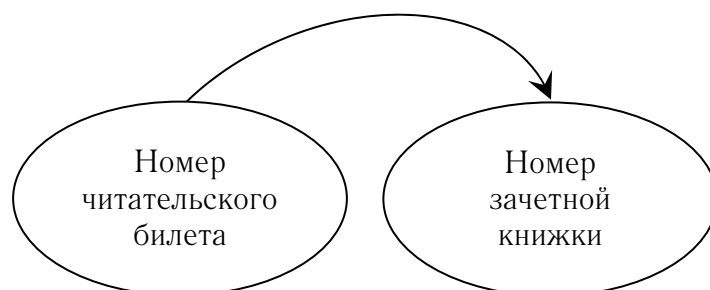


Рис. 1.6. Связь «один-к-одному»

2. **М:1**, или связь «много-к-одному», — множеству значений первого элемента данных соответствует одно значение второго элемента. Пример данной связи приведен на рис. 1.7: «Табельный номер» — «Должность сотрудника».

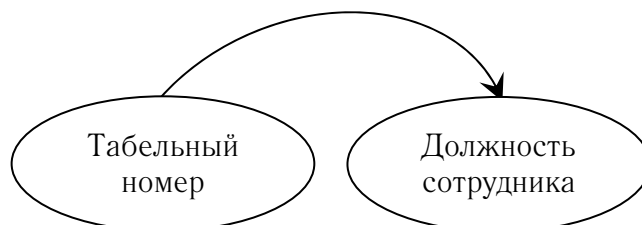


Рис. 1.7. Связь «много-к-одному»

3. **1:М**, или связь «один-ко-многим», — одному значению первого элемента данных соответствует множество значений второго. Пример данной связи приведен на рис. 1.8: «Должность сотрудника» — «Табельный номер».

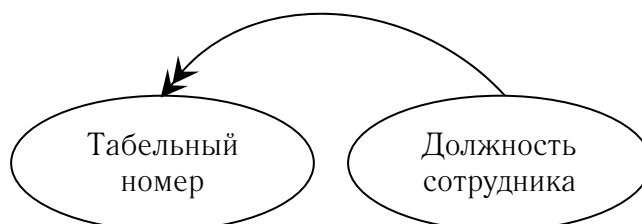


Рис. 1.8. Связь «один-ко-многим»

4. **М:М**, или связь «много-ко-многим», — множеству значений первого элемента данных соответствует множество значений второго элемента данных. Пример данной связи приведен на рис. 1.9: «Должность сотрудника» — «Оклад сотрудника».

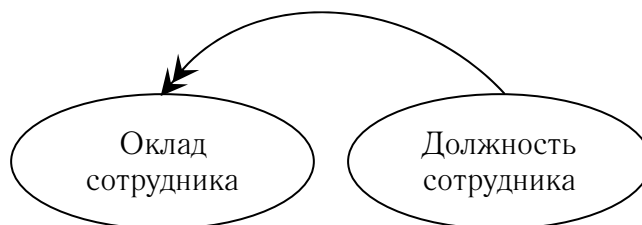


Рис. 1.9. Связь «много-ко-многим»

Оформление схемы выполняется графически, атрибуты обозначаются овалами с вписанными в них именами, связи — линиями.

2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Проще всего представить себе БД в виде таблицы с некоторым фиксированным числом столбцов и достаточно большим неопределенным числом строк. Простейший пример БД – телефонный справочник. Его поля фиксированы: «Фамилия», «Имя», «Отчество», «Номер телефона». Набор записей телефонного справочника не является постоянным. Телефонный справочник может непрерывно пополняться, записи в нем могут изменяться при смене телефонного номера.

Что останется в БД, если удалить из нее все записи? В ней останутся только названия полей, а точнее говоря, структура БД, которую определяет совокупность полей и их свойства.

БД с нулевым количеством записей все равно остается базой, поскольку имеет структуру. Точно так же книжка для записи телефонных номеров не перестает быть книжкой, даже если в ней пока нет ни одной записи.

Таким образом, создание БД состоит как бы из двух этапов: создания ее структуры и наполнения структуры информацией. И ту, и другую функцию выполняет СУБД.

2.1. Проектирование базы данных

Проектирование данных БД представляет собой процесс отображения исследуемых явлений реального мира в виде данных в памяти ЭВМ (рис. 2.1).

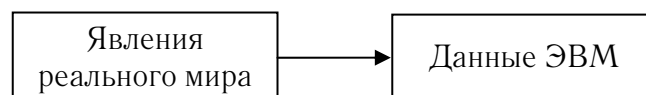


Рис. 2.1. Процесс отображения

Конкретные явления реального мира, представляющие интерес для проводимого исследования, называется предметной областью.

Проектирование БД — это процесс создания БД, предназначенной для поддержки функционирования информационного объекта и способствующей достижению его целей. Оно представляет собой трудоемкий процесс, требующий совместных усилий аналитиков, проектировщиков и пользователей.

При проектировании БД необходимо обеспечить соблюдение комплекса требований:

- целостность БД — требование полноты и непротиворечивости данных;
- многократное использование данных;
- быстрый поиск и получение информации по запросам пользователей;
- простота обновления данных;
- минимизация избыточности данных;
- защита данных от несанкционированного доступа, искажения и уничтожения.

2.2. Этапы проектирования базы данных и их процедуры

Проектирование БД осуществляется в три этапа:

- 1) концептуальное проектирование;
- 2) логическое проектирование;
- 3) физическое проектирование.

Цель этапа концептуального проектирования — создание концептуальной модели данных исходя из представлений пользователей о предметной области. Для ее достижения выполняется ряд последовательных процедур:

- определение сущностей и их документирование;
- определение связей между сущностями и их документирование;
- создание модели предметной области;
- определение атрибутов и их документирование;
- определение значений атрибутов и их документирование;
- определение первичных ключей для сущностей и их документирование.

Цель этапа логического проектирования — преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависящую от особенностей используемой

в дальнейшем СУБД для физической реализации БД. Для ее достижения выполняются следующие процедуры:

- выбор модели данных;
- определение набора таблиц и их документирование;
- нормализация таблиц;
- определение требований поддержки целостности данных и их документирование.

Цель этапа физического проектирования — описание конкретной реализации БД, размещаемой во внешней памяти компьютера. Для ее достижения выполняются следующие процедуры:

- проектирование таблиц БД средствами выбранной СУБД;
- проектирование физической организации БД;
- разработка стратегии защиты БД.

2.3. Способы описания предметной области

Введем основные понятия, с помощью которых описывается предметная область.

Сущность, или *объект*, — то, о чем будет накапливаться информация в информационной системе (нечто такое, за чем пользователь хотел бы наблюдать).

Если в системе обрабатывается информация о студентах, то сущностью является СТУДЕНТ, если обрабатывается информация об экзамене, то сущность — ЭКЗАМЕН и т. п. Каждая сущность обладает определенным набором свойств (рассматриваем только свойства, представляющие интерес для пользователей в рамках проводимого исследования), которые запоминаются в информационной системе.

Так, например, в качестве свойств сущности СТУДЕНТ можно указать фамилию, дату рождения, место рождения, в качестве свойств сущности ЭКЗАМЕН можно указать предмет, дату проведения экзамена, экзаменаторов.

Совокупность сущностей, характеризующихся в информационной системе одним и тем же перечнем свойств, называется *классом сущностей (набором объектов)*. Так, например, совокупность всех сущностей СТУДЕНТ составляет класс сущностей СТУДЕНТ, совокупность всех сущностей ЭКЗАМЕН составляет класс сущностей ЭКЗАМЕН.

Класс сущностей описывается перечнем свойств сущностей, составляющих этот класс.

Экземпляр сущности будем называть конкретную сущность (сущность с конкретными значениями соответствующих свойств). Пример класса сущностей СТУДЕНТ и конкретного экземпляра сущности показан на рис. 2.2.

| Класс сущностей СТУДЕНТ | Экземпляр сущности |
|-------------------------|--------------------|
| Фамилия | Иванов |
| Дата рождения | 18.05.47 |
| Место рождения | Минск |

Рис. 2.2. Класс сущностей и экземпляр сущности

Взаимоотношения сущностей выражаются связями. Различают классы связей и экземпляры связей. *Классы связей* — это взаимоотношения между классами сущностей, а *экземпляры связи* — взаимоотношения между экземплярами сущностей.

Класс связей может затрагивать несколько классов сущностей. Число классов сущностей, участвующих в связи, называется *степенью связи* ($n = 2, 3, \dots$). Так, например, класс сущностей СТУДЕНТ связан с классом сущностей ЭКЗАМЕН связью «сдает». Степень этой связи равна двум, и такая связь называется бинарной. В качестве примера связи третьей степени можно указать связь «родители» между тремя классами сущностей МАТЬ, ОТЕЦ, РЕБЕНОК.

2.4. Описание информационной модели предметной области

Как уже отмечалось, в качестве основного понятия для описания предметной области используется понятие сущности (объекта), характеризуемого набором определенных свойств. Для информационного описания сущности вводится понятие атрибута.

Атрибут — поименованное свойство (характеристика) сущности. Атрибут представляет собой информационное отображение свойства сущности. Атрибут объекта принимает конкретное значение из множества допустимых значений. Так, например, для сущ-

ности СТУДЕНТ атрибут «Фамилия» у конкретного экземпляра сущности принимает конкретное значение «Иванов».

Таким образом, атрибут представляет собой информационное представление количественных или качественных свойств сущности, описывает состояние сущности, позволяет идентифицировать ее. Информация о сущности представляется совокупностью атрибутов. Такую совокупность атрибутов часто называют записью об объекте.

Другим основным понятием для описания предметной области является понятие связи. В частности, для представления связей между экземплярами сущностей могут использоваться атрибуты. В этом случае связь устанавливается путем включения в совокупность атрибутов сущности атрибута, однозначно идентифицирующего экземпляр сущности, находящийся в отношении с исходным экземпляром сущности.

Так, рассмотрим класс сущностей ФАКУЛЬТЕТ, представленный совокупностью атрибутов («Название», «Номер») и класс сущностей РАСПИСАНИЕ ЭКЗАМЕНОВ НА ФАКУЛЬТЕТ, представленный совокупностью атрибутов («Название экзамена 1», «Дата экзамена 1», «Название экзамена 2», «Дата экзамена 2», «Название экзамена 3», «Дата экзамена 3»). Для представления связи «экзамены» (тип связи 1:1) в совокупность атрибутов РАСПИСАНИЕ ЭКЗАМЕНОВ НА ФАКУЛЬТЕТ можно включить атрибут «Название факультета».

2.5. Нормализация отношений в реляционной базе данных

Самым простым примером реляционной БД является БД, состоящая всего из одного отношения — одной таблицы. Однако в таком случае, как правило, не будут выполняться основные требования, предъявляемые к структуре БД из-за возникающих проблем избыточности информации, нарушения целостности данных, сложности редактирования данных, низкой скорости обработки информации и т. п.

Пример. В таблице «Поставки товаров» есть сведения о товарах, их цене, количестве и стоимости, а также о поставщиках, их адресах и расчетных счетах:

Поставки товаров

| Товар | Цена | Количество | Стоимость | Поставщик | Адрес | Счет |
|--------|--------|------------|-----------|------------|-----------------------------------|---------|
| Стол | 12 000 | 100 | 1 200 000 | Пинскдрев | 226000, Брестская обл., г. Пинск | 1100022 |
| Стул | 6 000 | 800 | 4 800 000 | Орбита | 220111, Минская обл., г. Слуцк | 2211003 |
| Кресло | 20 000 | 200 | 4 000 000 | Столиндрев | 226100, Брестская обл., г. Столин | 3322004 |
| Диван | 30 000 | 80 | 2 400 000 | Пинскдрев | 226000, Брестская обл., г. Пинск | 1100022 |

Анализируя структуру таблицы, необходимо, прежде всего, отметить, что в ней имеется повторяющаяся информация о поставщике. Кроме того, стоимость товара является избыточной информацией, т. к. всегда может быть получена на основе цены товара и его количества. Далее, атрибуты «Адрес» и «Счет» характеризуют только поставщика и, вообще говоря, не связаны с поставляемым товаром. Существуют и другие более тонкие недостатки в структуре такой БД.

Таким образом, на первом этапе проектирования реляционной БД важнейшим является вопрос, какую выбрать схему отношений для данной БД из множества альтернативных вариантов, т. е. какую систему таблиц и с каким набором столбцов в каждой таблице выбрать для данной БД. Как правило, БД содержит объекты разных типов, и для каждого типа объектов создается своя таблица с соответствующим набором столбцов-атрибутов объекта.

Процесс создания оптимальной схемы отношений для реляционной БД строго формализован и называется нормализацией БД. *Нормализация* — это формализованная процедура, в процессе выполнения которой атрибуты данных группируются в таблицы, а таблицы, в свою очередь, в БД.

Цели нормализации следующие:

- исключить дублирование информации;
- исключить избыточность информации;
- обеспечить возможность проведения непротиворечивых и корректных изменений данных в таблицах;
- упростить и ускорить поиск информации в БД.

Процесс нормализации состоит в приведении таблиц реляционной БД к так называемым нормальным формам. Всего существует пять нормальных форм, которые удовлетворяют соответствующим правилам нормализации. При этом в большинстве случаев оптимальная структура БД достигается при выполнении уже первых трех правил нормализации, которые были сформулированы для реляционных БД Э. Ф. Коддом в 1972 г.

Чтобы таблица, а вместе с ней и БД, соответствовала *первой нормальной форме*, необходимо, чтобы все значения ее полей были атомарными (неделимыми) и невычисляемыми, а все записи — уникальными (не должно быть полностью совпадающих строк). Выполняя это правило, преобразуем первоначальную таблицу к следующему виду:

Поставки товаров

| Товар | Цена | Количество | Поставщик | Индекс | Область | Город | Счет |
|--------|--------|------------|------------|--------|-----------|--------|---------|
| Стол | 12 000 | 100 | Пинскдрев | 226000 | Брестская | Пинск | 1100022 |
| Стул | 6 000 | 800 | Орбита | 220111 | Минская | Слуцк | 2211003 |
| Кресло | 20 000 | 200 | Столиндрав | 226100 | Брестская | Столин | 3322004 |
| Диван | 30 000 | 80 | Пинскдрев | 226000 | Брестская | Пинск | 1100022 |

Чтобы таблица соответствовала *второй нормальной форме*, необходимо, чтобы она уже находилась в первой нормальной форме и все неключевые поля полностью зависели от ключевого. В данной таблице на роль ключевого поля может претендовать только поле (атрибут-признак) «Товар», значения которого в таблице не повторяются. Из других полей только поле «Поставщик» непосредственно связано с поставляемым товаром (полем «Товар»), а поля «Индекс», «Область», «Город» и «Счет» характеризуют только самого поставщика. Поэтому, удовлетворяя второму правилу нормализации, необходимо разбить (или разложить) исходную таблицу на две — соответственно «Товары» и «Поставщики».

Товары

| Товар | Цена | Количество | Поставщик |
|--------|--------|------------|------------|
| Стол | 12 000 | 100 | Пинскдрев |
| Стул | 6 000 | 800 | Орбита |
| Кресло | 20 000 | 200 | Столиндрав |
| Диван | 30 000 | 80 | Пинскдрев |

Поставщики

| Поставщик | Индекс | Область | Город | Счет |
|------------|--------|-----------|--------|---------|
| Пинскдрев | 226000 | Брестская | Пинск | 1100022 |
| Орбита | 220111 | Минская | Слуцк | 2211003 |
| Столиндрав | 226100 | Брестская | Столин | 3322004 |

Проведенное преобразование называется разложением, или проектированием, БД и является обратимой операцией. Причем проектирование исходной таблицы привело, с одной стороны, к уменьшению записей (строк) во второй таблице, однако, с другой стороны, для организации связей между отдельными таблицами и обеспечения таким образом целостности БД поле «Поставщик» появилось уже в обеих таблицах, привнося этим некоторую неизбежную избыточность информации. Очевидно, что в больших БД, где реально существуют сотни и тысячи записей, эта избыточность во много раз будет перекрыта уменьшением общего размера таблиц, полученных из исходной таблицы при ее разложении.

Заметим, что на роль ключевого поля таблицы «Поставщики» подходит поле «Поставщик», значения которого в этой таблице уже не повторяются. Можно отметить, что на эту роль вполне подходит и поле «Счет», значения которого также не будут повторяться, а само поле, хотя и выглядит как набор цифр, все же является не количественной, а качественной характеристикой объекта, т. е. является атрибутом-признаком, что необходимо для ключевого поля (см. выше).

Чтобы теперь перейти к *третьей нормальной форме*, необходимо, прежде всего, обеспечить, чтобы все таблицы БД находились во второй нормальной форме и все неключевые поля в таблицах зависели только от ключа таблицы и не зависели непосредственно друг от друга.

Анализируя таблицу «Поставщики», можно заметить, что поля «Область» и «Город» являются зависимыми от поля «Индекс», и поэтому эта таблица не находится в третьей нормальной форме. В связи с этим необходимо разбить таблицу на две: оставить в таблице «Поставщики» только два («Поставщик» и «Счет»), а также поле «Индекс» для обеспечения связи между таблицами, а остальные поля выделить в новую таблицу «Адреса», в которой поле «Индекс», естественно, будет ключевым, т. к. его значения в таблице не повторяются.

Поставщики

| Поставщик | Индекс | Счет |
|------------|--------|---------|
| Пинскдрев | 226000 | 1100022 |
| Орбита | 220111 | 2211003 |
| Столиндрев | 226100 | 3322004 |

Адреса

| Индекс | Область | Город |
|--------|-----------|--------|
| 226000 | Брестская | Пинск |
| 220111 | Минская | Слуцк |
| 226100 | Брестская | Столин |

Приведение БД к *четвертой* и *пятой* нормальным формам является необходимой операцией в специальных случаях, когда между элементами БД существуют связи типа «многие-ко-многим» и при этом необходимо обеспечить возможность точного восстановления исходной таблицы из таблиц, на которые она была спроектирована. Как уже говорилось, этими правилами нормализации при проектировании БД в большинстве случаев можно пренебречь.

2.6. Рекомендации по проектированию баз данных

Все возникающие вопросы по хранению данных, отслеживанию уникальности значений, поиску требуемых данных, а также многие другие — забота программного механизма СУБД. Но для того, чтобы СУБД могла корректно решать эти вопросы, необходимо правильно разработать структуру данных, т. е. корректно разработать информационную модель, придерживаясь основных требований.

1. *Выносите повторяющиеся данные в отдельные таблицы (избавляйтесь от зависимостей).* Основная идея нормализации — информация о каждом объекте или атрибуте появляется в БД только один раз. Если информацию необходимо использовать в нескольких местах (в разных таблицах или отчетах), то используют ссылки, но не вводят ничего повторно.

Проанализируйте данные, которые будут храниться в вашей таблице. Столбец с большим количеством повторений — повод задуматься о вынесении значений из него в отдельную таблицу.

2. Всегда задавайте первичный ключ. Помните, что для каждой таблицы необходимо задать ключевой столбец, т. е. такое поле, по значениям в котором можно однозначно определить строку таблицы (выбрать, найти, позиционировать). При этом ключ должен быть уникальным, неповторяющимся. Вообще, в таблице может быть несколько уникальных ключей, но только один из них объявляется (выбирается) первичным, после чего все операции с данными будут идти с учетом этого ключа. Внутренние функции любой СУБД лучше работают с числовыми данными, поэтому рекомендуется указывать в качестве первичного ключа — числовой. Цель использования первичного ключа — получить однозначную внутреннюю разбивку строк таблицы. По остальным полям строки могут совпадать как угодно, но это поле будет всегда уникально.

Первичный ключ не должен быть связан с данными, т. к. любой объект реального мира подвержен изменениям, иногда кардинальным. Очень многое из того, что кажется незыблемым, может меняться и довольно быстро. Трудно найти действительно уникальный естественный ключ. Возьмем, например, таблицу рабочих предприятия. Что взять в качестве первичного ключа?

Составной: «Фамилия» + «Имя» + «Отчество»? Но полных однофамильцев не так уж мало, кроме того, люди иногда меняют фамилии. Номер паспорта? Проведенная не так давно смена паспортов показала непостоянность этого параметра. К тому же они иногда теряются — люди получают новые. Взять за основу уникальный табельный номер работника (ведь для того его и придумали)? А теперь представьте ситуацию — предприятие работает долго, имеет обширные архивы, база эксплуатируется практически непрерывно. Руководство отдела кадров решило, что теперь табельный номер должен быть не числовой, а текстовый и нести в себе некую дополнительную информацию, что-нибудь вроде 12345/45-КАУ-07Б. Причем с точки зрения программиста ситуация весьма нестандартная, а с точки зрения отдела кадров — проста, и менять свои взгляды на формат табельного номера отдел кадров может раз в неделю. Предыдущий ключ хранился в числовом поле, на него ссылались записи из других таблиц, по нему вычислялись условия запросов — всю БД придется полностью переписывать. От такой проблемы спасает изначальное использование абстрактного первичного ключа, причем, учитывая особенности реальных СУБД, — числового.

3. Не надо гнаться за полной независимостью (нормализацией) данных — существует понятие разумной денор-

мализации. Продолжим анализ предыдущей задачи о хранении информации по сотрудникам предприятия, а точнее — хранении их фамилий, имен и отчеств. Как лучше — все в одном поле через пробел или в трех разных полях? И тогда уж пойти дальше и вынести три поля в отдельные таблицы-справочники: Справочник имен, Справочник отчеств, Справочник фамилий?

Все зависит от задач, для решения которых требуется информация о ФИО. Вариант с хранением в одном поле — самый простой, но когда в отчете потребуется вывести только фамилию — придется выполнять дополнительную работу по анализу строки «Фамилия Имя Отчество» и выделять часть, в которой хранится фамилия. С другой стороны, при хранении фамилии, имени и отчества в отдельных полях или таблицах также придется проводить дополнительную работу по сведению этих полей в одну строку. Однако есть некоторое преимущество в разделении этой информации на три поля и вынесении их в отдельные таблицы — контроль ввода данных и автоматическое исправление ошибок.

Рассмотрим особенности такого решения на задаче хранения адресов работников предприятия. Требуется разработать структуру для хранения атрибутов: «Улица», «Номер дома», «Квартира».

На первый взгляд — очень простая задача. Насколько она сложная, вы скоро узнаете, но стоит потратить пару минут на самостоятельный анализ. Прямо сейчас подумайте и разработайте таблицу (или таблицы) для хранения данных о названии улицы, номере дома и номере квартиры. Просто прикиньте в уме: какие поля, какого типа вам понадобятся и как их разместить. Интересно, какие проблемы вы сумеете предусмотреть?

Первая проблема — как занести информацию о доме №10/2? А о доме №10/2-Б? В номере квартиры, как в числовом типе, тоже нельзя быть уверенным. Например, встречается нумерация с использованием этажа — «5-15». Допустим, на этаже 50 квартир, 551-й квартиры в таком доме нет, а дальше идет сразу 601-я на 6-м этаже. Поэтому, хотя многие и привыкли считать, что номер квартиры — число, его нельзя хранить в числовом поле. Как и номер дома. Тогда сменим тип поля на текстовый. Решите сами, сколько символов надо предусмотреть для этого. Если подумать, то разделение информации о номере дома и номере квартиры необходимо только для операций с квартирами одного дома или с жителями всех домов, квартира которых имеет заданный номер. В таких операциях могут быть заинтересованы какие-нибудь

административные учреждения, но для нашей задачи о хранении информации по работникам предприятия такая возможность не требуется. Объединим эти два, теперь уже текстовых поля, в одно: «Номера дома и квартиры», соответственно поправив размерность.

Теперь обратим внимание на поле «Название улицы» — как многие уже вспомнили, кроме улиц бывают еще переулки, площади, проспекты. И чтобы отличить переулок Лазо от улицы Лазо, придется заносить также информацию о типе улицы. Куда? В поле, которое предназначено для хранения названия улицы.

Такое решение позволяет точнее контролировать информацию и предоставляет больше возможностей (хотя и за счет усложнения структуры базы), а также отвечает идеям нормализации данных.

4. Максимально контролируйте правильность вводимых данных. Людям свойственно делать ошибки. Компьютер оперирует точными значениями. Поэтому две строки «Иванов» и « Иванов» для человека содержат одинаковую информацию, для операции сравнения строк — вторая длиннее на один символ — пробел в начале строки. Предположим, оператор случайно нажал на пробел, и в базу попал второй вариант — « Иванов». При последующей попытке найти нужную фамилию оператор набирает первый вариант — «Иванов». Но ничего найдено не будет. Для функции поиска в базе просто не существует требуемой строки. Выход есть (и это, кстати, правило хорошего тона для программиста) — отсекают пробелы слева и справа от значимой части с помощью специальной функции.

Можно придумать дополнительные правила контроля, например, сводить количество пробелов между словами до одного. Тогда при хранении фамилии, имени и отчества в одном поле лишние пробелы между ними, случайно сделанные оператором, будут удалены. В случае с названиями улиц можно проверять наличие пробела после точки, как этого требуют правила делопроизводства. Пробелы ставятся между сокращениями и следующим словом, как, например, в строке «ул. Ф. Лыткина». При выполнении таких правил легче осуществлять поиск по образцу.

5. Не теряйте информацию при изменении данных. Предположим, вы разработали БД для инструментального магазина. Продаются отвертки, молотки, стамески. Кроме «Названия» объекты будут обладать атрибутом «Цена». Ну, и еще надо хранить сведения о количестве проданного. Для этого введем таблицы «Справочник товаров» и «Продажи». Пусть в «Справочнике товаров» хранятся название и цена, а в «Продажах» — дата и количество.

Такая структура будет работать до первого изменения цены. Представьте себе — работает магазин уже неделю, по данным из БД созданы десятки документов (расходные, накладные, отчеты, прайс-листы), но тут меняется цена изделия. А у нас цена хранится в справочнике. Если мы изменим ее, то все созданные ранее документы потеряют данные, на основании которых их создавали. Цена-то теперь указана другая. Месяц назад в отчете о продажах за месяц получилась одна сумма, сегодня в отчете за тот же месяц — другая. Очевидное решение — хранить цену в таблице о продажах. Поскольку там есть поле «Дата продажи», то цена будет указана индивидуально для каждой даты. Можно предложить более сложный вариант — вынести цену в отдельный справочник, добавив туда поле «Дата начала действия цены».

Это позволит менять цену с точностью до минуты. Минуту назад торговали по одной цене, сейчас — по другой. Случай кажется довольно нестандартным, пока мы не вспомним о бирже и скорости изменения котировок акций. Заметьте, что из таблицы продаж исчезло поле «Код названия». Все правильно, оно теперь доступно через таблицу «Справочник цен на товары». Мы всегда можем по строке из таблицы «Продажи» узнать название проданного товара, составив соответствующий запрос к трем таблицам.

Наш магазин очень простой, в реальной задаче атрибутов будет гораздо больше, и вот тут-то программист должен будет выбирать наиболее подходящую структуру данных. Вполне можно представить такие случаи, когда любой из этих вариантов будет оптимальным для своего случая.

Формулируя требование более строго, можно сказать: хорошо спроектированная БД хранит состояние данных на каждый необходимый период времени.

Если в нашей БД у изделия может измениться название, то в таком случае правильнее будет сразу завести для каждого изделия уникальный артикул, а название хранить в отдельной таблице с датой использования этого названия. Тогда все отчетные документы смогут использовать то название, которое действовало в требуемый период времени.

Или, например, улицы, которые время от времени переименовывают. Если система работает с историческими данными (т. е. данными, охватывающими большой период времени), то ей потребуется для одного и того же объекта (улица) хранить все варианты названия и учитывать период действия каждого из них.

3. СУБД ACCESS 2007

Программа Microsoft (MS) Access 2007 является преемником MS Access 2003, а также членом дружного семейства MS Office 2007. Ряд новых возможностей связан с родовыми чертами MS Office 2007, остальные же свойства связаны с диалектическим законом перехода количества в качество, проявившимся в процессе долгой эволюции от MS Access 2003 к MS Access 2007.

Прежде чем устанавливать MS Office 2007, выберите выпуск программного пакета, в состав которого входит MS Access 2007, из следующего списка:

- MS Office Professional 2007;
- MS Office Professional Plus 2007;
- MS Office Enterprise 2007.

MS Access 2007 является частью MS Office System, так что базовые объекты интерфейса — меню, панели инструментов, диалоговые окна — работают точно так же, как в других продуктах Office 2007 или других приложениях для MS Windows.

3.1. Новые функциональные возможности СУБД Access 2007

В программе Access 2007 используется *новый интерфейс пользователя*, который обеспечивает высокую эффективность и производительность работы. Главным элементом интерфейса является особая панель — **Лента инструментов**, заменившая все меню и панели инструментов, которые были в предыдущих версиях программ Office.

На **Ленте** собраны вместе все команды, которые вы можете выполнять. Они сгруппированы по тематическим вкладкам. **Лента** имеет вкладки двух типов:

- *основные* (**Главная, Создание, Внешние данные, Работа с базами данных**), на которых собраны команды и инструменты, доступные в любой момент;

- *контекстные* (**Конструктор, Режим таблицы, Формат, Упорядочить** и т. д.), которые появляются на **Ленте** по мере необходимости. Количество отображаемых контекстных вкладок и их

названия связаны с текущим (выделенным) объектом БД, его режимом или задачами, которые можно выполнить.

Начинается работа в программе Access 2007 со страницы **Приступая к работе с Microsoft Office Access**.

Здесь вы не только можете создать новую (из шаблона или пустую) или открыть существующую БД, но также просмотреть информацию из справочной системы или из Интернета. Единственная традиционная панель инструментов, которая осталась в Access 2007 — это **Панель быстрого доступа**. На ней расположены кнопки чаще всего используемых команд (**Сохранить**, **Отменить** и др.). Вы можете добавлять на нее кнопки любых команд по своему усмотрению.

Плавающее окно БД было заменено **Областью переходов**, прикрепленной к левому краю окна программы.

Область переходов можно сворачивать. Она содержит все объекты текущей БД, но в ней не осталось средств для создания объектов и управления ими (кроме, естественно, контекстного меню, которое можно открыть для любого объекта). Способ отображения объектов может быть настроен. Из **Области переходов** можно открыть любой объект БД в любом режиме представления.

Наконец, объекты БД открываются на вкладках в рабочей области (а не в отдельных окнах). Вверху вкладки каждого открытого объекта отображается его ярлык.

Это позволяет легко видеть все объекты, с которыми вы работаете, и не терять из виду те, которые закрыты другими. Однако остается возможность вернуться к старому, оконному способу отображения объектов.

Новые форматы БД. Формат БД программы Access 2007 — .accdb (и производные форматы .accde, .accdt, .accdc и .accdr) — обеспечивает поддержку новых возможностей программы. При этом старые файлы (.mdb и др.), как и их упраздненные возможности, продолжают полноценно работать в новой версии программы. Более того, в Access 2007 вы можете создавать БД и проекты Access в старых форматах. Работа с БД в формате .accdb в более ранних версиях программы не обеспечивается.

Новый режим представления форм и отчетов — режим **Макета**. Он позволяет редактировать структуру формы или отчета вместе с данными. Здесь легко добавлять элементы, выравнивать и форматировать их, применяя специально предназначенные для этого средства. Элементы управления можно объединять в макеты

двух типов: в столбик или ленточный. Средства редактирования применяются сразу ко всему макету.

Тематические шаблоны нового типа. Новые шаблоны БД доступны непосредственно со страницы **Приступая к работе с Microsoft Office Access**. Они позволяют быстро создавать структурно законченные тематические БД формата Access 2007. Каждая БД включает в себя таблицы, формы, отчеты, запросы, макросы и межтабличные связи, но не заполнена никакими данными. Некоторое количество шаблонов входит в установку программы Access 2007.

Средства быстрого создания объектов. Вкладка **Создание ленты** собрала все средства (в том числе и вновь появившиеся) для создания объектов БД (таблиц, форм, отчетов, запросов, макросов, модулей, списков SharePoint и др.). Средства для создания объектов значительно улучшены и хорошо согласованы. Например, если открыта или выделена таблица, то создать на ее основе форму можно всего лишь двумя щелчками мыши. Автоматически созданные формы и отчеты получают единообразный вид, включающий, например, общую эмблему, цветовой стиль, дату и т. д.

Быстрое заполнение таблиц. Программа Access 2007 научилась автоматически распознавать данные, которые вводятся в бланк таблицы, и на основе этого задавать тип данных для нового поля. Поэтому во многих случаях процесс создания таблиц значительно упростился — таблицу можно создавать сразу в режиме **Таблицы**, минуя режим **Конструктора**. Также средство распознавания типа данных работает при вставке таблицы из программы Excel.

Строки итогов в таблицах. Еще одна новая возможность заключается в следующем: в режиме **Таблицы** можно добавлять строки с итоговыми значениями полей. В ячейках итогов могут размещаться суммы, средние значения, количества значений, максимальные и минимальные значения, стандартные отклонения и дисперсии. Это удобно для быстрого анализа данных таблиц.

Готовые шаблоны полей. Пользователи могут не утруждать себя созданием полей и заданием их свойств — в программе Access 2007 есть панель **Шаблоны полей**, в которой имеется 70 готовых полей в 7 различных категориях. Выберите поле в панели и перетащите его в таблицу, которую вы создаете. Такой способ гарантирует единообразие полей. Поэтому его удобно принимать для обязательного применения при совместной деятельности в составе рабочей группы.

Разделенная форма. В Access 2007 добавлен новый тип форм, который создается автоматически — разделенная форма. Она комбинирует в себе для одной и той же таблицы (или запроса) режим **Таблицы** и режим **Формы**. Такой тип формы удобно использовать для ввода данных в таблицу. С помощью специального свойства формы можно задать, где должно располагаться табличное представление: слева, справа, сверху или снизу.

Новый тип данных «Вложение». В Access 2007 добавлен новый тип данных — «Вложение». В полях такого типа информация хранится в двоичной кодировке и сжатом виде. Таким образом в БД удобно сохранять изображения, графику, файлы программ Office или любые другие типы файлов. В одной записи может быть несколько полей с этим типом данных. В каждом поле могут храниться несколько файлов. Из своей ячейки файл легко открывается и обрабатывается в Windows в той программе, в которой он был создан (или к которой он приписан). В такие поля удобно, например, сбрасывать промежуточные файлы при совместной работе над проектом.

3.2. Объекты Access 2007

БД — это компьютерный эквивалент организованного списка информации. Преимущество БД состоит в вашей возможности быстро получать из нее точную информацию. В Access данные организованы в виде таблиц. Access 2007 является реляционной БД, так что вы можете считать несколько таблиц одной БД как единое место хранения и легко извлекать информацию из различных таблиц в том порядке и формате, который наиболее вам подходит.

Таблица — один из объектов, с которыми вы можете работать в Access. Таблицы являются ключевыми объектами БД, а целью других объектов (запросов, форм, отчетов, страниц доступа к данным, макросов и модулей) является взаимодействие с одной или несколькими таблицами.

Каждый объект Access 2007 имеет два или более способов представления. Например, вы просматриваете данные из таблицы в режиме **Таблицы** и определяете, как эти данные отображаются в режиме **Конструктора**.

Одним из способов поиска информации в БД Access является создание и запуск запроса. Вы используете запросы для выборки

информации так, что затем ее можно просматривать, изменять или анализировать. Запросы могут просматриваться в режиме **Таблицы** или в режиме **Конструктора**, но вы также можете использовать результаты запроса как основу для других объектов Access, таких как форма или отчет.

Формы облегчают пользователям ввод, получение, отображение и печать информации, хранящейся в таблицах. Форма – это окно, в котором вы можете поместить элементы управления, которые либо предоставляют пользователям информацию, либо получают информацию, которую они вводят. Формы могут просматриваться в режиме **Формы**, в режиме **Таблицы** или в режиме **Конструктора**.

Отчеты отображают информацию из ваших таблиц на экране компьютера или на бумаге в отформатированном виде. Отчет может включать элементы информации, выбранные из нескольких таблиц и запросов, значения, вычисленные на основе информации из БД, и форматирующие элементы, такие как заголовки, колоннотитулы, названия и «шапки». Отчеты могут просматриваться в режиме **Конструктора**, в режиме **Предварительного просмотра** и в режиме **Просмотра образца**.

Страницы доступа к данным, макросы и модули значительно расширяют возможности Access.

Страницы доступа к данным позволяют просматривать и манипулировать информацией из вашей БД через Интранет или Интернет.

Макросы могут использоваться для того, чтобы сделать часто повторяющиеся действия доступными в виде командных кнопок на формах, которые помогают менее опытным пользователям работать с вашей БД.

Модули – это программы на Microsoft Visual Basic for Applications (VBA). В то время как макросы могут автоматизировать многие действия, VBA может использоваться для выполнения задач, которые слишком сложны для выполнения с помощью макросов.

3.3. Физическая структура данных

Для создания физической структуры данных следует указать тип данных для каждого поля таблицы. Различные СУБД могут хранить и обрабатывать разные типы данных. В Access 2007 используются типы данных, представленные в табл. 3.1.

Типы данных полей таблиц БД

| Тип данных | Описание | Размер |
|--------------------|---|--|
| Текстовый | Алфавитно-цифровые знаки. Используется для текста или текста и чисел, не применяемых в расчетах | До 255 знаков |
| Поле МЕМО | Алфавитно-цифровые знаки или форматированный текст. Используется для текста длиннее 255 знаков | До 1 Гб знаков или 2 Гб памяти |
| Числовой | Используется для хранения числовых данных, используемых в вычислениях, за исключением денежных значений | 1, 2, 4 или 8 байт (или 16 байт, когда поле используется для кодов репликации) |
| Дата/время | Используется для хранения значений даты и времени | 8 байт |
| Денежный | Используется для хранения денежных значений (валюты) | 8 байт |
| Счетчик | Уникальное числовое значение, которое при добавлении новой записи автоматически вводит Office Access 2007 | 4 или 16 байт, если используется для кодов репликации |
| Логический | Используется для полей, которые могут содержать одно из двух значений, например «Да» и «Нет» | 1 бит (8 бит = 1 байт) |
| Поле объекта OLE | Используется для хранения OLE-объектов других программ приложений MS Windows | До 1 Гб |
| Вложение | Рисунки, изображения, двоичные файлы, файлы MS Office. Стандартный тип данных для сохранения цифровых изображений | Для сжатых вложений – 2 Гб. Для несжатых – примерно 700 Кб |
| Гиперссылка | Используется для хранения гиперссылок вызова веб-страниц и на объекты Access | До 1 Гб знаков или 2 Гб памяти |
| Мастер подстановок | Используется для запуска мастера подстановок, с помощью которого можно создать поле, позволяющее выбрать значение из другой таблицы, запроса или списка значений, используя поле со списком | На основе таблицы или запроса – размер привязанного столбца. На основе значения – размер текстового поля, содержащего значение |

При вводе типа данных одновременно можно задать и свойства поля (табл. 3.2), не совпадающие со значением, присваиваемым Access 2007 по умолчанию. С помощью значений свойств полей можно управлять отображением данных, предотвращать ввод неверных значений, задавать значения по умолчанию, ускорять поиск и сортировку, а также управлять другими функциональными характеристиками и внешним видом полей.

Таблица 3.2

Свойства полей таблиц БД

| Свойство поля | Использование |
|-------------------------|--|
| Размер поля | Задание максимального размера данных, сохраняемых в полях с типом данных «Текстовый», «Числовой» или «Счетчик» |
| Формат | Настройка формата данных поля для отображения или печати |
| Число десятичных знаков | Задание количества отображаемых знаков в дробной части для числовых значений |
| Новые значения | Определение способа присвоения значений для поля «Счетчик»: последовательное увеличение или случайные значения |
| Маска ввода | Отображение специальных знаков для управления вводом данных |
| Подпись | Определение текста, отображаемого по умолчанию в надписях для форм, отчетов или запросов |
| Значение по умолчанию | Автоматическое назначение полю значения по умолчанию при добавлении новых записей |
| Условие на значение | Позволяет контролировать ввод, задает ограничения на вводимые значения. При нарушении заданного условия программа запрещает ввод и выводит сообщение, текст которого задается в свойстве «Сообщение об ошибке» |
| Сообщение об ошибке | Ввод текста, который будет отображаться при нарушении значения правила «Условие на значение» |
| Индексированное поле | Ускорение доступа к данным в этом поле путем создания и использования индекса |

3.4. Практическая работа № 1 Концептуальное проектирование

Цель работы: приобрести навыки анализа предметной области; выделить основные абстракции в предметной области и определить их параметры; научиться создавать информационную модель данных.

Рассмотрим задачу о зачислении абитуриентов на бюджетные места в некоторый вуз. Абитуриенты сдают экзамены на один или несколько факультетов вуза. Известно расписание экзаменов: дата, предмет, факультет, на который экзамен сдается. По результатам экзаменов абитуриенты получают оценки. По каждому абитуриенту хранятся некоторые данные.

Требуется построить различные варианты информационной модели данных (представления данных), сравнить предложенные варианты.

Рассмотрим несколько вариантов модели.

Вариант 1.

Представим всю информацию как характеристики одного объекта – экзаменационной оценки:

Оценка

| |
|-----------------------|
| Предмет экзамена |
| Дата экзамена |
| Факультет |
| Фамилия |
| Имя |
| Отчество |
| Номер аттестата |
| Дата выдачи аттестата |
| Значение оценки |

Видно, что информация об абитуриенте дублируется, т. к. при внесении информации о новой оценке мы должны заново вносить уже введенную ранее информацию по абитуриенту (фамилию, имя, отчество, номер аттестата, дату выдачи аттестата). При вводе одной и той же информации можно допустить ошибки, а значит БД перейдет в противоречивое состояние. Предположим, что Сергеев

Сергей Петрович сдал экзамен по математике на факультет ВМК на оценку 5. Мы внесли информацию об этом (математика, 22 июля 2003 года, ВМК, Сергеев, Сергей, Петрович, 123123, 21 июня 2003 года, 5) в нашу БД. Через некоторое время данный абитуриент сдает информатику. Мы вносим информацию (информатика, 25 июля 2003 года, ВМК, Сергеев, Сергей, Петрович, 123123, 22 июня 2003 года, 5). При вводе была допущена ошибка — мы неправильно ввели дату выдачи аттестата. Таким образом, наша БД дает противоречивую информацию: по одним данным аттестат был получен Сергеевым 21 июня, по другим — 22 июня.

Кроме того, даже если информация была бы введена правильно, мы увеличиваем объем информации, что приводит к необходимости увеличения ресурсов и замедлению работы программного обеспечения. Избавимся от данного недостатка и построим другую информационную модель.

Вариант 2.

Представим эту информацию как характеристики двух объектов — абитуриента и оценки:

Абитуриент

| |
|-----------------------|
| Фамилия |
| Имя |
| Отчество |
| Номер аттестата |
| Дата выдачи аттестата |

Оценка

| |
|------------------|
| Предмет экзамена |
| Дата экзамена |
| Факультет |
| Значение оценки |

Между двумя сущностями должна существовать связь (абитуриент получает оценки).

АБИТУРИЕНТ <получает> ОЦЕНКА.

Для каждой полученной каждым абитуриентом оценки дублируется информация о предмете экзамена, дате экзамена, факультете.

Попробуем избавиться и от этого недостатка, представив информационную модель в виде следующих таблиц.

Вариант 3.

Абитуриент

| |
|-----------------------|
| Фамилия |
| Имя |
| Отчество |
| Номер аттестата |
| Дата выдачи аттестата |

Экзамен

| |
|------------------|
| Предмет экзамена |
| Дата экзамена |
| Факультет |

Оценка

| |
|-----------------|
| Значение оценки |
|-----------------|

Между тремя сущностями существует две связи:

АБИТУРИЕНТ <получает> ОЦЕНКА;

АБИТУРИЕНТ <сдает> ЭКЗАМЕН.

В данной модели данных нет недостатков, отмеченных в предыдущих двух. Возьмем этот вариант за основной при построении реляционной модели.

Реляционная модель будет представлена следующими отношениями:

Экзамены (Код экзамена, Предмет, Факультет, Дата).

Абитуриенты (Код абитуриента, Фамилия, Имя, Отчество, Номер аттестата, Дата выдачи аттестата).

Оценки (Код экзамена, Код абитуриента, Значение оценки).

Поля **Код экзамена** и **Код абитуриента** в таблице **Оценки** являются полями для реализации связи с соответствующими таблицами.

Задания по анализу предметных областей, выделению основных абстракций, определению их параметров и созданию информационной модели данных представлены в приложении.

3.5. Практическая работа № 2 Интерфейс СУБД Access 2007

Цель работы: научиться запускать и работать с основными объектами СУБД Access 2007; ознакомиться со средой.

Система MS Access 2007 входит в профессиональный программный комплекс MS Office и представляет собой мощное средство для работы с БД.

В программе Access 2007 можно создавать БД следующих версий:

- Access 2000 (формат файла *.mdb);
- Access 2002–2003 (формат файла *.mdb);
- Access 2007 (формат файла *.accdb).

Открывать и преобразовывать можно БД, созданные и в более ранних версиях.

Файлы БД отличаются от файлов других приложений MS Office. К ним предъявляются особые требования по безопасности. От достоверности информации, которая в них содержится, очень часто зависит благополучие множества людей (например, база регистрации автомобилей в ГАИ или база паспортного контроля в ОВИРЕ). Поэтому целостность содержимого БД не может и не должна зависеть ни от конкретных действий некоего пользователя, забывшего сохранить файл, ни от перебоев электричества.

Для сохранения информации используется двойной подход. Операции по изменению структуры БД, созданию таблиц или иных объектов происходят при сохранении файла БД. Операции же по изменению содержимого данных, не затрагивающие структуру базы, выполняются автоматически, как только вы внесли изменения.

Запустить программу можно несколькими способами, предусмотренными для всех программ пакета MS Office.

При запуске MS Access 2007 на экране появится стартовое окно **Приступая к работе с Microsoft Office Access**, показанное на рис. 3.1.

С помощью данного окна можно выбрать, в каком ключе вы будете работать:

- создавать новую БД «с нуля»;
- открыть уже созданную ранее БД;
- воспользоваться одним из предлагаемых шаблонов и создать новую БД на основе одного из них.

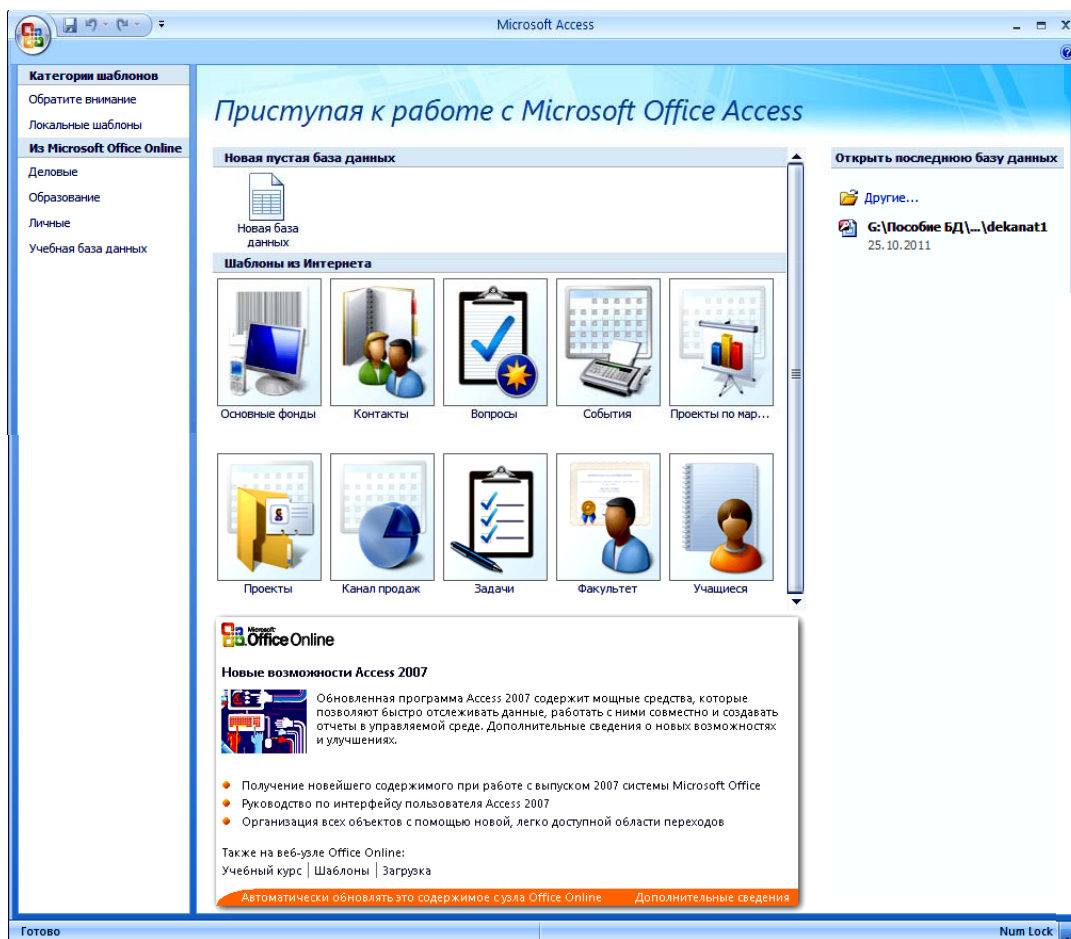


Рис. 3.1. Окно **Приступая к работе с Microsoft Office Access**

Для создания новой БД нужно выбрать **Новая база данных** и щелкнуть по появившейся кнопке «**Создать**», предварительно определив местоположение файла вашей БД на диске компьютера и задав ему имя. Выбрать одну из недавно открывавшихся БД можно в области **Открыть последнюю базу данных**, которая находится в правой части окна.

Для создания новой БД на основе шаблона нужно выбрать шаблон, а затем в правой части окна щелкнуть по кнопке «**Создать**», предварительно задав имя файла и его местоположение.

При открытии ранее созданной БД появится **Предупреждение системы безопасности**, говорящее о необходимости нажать кнопку «**Параметры**» и задать настройки для корректного открытия базы (рис. 3.2). В появившемся диалоговом окне **Параметры безопасности Microsoft Office** следует перевести переключатель в положение **Включить это содержимое** и затем активизировать кнопку «**ОК**».

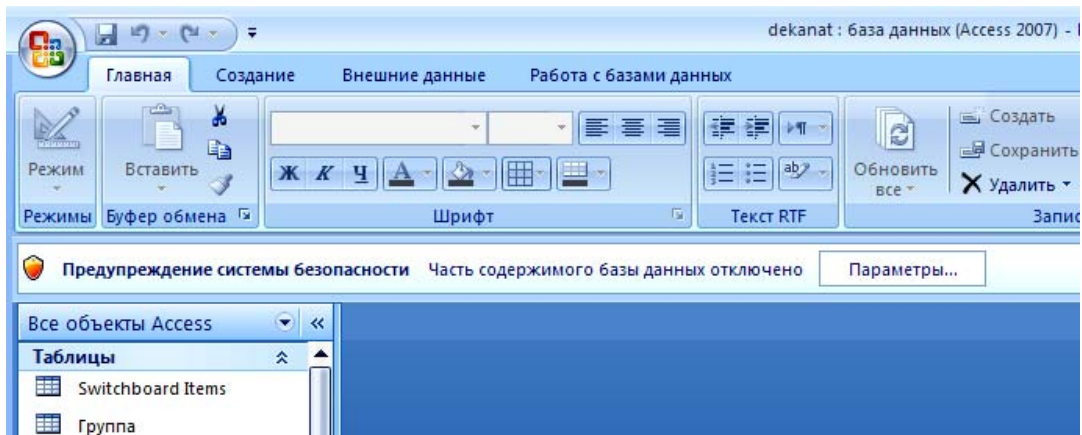


Рис. 3.2. Предупреждение системы безопасности

Вверху окна Microsoft Office Access 2007 (рис. 3.3) располагается **Лента инструментов**, а над ней — **Панель быстрого доступа**. Центральную часть занимает окно открытой БД с ее элементами. Внизу располагается **Строка состояния**.

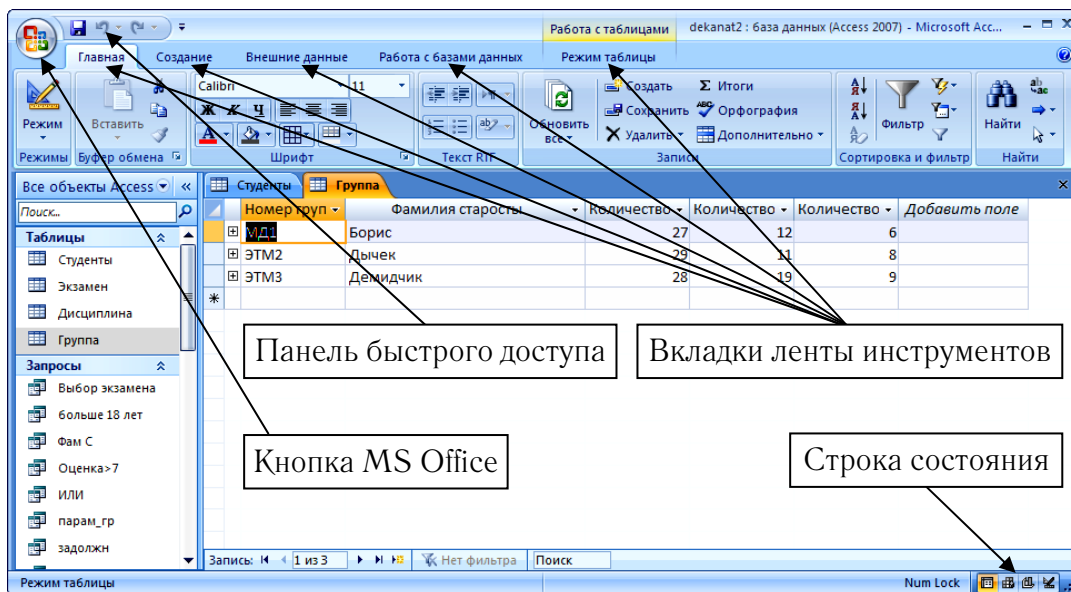


Рис. 3.3. Окно БД

На **Ленте инструментов** располагаются следующие вкладки:

- **Главная** — данная вкладка доступна по умолчанию и содержит команды, позволяющие выбрать режим представления БД (**Таблицы** или **Конструктора**), вырезать / вставить / скопировать данные с одного места на другое, задать шрифтовое оформление, произвести некоторые основные операции с записями в БД, а также фильтрацию и сортировку данных.

- **Создание** — на этой вкладке размещены команды создания всевозможных элементов, объектов БД — таблиц, форм, отчетов и т. п.

- **Внешние данные** — команды данной вкладки призваны обеспечить преобразование данных из БД, например, в таблицы Excel и наоборот — импорт данных из источников различного происхождения.

- **Работа с базами данных** — предназначена для производства команд различного рода общих работ с объектами БД, такие как команды отображения схемы данных, показа зависимостей между объектами, анализа данных и т. п.

Может присутствовать вкладка **Режим таблицы**, появляющаяся при создании таблицы БД, а также другие всевозможные контекстные вкладки в зависимости от того, с каким объектом БД вы в данный момент работаете.

Над **Лентой** располагается **Панель быстрого доступа**. На ней располагаются инструменты, доступные в любой момент и видимые в окне независимо от того, на какие вкладки **Ленты** вы переходите. По умолчанию на данной панели размещено всего три инструмента: **Сохранить**, **Отменить** и **Вернуть**. Однако можно добавить на нее и другие. Для этого используют расположенную справа стрелку и в раскрывшемся списке выбирают необходимый инструмент (рис. 3.4).

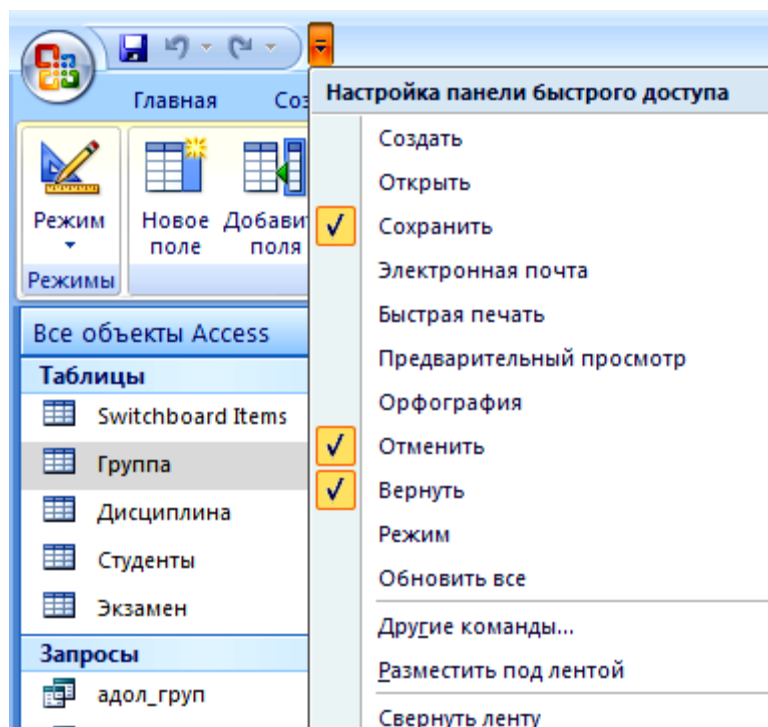


Рис. 3.4. Настройка **Панели быстрого доступа**

По левому краю окна Access 2007 имеется один специфический элемент, называемый **Областью переходов**. По умолчанию **Область перехода** находится в свернутом виде — видно одно ее название. Чтобы ее раскрыть, необходимо щелкнуть мышкой по двойной стрелочке вверх. В результате станет доступен список объектов текущей БД. С помощью **Области переходов** можно открывать, переименовывать, копировать и удалять объекты (рис. 3.5).

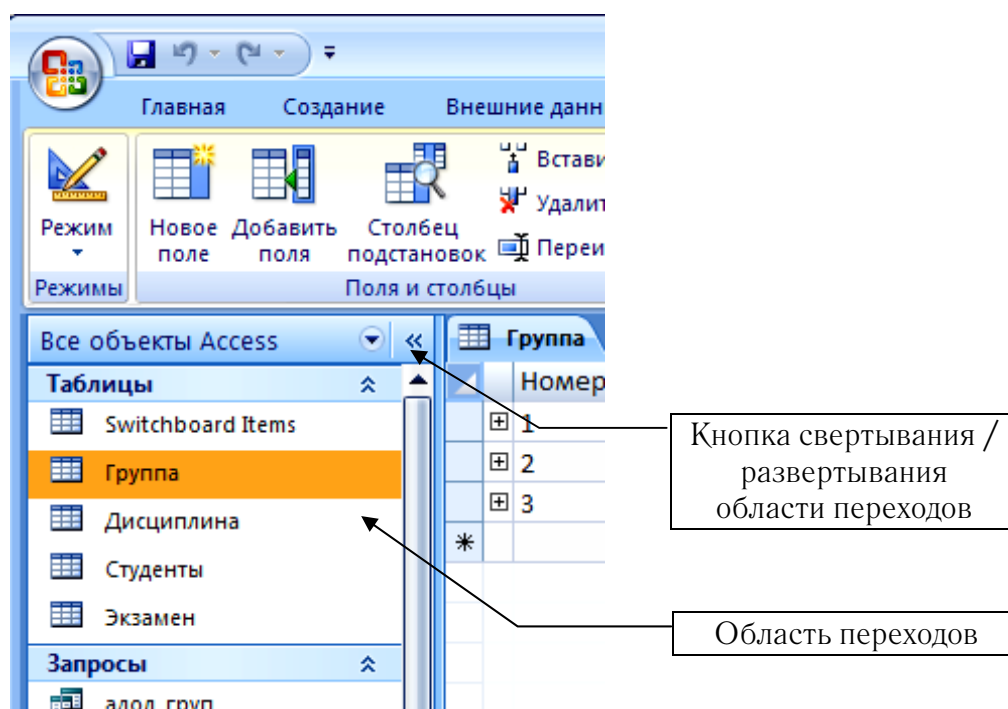


Рис. 3.5. Область переходов

Отображаемый в области переходов список объектов может быть отсортирован по различным критериям: по дате создания, дате изменения, типу и т. п.

3.6. Практическая работа № 3 Создание таблиц базы данных

Цель работы: изучить назначение составных элементов таблицы: полей, записей, ключей; научиться определять тип данных полей и их свойства; освоить способы создания таблиц в разных режимах (**Таблица**, **Шаблоны таблиц**, **Конструктор таблиц**, с помощью импорта внешних данных).

БД MS Access состоит из взаимосвязанных реляционных таблиц. Реляционная таблица является двумерной и содержит информацию об отдельном объекте БД.

Таблицу можно создать с помощью создания новой БД, формируя таблицу с помощью шаблона, вставки таблицы в существующую БД, а также импорта или создания ссылки на таблицу из другого источника данных, такого как книга Excel, документ Word, текстовый файл или другая БД.

3.6.1. Постановка задачи

Рассмотрим конкретный пример. Декан факультета хочет иметь такую БД, в которой он по фамилии студента сможет узнать сведения о его успеваемости, анкетные данные, сводку сданных и несданных экзаменов, когда студент их сдавал и кому. Возможно, ему будет полезна информация о количестве пропущенных занятий.

Однако, собрав все пожелания в одну структуру, легко получить совершенно нежизнеспособную конструкцию. Попробуем удовлетворить запросы деканата, создав четыре разные таблицы.

Таблицы должны содержать сведения:

- о контингенте студентов (**Студенты**) — фамилия, номер зачетной книжки, дата рождения, фотография, мобильный телефон, номер группы;
- об учебном плане (**Дисциплина**) — наименование дисциплины, фамилия преподавателя, семестр обучения;
- об успеваемости студентов (**Экзамен**) — дисциплина, номер зачетной книжки, оценка, дата сдачи экзамена, количество пропущенных часов.
- о группе (**Группа**) — номер группы, фамилия старосты, количество студентов, количество проживающих в общежитии, количество минчан.

Конечно, в реальной БД гораздо больше полей, но для учебного примера рассмотрим только эти.

При этом встает вопрос, как связать таблицы данных между собой. Для этой цели надо искать поле, которое обладает свойством уникальности. Эта уникальность состоит в том, что в таблице не может быть двух записей, содержащих одинаковое значение в данном поле. Такое уникальное поле, используемое для связи, называют полем первичного ключа или просто первичным ключом.

В нашем примере в таблице **Студенты** каждому студенту присвоен некий шифр (номер зачетной книжки). Обычно такие шифры в учебных заведениях образуются из цифр года поступления, номеров учебных групп и других подобных данных. Он и будет первичным ключом.

В таблице **Дисциплина** уникальным является название дисциплины, в таблице **Группа** — номер группы. Таблица **Экзамен** не имеет первичных ключей, т. к. все поля могут содержать повторяющиеся записи.

В приведенном примере поля в таблицах должны быть скоординированы таким образом, чтобы отображать сведения об одном и том же студенте. Эта координация осуществляется путем установления связей между таблицами.

3.6.2. Элементы объекта «Таблица»

Предположим, что нам надо изготовить самый простой вариант БД, когда вся информация хранится в одной таблице. В простой БД, такой как **Информация о студентах**, можно использовать всего одну таблицу — **Студенты**, где будут храниться фамилия студента, номер зачетной книжки, дата рождения, фотография, мобильный телефон, номер группы.

Таблица состоит из:

- *записей* (строк). Каждая запись содержит данные только об одном объекте (студенте), дублирование записей запрещается;
- *полей* (столбцов). Каждый столбец характеризуется определенным типом и размером хранимой информации (текстовая, числовая, денежная, дата, графическая). Число столбцов в таблице определяется числом выбранных атрибутов объекта. Например, объект **Студент** имеет атрибуты (названия полей): **Фамилия, Дата рождения, Мобильный телефон, Номер группы** и т. д.

Для однозначного определения каждой записи таблица должна иметь уникальный (первичный) ключ. Ключ может состоять из одного или нескольких полей. Связь между отдельными таблицами обеспечивается одинаковыми полями в них — ключом связи.

3.6.3. Создание таблицы базы данных в режиме Таблицы

После открытия БД Access автоматически активизирует подготовку новой таблицы (рис. 3.6) только с одним именем поля — **Код**.

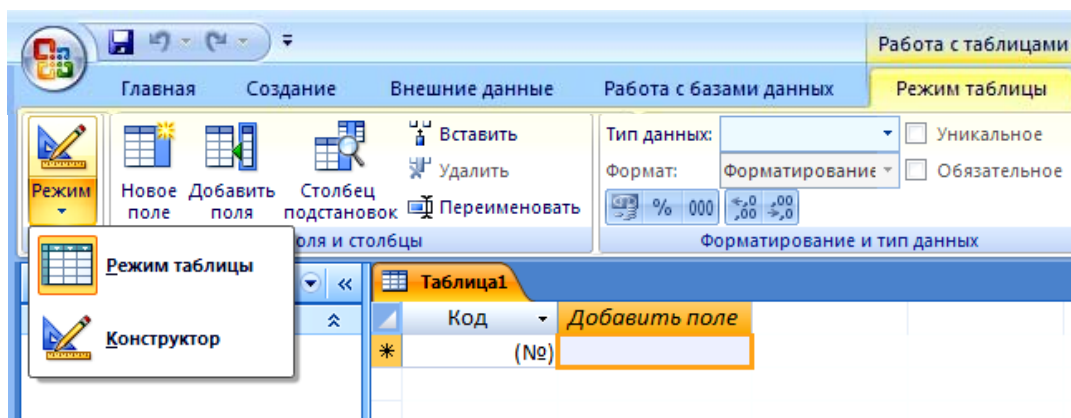


Рис. 3.6. Создание таблицы на основе пустой таблицы

Если ввести значение в пустой столбец, то появится новое имя **Поле1**. Access 2007 автоматически будете определять тип данных в зависимости от того, что вы будете вводить. Не распознанным данным присваивается тип «Текстовый».

Иногда требуется переопределить тип данных, назначенный автоматически. Для этого используются команды на вкладке **Режим таблицы** в группе **Форматирование и тип данных** (рис. 3.6).

Переименовывать, удалять, вставлять и добавлять поля можно с помощью команд вкладки **Режим таблицы** группы **Поля и столбцы** (рис. 3.7).

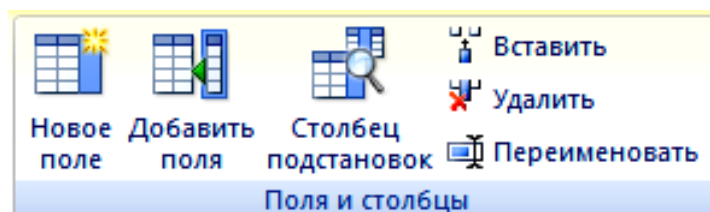


Рис. 3.7. Группа инструментов **Поля и столбцы**

MS Access 2007 имеет набор встроенных шаблонов, позволяющих значительно сократить время, затрачиваемое на создание полей. Можно выбрать поле из списка области **Шаблоны полей**. Для этого на вкладке **Режим таблицы** в группе **Поля и столбцы** щелкните кнопку **Новое поле**.

Появится область **Шаблоны полей**. Выберите одно или несколько полей и перетащите их в таблицу. Когда появится линия вставки, поместите поля в выбранное место. Эти поля появятся в таблице.

3.6.4. Создание таблицы базы данных в режиме Конструктор

Конструктор таблиц используется как для создания и настройки свойств новых таблиц, так и для внесения изменений в свойства уже существующих таблиц. Для вызова конструктора используют кнопку **Конструктор таблиц** на вкладке **Создание** в группе инструментов **Таблицы** (рис. 3.8).

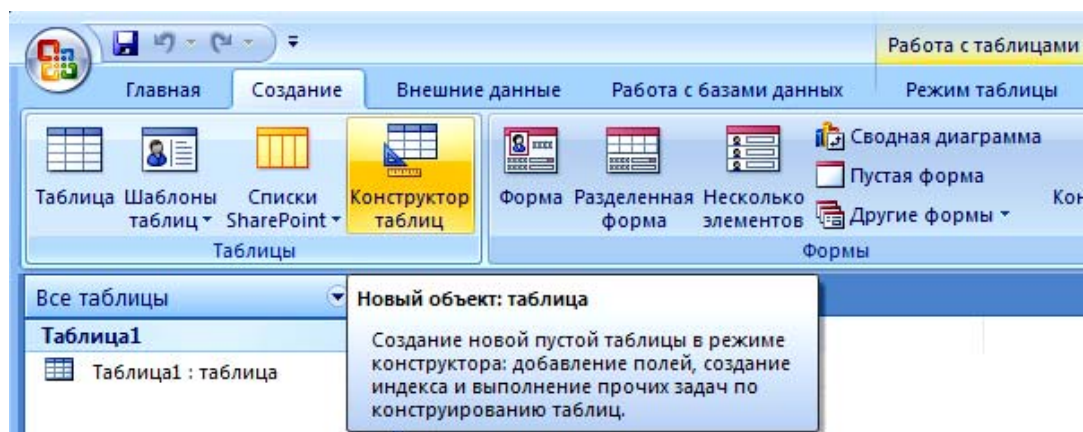


Рис. 3.8. Создание таблицы в режиме **Конструктор таблиц**

При первом открытии таблицы в режиме **Конструктор** появляется диалоговое окно **Сохранение** (рис. 3.9), где нужно задать имя таблицы, описывающее содержащиеся в ней данные (в нашем случае **Студенты**). Имя таблицы может содержать до 64 символов (букв или цифр) включая пробелы.

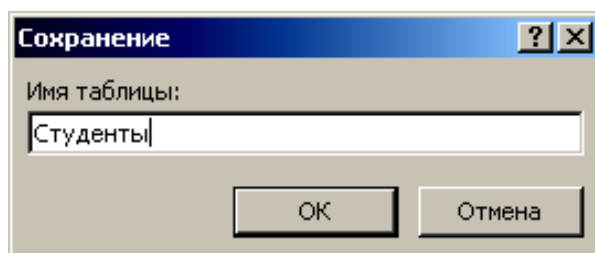


Рис. 3.9. Диалоговое окно **Сохранение**

Конструктор таблиц содержит в верхней части область для создания полей таблицы и в нижней — область для определения свойств каждого из этих полей (рис. 3.10). При включении нового поля в таблицу ему последовательно задают **Имя**, определяют **Тип данных** и заносят при необходимости комментарии в графу **Описание**. Эта графа не является обязательной.

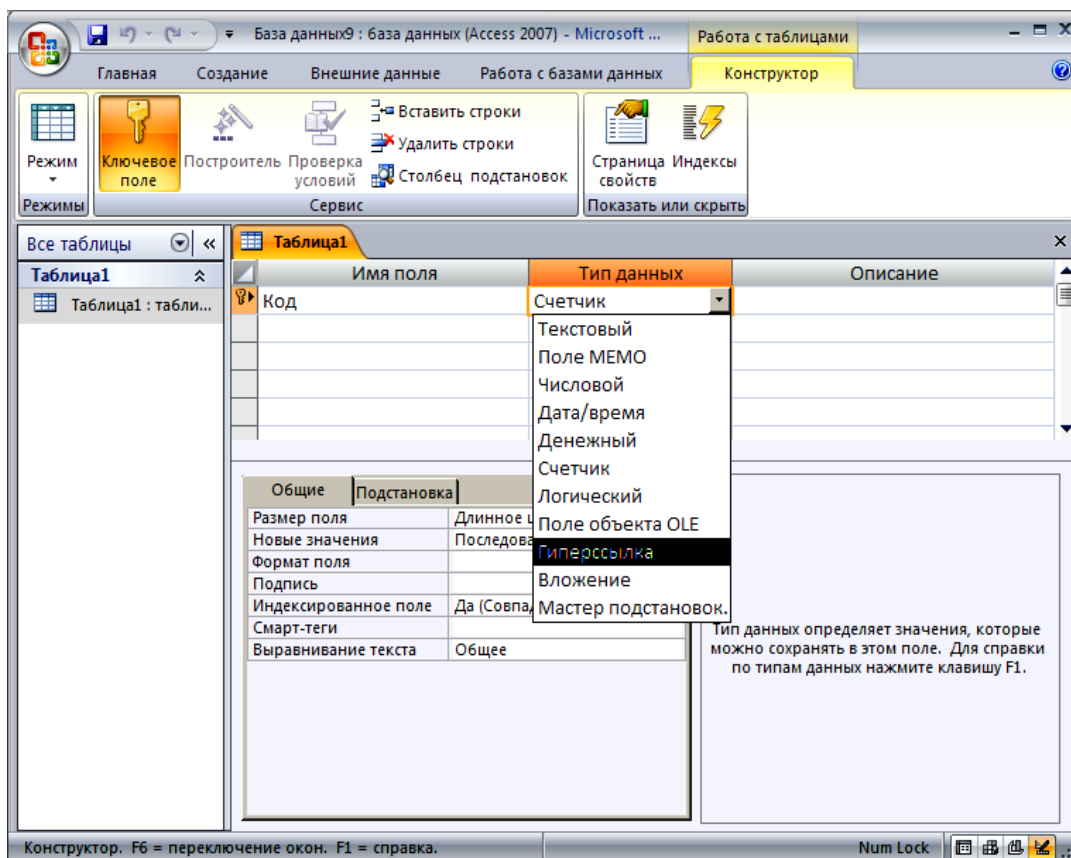


Рис. 3.10. Окно **Конструктор таблиц**

Имя поля состоит из комбинации букв, цифр, пробелов и специальных символов за исключением «.», «!», «'», «[]». Максимальная длина имени — 64 символа с учетом пробелов. Для элементов управления, которые могут быть расположены на формах и отчетах БД, — не более 255 символов.

Тип данных определяется видом хранимой в поле информации.

Для телефонных, инвентарных и других номеров, которые не используются в математических вычислениях, вместо числового выбирают текстовый тип данных.

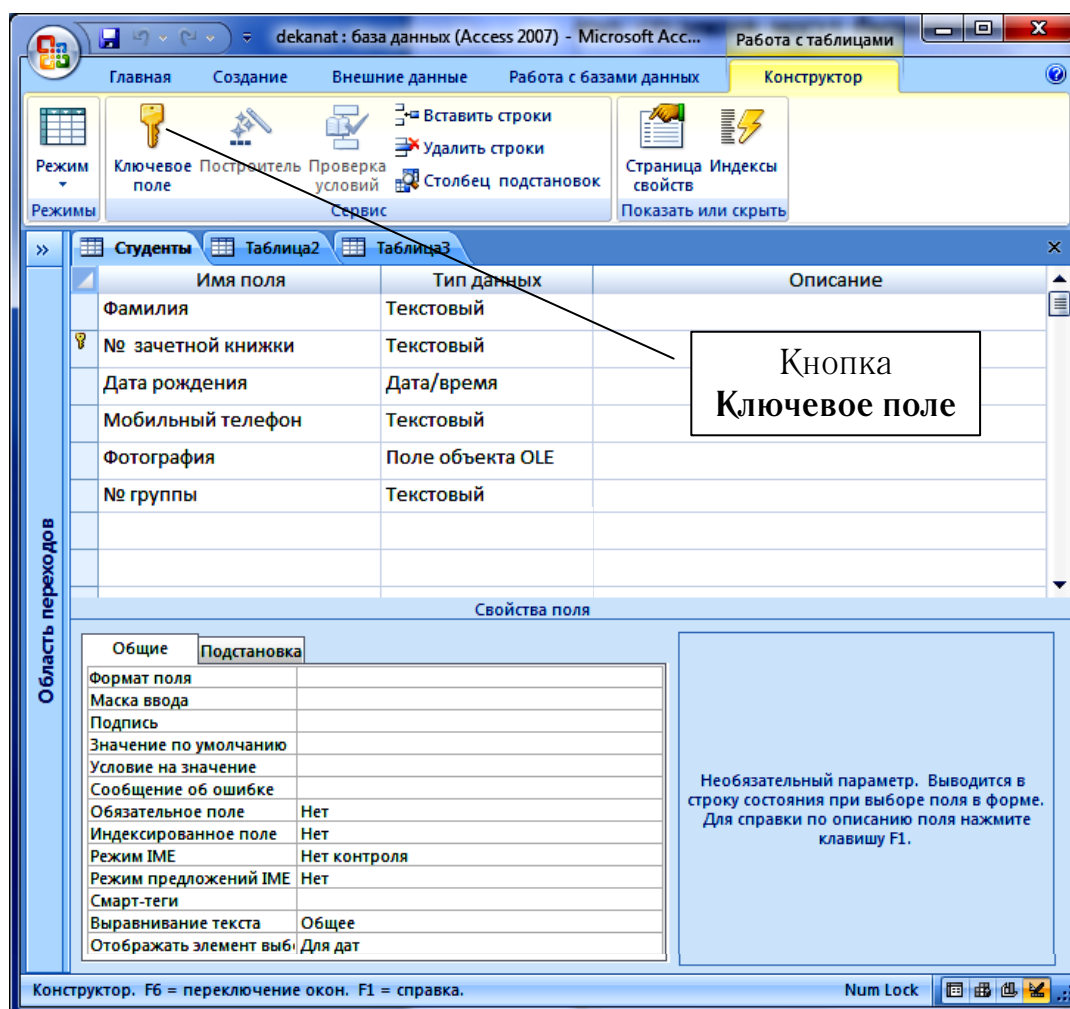
При вводе типа данных одновременно можно задать и свойства поля, не совпадающие со значением, присваиваемым Access 2007 по умолчанию. С помощью значений свойств полей можно управлять отображением данных, предотвращать ввод неверных значений, задавать значения по умолчанию, ускорять поиск и сортировку, а также управлять другими функциональными характеристиками и внешним видом полей.

Для таблицы **Студенты** зададим следующие поля и их типы (табл. 3.3, рис. 3.11).

Свойства полей и типы данных таблицы «Студенты»

| Имя поля | Тип данных | Свойства |
|-------------------|------------------|----------------|
| Фамилия | Текстовый | 255 |
| № зачетной книжки | Текстовый | 6 |
| Дата рождения | Дата/время | Краткий формат |
| Мобильный телефон | Текстовый | 12 |
| Фотография | Поле объекта OLE | |
| № группы | Текстовый | 6 |

Одно из полей таблицы может быть определено как ключевое. Можно было бы выбрать в качестве ключевого поле **Фамилия**. Однако оно не отвечает требованию уникальности данных. У разных студентов могут быть одинаковые фамилии. В нашем случае ключевым полем является **№ зачетной книжки**.

Рис. 3.11. Определение полей, их типов и свойств в режиме **Конструктор**

Ключевое поле таблицы задается с помощью контекстного меню при выделении его мышью и выполнении команды **Ключевое поле** либо нажатием кнопки **Ключевое поле** (рис. 3.11).

Поле **№ зачетной книжки** является первичным ключом этой таблицы. При необходимости можно удалить или заменить первичный ключ. Для этого нужно выделить поле и нажать на кнопку **Ключевое поле**, индикатор ключа удаляется из поля, которое ранее было задано в качестве первичного ключа.

Сохранение таблицы. После добавления полей в таблицу необходимо сохранить ее структуру. При закрытии режима **Конструктора** появляется окно, предлагающее сохранить изменения макета или структуры таблицы.

Если при создании структуры таблицы первичный ключ не был установлен, то при ее сохранении будет предложено создать этот ключ. Если ответить «**Да**», приложение Access 2007 создаст поле **Код** с типом данных «Счетчик» для сохранения уникального значения для каждой записи. Если в таблице уже есть поле с таким типом данных, оно будет использовано в качестве первичного ключа. Можно ответить «**Нет**», в этом случае таблица будет сохранена без первичного ключа.

3.6.5. Создание таблицы на основе шаблона

Если вас не устраивает «медленный» способ создания структуры таблицы с последующим добавлением требуемых полей, то можно создать ее с помощью шаблона таблицы. Мы уже упоминали шаблоны для создания БД и полей. Теперь возвращаемся к ним, но уже на уровне объекта БД — таблицы.

Для создания таблицы на основе шаблона необходимо на вкладке **Создание** в группе **Таблицы** щелкнуть **Шаблоны таблицы** и затем выбрать из списка один из доступных шаблонов. Отобразится следующий перечень шаблонов таблиц: **Контакты, Задачи, Вопросы, События, Основные фонды**.

Необходимо выбрать шаблон, структура которого наиболее близка к структуре вашей таблицы. Например, для создания таблицы **Адреса студентов**, состоящей из полей **Фамилия, Индекс, Город, Адрес, Домашний телефон и Мобильный телефон**, можно выбрать шаблон **Контакты**. Используя контекстное меню (щелчком ПКМ по имени поля), можно удалять, переименовывать, вставлять поля. В результате появится таблица (рис. 3.12).

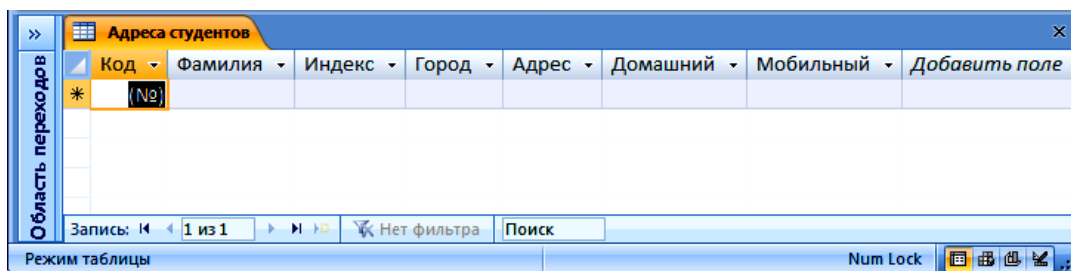


Рис. 3.12. Таблица Адреса студентов

3.6.6. Создание таблицы с помощью импорта внешних данных

В Access 2007 предусмотрена такая возможность как импорт информации из других источников, таких как электронных таблиц Excel, контактов Outlook, текстового редактора Word либо из ранее созданной БД. При импорте данных в Access создается их копия в новой или существующей таблице; при этом исходный файл не изменяется. Причем можно импортировать как всю базу, так и отдельные ее объекты.

Допустим, в деканате существует список всех студентов факультета, созданный в электронной таблице Excel (рис. 3.13).

| FAM | | Фамилия | | | | |
|-----|-------|------------|------------|---------------|------|--------|
| | A | B | C | D | E | F |
| | № п/п | Фамилия | Имя | Отчество | Курс | Группа |
| 1 | | | | | | |
| 2 | 1 | Борис | Илья | Леонидович | 1 | 3 |
| 3 | 2 | Войнич | Оксана | Ильинична | 1 | 3 |
| 4 | 3 | Демидчик | Мария | Викторовна | 1 | 3 |
| 5 | 4 | Дычек | Наталья | Владимировна | 1 | 3 |
| 6 | 5 | Ждан | Екатерина | Ивановна | 1 | 3 |
| 7 | 6 | Иваненко | Светлана | Ивановна | 1 | 3 |
| 8 | 7 | Кашлей | Екатерина | Феликсовна | 1 | 3 |
| 9 | 8 | Климович | Александра | Владимировна | 1 | 3 |
| 10 | 9 | Кондрашова | Вероника | Викторовна | 1 | 3 |
| 11 | 10 | Кужаль | Кристина | Васильевна | 1 | 3 |
| 12 | 11 | Лазаревич | Мargarита | Станиславовна | 1 | 3 |
| 13 | 12 | Лащ | Виктория | Леонтьевна | 1 | 3 |
| 14 | 13 | Лучкина | Александра | Васильевна | 1 | 3 |
| 15 | 14 | Лысёнок | Татьяна | Евгеньевна | 1 | 3 |
| 16 | 15 | Митрахович | Анна | Петровна | 1 | 3 |

Рис. 3.13. Лист Excel со списком студентов факультета

Нам требуется перенести список фамилий студентов в БД, т. е. импортировать содержимое листа Excel в существующую или новую таблицу Access.

Для этого необходимо открыть исходный файл Excel и выделить лист с данными, которые требуется импортировать в Access. Если необходимо импортировать лишь часть данных листа, то нужно определить именованный диапазон, который содержит только те ячейки, которые требуется импортировать (в нашем случае мы выделяем диапазон с фамилиями и присваиваем ему имя **FAM**).

Затем необходимо открыть БД Access, в которой будут храниться импортируемые данные. На вкладке **Внешние данные** в группе **Импорт** выберите команду **Excel** (рис. 3.14).

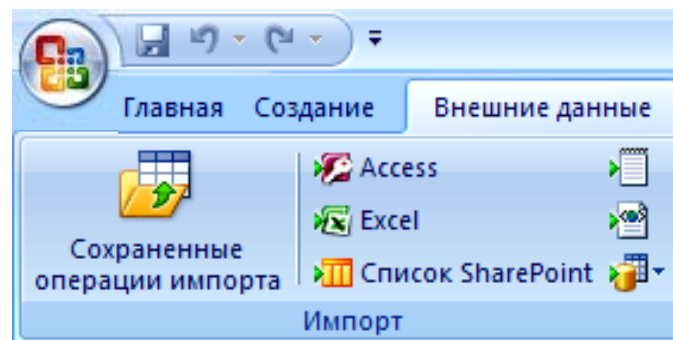


Рис. 3.14. Группа инструментов **Импорт**

В появившемся диалоговом окне в поле **Имя файла** с помощью кнопки **Обзор** необходимо указать имя файла Excel, который содержит данные для импорта, и способ сохранения импортируемых данных:

- **Импортировать данные источника в новую таблицу в текущей базе данных** — чтобы сохранить данные в новой таблице.
- **Добавить копию записей в конец таблицы** — чтобы добавить данные в существующую таблицу. Этот вариант недоступен, если БД не содержит таблиц.
- **Создать связанную таблицу для связи с источником данных** — если требуется хранить данные в листах Excel, но иметь возможность использовать удобные функции запросов и отчетов Access.

Для нашего случая выберем первый пункт — **Импортировать данные источника в новую таблицу в текущей базе данных**.

Затем будет открыт **Мастер импорта электронных таблиц**, с помощью которого выполняется пошаговый процесс импорта (выбор необходимых листов, заголовков столбцов, просмотра

свойств полей, имени таблицы и т. д.). Мастер предложит просмотреть свойства полей. Типы данных Access присваивает автоматически, анализируя вводимую информацию. При необходимости его можно изменить. Если импорт успешно завершен, то в результате получим новую таблицу (рис. 3.15).

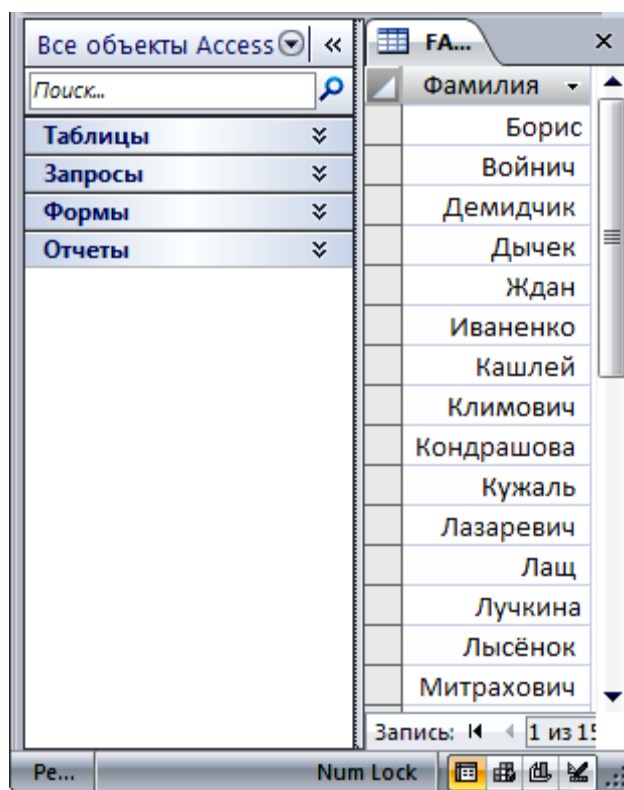


Рис. 3.15. Импортированная из Excel таблица

Аналогичным образом можно импортировать данные и из других источников.

3.6.7. Модификация структуры таблицы

Для изменения типа данных и их свойств используют раскрывающиеся меню, которые вызываются щелчком мыши на кнопке вызова списка соответствующего элемента таблицы.

Если при создании макета таблицы были допущены ошибки, то их можно исправить (используя контекстное меню либо соответствующие кнопки), например:

- изменить порядок расположения полей в таблице можно, перенеся строку с помощью мыши на нужное место;
- удалить поле можно, используя клавишу «**Del**» либо контекстное меню.

3.7. Практическая работа № 4 Организация связей между таблицами

Цель работы: изучить реальные отношения между таблицами; научиться устанавливать связи между ними.

Исходя из постановки задачи, одной из целей создания хорошей структуры БД является устранение избыточности данных (повторяющихся данных). Чтобы корректно выполнить это действие, нужно сначала понять взаимосвязи между таблицами и описать эти взаимосвязи в БД.

После создания макетов таблиц следует установить связи между ними. Access сможет использовать эти связи для удобства поиска информации в разных таблицах БД.

3.7.1. Организация связей

Связь между таблицами устанавливается с помощью поля (полей), которое содержит одинаковые значения для разных таблиц. Эта связь создается посредством первичного ключа одной таблицы с полем (внешним ключом), содержащим совпадающие значения данных, другой таблицы.

Вначале определяется, какая таблица является главной, а какая — подчиненной. Связываемое поле главной таблицы должно быть ключевым, т. е. первичным ключом таблицы. Связываемое поле подчиненной таблицы обычно является полем вторичного, или внешнего, ключа, тип данных и размер которого совпадают с полем первичного ключа главной таблицы.

В процедуре установки связи можно выделить три шага:

- открытие окна **Схема данных**;
- выбор таблиц или запросов, между которыми следует установить связь;
- непосредственно установка связи между полями таблиц с заданием для нее свойств.

Открытие диалогового окна **Схема данных** осуществляется с помощью одноименной кнопки на вкладке **Работа с базами данных** в группе инструментов **Показать или скрыть** (рис. 3.16).

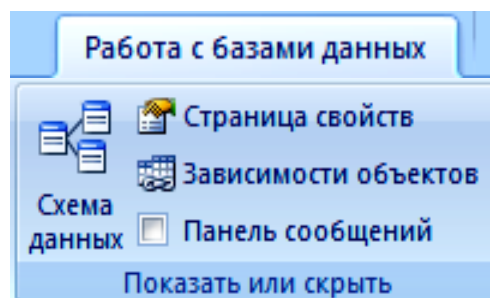


Рис. 3.16. Группа инструментов **Показать или скрыть**

Если связи просматривались или изменялись ранее, то диалоговое окно будет содержать последнюю сохраненную схему.

При первом открытии диалогового окна оно будет пустым, а Access откроет окно **Добавление таблицы** (рис. 3.17). Если оно не появится, необходимо в группе инструментов **Связи** нажать кнопку **Отобразить таблицу**. Затем в списке **Таблицы и запросы** выделить нужную таблицу и нажать кнопку «**Добавить**». Самый быстрый способ выбора таблицы (или запроса) для установки связи состоит в ее переносе из окна БД в диалоговое окно **Схема данных** с помощью мыши.

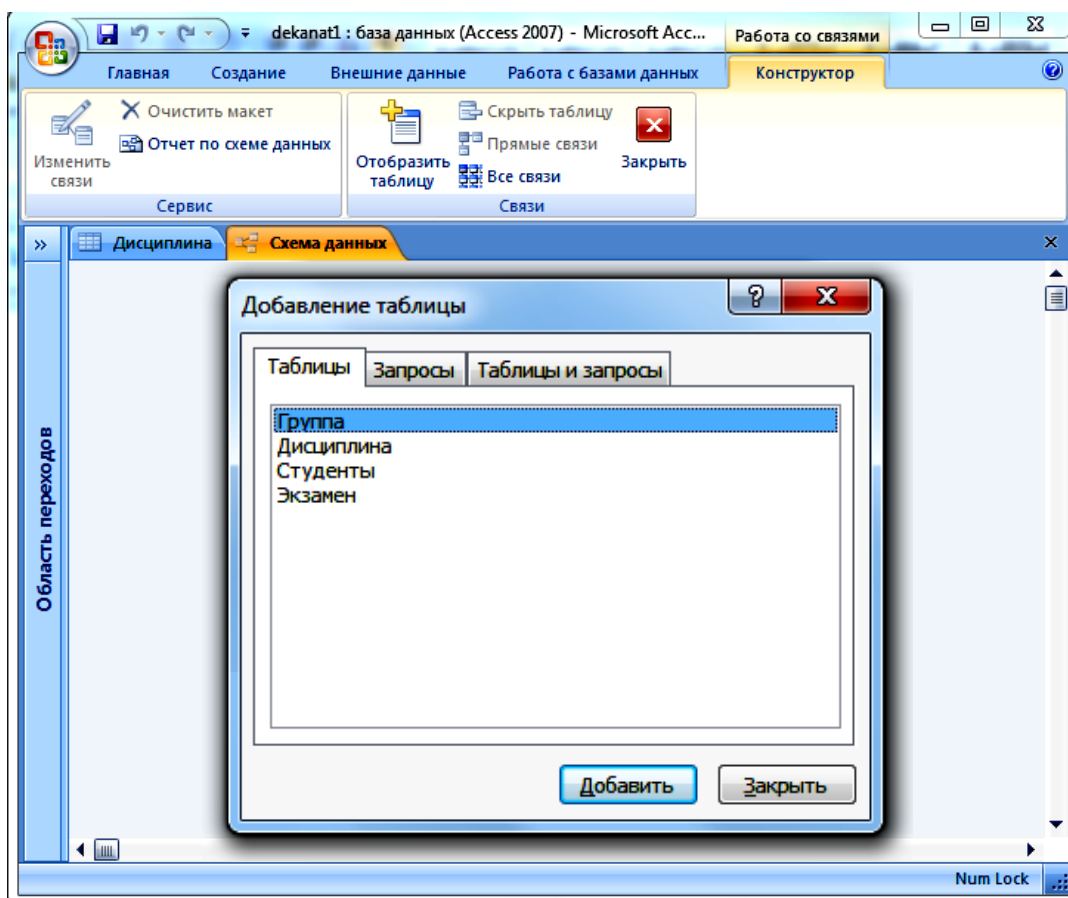


Рис. 3.17. Окно **Схема данных**

Установку связей между таблицами выполняют с помощью мыши переносом ключевых полей из списка полей одной таблицы к соответствующему полю другой таблицы. Поля первичного ключа в списке полей выделяются полужирным начертанием. При этом поле, которое переносится, принадлежит главной таблице, а таблица, куда переносится поле, является подчиненной.

После этого появится диалоговое окно **Изменение Связей** (рис. 3.18).

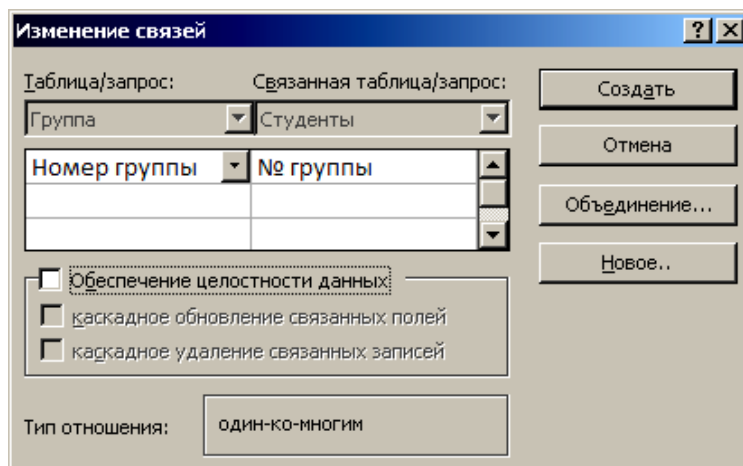


Рис. 3.18. Окно **Изменение связей**

Выберите требуемые параметры связи из предлагаемого набора: **Обеспечение целостности данных**, **Каскадное обновление связанных полей**, **Каскадное удаление связанных записей** и нажмите кнопку «**Создать**». Access установит указанную связь и проведет линию между связанными полями в двух списках полей. На линии будет указан тип отношения (рис. 3.19).

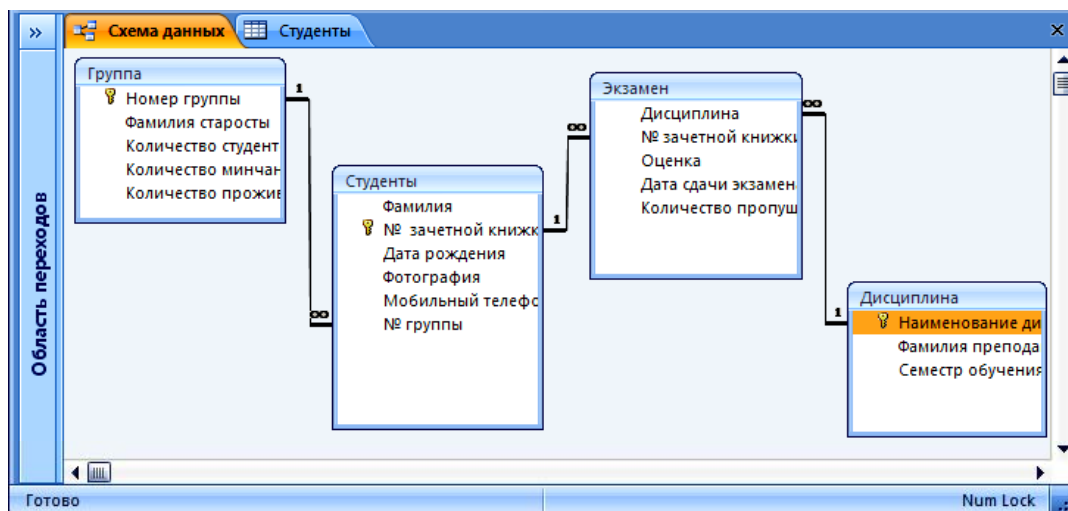


Рис. 3.19. Схема данных базы **dekanat**

Понятие о целостности данных. При создании БД сведения распределяются по многим тематически организованным таблицам, чтобы минимизировать избыточность данных.

Предположим, между таблицами **Студенты** и **Экзамен** существует отношение «один-ко-многим». Одного студента отчислили, и нужно удалить запись об этом студенте из таблицы **Студенты**. Если этот студент сдавал экзамен и у него есть записи в таблице **Экзамен**, эти записи станут непарными после удаления записи в таблице **Студенты**. В таблице **Экзамен** останется № зачетной книжки этого студента, но он будет недействителен, поскольку запись, на которую он ссылается, уже не существует.

Задача сохранения целостности данных состоит в недопущении непарных записей и поддержании ссылок в синхронизированном состоянии, чтобы описанная выше ситуация никогда не возникла.

Если при обеспечении целостности данных выбрать параметр **Каскадное обновление связанных полей**, а затем обновить первичный ключ, Access автоматически обновит все поля, ссылающиеся на этот первичный ключ.

Если при обеспечении целостности данных выбрать параметр **Каскадное удаление связанных записей**, Access при удалении записи, содержащей первичный ключ, автоматически удалит все записи со ссылкой на этот первичный ключ.

3.7.2. Изменение существующей связи

Для изменения существующей связи служит кнопка **Изменить связи** на вкладке **Конструктор** (рис. 3.17). Предварительно необходимо указать на линию связи, которую следует изменить. На экране появится диалоговое окно **Изменение связей** (рис. 3.18). После установки нужных параметров связи следует нажать кнопку **«Создать»**.

Для удаления связи необходимо выделить линию связи и нажать клавишу **«Del»**.

Для удаления таблицы необходимо выделить таблицу и активизировать команду **Скрыть таблицу** (рис. 3.17) или нажать клавишу **«Del»**.

3.8. Практическая работа № 5 Ввод, редактирование и сохранение данных в таблице

Цель работы: научиться вводить и редактировать информацию в СУБД Access 2007 различными способами.

СУБД Access представляет собой набор объектов — таблиц, форм, отчетов, запросов и т. д., — которые для правильной работы БД должны взаимодействовать между собой. Кроме того, эти объекты должны подчиняться определенному набору принципов разработки, иначе БД будет работать плохо или не будет работать вовсе. В свою очередь, от этих принципов зависят способы ввода данных.

В Access существуют следующие способы ввода данных:

- добавление записей непосредственно в таблицу в режиме **Таблицы**. В прежних версиях Access нужно было разработать и создать хотя бы одну таблицу, только после этого можно было вводить данные. Необходимо было решить, из каких полей будет состоять таблица, и задать тип данных для каждого поля. В Access 2007 можно открыть пустую таблицу и сразу начать вводить данные. Тип данных для полей будет выбран автоматически исходя из того, что введено в поле;

- добавление записей с использованием формы. С помощью форм ввода данных можно сделать процесс ввода более простым, быстрым и точным. Также формы создаются, чтобы скрыть некоторые поля в таблице, упростить работу с БД и обеспечить точность ввода данных пользователями;

- изменение элементов в поле подстановок. Поле подстановок представляет собой список данных, в котором пользователи могут выбирать один или несколько элементов.

3.8.1. Добавление записей непосредственно в таблицу в режиме Таблицы

Для заполнения таблицы данными необходимо открыть таблицу в режиме **Таблицы**: имена полей будут записаны в качестве наименования столбцов. Порядок заполнения таблиц зависит от связей в таблицах. Вначале заполняются таблицы на стороне «один», затем — на стороне «многие».

Если между таблицами определены связи, то можно просматривать и вводить данные одновременно в несколько таблиц. Для раскрытия связанной записи надо щелкнуть по значку развертывания записи (+).

После окончания ввода данных таблица закрывается кнопкой закрытия окна. Данные сохраняются автоматически.

Редактирование данных осуществляется обычными средствами редактирования.

Access автоматически следит за обеспечением целостности данных. Запись в подчиненной таблице будет сохранена лишь в том случае, если введенное значение связующего поля присутствует в главной таблице.

3.8.2. Добавление записей с использованием формы

Структура формы определяет, как именно эта форма может использоваться для изменения данных. Форма может содержать любое количество элементов управления — списков, полей, кнопок и таблиц, которые выглядят как листы Excel. Каждый элемент управления в форме может считывать данные из поля базовой таблицы или записывать данные в него. Поведение каждого конкретного элемента управления зависит от типа данных, заданных для поля базовой таблицы, свойств этого поля, и, возможно, от нескольких свойств, которые разработчик БД задал для этого элемента управления. Работа с формами будет рассмотрена в п. 3.11.

3.8.3. Изменение элементов в поле подстановок

Можно создавать два типа списков подстановок:

- списки значений — содержат постоянный набор значений, которые вводятся вручную. Эти значения хранятся в свойстве **Источник строк** поля подстановок;
- списки подстановок — используют запрос для извлечения значений из другой таблицы. Свойство **Источник строк** такого поля вместо постоянного списка значений содержит запрос.

По умолчанию данные подстановки отображаются в поле со списком, хотя для этого можно выбрать элемент управления **Список**. Поле со списком открывается для отображения списка и закрывается после того, как сделан выбор. Элемент управления **Список**, напротив, остается открытым все время.

Поле подстановок можно создавать в форме и в таблице. Рассмотрим на примере, как можно создать поле подстановок в таблице.

В рассмотренном выше примере БД **dekanat** имеются следующие таблицы: **Студенты** с информацией о студентах и **Дисциплина** с информацией о дисциплинах. Данными этих таблиц можно воспользоваться при заполнении полей таблицы **Экзамен**.

Для этого откроем таблицу **Экзамен** в режиме **Конструктор**.

В верхней части конструктора в области создания полей для поля **Дисциплина** необходимо выбрать тип данных «Текстовый».

В области **Свойства поля** выбрать вкладку **Подстановка** (рис. 3.20) и установить значения свойств, приведенные в табл. 3.4.

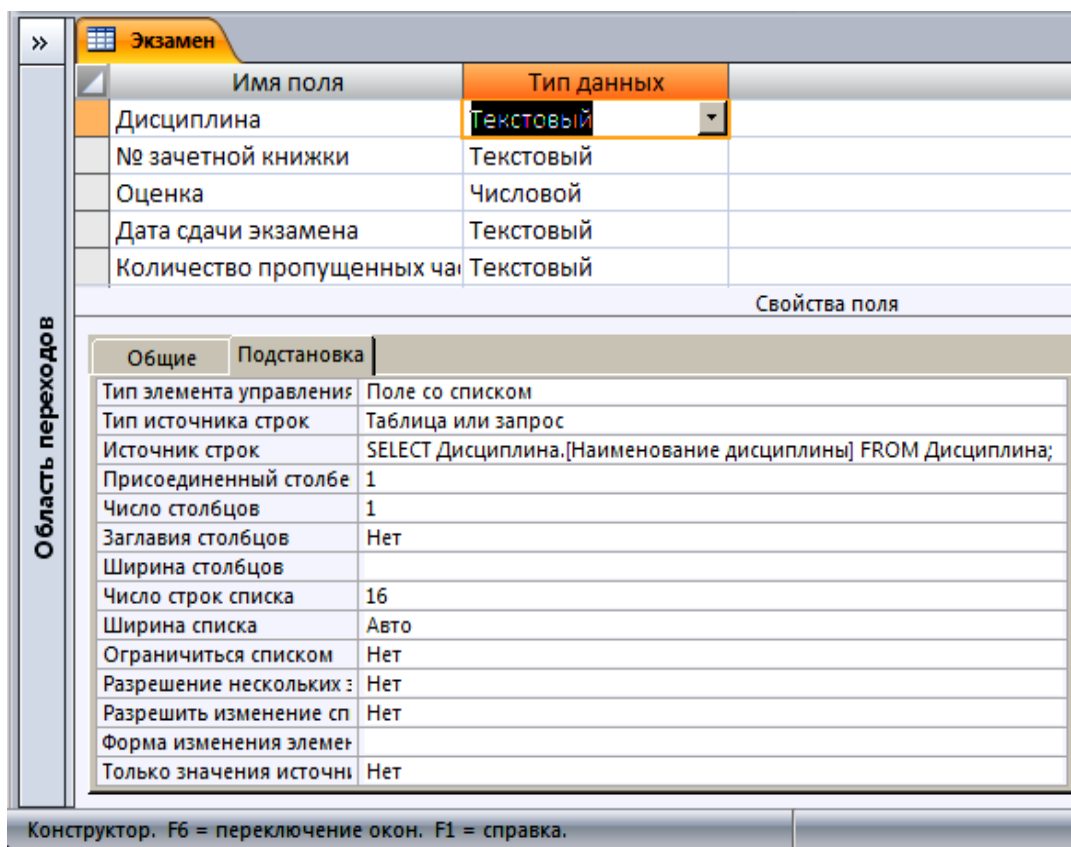



Рис. 3.20. Окно создания поля со списком

Аналогичные действия необходимо выполнить и для поля **№ зачетной книжки**, используя в качестве строк соответственно таблицу **Студенты**.

Таблица 3.4

Значения свойств для поля подстановок

| Свойства | Значение |
|-------------------------|---|
| Тип элемента управления | Поле со списком |
| Тип источника строк | Таблица или запрос |
| Источник строк | <ul style="list-style-type: none"> щелкнуть по кнопке ; добавить таблицу Дисциплина; закрыть окно Добавление таблицы; перетащить с помощью мыши поле Наименование дисциплины в область бланка запроса; в поле Вывод на экран установить <input checked="" type="checkbox"/> |

Для заполнения поля **Оценка** таблицы **Экзамен** создадим поле **Список** (рис. 3.21).

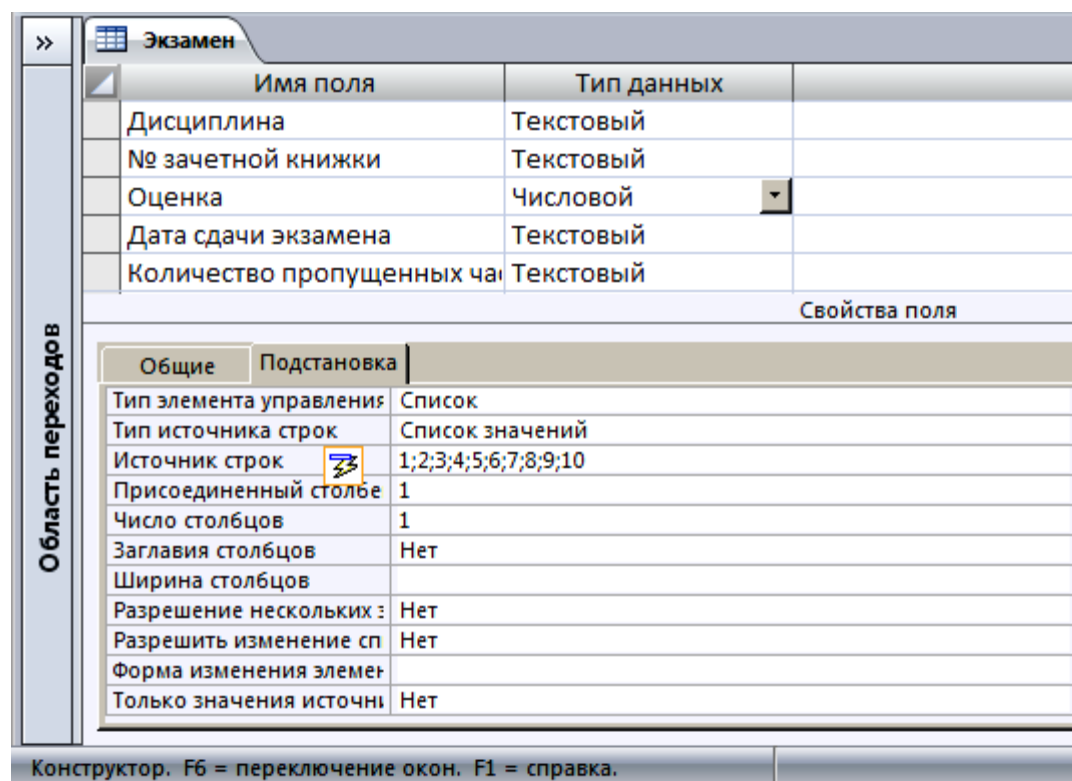


Рис. 3.21. Окно создания поля **Список**

Для используемого списка значений изменим свойства полей так, как показано на рис. 3.21. В свойстве **Источник строк** элементы списка должны отделяться друг от друга точкой с запятой.

3.9. Практическая работа № 6 **Работа с таблицами.** **Поиск информации в базе данных**

Цель работы: изучить основные средства и способы работы с таблицами; научиться производить поиск и замену, сортировку и фильтрацию данных.

Формирование таблиц и заполнение их данными — это основной этап в создании БД. Хотя этот процесс необходим и достаточен с точки зрения хранения данных, он отнюдь не достаточен с точки зрения доступа и использования информации. А ведь информация

для того и хранится в БД, чтобы ее можно было быстро извлечь и при необходимости обработать соответствующим образом. В больших по размеру таблицах, а для современных БД это совершенно естественно, простым просмотром нелегко отыскать какие-то конкретные данные. А это бывает нужно, например, для их замены или исправления.

В Access 2007 имеется целый ряд интерактивных средств работы с таблицами: диалоговое окно поиска и замены, разнообразные фильтры, средства сортировки записей. Все эти операции можно выполнить как в режиме **Таблицы**, так и в режиме **Формы**.

3.9.1. Поиск и замена данных

С помощью диалогового окна **Поиск и Замена** (рис. 3.22) можно найти конкретные записи или определенные значения в полях, а также произвести их замену на другие значения.

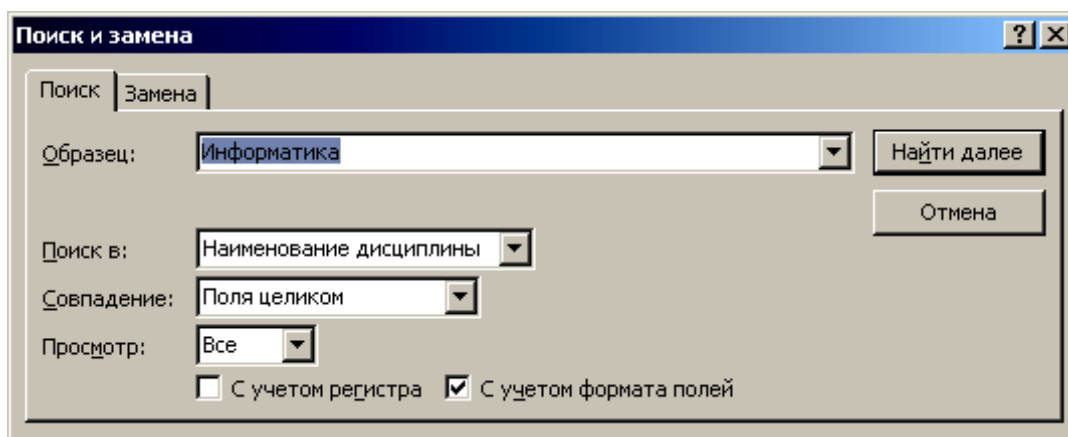


Рис. 3.22. Окно **Поиск и Замена**

Доступ к окну **Поиск и замена** можно получить из контекстного меню поля таблицы, открытой в режиме **Таблицы** (команда **Найти**), или из группы **Найти** (рис. 3.23) вкладки **Главная** (команды **Найти** и **Заменить**).

Диалоговое окно **Поиск и замена** имеет две вкладки: **Поиск** и **Замена**. Внешне они почти не отличаются. На вкладке **Замена** есть дополнительное поле **Заменить на**, в котором указываются данные, заменяющие искомые, и еще две кнопки **Заменить** и **Заменить все**. Поэтому вкладку **Поиск** используют, например,

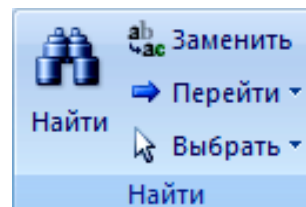


Рис. 3.23. Группа **Найти**

когда нужно заменить данные в одном или нескольких полях (ошибки или опечатки), при этом сами исправления делаются вручную. Вкладка **Замена** автоматизирует весь процесс: с помощью кнопки **Заменить** исправляется конкретное найденное значение, а с помощью кнопки **Заменить все** — сразу все значения в поле или таблице.

При поиске нужной информации команда **Найти** может быть использована как для таблицы, открытой в режиме **Таблицы**, так и для работы с формой, созданной для этой таблицы.

Существует, между тем, одно «но» — выполняя поиск, команда отображает только первое значение, отвечающее условиям поиска. При наличии в таблице множества таких значений поиск придется производить несколько раз. Для нахождения и отображения группы записей, в которых содержатся искомые данные, целесообразнее использовать фильтры.

3.9.2. Сортировка и фильтрация данных

Сортировка позволяет изменить порядок следования записей в таблице и отображать их с учетом значений некоторого определенного поля, т. е. выстроить все записи таблицы в порядке следования значений в этом поле. Упорядоченные записи легче просматривать и обрабатывать. Действительно, вряд ли кто станет отрицать, что со списком фамилий (студентов) гораздо удобнее работать, если он составлен в алфавитном порядке. Аналогично численные значения легче воспринимаются, если они перечислены по порядку. В программе Access есть широкие возможности автоматической сортировки данных различных типов.

Сортировка может быть простой или более сложной, когда записи упорядочиваются не по одному, а по нескольким полям, например, сначала по полю **Фамилия**, а затем — **Дата рождения**.

Для сортировки и фильтрации используется группа инструментов **Сортировка и фильтр** (рис. 3.24), расположенная на вкладке **Главная**.

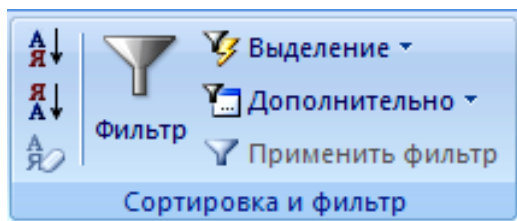


Рис. 3.24. Группа
Сортировка и фильтр

Три кнопки, расположенные в левой части группы, обеспечивают стандартные условия сортировки: по возрастанию, по убыванию и отмена сортировки.

Сортировка может применяться к разным столбцам. Признак сортировки — стрелочка, которая расположена справа от названия поля.

Можно сказать, что фильтры временно «удаляют» некоторый набор записей, удовлетворяющих определенному условию на значения полей. Их можно применять перед проведением поиска, чтобы «уменьшить» количество просматриваемых записей и таким образом ускорить работу. Более того, правильно подобранные фильтры могут полностью решить задачу поиска конкретных данных.

Фильтр используется как инструмент работы с данными, их просмотра, проверки и редактирования. Фильтр — это фактически набор условий для отбора подмножества данных (или для сортировки данных). Фильтр иногда образно называют «одноразовым» или «мимолетным» запросом, т. к. его можно наложить на таблицу (запрос) и удалить нажатием всего лишь одной кнопки. Разные фильтры можно «наслаивать» друг на друга, постепенно приближаясь к желаемому результату. У фильтра нет имени (но в программе Access созданный фильтр можно сохранить в виде запроса).

В Access можно применять фильтры различных типов.

- *Простой фильтр* (фильтр по значению поля) применяется непосредственно из заголовка поля в режиме **Таблицы**. В заголовках полей таблиц имеется маленькая кнопка со стрелкой, направленной вниз. Именно она и открывает панель фильтра для данного поля.

- *Фильтр по выделенному* (кнопка **Выделение** на рис. 3.24) «реагирует» на содержание всего поля (того, в котором установлен курсор ввода) или на выделенный в нем фрагмент данных. Кнопка **Выделение** открывает меню команд для применения фильтра, а контекстное меню просто содержит эти команды. Фильтр по выделенному удобно применять несколько раз подряд, последовательно приближаясь к искомому значению поля (или полей), а также в комбинации с другими фильтрами. Например, чтобы вывести все фамилии, начинающиеся на букву Д, надо выделить букву Д и затем воспользоваться кнопкой **Выделение**, в отобразившемся меню выбрать **Начинается с Д**.

• *Обычный фильтр* (фильтр по форме) может содержать сразу несколько условий по нескольким полям таблицы. Чтобы их задать, необходимо заполнить ячейки в специальном окне-форме фильтра. Поэтому этот тип фильтра еще называют «фильтром по форме». Условия отбора выбираются либо из списка всех возможных значений полей, либо вводятся в виде логических выражений (рис. 3.25).

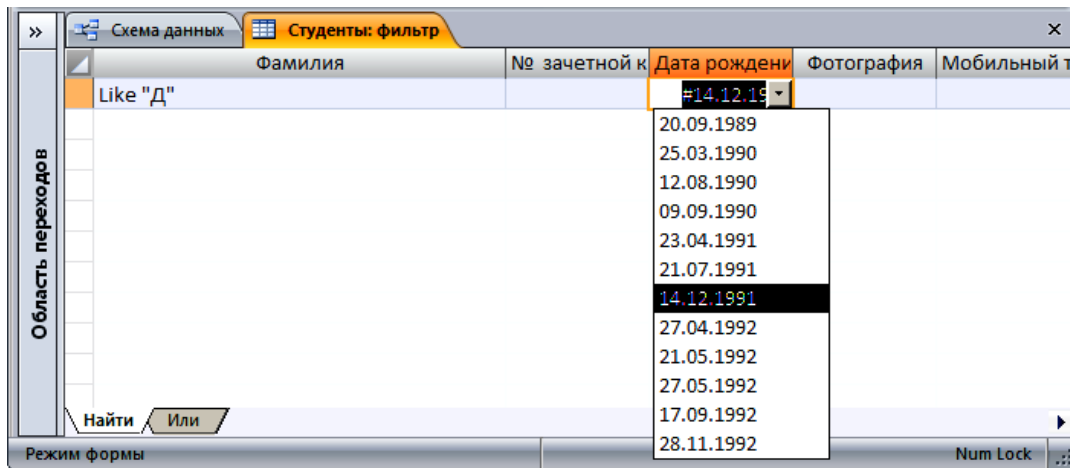


Рис. 3.25. Окно формы обычного фильтра

Чтобы открыть окно обычного фильтра, необходимо выбрать кнопку **Дополнительно** в группе **Сортировка и фильтр** вкладки **Главная**, а затем в раскрывшемся меню выбрать команду **Изменить фильтр**.

Окно обычного фильтра позволяет объединять условия отбора с помощью логических операторов **And** (**И**) или **Or** (**ИЛИ**). Сама форма фильтра содержит несколько вкладок, ярлыки которых расположены внизу (рис. 3.25).

Если на одной вкладке фильтра значения (или выражения) заданы в разных ячейках, но в одной записи, то это аналогично оператору **And** (**И**). Это означает, что будут возвращены только записи, отвечающие условиям отбора, указанным во всех ячейках. Если выражения находятся в разных вкладках, это аналогично оператору **Or** (**ИЛИ**), т. е. означает, что будут возвращены записи, отвечающие условиям отбора, заданным на любой из вкладок.

Чтобы закрыть форму фильтра и применить его к таблице, необходимо выбрать кнопку **Применить фильтр** в группе **Сортировка и фильтр** или воспользоваться контекстным меню формы фильтра, где есть одноименная команда. Результат применения фильтра, описанного выше, показан на рис. 3.26. Обратите внима-

ние на то, что в заголовках двух полей появились значки фильтров, которые обозначают поля, по которым проведен отбор записей.

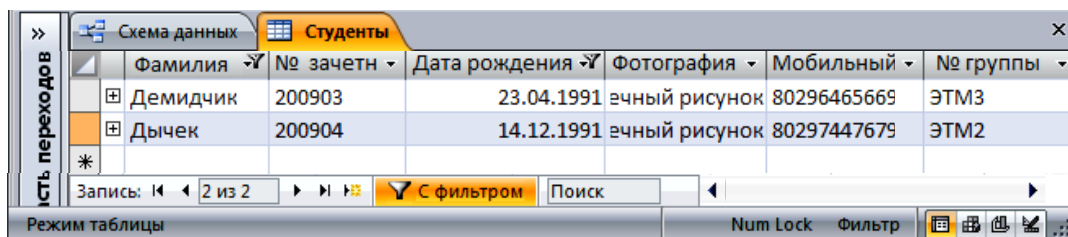


Рис. 3.26. Результат применения фильтра по форме

• *Расширенный фильтр* имеет окно, очень похожее на окно запроса (рис. 3.27). И открывается оно на собственной вкладке в рабочей области программы.

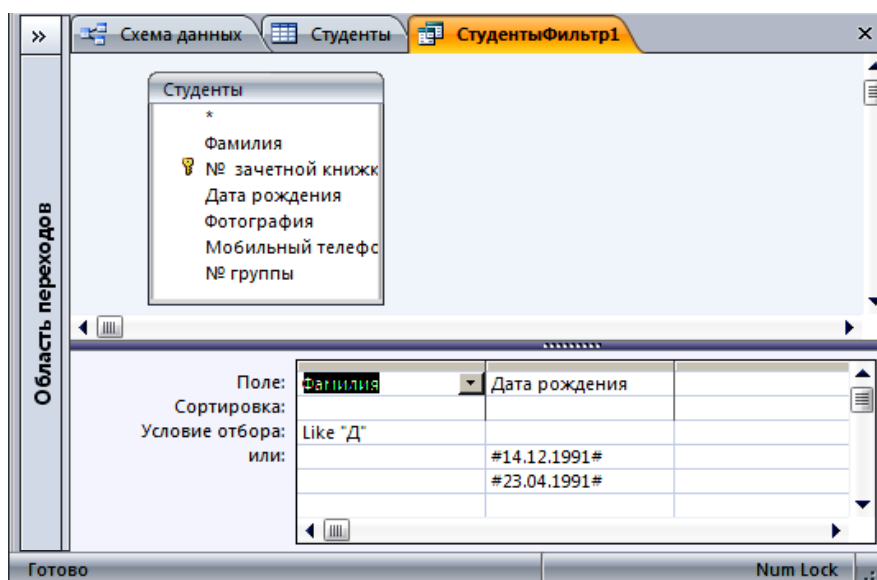


Рис. 3.27. Окно расширенного фильтра

Чтобы открыть окно расширенного фильтра, необходимо выбрать кнопку **Дополнительно** в группе **Сортировка и фильтр**, а затем в открывшемся меню выполнить команду **Расширенный фильтр** (при этом таблица должна быть открыта в режиме **Таблицы** и активна).

В верхней части окна расширенного фильтра (рис. 3.27) находится макет таблицы (со списком полей), для которой он строится (в отличие от запроса, в это окно нельзя добавлять другие таблицы или запросы). В нижней части расположены столбцы условий фильтрации, которые строятся на основе полей таблицы. Любое

поле таблицы из верхней части окна можно просто перетащить указателем мыши в свободный столбец фильтра (если перетащить из списка знак звездочки (*), то в фильтр будут перемещены все поля таблицы). В каждом столбце условий указываются:

- поле таблицы;
- сортировка (выбирается из списка: по возрастанию, по убыванию, отсутствует);
- условия отбора;
- дополнительные условия отбора (при необходимости), связанные с первой строкой условий отбора логическим оператором ИЛИ.

Новое, еще не рассмотренное, свойство, присущее только этому типу фильтров – сортировка. Выражения для условий отбора данного фильтра могут быть созданы также с помощью **Построителя выражений** (см. пп. 3.10.2), окно которого открывается с помощью команды **Построить** контекстного меню ячейки условий.

Сохранить расширенный фильтр в виде запроса можно с помощью команды контекстного меню или меню кнопки **Дополнительно** (рис. 3.24). В этом меню есть также команда **Загрузить из запроса**, с помощью которой можно вернуться к сохраненному в виде запроса расширенному фильтру.

Запустить расширенный фильтр можно или из контекстного меню верхней части окна фильтра или кнопкой **Применить фильтр** (она расположена ниже кнопки **Дополнительно**) в группе инструментов **Сортировка и фильтр** (рис. 3.24).

3.10. Практическая работа № 7 Создание запросов в СУБД Access 2007

Цель работы: изучить назначение и виды запросов; научиться создавать различные виды запросов в режиме **Мастер запросов** и в режиме **Конструктор запросов**; научиться создавать условия отбора для различных типов данных; изучить возможности **Построителя выражений**.

Запросы обеспечивают быстрый и эффективный доступ к хранящимся в таблицах данным. Они также используются для обработки информации, хранящейся в таблицах: сортировки данных, проведения вычислений с ними, создания таблиц с выборочными данными, подведения итоговых результатов и т. п.

Access позволяет выполнять следующие виды запросов:

- запрос на выборку данных;
- запрос с условием;
- запрос с вычисляемыми полями;
- итоговый запрос;
- параметрический запрос;
- перекрестный запрос;
- запрос на изменение записей.

Запросы можно создавать в режиме **Мастер запросов** и в режиме **Конструктор запросов**. При этом в режиме **Конструктор запросов** запрос создается полностью самим пользователем на основе специальной таблицы **QBE** (Query by Example – запрос по образцу). С помощью **Мастера запросов** можно создать запрос путем выбора предлагаемого готового варианта и по наводящим подсказкам.

3.10.1. Создание запроса с помощью Мастера

Запрос на выборку данных. Данный запрос выбирает для просмотра только необходимые поля таблицы. Его можно создавать как в режиме **Конструктор запросов**, так и с помощью **Мастера запросов**.

Рассмотрим создание этого запроса на примере. У нас имеется таблица **Студенты** (рис. 3.28).

| Фамилия | № зачетной | Дата рожде | Мобильный | Фотография | № группы |
|------------|------------|------------|-------------|--------------|----------|
| Борис | 200901 | 17.09.1992 | 80294445689 | Bitmap Image | ЛХ1 |
| Войнич | 200902 | 25.03.1990 | 80293745684 | Bitmap Image | ЛХ2 |
| Демидчик | 200903 | 23.04.1991 | 80296465669 | Bitmap Image | СПС3 |
| Дычек | 200904 | 14.12.1991 | 80297447679 | Bitmap Image | ЛХ2 |
| Ждан | 200905 | 28.11.1992 | 80295455659 | Bitmap Image | ЛХ2 |
| Иваненко | 200906 | 20.09.1989 | 80295445589 | Bitmap Image | СПС3 |
| Кашлей | 200907 | 12.08.1990 | 80296465669 | Bitmap Image | ЛХ1 |
| Климович | 200908 | 27.05.1992 | 80296443629 | Bitmap Image | ЛХ2 |
| Кондрашова | 200909 | 21.07.1991 | 80291441681 | Bitmap Image | ЛХ1 |
| Кужаль | 200912 | 09.09.1990 | 80291441619 | Bitmap Image | ЛХ1 |
| Лазаревич | 200913 | 27.05.1992 | 80295455685 | | СПС3 |
| Лащ | 200914 | 21.05.1992 | 80296446686 | | СПС3 |
| Лучкина | 200915 | 27.04.1992 | 80294545649 | | СПС3 |

Рис. 3.28. Таблица **Студенты**

Деканату необходимо получить следующие данные: фамилию студента и его мобильный телефон. Для этого выберем на вкладке **Создание** в группе **Другие** кнопку **Мастер запросов** (рис. 3.29).

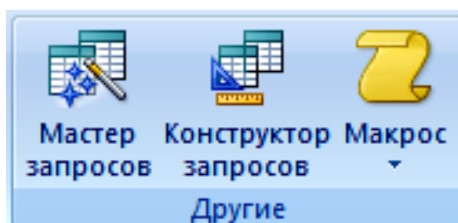


Рис. 3.29. Группа **Другие**

В диалоговом окне **Новый запрос** выберем **Простой запрос** и нажмем кнопку «**ОК**».

В группе **Таблицы и запросы** (рис. 3.30) выберем таблицу, содержащую нужные данные (в нашем случае – **Студенты**).

В группе **Доступные поля** выберем поля **Фамилия** и **Мобильный телефон**. При этом они добавятся в список **Выбранные поля**. После добавления полей нажмем кнопку «**Далее**».

Присвоим запросу имя **Телефоны студентов**, а затем нажмем кнопку «**Готово**».

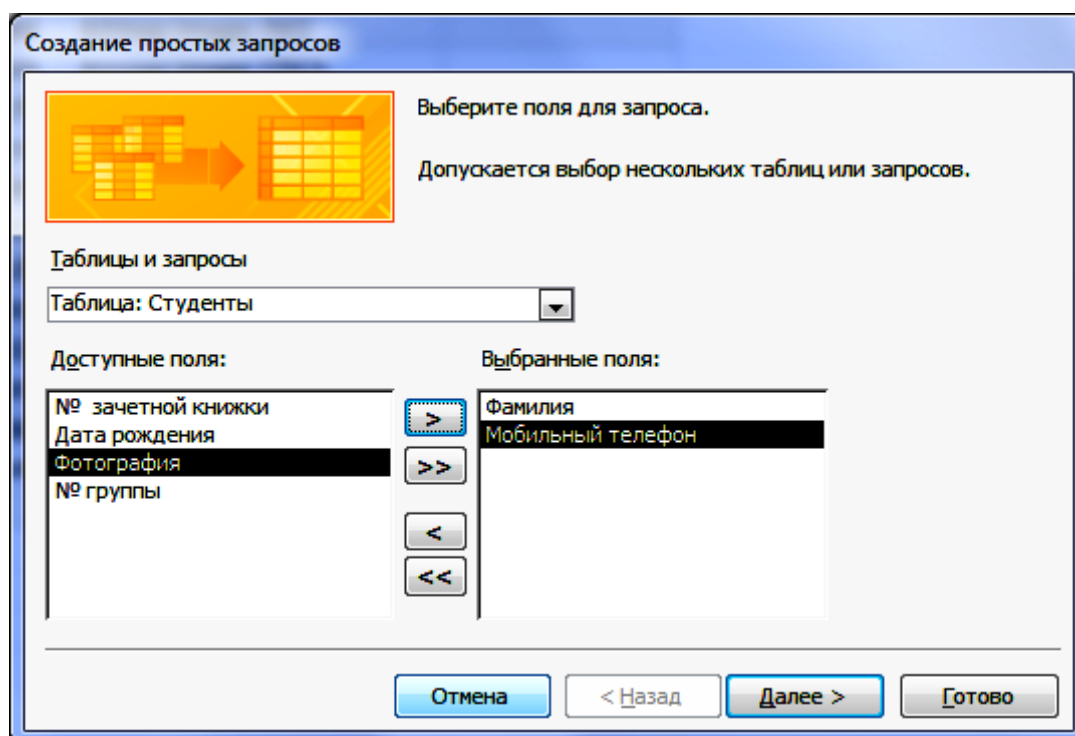
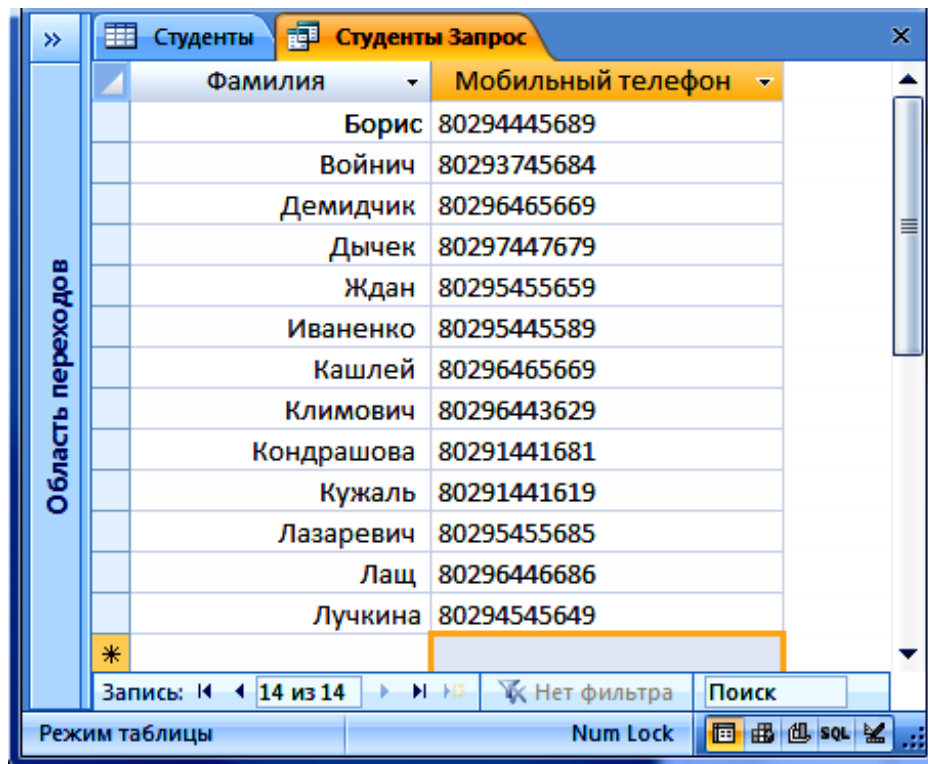


Рис. 3.30. Окно **Создание простых запросов**

В результате мы получим таблицу с двумя полями, указанными в запросе (рис. 3.31).



The screenshot shows a window titled 'Студенты Запрос' (Students Query) displaying the result of a query. The window has a vertical sidebar on the left labeled 'Область переходов' (Navigation Area). The main area contains a table with two columns: 'Фамилия' (Surname) and 'Мобильный телефон' (Mobile phone). The table lists 14 records. At the bottom of the window, there is a status bar with 'Запись: 14 из 14' (Record: 14 of 14), 'Нет фильтра' (No filter), and a search field labeled 'Поиск'. The bottom-most bar shows 'Режим таблицы' (Table mode) and 'Num Lock'.

| Фамилия | Мобильный телефон |
|------------|-------------------|
| Борис | 80294445689 |
| Войнич | 80293745684 |
| Демидчик | 80296465669 |
| Дычек | 80297447679 |
| Ждан | 80295455659 |
| Иваненко | 80295445589 |
| Кашлей | 80296465669 |
| Климович | 80296443629 |
| Кондрашова | 80291441681 |
| Кужаль | 80291441619 |
| Лазаревич | 80295455685 |
| Лащ | 80296446686 |
| Лучкина | 80294545649 |

Рис. 3.31. Результат простого запроса

С помощью **Мастера запросов** можно создавать и другие виды запросов: перекрестный, повторяющиеся записи, записи без подчиненных.

3.10.2. Создание запроса в режиме Конструктора

Запрос на выборку данных. Рассмотрим создание запроса на выборку на примере. Деканату необходимо получить следующие данные: № группы и количество студентов в группе.

Для этого на вкладке **Создание** в группе **Другие** необходимо выбрать **Конструктор запросов** (рис. 3.29). Затем с помощью окна **Добавление таблицы** (рис. 3.32) выбрать таблицу, на основе которой будет создаваться запрос (в нашем случае таблицу **Группа**).

На экране появится диалоговое окно запроса, состоящее из двух частей: верхней, где представлены таблицы, на основе которых строится запрос, и нижней, где выбираются поля из таблиц и где, собственно говоря, непосредственно формируется запрос (рис. 3.33).

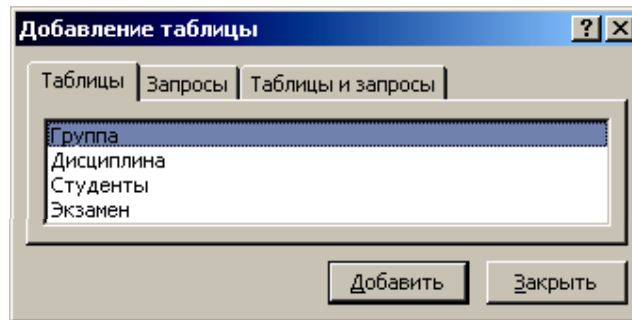


Рис. 3.32. Окно **Добавление таблицы**

Нижняя часть называется бланком запроса по образцу, иначе QBE-бланком или QBE-областью. При этом количество строк в QBE-бланке может меняться в зависимости от вида запроса.

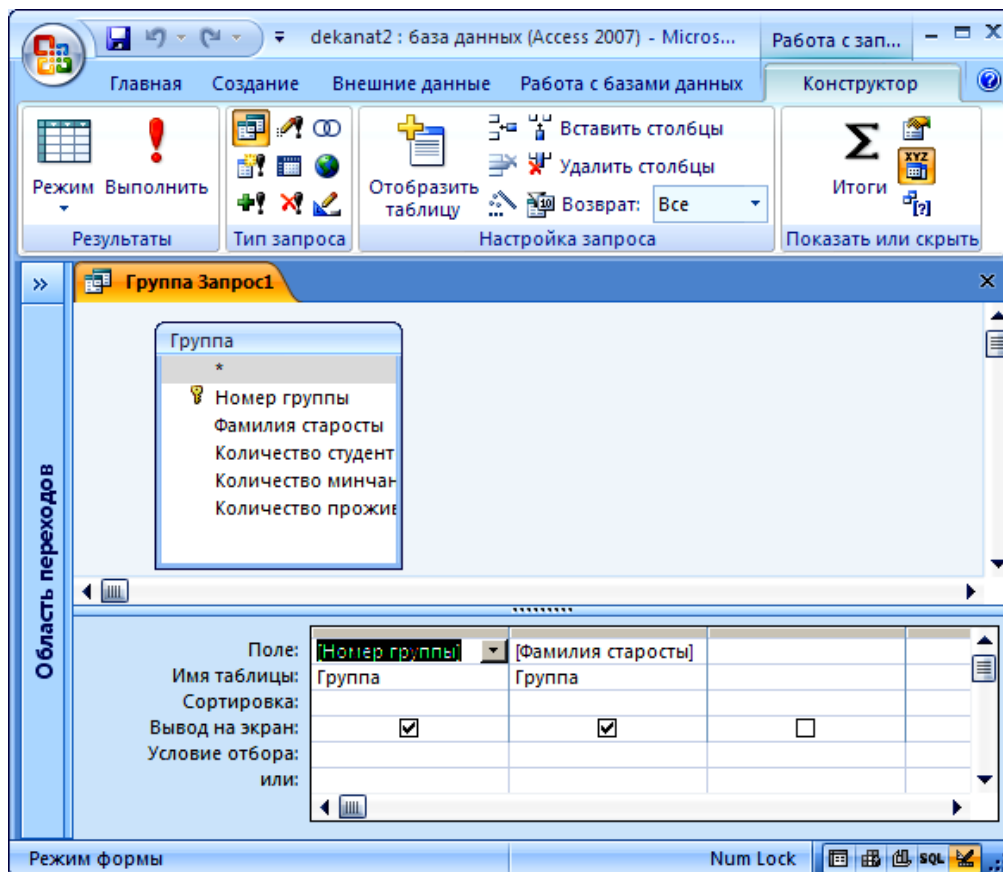


Рис. 3.33. Окно **Бланк запроса**


Бланк запроса содержит строки:

- **Поле** — для запрашиваемых полей;
- **Имя таблицы** — для вывода имени таблицы, из которой выбирается поле;

- **Сортировка** — для проведения сортировки по этим полям;
- **Вывод на экран** — для вывода или нет выбранных запросом полей на экран;
- **Условие отбора** — для ввода условий на выбор записей в соответствии с заданными условиями, причем условия по отдельным полям в этой строке соединяются операцией «И» (все условие верно только, если все составляющие его условия верны);
- **Или** — для ввода условий, которые соединяются с условием в вышерасположенной строке **Условие отбора** по принципу «ИЛИ» (все условие верно, если хотя бы одно из составляющих его условий верно).

Для создания простого запроса необходимо в бланк запроса перенести мышью нужные поля.

После этого активизируется вкладка **Конструктор**, которая содержит группы инструментов **Результаты**, **Тип запроса**, **Настройка запроса**, **Показать или скрыть** (рис. 3.33).

Для выполнения запроса необходимо активизировать кнопку **Выполнить**  в пиктографическом меню. В результате на экране в виде таблицы отобразится выполненный запрос. Сохраните его.

Запрос с условием. Условие запроса — это правило, определяющее, какие записи требуется включить в результаты запроса.

Для подготовки запроса с условием необходимо в строке **Условие отбора** бланка запроса записать выражение, которое состоит из операндов (табл. 3.5), операторов (табл. 3.6–3.9), функций, ссылок на имена полей, позволяющих выбирать необходимую информацию по заданному критерию отбора.

Таблица 3.5

Операнды, используемые в критериях отбора

| Операнды | Описание |
|-------------------------|---|
| Литералы | Конкретные значения: — числа (любые): 5, 20 и т. п.; — текст (заключается в двойные кавычки): "Иванов", "Минск"; — даты (заключаются в символы #): #1.01.03#, #9-Июнь-03# |
| Константы | Неизменяющиеся значения, определенные в Access: True, False, Null, Да, Нет |
| Идентификаторы (ссылки) | Имена полей, таблиц, форм, отчетов и т. д. (заклучаются в квадратные скобки, восклицательный знак используется при указании ссылки на поле в конкретном объекте БД): [Группа]![ФИО] |

**Оператор выбора по шаблону,
используется для полей с типами данных «Текстовый», «Поле МЕМО» и «Гиперссылка»**

| Оператор | Описание | Пример |
|---|--|--|
| Like | Для отбора данных в текстовых полях по шаблону, заключенному в кавычки. Шаблоном может быть слово, по которому будет производиться поиск и отбор записей или набор символов: | Like"С*" выбирает все записи из заданного поля, начинающиеся на букву С |
| | ? любой одиночный символ в данной позиции; | Like"?#[5-8][!1-3]A*" выбирает все записи из заданного поля со значением: |
| | * любое количество символов в данной позиции; | • в первой позиции — произвольный символ; |
| | # любая цифра в данной позиции; | • во второй позиции — произвольная цифра; |
| | [] включает допустимый диапазон символов; | • в третьей позиции — любое число от 5 до 8 включительно; |
| [!] включает недопустимый диапазон символов | • в четвертой позиции — любое число, кроме цифр от 1 до 3 включительно; | |
| | | • в пятой позиции — буква А, после — произвольные символы в любом количестве |

Логические операторы

| Оператор | Описание | Пример |
|---------------------|--|---|
| =, >, <, >=, <=, <> | Равно, больше, меньше, больше или равно, меньше или равно, неравно | >#02.02.2006# выбирает записи, совершенные после 2 февраля 2006 г. |
| And | Логическое И, задает интервал отбора из выражений | >10 And <=20 в числовом поле выбирает записи из интервала [10–20] |
| Or | Логическое ИЛИ, задает альтернативы отбора из выражений | 10 Or 20 Or 30 в числовом поле выбирает записи, равные 10, или 20, или 30 Like"М*" Or Like"К*" в текстовом поле выбирает записи, начинающиеся с букв М или К |
| Not | Логическое НЕ (отрицание) | Not Like"белый" в текстовом поле выбирает все записи, кроме белый |

Таблица 3.8

Операторы условий для полей с типом данных «Дата/Время»

| Оператор | Описание | Пример |
|----------|---|--|
| DateDiff | Определяет интервал времени между двумя датами. Например, для вычисления числа лет между двумя датами или числа дней между сегодняшним днем и концом года. Интервал может принимать значения: | DateDiff("yyyy";[Студенты]![Дата рождения];Date())>18 выводит фамилии студентов старше 18 лет |
| | "yyyy" | определяет число лет; |
| | "m" | определяет число месяцев; |
| | "d" | определяет число дней; |
| | Date() | определяет текущую дату; |
| | Date()-1 | записи об операциях, совершенных за один день до текущей даты; |
| Date()+1 | записи об операциях, совершенных на следующий день после текущей даты | |
| DatePart | Определяет значение, содержащее указанную часть заданной даты. Например, год, месяц или день в текущей дате. Интервал принимает те же значения, что и в функции DateDiff | DatePart("m";[Студенты]![Дата рождения])=5 выводит даты рождения всех студентов, родившихся в мае |

Таблица 3.9

Операторы выбора из диапазона

| Оператор | Описание | Пример |
|----------------------|---|--|
| Between X1 And X2 | Позволяет задать интервал для числового значения от X1 до X2 включительно | Between Date() And Date()-6 выбирает записи об операциях, совершенных в течение последних 7 дней |
| | | Between 10 And 20 в числовом поле выбирает записи из интервала от 10 до 20 включительно |
| In | Позволяет выполнить проверку на равенство любому значению из списка, который задается в круглых скобках. Выбирает записи из полей со значениями | In("Минск";"Омск";"Орша") Минск, или Омск, или Орша |
| | | In(10;20;30) 10, или 20, или 30 |

Рассмотрим пример. Деканату необходимо получить фамилии студентов, не сдавших экзамен (оценка ниже 4) в первом семестре.

Для получения запроса нам понадобятся таблицы: **Студенты** (поле **Фамилия**), **Дисциплина** (поля **Семестр обучения**, **Наименование дисциплины**) и **Экзамен** (поле **Оценка**). Добавим эти поля в бланк запроса. В строке **Условие отбора** для поля **Семестр** необходимо записать "1"; для поля **Оценка** – <4 (рис. 3.34).

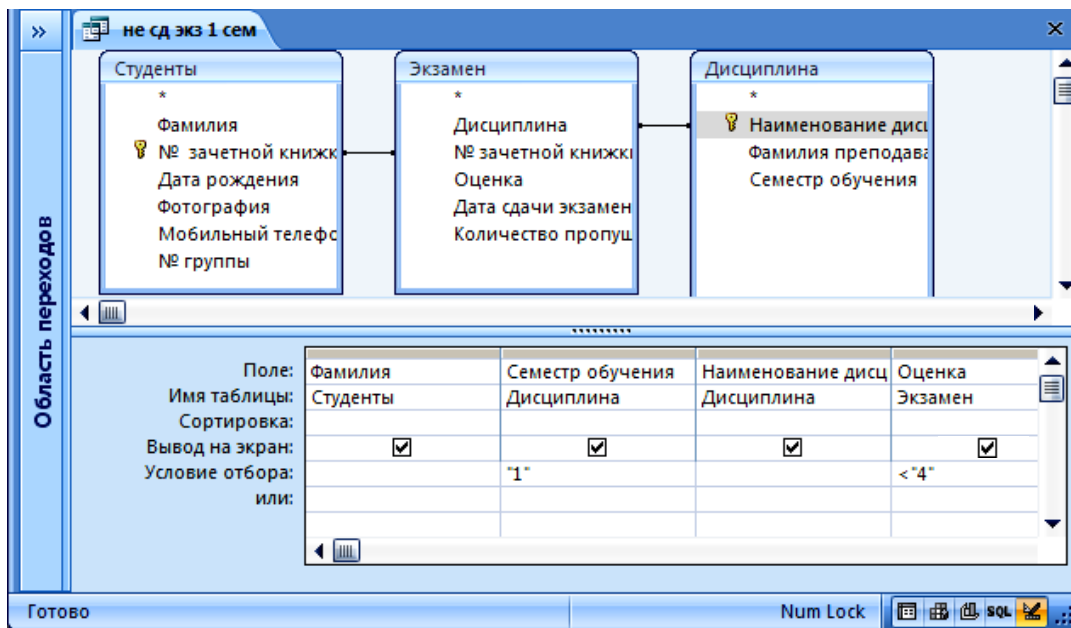


Рис. 3.34. Построение запроса в режиме **Конструктора запросов**

В результате мы получим (рис. 3.35):

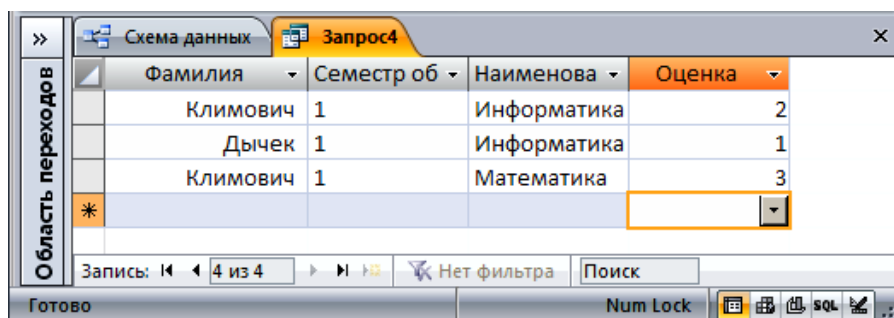



Рис. 3.35. Результат запроса

Если условие налагается на несколько полей и они связаны логическим оператором **И**, то они вводятся в одной строке под нужными полями, если логическим оператором **ИЛИ** – то в разных строках под нужными полями.

Сохраненным запросом можно воспользоваться в любой момент и после внесения изменений и дополнений в исходную таблицу.

Запросы с вычисляемыми полями. В таблицах БД не может быть полей, значения которых являются производными от других полей таблицы, т. к. это нарушает правила нормализации. Для получения таких полей используются запросы, а именно — вычисляемые поля в запросах.

Для построения вычисляемого поля нужно в пустую ячейку строки **Поле бланка запроса** ввести выражение. Выражения могут быть арифметическими, логическими или текстовыми.

Во избежание ошибок ввода для построения выражения (как и для записи условий отбора в предыдущих запросах) лучше использовать **Построитель выражений** (рис. 3.36), который открывается при нажатии на кнопки **Построитель**  вкладки **Конструктор**.

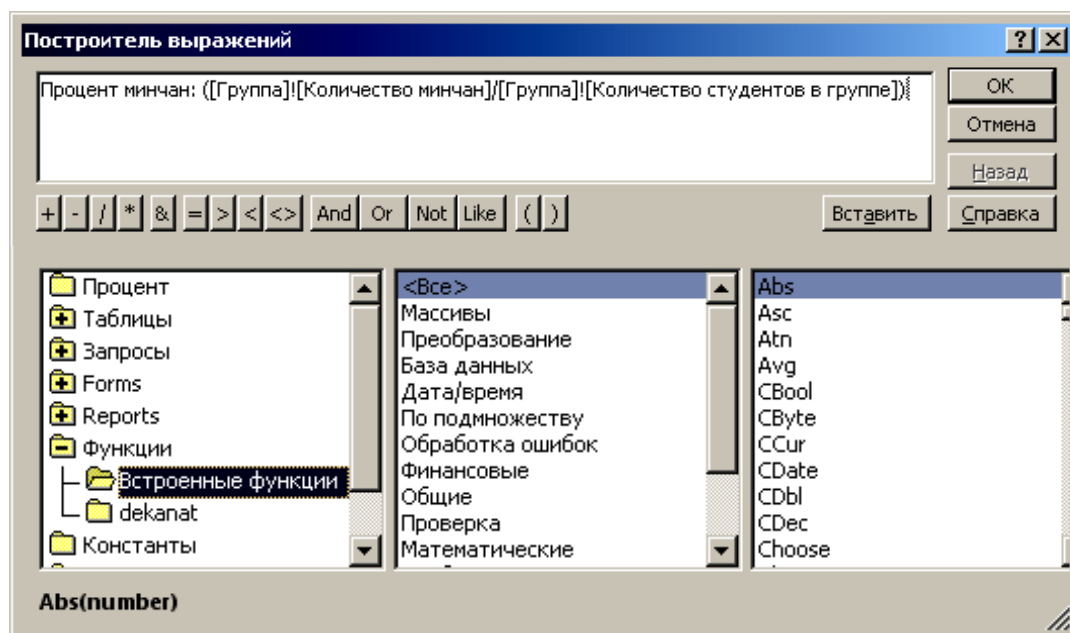


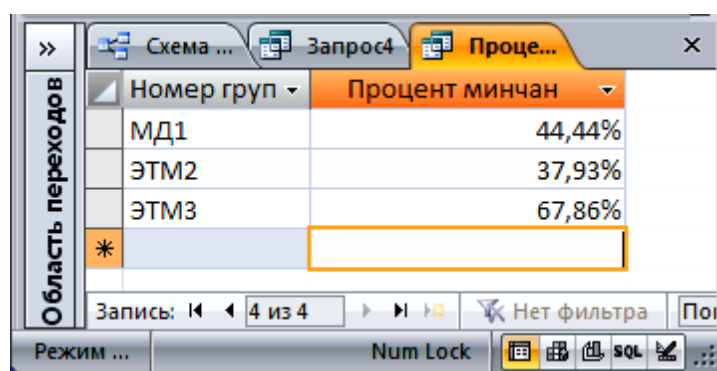
Рис. 3.36. Окно мастера **Построитель выражений**

В верхней части окна **Построителя выражений** расположена область ввода выражения. В нижней — три списка для поиска имен полей и встроенных функций, необходимых для создания выражения.

Для создания выражения нужно выбрать необходимую таблицу, в ней — нужные для расчета поля и произвести между ними вычисления, используя кнопки соответствующих операторов. Имя создаваемого в процессе запроса вычисляемого поля вводится с двоеточием. Это имя появится в качестве заголовка поля.

Создадим запрос, определяющий процент минчан в каждой группе студентов. Для этого из таблицы **Группа** выберем поле **Номер группы** и в свободном поле бланка запроса введем выражение: **Процент минчан:([Группа]![Количество минчан]/[Группа]![Количество студентов в группе])**.

Результат расчета должен быть выведен в процентах (рис. 3.37). Поэтому выберем на вкладке **Конструктор** в группе **Показать или скрыть** кнопку **Страница свойств** и установим в открывшемся **Окне свойств** для данного поля формат «Процентный».



| Номер групп | Процент минчан |
|-------------|----------------|
| МД1 | 44,44% |
| ЭТМ2 | 37,93% |
| ЭТМ3 | 67,86% |
| * | |

Рис. 3.37. Запрос с вычисляемым полем

Запрос с параметром. В условиях отбора бланка запроса мы вводим конкретные значения (константы). Но иногда условия отбора необходимо изменять. Для этого используется параметр запроса, который делает поле переменной величиной. При каждом выполнении запроса значение параметра будет запрашиваться.

Запросами с параметром являются запросы, в которых конкретное значение параметра, входящего в условие на выборку, формируется в диалоговом режиме через специальное окно запроса.

Для создания такого запроса в строку **Условие отбора** вводится фраза в квадратных скобках, которая будет выводиться в качестве «подсказки» в процессе диалога, например, [Введите фамилию]. Таких параметров может быть несколько, каждый для своего поля.

Создадим запрос с параметром, чтобы определить, какие оценки получили студенты по конкретному предмету. Для этого на основе таблиц **Студенты** и **Экзамен** создадим запрос, включив в него необходимые поля этих таблиц. Для поля **Дисциплина** в условии отбора установим в квадратных скобках параметр **[Введите наименование дисциплины]** (рис. 3.38).

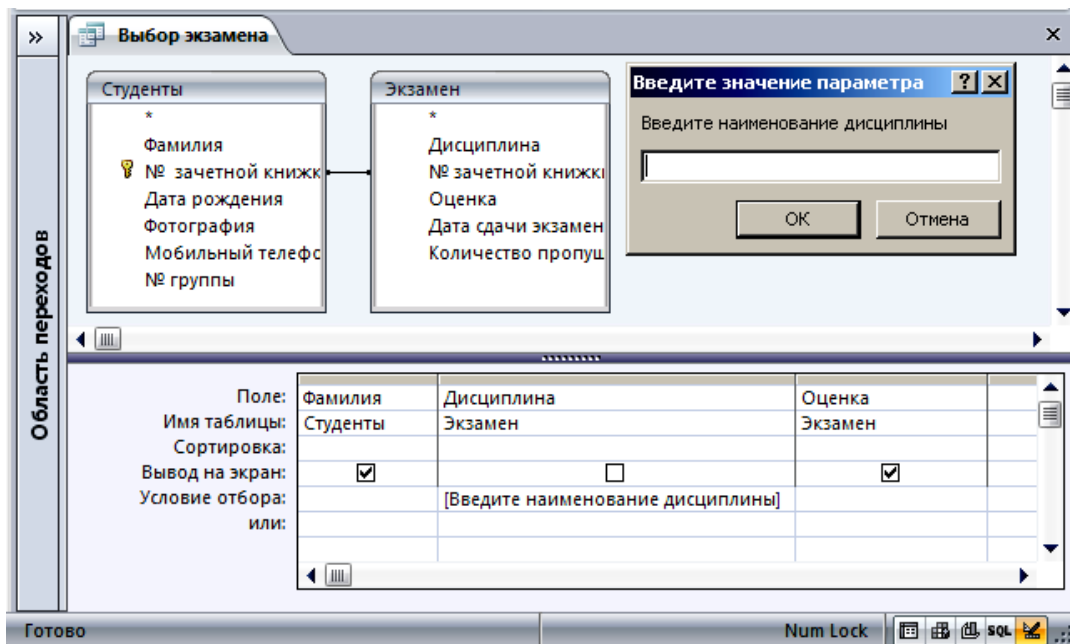


Рис. 3.38. Бланк запроса с параметром

Запустив запрос на выполнение, мы сначала получим диалоговое окно, в котором укажем, какая именно дисциплина нас интересует. Результат представлен на рис. 3.39.

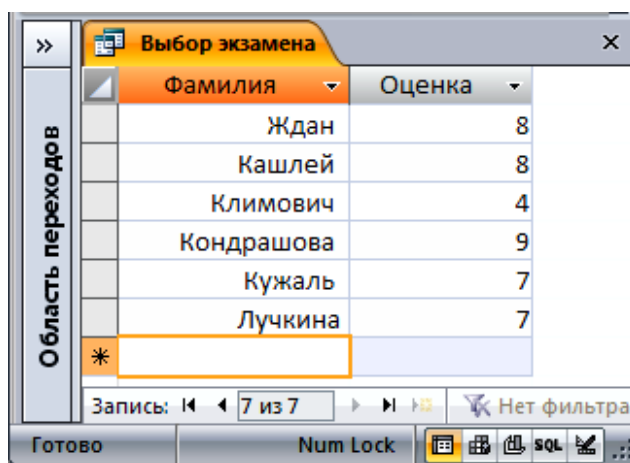


Рис. 3.39. Результат запроса с параметром

Итоговые запросы. Итоговые запросы используются в том случае, если необходимо сгруппировать записи, выбранные согласно заданным условиям, по совпадающим значениям поля, а по несовпадающим значениям вычислить итоговые значения. В таких запросах используются два типа полей: по одним полям осуществляется группировка данных, по другим — вычисления.

Для выполнения вычислений в итоговых запросах используются следующие функции (табл. 3.10):

Таблица 3.10

Таблица функций

| Функция | Описание |
|---------|---|
| Sum | Вычисляет сумму всех значений заданного поля в каждой группе |
| Avg | Вычисляет среднее арифметическое всех значений заданного поля в каждой группе |
| Min | Возвращает наименьшее значение, найденное в заданном поле внутри каждой группы |
| Max | Возвращает наибольшее значение, найденное в заданном поле внутри каждой группы |
| Count | Возвращает число записей, найденное в заданном поле внутри каждой группы, отличное от Null (пустого значения) |
| First | Возвращает первое значение, найденное в заданном поле внутри каждой группы |
| Last | Возвращает последнее значение, найденное в заданном поле внутри каждой группы |
| Stdev | Возвращает среднееквадратичное отклонение от среднего значения поля в группе |
| Var | Возвращает дисперсию значений поля в группе |

Создадим запрос, определяющий средний балл, полученный студентами по разным дисциплинам на факультете (рис. 3.40).

Для этого из таблицы **Экзамен** выберем поля **Дисциплина** и **Оценка** и затем на вкладке **Конструктор** в группе **Показать или скрыть** нажмем кнопку **Итоги**. При этом в бланке запроса появится строка **Групповая операция**, и в этой строке будет выведена установка **Группировка для каждого поля, внесенного в бланк**. Для поля **Дисциплина** значение **Группировка** оставим без изменения, для поля **Оценка** из раскрывающегося списка выберем функцию Avg.

В полученной таблице для поля **Оценка** установлены свойства: формат поля — фиксированный, число десятичных знаков — 1.

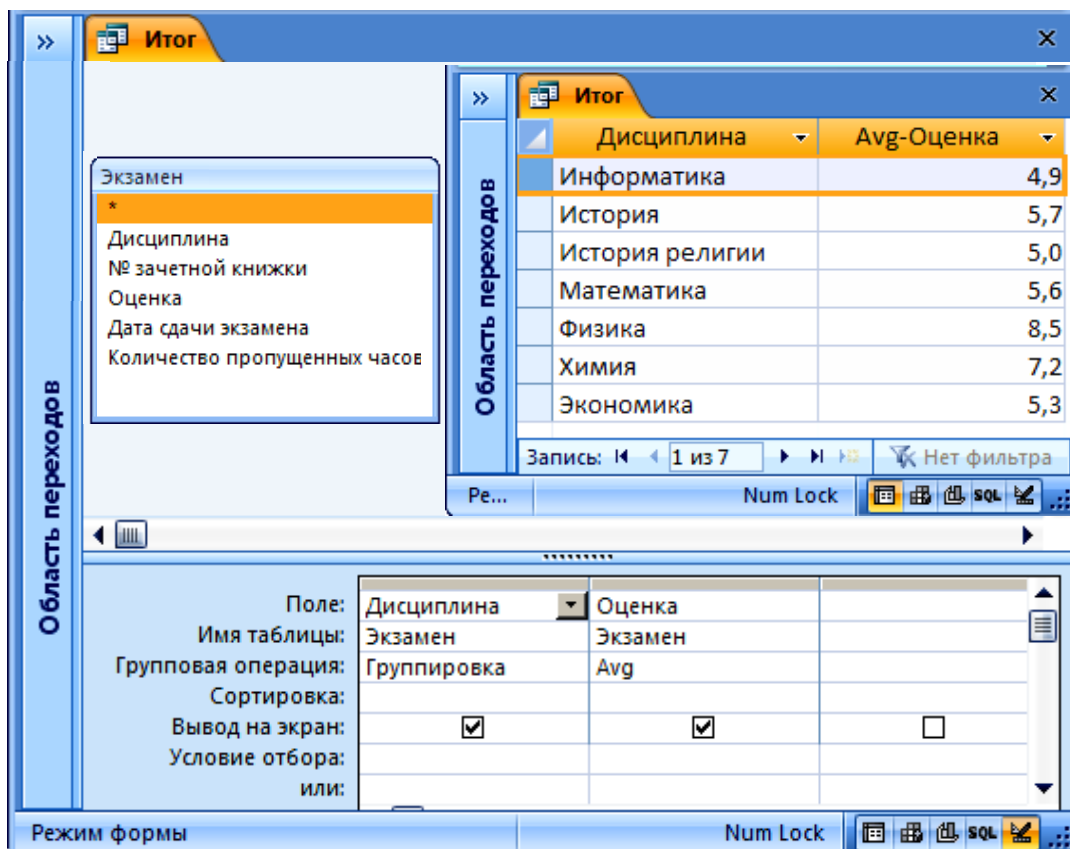


Рис. 3.40. Итоговый запрос

Перекрестные запросы. Перекрестные запросы используются в тех случаях, когда необходимо разгруппировать имеющиеся данные по определенным критериям отбора. В перекрестных запросах, кроме обычных операций группировки выбранных данных, производится такое их расположение в таблице запроса, которое позволяет более компактно и наглядно отображать выбранную из БД информацию. Таблица перекрестного запроса выглядит примерно так же, как стандартная электронная таблица, где обычно все строки и столбцы имеют свои названия или номер, а на их пересечении размещается соответствующее выбранной ячейке значение.

Рассмотрим на примере. Необходимо определить среднюю оценку за экзамен по всем дисциплинам в каждой группе.

Для построения запроса после добавления необходимых таблиц и полей (в нашем случае **№ группы**, **Дисциплина**, **Оценка**) нужно на вкладке **Конструктор** в группе **Тип запроса** нажать кнопку **Перекрестный**, при этом в бланке запроса появится строка **Перекрестная таблица** (рис. 3.41).

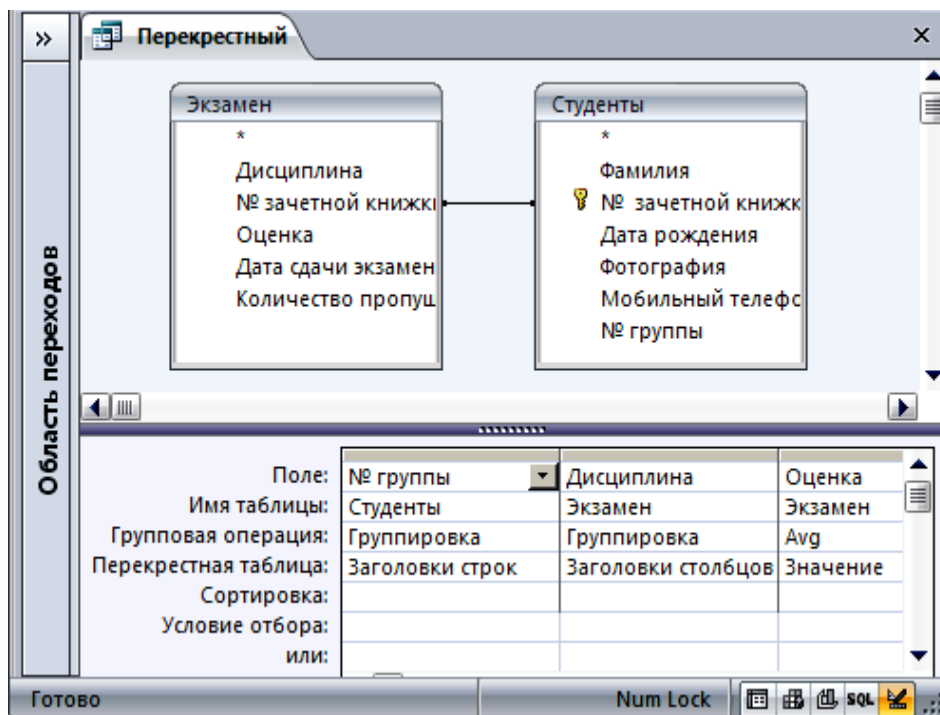


Рис. 3.41. Перекрестный запрос

Для поля **№ группы** в строке **Перекрестная таблица** выбрать значение **Заголовки строк**, а в строке **Групповая операция** — значение **Группировка**.

Для поля **Дисциплина** в строке **Перекрестная таблица** выбрать значение **Заголовки столбцов**, а в строке **Групповая операция** — значение **Группировка**.

Для поля **Оценка** в строке **Перекрестная таблица** выбрать **Значение**, а в строке **Групповая операция** — статистическую функцию **Avg**.

В результате получим информацию следующего вида (рис. 3.42):

| № группы | Информатика | История | Математика | Физика | Химия |
|----------|-------------|---------|------------|--------|-------|
| МД1 | 6,67 | | 6,25 | 8,00 | 8,00 |
| ЭТМ2 | 2,67 | 6,00 | 4,00 | 9,00 | 6,00 |
| ЭТМ3 | 5,50 | 5,00 | 6,00 | | 7,00 |

Рис. 3.42. Результат перекрестного запроса

3.11. Практическая работа № 8 Создание форм в СУБД Access 2007

Цель работы: освоить способы создания форм в разных режимах (**Мастера форм**, **Конструктора форм**, на основе шаблона).

Формы в Access предназначены для отображения в удобном виде на экране монитора данных, хранящихся в таблицах. Фактически на основе форм создается тот необходимый и удобный пользовательский интерфейс, в котором и происходит вся работа с БД.

Другим важным назначением форм является обеспечение безопасности структуры БД, т. к. производимые с помощью форм операции по вводу и редактированию данных не затрагивают структуры исходных таблиц. Рядовой пользователь БД, в принципе, не должен иметь доступа непосредственно к самим таблицам данных. Он должен иметь право только «заглянуть» в их содержимое и при необходимости его отредактировать. Грамотно построенная БД предоставляет пользователям возможность вообще не обращаться к самой программе Access, т. к. все необходимые им функции реализованы с помощью форм.

Большинство форм обычно присоединены к одной или нескольким таблицам или запросам БД. Источником данных, отображаемых в них, являются поля в базовых таблицах и запросах. Очень часто в формы добавляются элементы управления. Именно поэтому они разнообразнее других объектов. Поэтому, может быть, некоторые пользователи тратят много времени и сил на создание удобных и красивых форм в своей БД.

3.11.1. Режимы формы

Форма имеет наибольшее количество режимов из всех объектов БД.

Режим Формы (рис. 3.43) — основной режим, т. е. режим, в котором пользователи работают с формой.

Режим Макета — предназначен для интерактивной настройки макета формы. В режиме **Макета** элементы управления формы представлены в таком же виде, как и в режиме **Формы** (вместе с данными). В отличие от режима **Конструктора** здесь нет возможности добавлять новые элементы управления (кроме связанных полей) и настраивать некоторые их свойства.

Режим Таблицы. В этом режиме поля данных, имеющиеся в форме, представляются в виде таблицы.

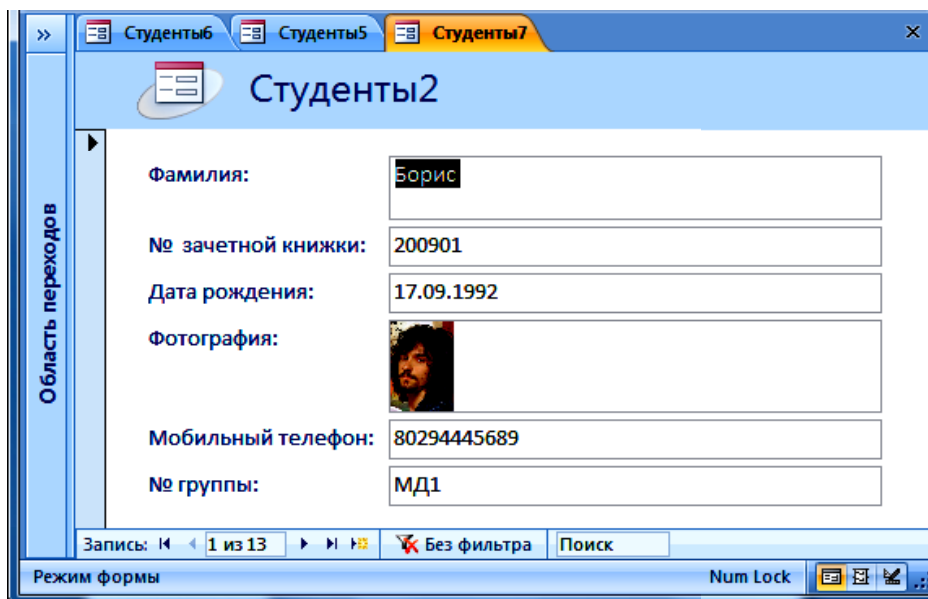


Рис. 3.43. Форма в режиме **Формы**

Режим Конструктора (рис. 3.44) — предназначен для создания и редактирования форм. В нем отображаются разделы и полосы их заголовков, а также линейки и сетки. При открытии формы в этом режиме становятся доступными специальные инструменты, сосредоточенные на контекстных вкладках **Конструктор** и **Упорядочить**.

Режимы Сводной таблицы и Сводной диаграммы. В этих режимах данные формы представляются в виде сводной таблицы или сводной диаграммы. Эти режимы удобно использовать для анализа данных.

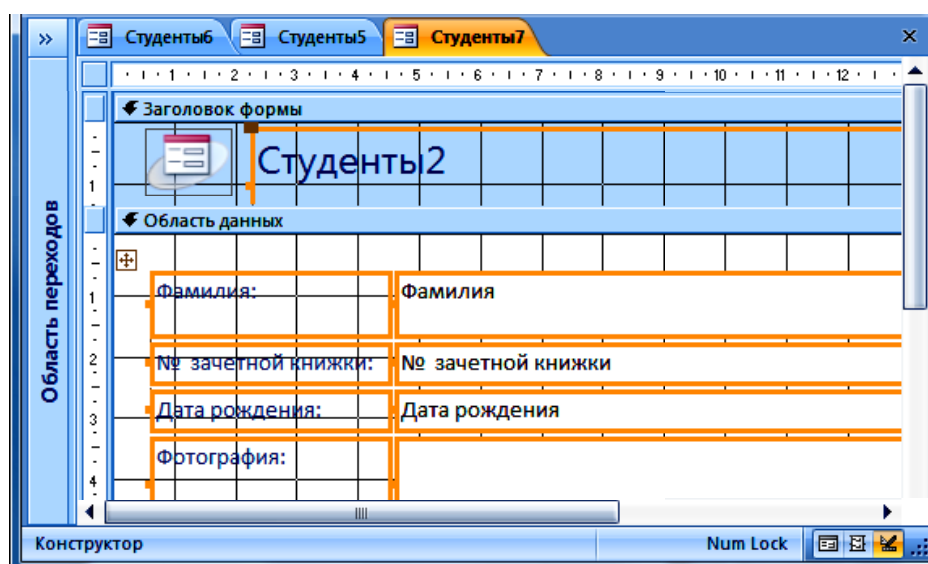


Рис. 3.44. Форма в режиме **Конструктора**

Форма открывается (например, двойным щелчком в области переходов) в режиме, заданном в ее свойстве **Режим по умолчанию**. Обычно это режим **Формы** (в один столбец или ленточная форма).

3.11.2. Создание форм на основе мастера и шаблонов

Для создания формы используют кнопки в группе инструментов **Формы** на вкладке **Создание** (рис. 3.45).

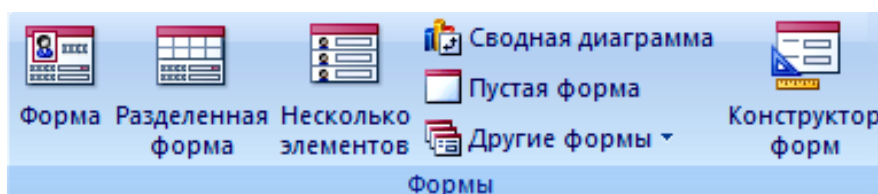


Рис. 3.45. Группа инструментов **Формы**

При помощи инструмента **Форма** можно создать форму одним щелчком мыши. При использовании этого средства все поля базового источника данных размещаются в форме. Можно сразу же начать использование новой формы либо при необходимости изменить ее в режиме **Макета** или **Конструктора**.

Разделенная форма позволяет одновременно отображать данные в двух представлениях — в режиме **Формы** и в режиме **Таблицы** (рис. 3.46).

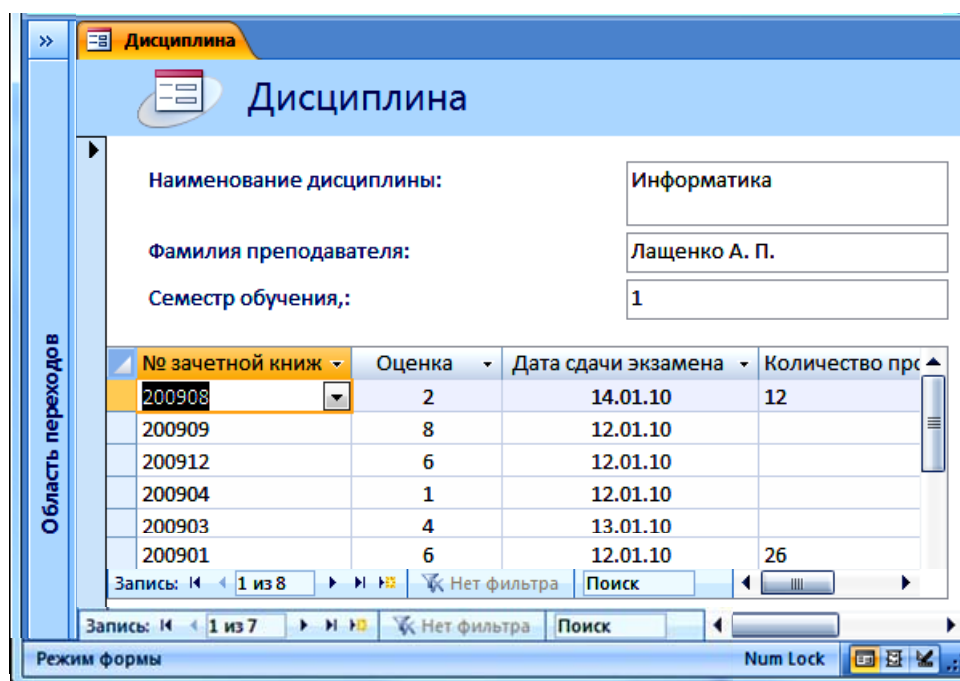


Рис. 3.46. Разделенная форма

Кнопка **Другие формы** позволяет вызвать **Мастер форм**. Процедура создания формы с помощью **Мастера форм** разделена на несколько шагов, при выполнении которых производится выбор нужной таблицы и тех полей в ней, которые необходимо просматривать или редактировать. Кроме того, допускается добавление полей и из других таблиц и размещение их на данной форме. Конечно, добавлять поля из других таблиц в создаваемую форму имеет смысл только в том случае, если обе таблицы связаны между собой.

Мастера форм создают формы, соответствующие их названию: в один столбец, ленточный, табличный, выровненный на основе одной заранее определенной таблицы или запроса. При этом форма строится в режиме диалога: последовательно открываются окна выбора внешнего вида формы, стиля оформления и имени, после чего создается форма со всеми остальными настройками, принятыми по умолчанию. Затем необходимо сохранить созданную форму под каким-либо именем.

Редактирование и ввод данных осуществляется таким же образом, как и в режиме **Таблицы**.

3.11.3. Создание формы в режиме Конструктора

Режим **Конструктора** предоставляет максимальные возможности для создания любых нестандартных форм, требующихся разработчику БД. При проектировании формы в режиме **Конструктора** можно использовать расширенный набор элементов управления, который недоступен в обычном режиме редактирования макета формы. Также можно настраивать внешний вид формы и расположенных на ней элементов управления в соответствии со своими требованиями и предпочтениями (поменять цвет, стиль оформления, положение и реакцию на действия пользователя у любого элемента управления, а также настроить любые свойства самой формы).

Режим **Конструктор** используется также для редактирования формы, созданной ранее с помощью **Мастера формы**, если он по каким-то причинам не удовлетворяет вас в полной мере.

Основные элементы управления. Каждая форма обязательно содержит несколько элементов управления. Элементы управления — это объекты, улучшающие интерфейс пользователя, которые используются для отображения, просмотра и работы с данными, а также для выполнения других действий.

Наиболее широко используемые элементы управления — поле (текстовое или числовое), кнопки, флажки, переключатели,

списки, надписи, а также рамки объектов для отображения графики и объектов OLE.

Программное управление формами и размещенными на них элементами управления осуществляется с помощью процедур, написанных на Visual Basic — встроенном языке программирования Access 2007.

По функциональному признаку любой элемент управления можно отнести к одной из трех следующих групп:

- присоединенные — элементы управления, источником данных которых служит поле таблицы или запроса;
- свободные — не имеющие источника данных (например, поля или выражения). Используются для вывода на экран сведений, пояснений, линий, прямоугольников и рисунков;
- вычисляемые — элементы управления, источником данных которых является результат вычисления заданного пользователем выражения, а не поле какой-либо таблицы БД.

Группа **Элементы управления** вкладки **Конструктор** состоит из трех секций (рис. 3.47).

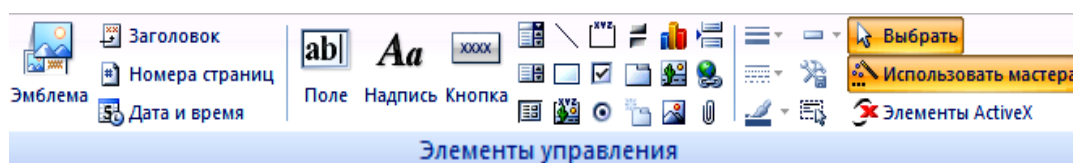


Рис. 3.47. Инструменты группы **Элементы управления**

В левой секции расположены четыре кнопки для добавления в форму «стандартных» элементов: **Эмблема**, **Заголовок**, **Номера страниц**, **Дата и время**. Кнопки открывают диалоговые окна настройки, а затем добавляют элементы в заранее «обговоренные» позиции.

В центральной секции находятся кнопки «классических» элементов управления, которые хорошо известны по диалоговым окнам в операционной системе MS Windows.

В правой секции находятся кнопки для форматирования рамок элементов управления и выполнения с ними других действий.

Конструирование формы. Каждая форма БД может включать следующие области:

- **Заголовок формы** — верхняя часть формы. Сюда можно поместить текст, графику и другие элементы управления. При печати отображается только на первой странице;

- **Верхний колонтитул** — отображается только в режиме **Предварительного просмотра**. При печати — вверху каждой страницы;
- **Область данных** — основная часть формы, может содержать элементы управления, отображающие данные из таблиц и запросов, а также неизменяемые данные (например, пояснительные надписи);
- **Нижний колонтитул** — отображается только в режиме **Предварительного просмотра**;
- **Примечание формы** — нижняя часть формы. Добавляется в форму вместе с разделом заголовка. При печати будет отображено только внизу последней страницы.

Для создания новой формы в режиме **Конструктора** необходимо на вкладке **Создание** в группе **Формы** выбрать кнопку **Конструктор форм** (рис. 3.45).

Access открывает пустую форму в режиме **Макета** и отображает область **Список полей** (рис. 3.48).

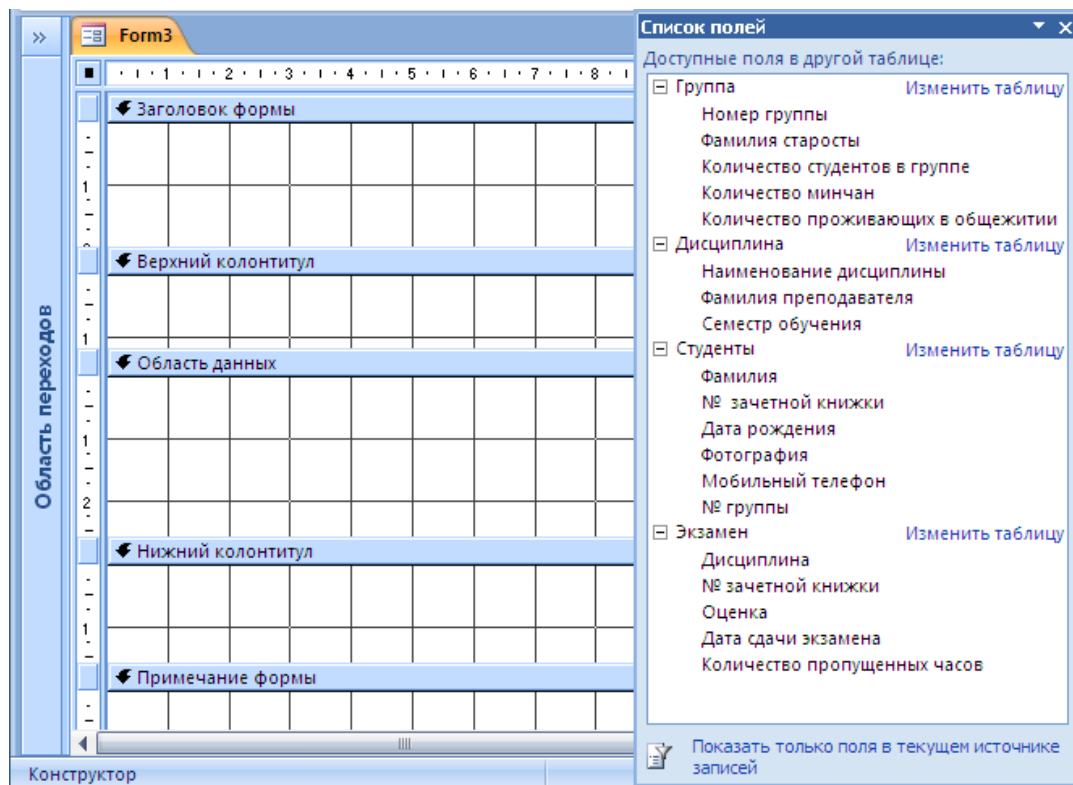


Рис. 3.48. Форма в режиме **Макета**

Если область **Список полей** не активизировалась, то необходимо на вкладке **Конструктор** в группе **Сервис** (рис. 3.49) щелкнуть по кнопке **Добавить поля**. Эта же вкладка содержит кнопку

Страница свойств, которая вызывает окно свойств формы или выделенного элемента управления.

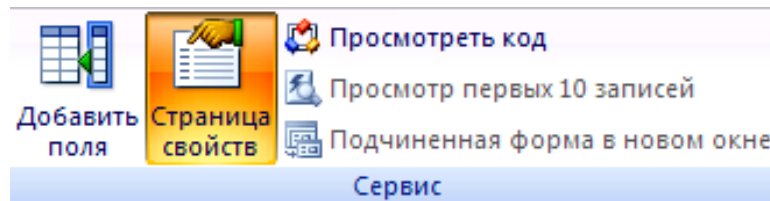


Рис. 3.49. Группа инструментов **Сервис**

Разрабатывая форму, необходимо решить ряд задач:

- определить источник данных (обычно таблицу или запрос);
- разместить на ней необходимые элементы управления;
- при необходимости добавить программы для обработки данных.

Для определения источника используется область **Список полей**, для выбора элементов управления — группа инструментов **Элементы управления**. Размещаются они перетаскиванием необходимых полей или элементов управления на форму.

Рассмотрим использование режима **Конструктора** на примере (рис. 3.50). Деканату требуется информация о том, как студенты конкретной группы (вводимой по запросу) сдали экзамен по конкретному предмету (также вводимому по запросу). А также определить средний балл по этому предмету в данной группе.

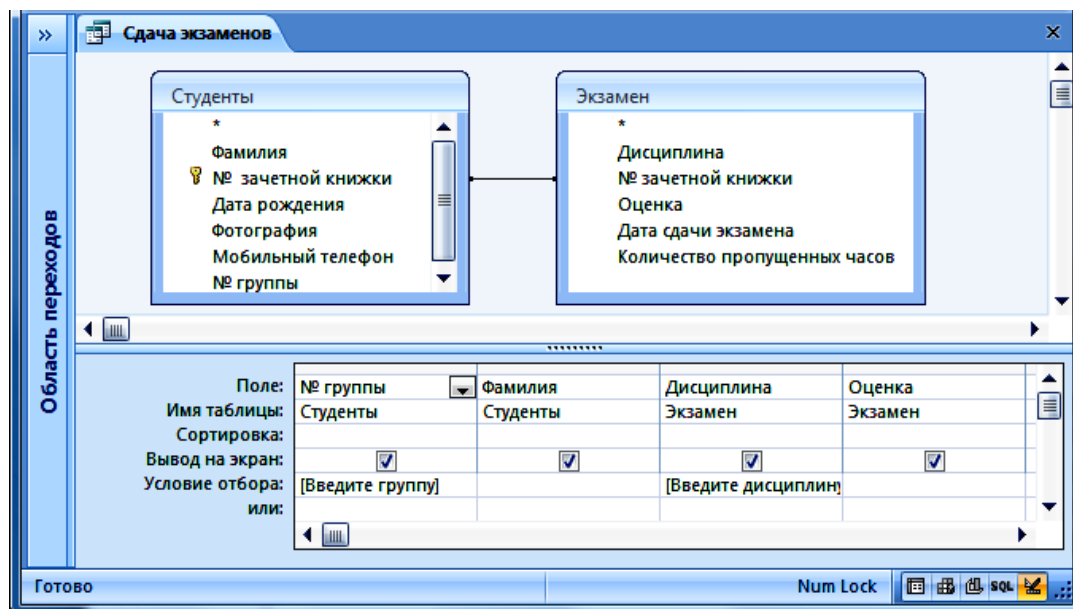


Рис. 3.50. Создание параметрического запроса в режиме **Конструктора**

Для этого вначале необходимо создать параметрический запрос, в который будут включены поля **Фамилия** и **№ группы** из таблицы **Студенты**, а также поля **Дисциплина** и **Оценка** из таблицы **Экзамен**.

Для полей **№ группы** и **Дисциплина** в строке **Условие отбора** введем соответствующие параметры (рис. 3.50).

Сохраним запрос под именем **Сдача экзаменов**.

С помощью **Мастера формы** создадим форму (рис. 3.51) и в режиме **Конструктора** и доработаем ее следующим образом: изменим заголовок формы, заменим поле оценок текстовым полем (вместо поля со списком), а также изменим размер шрифта, цвет фона и цвет подписей в именах полей.

Рис. 3.51. Форма, созданная с помощью **Мастера формы**

Кроме того, нам необходимо добавить вычисляемое поле для расчета среднего балла. Рассмотрим это более подробно.

Чтобы рассчитать средний балл, поместим на форму еще один элемент — **Поле**, выбрав его на вкладке **Конструктор** в группе **Элементы управления**.

Откроем окно свойств этого элемента, нажав клавишу F4 (либо выбрав команду **Свойства** из контекстного меню либо команду **Страница свойств** в группе **Сервис**). Перейдем на вкладку **Все** и зададим значения свойств в соответствии с приведенной таблицей (табл. 3.11).

Таблица 3.11

Значения свойств вычисляемого поля

| Свойство | Значение |
|-------------------------|--------------------------|
| Имя | Средний балл по предмету |
| Данные (ControlSource) | =Avg([Экзамен]![Оценка]) |
| Формат | Фиксированный |
| Число десятичных знаков | 2 |

Для создания выражения удобнее воспользоваться **Построителем выражений**, который вызывается нажатием кнопки [...] рядом с ячейкой свойства **Данные**. Чтобы сохранить изменения, необходимо нажать клавиши «CTRL+S».

Сохраним форму (рис. 3.52).

Рис. 3.52. Форма, отредактированная в режиме **Конструктор**

Подчиненная форма. Одна форма рассчитана на использование одного набора данных: таблицы, связанных таблиц или запросов. Для использования же в форме других данных, например размещения данных из связанных таблиц, создаются подчиненные формы.

Подчиненной называют форму, вставленную в другую форму. Первичная форма называется главной формой. Подчиненные формы удобны для отображения данных из таблиц или запросов, имеющих отношение «один-ко-многим».

В главной форме отображаются данные на стороне отношения «один». В подчиненной — на стороне отношения «многие».

Например, деканату требуется информация, где для каждой конкретной группы была бы выведена фамилия ее старосты, а также фамилии студентов этой группы, их телефоны, фотографии и даты рождения.

Получить такую информацию можно с помощью создания подчиненной формы. Для этого в основную форму включим поля **Группа** и **Фамилия старосты**, в подчиненную — **Фамилия**, **Мобильный телефон** и **Дата рождения**.

Создавать подчиненную форму будем с использованием мастера форм.

Для этого на вкладке **Создание** в группе **Формы** выберем кнопку **Другие формы**, а затем в списке пункт **Мастер форм**.

На первой странице мастера в раскрывающемся списке **Таблицы и запросы** выберем таблицу **Группа** и включим поля **№ группы**, **Фамилия старосты**.

На той же странице мастера в раскрывающемся списке **Таблицы и запросы** выберем другую таблицу (**Студенты**) и поля, которые требуется включить в форму (**Фамилия**, **Мобильный телефон** и **Дата рождения**) и нажмем кнопку **«Далее»**.

Установим переключатель в нижней части страницы мастера в положение **Подчиненные формы** и нажмем кнопку **«Далее»**.

Выберем внешний вид подчиненной формы и вариант: ленточный или табличный. В макетах обоих стилей данные подчиненной формы располагаются в виде строк и столбцов; табличный макет более компактен, а ленточный имеет больше возможностей настройки. В нем можно добавлять цвет, рисунки и другие элементы форматирования.

Выберем требуемый стиль форматирования формы и введем заголовки форм, а также укажем, в каком режиме должна открываться форма: **Формы** (для просмотра и ввода данных) или **Конструктора** (для возможности изменения ее структуры). Выбрав требуемые параметры, нажмем кнопку **«Готово»**.

Будет создано две формы: одна для главной формы, содержащей элемент управления подчиненной формы, а другая — для самой

подчиненной формы. Переход к другим записям формы осуществляется при помощи кнопок прокрутки (рис. 3.53).

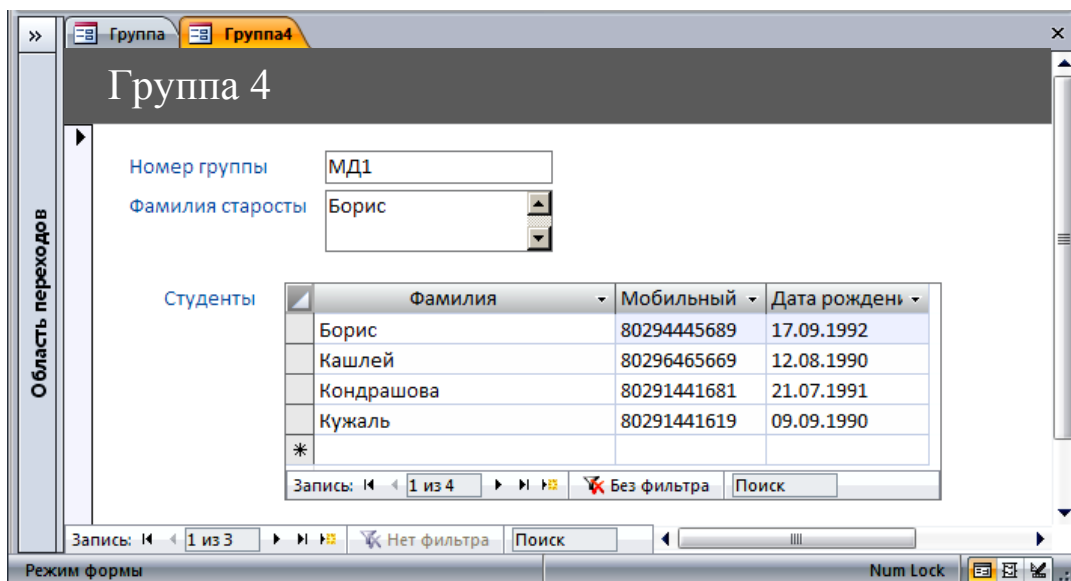


Рис. 3.53. Подчиненная форма в режиме **Формы**

Если необходимо доработать подчиненную форму, ее открывают в режиме **Конструктор**. Доработаем созданную нами форму следующим образом: изменим заголовок формы, вставим текущую дату, а также изменим размер шрифта, цвет фона и цвет подписей в именах полей. В результате получим (рис. 3.54):

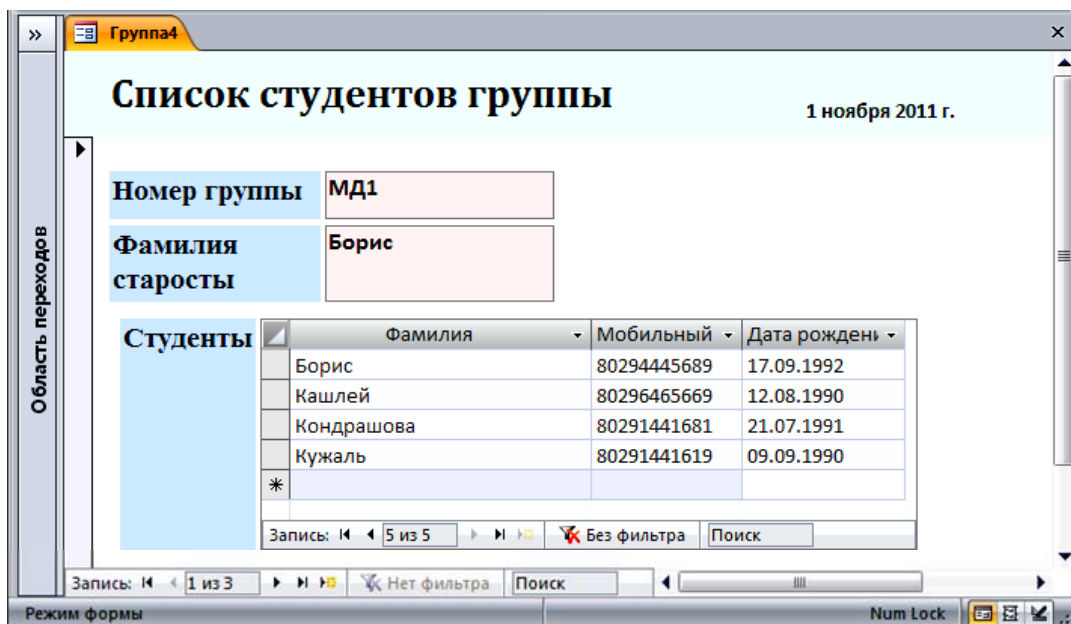


Рис. 3.54. Доработанная в режиме **Конструктора** подчиненная форма

Кнопочная форма. Кнопочные формы содержат только кнопки и предназначены для выбора основных действий, которые можно выполнять в БД. На кнопку в форме можно поместить текст или рисунок. Когда пользователь щелкает кнопку, запускается макрос или процедура обработки события.

Для создания кнопок в Access может использоваться мастер. Он упрощает и ускоряет процесс создания и настройки кнопок, автоматически выполняя все основные действия. С помощью мастера можно создавать более 30 различных типов кнопок для выполнения таких действий как: выполнение запроса, запуск макроса, запуск и закрытие приложения, применение фильтра, печать отчета, обновление данных в форме, поиск конкретной записи и др.

Для создания кнопочной формы в режиме **Конструктор** необходимо добавить на форму элемент управления **Кнопка**. После добавления будет открыт мастер **Создание кнопок**. Следуя указаниям мастера, выберем последовательно: в списке **Категории** – **Работа с формой**; в списке **Действия** для этой категории – **Открыть форму**; форму, которую нужно открыть при нажатии кнопки; текст или рисунок, которые нужно поместить на ней. На последней странице нажмем кнопку «**Готово**». Мастер создает кнопку и внедряет макрос **Нажатие кнопки** в свойство кнопки. Этот макрос содержит команды для выполнения задач, выбранных в мастере.

Создадим форму (рис. 3.55) которая при открытии БД **dekanat** будет осуществлять переход на другие формы (**Список студентов группы**, **Сдача экзаменов**) и также вызывать запрос (**Задолженности по экзаменам**).

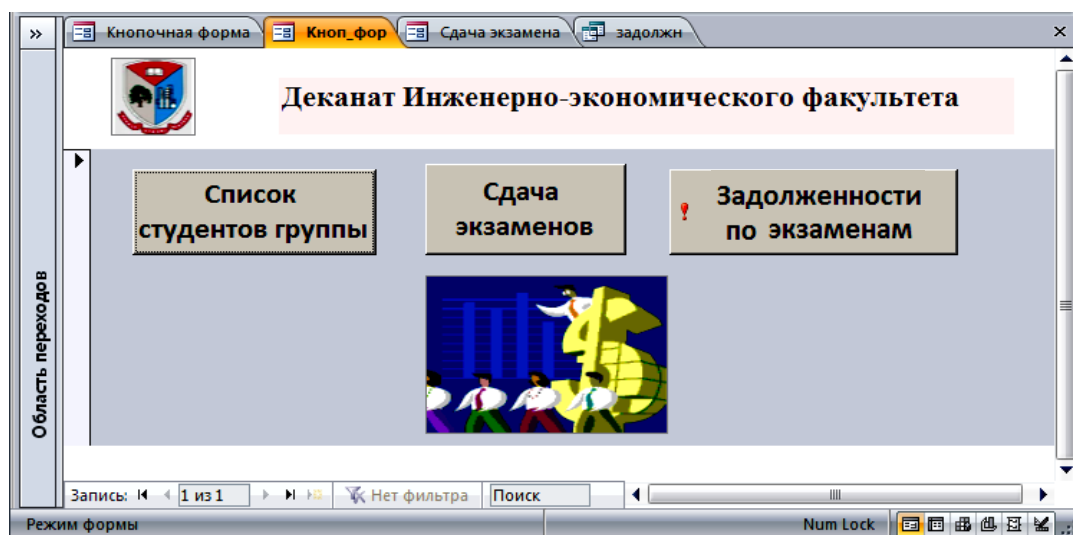


Рис. 3.55. Доработанная в режиме **Конструктора** подчиненная форма

3.12. Практическая работа № 9 Создание отчетов в СУБД Access 2007

Цель работы: изучить способы создания простых отчетов, отчетов с группировкой и сводных отчетов.

Отчеты используются для представления данных в легком для понимания и выразительном виде и предназначены, в основном, для вывода их на печать, а не для отображения на экране. По своей структуре отчеты похожи на формы, но в отличие от них в отчетах жестче контролируется расположение данных и с их помощью, естественно, нельзя вводить данные в БД.

Отчеты, как и формы, обычно присоединены к одной или нескольким таблицам (или запросам) БД. Связь осуществляется с помощью элементов управления, в которых отображаются надписи или данные. Как и форма, отчет не должен обязательно включать в себя все поля из каждой базовой таблицы или запроса.

Часто данные в отчетах располагаются в табличном формате. В отличие от распечаток таблиц или запросов, отчет дает более широкие возможности сортировки и группировки данных, в нем есть возможность добавлять итоговые значения, вычисляемые поля, а также поясняющие надписи, колонтитулы, номера страниц, стили и различные графические элементы. Выражения для вычисления значений полей, итоговых значений, а также надписи и графические элементы создаются и сохраняются в макете отчета.

Отчет может быть открыт в режиме **Предварительного просмотра**, в режиме **Представления отчета**, в режиме **Макета** или в режиме **Конструктора**. Первые два режима применяются для оценки того, как будет выглядеть отчет после его распечатки, а два последних — для его создания и редактирования.

Отчет может представлять собой как простой список, так и подробную сводку данных, сгруппированных по какому-либо параметру. Однако в любом случае необходимо сначала определить, в каких полях содержатся данные, которые должны войти в отчет, и в каких таблицах или запросах находятся эти поля.

3.12.1. Создание простого отчета

Для создания отчетов (также как и для таблиц и форм) используется на вкладке **Создание** группа инструментов **Отчеты** (рис. 3.56).

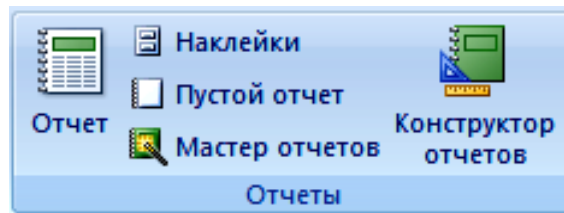


Рис. 3.56. Группа **Отчеты**

Кнопка **Отчет** одним щелчком создает простой отчет (авто-отчет), содержащий все поля выделенной таблицы (или запроса). К первому полю добавляется итоговое значение, соответствующее типу данных поля. Отчет содержит номера страниц, дату, время создания, заголовок, повторяющий имя таблицы или запроса, а также формальную эмблему. После создания отчет открывается в режиме **Макета**.

Кнопка **Наклейки** запускает мастер **Создание наклеек**, в диалоговых окнах которого вы задаете размер наклеек, их тип и количество, шрифт и его атрибуты, поля из выделенной таблицы (или запроса), которые следует разместить на наклейках, сортировку записей, а также имя отчета. Отчет после создания открывается в режиме **Предварительного просмотра**.

Кнопка **Пустой отчет** создает заготовку пустого отчета, которая открывается в режиме **Макета**. Для добавления полей открывается панель **Список полей**. Можно построить источник данных в виде запроса. Перед щелчком этой кнопки не нужно выделять таблицу или запрос. Отчет получает формальное имя **Отчет1**.

Кнопка **Мастер отчетов** загружает мастер **Создание отчетов**, который позволяет выбирать поля из различных таблиц или запросов, задавать вид представления данных, группировку и сортировку, стиль оформления и имя отчета. Мастер позволяет строить самые сложные отчеты. От пользователя требуется только корректно заполнять элементы управления в его диалоговых окнах. Отчет открывается в режиме **Предварительного просмотра** или **Конструктора**.

Кнопка **Конструктор отчетов** создает заготовку пустого отчета, которая открывается в режиме **Конструктора**. Для добавления полей открывается панель **Список полей**. Можно построить источник данных в виде запроса. Перед щелчком этой кнопки не нужно выделять таблицу или запрос. Отчет получает формальное имя **Отчет1**.

Любым из описанных выше способов отчет можно изменять или настраивать в режимах **Конструктор** и **Макет**.

На рис. 3.57 представлен отчет, созданный по таблице **Экзамен**.

| Дисциплина | № зачетной книжки | Дата сдачи экзамена | Оценка | Количество пропущенных часов |
|-------------|-------------------|---------------------|--------|------------------------------|
| Информатика | 201108 | 14.01.2010 | 2 | |
| Информатика | 201109 | 12.01.2010 | 8 | |
| Математика | 201109 | 07.01.2010 | 7 | 2 |
| Информатика | 201112 | 12.01.2010 | 6 | |
| Математика | 201112 | 07.01.2010 | 5 | |
| Химия | 201112 | 25.01.2010 | 7 | |
| Химия | 201109 | 25.01.2010 | 9 | |
| Химия | 201108 | 27.01.2010 | 4 | |
| Химия | 201107 | 25.01.2010 | 8 | |
| Информатика | 201104 | 12.01.2010 | 1 | |
| Экономика | 201103 | 12.06.2010 | 2 | 9 |
| Информатика | 201103 | 13.01.2010 | 4 | |
| Математика | 201102 | 08.01.2010 | 5 | |
| История | 201104 | 25.06.2010 | 4 | |
| Информатика | 201106 | 12.01.2010 | 7 | |

Рис. 3.57. Отчет, созданный с помощью кнопки **Отчет**

В **Мастере отчетов** предоставляется больше возможностей относительно выбора полей для включения в отчет. При этом можно указать способ группировки и сортировки данных, а также включить в отчет поля из нескольких таблиц или запросов, если отношения между этими таблицами и запросами заданы заранее. Если нужно добавить в отчет поля из нескольких таблиц (или запросов), то после выбора полей из первой таблицы процедура выбора полей для другой таблицы повторяется. Созданный отчет можно доработать в режиме **Конструктора**.

3.12.2. Добавление группировки, сортировки и итогов в отчет

Как правило, при создании отчетов добавляют различные виды сортировки и группировки, а также строки итогов. Простые операции сортировки, группировки и подведения итогов могут быть выполнены путем щелчка правой кнопки мыши на полях в режиме **Макета** и выбора нужной операции из контекстного меню.

Перейдем в режим **Макета** и установим с помощью контекстного меню: для поля **Дисциплина** — **Группировка**; для поля **Дата сдачи экзамена** — **Сортировка от А до Я**; для поля **Оценка** — **Итог оценка/Среднее**. В результате получим отредактированный отчет. Перейдем в режим **Предварительного просмотра**: имеем отчет (рис. 3.58).

| Дисциплина | № зачетной книжки | Дата сдачи экзамена | Оценка | Количество пропущенных часов |
|--------------------------|-------------------|---------------------|--------|------------------------------|
| Информатика | | | | |
| | 200906 | 12.01.2010 | 7 | |
| | 200909 | 12.01.2010 | 8 | |
| | 200912 | 12.01.2010 | 6 | |
| | 200904 | 12.01.2010 | 1 | |
| | 200901 | 12.01.2010 | 6 | 26 |
| | 200903 | 13.01.2010 | 4 | |
| | 200908 | 14.01.2010 | 2 | 12 |
| | 200902 | 14.01.2010 | 5 | |
| Средний балл по предмету | | | 4,9 | |
| История | | | | |
| | 200904 | 25.06.2010 | 8 | 35 |
| | 200902 | 25.06.2010 | 4 | |
| | 200913 | 26.06.2010 | 5 | |
| Средний балл по предмету | | | 5,7 | |

Рис. 3.58. Отчет в режиме **Предварительного просмотра**

ПОСТАНОВКА ЗАДАЧ ДЛЯ РАЗЛИЧНЫХ ПРЕДМЕТНЫХ ОБЛАСТЕЙ

1. Страховая компания

Описание предметной области

Вы работаете в страховой компании. Вашей задачей является отслеживание ее финансовой деятельности. Компания имеет различные филиалы по всей стране каждый со своим названием, адресом и телефоном. Деятельность компании организована следующим образом. Различные лица обращаются с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков, договор заключается по определенному виду страхования (например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора фиксируется дата заключения, страховая сумма, вид страхования, тарифная ставка и филиал, в котором заключался договор.

Таблицы

Филиал (Код филиала, Наименование филиала, Адрес, Телефон).

Вид страхования (Код вида страхования, Наименование).

Договоры (Номер договора, Дата заключения, Страховая сумма, Тарифная ставка, Код филиала, Код вида страхования).

Развитие постановки задачи

Нужно учесть, что договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон), нужно еще хранить филиал, в котором они работают. Кроме того, исходя из БД нужно иметь возможность рассчитывать заработную плату агентам. Заработная плата составляет некоторый процент от страхового платежа (страховой платеж — это страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

2. Гостиница

Описание предметной области

Вы работаете в гостинице. Вашей задачей является отслеживание финансовой стороны работы гостиницы. Ваша деятельность организована следующим образом. Гостиница предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортностью (люкс, полуплюкс, обычный) и ценой. Вашими клиентами являются различные лица, о которых Вы собираете определенную информацию (фамилия, имя, отчество и некоторый комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При заселении фиксируется его дата. При выезде из гостиницы для каждого места запоминается дата освобождения.

Таблицы

Клиенты (Код клиента, Фамилия, Имя, Отчество, Паспортные данные, Комментарий).

Номера (Код номера, Номер, Количество человек, Комфортность, Цена).

Поселение (Код поселения, Код клиента, Код номера, Дата поселения, Дата освобождения, Примечание).

Развитие постановки задачи

Необходимо хранить информацию не только по факту сдачи номера клиенту, но и осуществлять бронирование номеров. Кроме того, для постоянных клиентов, а также для определенных категорий клиентов предусмотрена система скидок. Скидки могут суммироваться.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

3. Реализация готовой продукции

Описание предметной области

Вы работаете в компании, занимающейся оптово-розничной продажей различных товаров. Вашей задачей является отслеживание финансовой стороны работы компании, деятельность которой организована следующим образом. Компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В компанию обращаются покупатели. Для каждого из них Вы фиксируете в БД стандартные данные (наименование,

адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с покупателем количество купленного им товара и дату покупки.

Таблицы

Товары (Код товара, Наименование, Оптовая цена, Розничная цена, Описание).

Покупатели (Код покупателя, Телефон, Контактное лицо, Адрес).

Сделки (Код сделки, Дата сделки, Код товара, Количество, Код покупателя, Признак оптовой продажи).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

4. Ведение заказов

Описание предметной области

Вы работаете в компании, занимающейся оптовой продажей различных товаров. Вашей задачей является отслеживание финансовой стороны работы компании. Деятельность фирмы организована следующим образом. Она торгует товарами из определенного спектра. Каждый из этих товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В компанию обращаются заказчики. Для каждого из них Вы фиксируете в БД стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с заказчиком количество купленного им товара и дату покупки.

Таблицы

Товары (Код товара, Цена, Доставка, Описание).

Заказчики (Код заказчика, Наименование, Адрес, Телефон, Контактное лицо).

Заказы (Код заказа, Код товара, Код заказчика, Количество, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что доставка разных товаров может производиться разными способами, различными по цене и скорости. Нужно хранить информацию по тому, какими

способами может осуществляться доставка каждого товара и информацию о том, какой вид доставки (и, соответственно, какую стоимость доставки) выбрал клиент при заключении сделки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

5. Бюро по трудоустройству

Описание предметной области

Вы работаете в бюро по трудоустройству. Вашей задачей является отслеживание финансовой стороны работы компании. Деятельность бюро организована следующим образом. Оно ищет работников для различных работодателей и вакансии для ищущих работу специалистов различного профиля. При обращении к Вам клиента-работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в БД. При обращении к Вам клиента-соискателя его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в БД. По каждому факту удовлетворения интересов обеих сторон составляется документ, в котором указываются соискатель, работодатель, должность и комиссионные (доход бюро).

Таблицы

Работодатели (Код работодателя, Название, Вид деятельности, Адрес, Телефон).

Соискатели (Код соискателя, Фамилия, Имя, Отчество, Квалификация, Вид деятельности, Иные данные, Предполагаемый размер заработной платы).

Сделки (Код соискателя, Код работодателя, Должность, Комиссионные).

Развитие постановки задачи

Оказалось, что БД не совсем точно описывает работу бюро. В базе фиксируется только сделка, а информация по открытым вакансиям не хранит. Кроме того, для автоматического поиска вариантов, необходимо вести справочник «Виды деятельности».

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

6. Нотариальная контора

Описание предметной области

Вы работаете в нотариальной конторе. Вашей задачей является отслеживание финансовой стороны работы компании. Деятель-

ность нотариальной конторы организована следующим образом. Фирма готова предоставить клиенту определенный комплекс услуг. Для наведения порядка Вы формализовали эти услуги, составив список с описанием каждой из них. При обращении к Вам клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в БД. По каждому факту оказания услуги клиенту составляется документ. В документе указываются услуга, сумма сделки, комиссионные (доход конторы), описание сделки.

Таблицы

Клиенты (Код клиента, Название, Вид деятельности, Адрес, Телефон).

Услуги (Код услуги, Название услуги, Описание).

Сделки (Код сделки, Код клиента, Код услуги, Сумма, Комиссионные, Описание).

Развитие постановки задачи

Теперь ситуация изменилась. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Кроме того, компания предоставляет в рамках одной сделки различные виды скидок. Скидки могут суммироваться.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

7. Фирма по продаже запчастей

Описание предметной области

Вы работаете в фирме, занимающейся продажей запасных частей для автомобилей. Вашей задачей является отслеживание финансовой стороны работы компании. Основная часть Вашей деятельности связана с работой с поставщиками. Фирма имеет определенный набор поставщиков, по каждому из которых известны название, адрес и телефон. У этих поставщиков Вы приобретаете детали. Каждая деталь наряду с наименованием характеризуется артикулом и ценой (считаем цену постоянной). Некоторые из поставщиков могут поставлять одинаковые детали (один и тот же артикул). Каждый факт покупки запчастей у поставщика фиксируется в БД, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей.

Таблицы

Поставщики (Код поставщика, Название, Адрес, Телефон).

Детали (Код детали, Наименование, Артикул, Цена, Примечание).

Поставки (Код поставщика, Код детали, Количество, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что цена детали может меняться от поставки к поставке. Поставщики заранее предупреждают о дате изменения цены и о ее новом значении. Нужно хранить не только текущее значение цены, но и всю историю ее изменений.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

8. Курсы по повышению квалификации

Описание предметной области

Вы работаете в учебном заведении и занимаетесь организацией курсов повышения квалификации. В Вашем распоряжении имеются сведения о сформированных группах студентов. Группы формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество студентов. Проведение занятий обеспечивает штат преподавателей. Для каждого из них у Вас в БД зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки Вы получаете информацию о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Кроме того, хранятся также сведения о виде проводимых занятий (лекции, практика), предмете и оплате за час.

Таблицы

Группы (Номер группы, Специальность, Отделение, Количество студентов).

Преподаватели (Код преподавателя, Фамилия, Имя, Отчество, Телефон, Стаж).

Нагрузка (Код преподавателя, Номер группы, Количество часов, Предмет, Тип занятия, Оплата).

Развитие постановки задачи

В результате работы с БД выяснилось, что размер почасовой оплаты зависит от предмета и типа занятия. Кроме того, каждый преподаватель может вести не все предметы, а только некоторые.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

9. Определение факультативов для студентов

Описание предметной области

Вы работаете в высшем учебном заведении и занимаетесь организацией факультативов. В Вашем распоряжении имеются

сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Преподаватели Вашей кафедры должны обеспечить проведение факультативных занятий по некоторым предметам. По каждому факультативу существует определенное количество часов и различные виды проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами у Вас появляется информация о том, кто из них записался на какие факультативы. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра Вы заносите информацию об оценках, полученных студентами на экзаменах.

Таблицы

Студенты (Код студента, Фамилия, Имя, Отчество, Адрес, Телефон).

Предметы (Код предмета, Название, Объем лекций, Объем практик, Объем лабораторных работ).

Учебный план (Код студента, Код предмета, Оценка).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

10. Туристическая фирма

Описание предметной области

Вы работаете в туристической компании, которая занимается продажей путевок клиентам. Вашей задачей является отслеживание финансовой стороны деятельности фирмы. Работа с клиентами в компании организована следующим образом. У каждого клиента, пришедшего к Вам, собираются некоторые стандартные анкетные данные: фамилия, имя, отчество, адрес, телефон. После этого Ваши сотрудники выясняют у клиента, куда он хотел бы поехать отдыхать. При этом ему демонстрируются различные варианты в зависимости от предпочтений (страны проживания, особенностей местного климата, отелей разного класса). Наряду с этим обсуждается возможная длительность пребывания и стоимость путевки. В случае если удалось договориться и найти для клиента

приемлемый вариант, Вы регистрируете факт продажи путевки (или путевок, если клиент покупает сразу несколько), фиксируя дату отправления. Иногда Вы решаете предоставить клиенту некоторую скидку.

Таблицы

Маршруты (Код маршрута, Страна, Климат, Длительность, Отель, Стоимость).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон).

Путевки (Код маршрута, Код клиента, Дата отправления, Количество, Скидка).

Развитие постановки задачи

Теперь ситуация изменилась. Фирма работает с несколькими отелями в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля. Скидки, которые предоставляет фирма, фиксированы. Например, при покупке более одной путевки предоставляется скидка 5%. Скидки могут суммироваться.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

11. Учет внутриофисных расходов

Описание предметной области

Вы работаете в бухгалтерии частной фирмы. Сотрудники фирмы имеют возможность осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Вашей задачей является отслеживание внутриофисных расходов. Фирма состоит из отделов, каждый из которых имеет название. В каждом отделе работает определенное количество сотрудников, которые могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел.

Таблицы

Отделы (Код отдела, Название, Количество сотрудников).

Виды расходов (Код вида, Название расхода, Описание, Предельная норма).

Расходы (Код расхода, Код вида, Код отдела, Сумма, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Оказалось, что нужно хранить данные о расходах не только по отделу в целом, но и по каждому сотруднику. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

12. Выдача банком кредитов

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра коммерческого банка, одним из существенных видов деятельности которого является выдача кредитов юридическим лицам. Вашей задачей является отслеживание динамики работы кредитного отдела. В зависимости от условий получения кредита, процентной ставки и срока возврата все кредитные операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить юридическое лицо (клиент), предоставивший при регистрации следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи.

Таблицы

Виды кредитов (Код вида, Название кредита, Условия получения, Ставка, Срок).

Клиенты (Код клиента, Название, Вид собственности, Адрес, Телефон, Контактное лицо).

Кредиты (Код вида, Код клиента, Сумма, Дата выдачи).

Развитие постановки задачи

Теперь ситуация изменилась. После проведения различных исследований выяснилось, что используемая система не позволяет отслеживать динамику возврата кредитов. Для устранения этого недостатка Вы приняли решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

13. Инвестирование свободных средств

Описание предметной области

Вы являетесь руководителем аналитического центра инвестиционной компании, которая занимается вложением денежных средств в ценные бумаги. Ваши клиенты — предприятия, которые доверяют Вам управлять их свободными денежными средствами в определенный период. Вам необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и Вам, и Вашему клиенту. При работе с клиентом для Вас необходимой является информация о его предприятии — название, вид собственности, адрес и телефон.

Таблицы

Ценные бумаги (Код ценной бумаги, Минимальная сумма сделки, Рейтинг, Доходность за прошлый год, Дополнительная информация).

Клиенты (Код клиента, Название, Вид собственности, Адрес, Телефон).

Инвестиции (Код инвестиции, Код ценной бумаги, Код клиента, Котировка, Дата покупки, Дата продажи).

Развитие постановки задачи

При эксплуатации БД стало понятно, что необходимо хранить историю котировок каждой ценной бумаги. Кроме того, помимо вложений в ценные бумаги, существует возможность вкладывать деньги в банковские депозиты.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

14. Платная поликлиника

Описание предметной области

Вы являетесь руководителем службы планирования платной поликлиники. Вашей задачей является отслеживание финансовых показателей работы поликлиники. В поликлинике работают врачи различных специальностей и разной квалификации. Каждый день в поликлинику обращаются больные. Все они проходят обязательную регистрацию, при которой в БД заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения). Каждый больной может обращаться в поликлинику несколько раз за различной медицинской помощью. Все обращения больных фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается дата обращения.

Таблицы

Врачи (Код врача, Фамилия, Имя, Отчество, Специальность, Категория).

Пациенты (Код пациента, Фамилия, Имя, Отчество, Год рождения).

Обращения (Код обращения, Код врача, Код пациента, Дата обращения, Диагноз, Стоимость лечения).

Развитие постановки задачи

В результате эксплуатации БД выяснилось, что при обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Общая стоимость лечения зависит от стоимости тех консультаций и процедур, которые назначены пациенту. Кроме того, для определенных категорий граждан предусмотрены скидки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

15. Анализ динамики показателей финансовой отчетности различных предприятий

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра крупного холдинга. Вашей задачей является отслеживание динамики показателей для предприятий холдинга, входящих в его структуру. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон, контактное лицо). Работа предприятия может быть оценена следующим образом. В начале каждого отчетного периода на основе финансовой отчетности по неким формулам вычисляется определенный набор показателей. Важность показателей характеризуется некоторыми числовыми константами. Значение каждого показателя измеряется в некоторой системе единиц.

Таблицы

Предприятия (Код предприятия, Название предприятия, Банковские реквизиты, Телефон, Контактное лицо).

Показатели (Код показателя, Название показателя, Важность, Единица измерения).

Динамика показателей (Код предприятия, Код показателя, Дата, Значение).

Развитие постановки задачи

В результате эксплуатации БД выяснилось, что некоторые показатели считаются в рублях, некоторые в долларах, некоторые

в евро. Для удобства работы с показателями нужно хранить изменения курсов валют относительно друг друга.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

16. Учет телекомпанией стоимости прошедшей в эфире рекламы

Описание предметной области

Вы являетесь руководителем коммерческой службы телевизионной компании. Вашей задачей является отслеживание расчетов, связанных с прохождением рекламы в телеэфире. Работа построена следующим образом: заказчики просят поместить свою рекламу в определенной передаче в определенный день. Каждый рекламный ролик имеет некоторую продолжительность. Для каждой организации-заказчика известны банковские реквизиты, телефон и контактное лицо для проведения переговоров. Передачи имеют определенный рейтинг. Стоимость минуты рекламы в каждой конкретной передаче известна (определяется коммерческой службой исходя из рейтинга передачи и прочих соображений).

Таблицы

Передачи (Код передачи, Название, Рейтинг, Стоимость минуты).

Заказчики (Код заказчика, Название организации, Банковские реквизиты, Телефон, Контактное лицо).

Реклама (Код рекламы, Код передачи, Код заказчика, Дата, Длительность в минутах).

Развитие постановки задачи

В результате эксплуатации БД выяснилось, что необходимо также хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

17. Интернет-магазин

Описание предметной области

Вы являетесь сотрудником коммерческого отдела компании, продающей различные товары через Интернет. Вашей задачей является отслеживание финансовой составляющей работы компании. Работа магазина организована следующим образом: на Интернет-сайте компании представлены (выставлены на продажу) товары. Каждый из них имеет название, цену и единицу измерения

(штуки, килограммы, литры). Для проведения исследований и оптимизации работы магазина Вы собираете данные о Ваших клиентах. При этом для Вас определяющее значение имеют стандартные анкетные данные, а также телефон и адрес электронной почты для связи. В случае приобретения товаров на сумму свыше 50 у. е. клиент переходит в категорию «постоянных клиентов» и получает скидку на каждую покупку в размере 2%. По каждому факту продажи Вы автоматически фиксируете клиента, купленные товары, их количество, дату продажи, дату доставки.

Таблицы

Товары (Код товара, Название, Цена, Единица измерения).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон, E-mail, Признак постоянного клиента).

Продажи (Код продажи, Код товара, Код клиента, Количество, Дата продажи, Дата доставки).

Развитие постановки задачи

В результате эксплуатации БД выяснилось, что иногда возникают проблемы, связанные с нехваткой информации о наличии на складе нужных товаров в нужном количестве. Кроме того, обычно клиенты в рамках одного заказа покупают не один вид товара, а несколько. Исходя из суммарной стоимости заказа компания предоставляет дополнительные скидки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

18. Грузовые перевозки

Описание предметной области

Вы работаете в компании, занимающейся перевозками грузов. Вашей задачей является отслеживание стоимости перевозок с учетом заработной платы водителей. Компания осуществляет перевозки по различным маршрутам. Для каждого маршрута Вы определили название, вычислили расстояние и установили оплату для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов Вы храните полную информацию о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия.

Таблицы

Маршруты (Код маршрута, Название, Дальность, Количество дней в пути, Оплата).

Водители (Код водителя, Фамилия, Имя, Отчество, Стаж).

Проделанная работа (Код маршрута, Код водителя, Дата отправки, Дата возвращения, Премия).

Развитие постановки задачи

Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему оплаты. Так, оплата водителям должна теперь зависеть не только от маршрута, но и от стажа водителя. Кроме того, нужно учесть, что перевозку могут осуществлять два водителя.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

19. Прокат автомобилей

Описание предметной области

Вы являетесь руководителем коммерческой службы в фирме, занимающейся прокатом автомобилей. Вашей задачей является отслеживание финансовых показателей работы пункта проката. В автопарк входит некоторое количество автомобилей различных марок, стоимости и типов. Каждый автомобиль имеет также стоимость проката. В пункт проката обращаются клиенты. Все клиенты проходят обязательную регистрацию, при которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Все обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата.

Таблицы

Автомобили (Код автомобиля, Марка, Стоимость, Стоимость проката, Тип).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные автомобили (Код автомобиля, Код клиента, Дата выдачи, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката, а также от года выпуска. Также нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

ЛИТЕРАТУРА

1. Microsoft Access 2003. Шаг за шагом: практ. пособие / пер. с англ. — М.: СП ЭКОМ, 2004. — 432 с.
2. Балдин, К. В. Информационные системы в экономике: учебник для вузов / К. В. Балдин, В. Б. Уткин. — М.: Дашков и К, 2005. — 442 с.
3. Гурин, Н. И. Работа с базами данных Access: учеб.-метод. пособие / Н. И. Гурин. — Минск: БГТУ, 2002. — 62 с.
4. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт; пер. с англ. — 7-е изд. — М.: Вильяме, 2001. — 1072 с.
5. Диго, С. М. Базы данных: проектирование и использование: учебник для вузов / С. М. Диго. — М.: Финансы и статистика, 2005. — 340 с.
6. Информатика для юристов и экономистов: учебник для вузов / С. В. Симонович [и др.]; под ред. С. В. Симоновича. — СПб.: Питер, 2007. — 965 с.
7. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг; пер. с англ. — М.: Вильяме, 2003. — 1440 с.
8. Леонтьев, В. П. Microsoft Office / В. П. Леонтьев. — М.: ОЛМА Медиа Групп, 2007. — 612 с.
9. Оскерко, В. С. Технологии баз данных: учеб. пособие для студентов экон. спец. вузов / В. С. Оскерко, З. В. Пунчик, О. А. Сосновский. — Минск: БГЭУ, 2007. — 172 с.
10. Тимошок, Т. В. Microsoft Access 2003. Самоучитель / Т. В. Тимошок. — М.: Вильяме, 2004. — 464 с.
11. Microsoft Office 2007. Все программы пакета. Самоучитель / А. Н. Тихомиров [и др.]. — СПб.: Наука и техника, 2008. — 608 с.
12. Уткин, В. Б. Информационные системы и технологии в экономике: учебник для вузов / В. Б. Уткин, К. В. Балдин. — М.: Юнити, 2005. — 341 с.
13. Хомоненко, А. Д. Базы данных: учебник для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; под ред. А. Д. Хомоненко. — 5-е изд., доп. — М.: Бином-Пресс; СПб.: КОРОНА принт, 2006. — 265 с.

ОГЛАВЛЕНИЕ

| | |
|--|----|
| ВВЕДЕНИЕ | 3 |
| 1. ТЕОРИЯ БАЗ ДАННЫХ | 5 |
| 1.1. Общие сведения о базах данных | 5 |
| 1.2. Категории баз данных | 7 |
| 1.3. Требования к базе данных | 10 |
| 1.3.1. Неизбыточность и непротиворечивость данных | 10 |
| 1.3.2. Защита данных от программных и аппаратных сбоев ... | 10 |
| 1.3.3. Мобильность прикладного программного обеспечения | 11 |
| 1.3.4. Секретность данных | 12 |
| 1.4. Представление и описание информации | 12 |
| 1.4.1. Плоские (двойные) файлы | 13 |
| 1.4.2. Ключи | 14 |
| 1.5. Модели данных | 14 |
| 1.5.1. Иерархическая модель | 15 |
| 1.5.2. Сетевая модель | 16 |
| 1.5.3. Реляционная модель | 16 |
| 1.6. Компоненты описания схемы данных | 18 |
| 2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ | 21 |
| 2.1. Проектирование базы данных | 21 |
| 2.2. Этапы проектирования базы данных и их процедуры | 22 |
| 2.3. Способы описания предметной области | 23 |
| 2.4. Описание информационной модели предметной области | 24 |
| 2.5. Нормализация отношений в реляционной базе данных | 25 |
| 2.6. Рекомендации по проектированию баз данных | 29 |
| 3. СУБД ACCESS 2007 | 34 |
| 3.1. Новые функциональные возможности СУБД Access 2007 | 34 |
| 3.2. Объекты Access 2007 | 37 |
| 3.3. Физическая структура данных | 38 |
| 3.4. Практическая работа № 1. Концептуальное проектирование | 41 |
| 3.5. Практическая работа № 2. Интерфейс СУБД Access 2007 ... | 44 |
| 3.6. Практическая работа № 3. Создание таблиц базы данных | 48 |

| | |
|---|---------|
| 3.6.1. Постановка задачи | 49 |
| 3.6.2. Элементы объекта «Таблица» | 50 |
| 3.6.3. Создание таблицы базы данных в режиме Таблицы | 50 |
| 3.6.4. Создание таблицы базы данных в режиме Конструктор | 52 |
| 3.6.5. Создание таблицы на основе шаблона | 55 |
| 3.6.6. Создание таблицы с помощью импорта внешних данных | 56 |
| 3.6.7. Модификация структуры таблицы..... | 58 |
| 3.7. Практическая работа № 4. Организация связей между таб- лицами | 59 |
| 3.7.1. Организация связей | 59 |
| 3.7.2. Изменение существующей связи..... | 62 |
| 3.8. Практическая работа № 5. Ввод, редактирование и сохра- нение данных в таблице | 62 |
| 3.8.1. Добавление записей непосредственно в таблицу в режиме Таблицы | 63 |
| 3.8.2. Добавление записей с использованием формы | 64 |
| 3.8.3. Изменение элементов в поле подстановок | 64 |
| 3.9. Практическая работа № 6. Работа с таблицами. Поиск ин- формации в базе данных | 66 |
| 3.9.1. Поиск и замена данных..... | 67 |
| 3.9.2. Сортировка и фильтрация данных | 68 |
| 3.10. Практическая работа № 7. Создание запросов в СУБД Access 2007 | 72 |
| 3.10.1. Создание запроса с помощью Мастера..... | 73 |
| 3.10.2. Создание запроса в режиме Конструктора | 75 |
| 3.11. Практическая работа № 8. Создание форм в СУБД Ас- cess 2007 | 87 |
| 3.11.1. Режимы формы..... | 87 |
| 3.11.2. Создание форм на основе мастера и шаблонов..... | 89 |
| 3.11.3. Создание формы в режиме Конструктора | 90 |
| 3.12. Практическая работа № 9. Создание отчетов в СУБД Access 2007 | 99 |
| 3.12.1. Создание простого отчета | 99 |
| 3.12.2. Добавление группировки, сортировки и итогов в отчет | 102 |
| ПРИЛОЖЕНИЕ. ПОСТАНОВКА ЗАДАЧ ДЛЯ РАЗЛИЧНЫХ ПРЕДМЕТНЫХ ОБЛАСТЕЙ | 103 |
| ЛИТЕРАТУРА | 117 |
| | 119 |

Учебное издание

**Лащенко Анатолий Павлович
Кишкурно Татьяна Владимовна**

**ПРОЕКТИРОВАНИЕ
БАЗ ДАННЫХ И СУБД
ACCESS 2007**

Лабораторный практикум

Редактор *П. В. Прохоровская*
Компьютерная верстка *П. В. Прохоровская*

Подписано в печать 25.11.2011. Формат 60×84¹/₁₆.
Бумага офсетная. Гарнитура Literaturna. Печать офсетная.
Усл. печ. л. 7,0. Уч.-изд. л. 7,2.
Тираж 500 экз. Заказ .

Издатель и полиграфическое исполнение:
УО «Белорусский государственный технологический университет».
ЛИ № 02330/0549423 от 08.04.2009.
ЛП № 02330/0150477 от 16.01.2009.
Ул. Свердлова, 13а, 220006, г. Минск.