Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
Российский государственный
гидрометеорологический университет

Т.М. Татарникова

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Учебное пособие

Санкт-Петербург
2004

УДК 004.65

Татарникова Т.М. Системы управления базами данных. Учебное пособие. СПб., изд. РГГМУ, 2004. – с.

Рецензент: О.И. Кутузов, д-р техн. наук, проф. РЭТУ (ЛЭТИ)

Учебное пособие написано по разделу дисциплине «Системы управления базами данных». Даны основы проектирования реляционных баз данных

Предназначено для подготовки морских инженеров по специальности 141000 – морские информационные системы, 075600 – информационная безопасность телекоммуникационных систем, а так же для студентов других специальностей РГГМУ, желающих изучить проектирование баз данных.

Введение

Современный мир информационных технологий трудно представить себе без использования баз данных. Практически все системы в той или иной степени связаны с функциями долговременного хранения и обработки информации. Фактически информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных, и теперь уже большинство операций не может быть выполнено вручную, они требуют наиболее перспективных компьютерных технологий. Любые административные решения требуют четкой и точной оценки текущей ситуации и возможных перспектив ее изменения. И если раньше в оценке ситуации участвовало несколько десятков факторов, которые могли быть вычислены вручную, то теперь таких факторов сотни и сотни тысяч, и ситуация меняется не в течение года, а через несколько минут, а обоснованность принимаемых решений требуется большая, потому что и реакция на неправильное решение более серьезная, более быстрая и более мощная, чем раньше. И, конечно, обойтись без информационной модели производства, хранимой в базе данных, в этом случае невозможно.

История развития систем управления базами данных (СУБД) насчитывает более 30 лет. Условно можно выделить четыре этапа в развитии технологии баз данных. Первый этап развития СУБД связан с организацией баз данных на больших машинах типа IBM 360/370, ЕС ЭВМ и мини-ЭВМ типа PDP11. Базы данных (БД) хранились во внешней памяти центральной ЭВМ, пользователями этих БД были задачи, запускаемые в основном в пакетном режиме. Интерактивный режим доступа обеспечивался с помощью терминалов, которые не обладали собственными вычислительными ресурсами и служили только устройствами ввода/вывода для центральной ЭВМ. На втором этапе, в эпоху персональных компьютеров, все СУБД были рассчитаны на создание БД в основном с монопольным доступом. И это понятно. Компьютер персональный, он не был подсоединен к сети, и БД на нем создавалась для одного пользователя. Хорошо известно, что история развивается по спирали, поэтому после процесса «персонализации» начался обратный процесс – интеграция. Множится количество локальных сетей, остро встает задача согласованности данных, хранящихся и обрабатывающихся в разных местах, но логически друг с другом связанных, возникают задачи, связанные с параллельной обработкой транзак-

ций. Успешное решение этих задач приводит к третьему этапу – появлению распределенных БД, сохраняющих все преимущества настольных СУБД и в то же время позволяющих организовать параллельную обработку информацию и поддержку целостности БД. Четвертый этап характеризуется появлением новой технологии доступа к данным Интернета. Основное отличие этого подхода от технологии клиент-сервер состоит в том, что отпадает необходимость использования специализированного клиентского программного обеспечения. Для работы с удаленной БД используются стандартный броузер Интернета, например, Microsoft Internet Explorer, и для конечного пользователя процесс обращения к данным происходит аналогично работе в сети Интернет. При этом встроенный в загружаемые пользователем HTML-страницы код, написанный обычно на языке Java, Java-script и других, отслеживает все действия пользователя и транслирует их в низкоуровневые SQL-запросы к БД, выполняя, таким образом, ту работу, которой в технологии клиент-сервер занимается клиентская программа.

1. Основные понятия и определения

База данных (БД) – электронная картотека – это именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Объект – человек, предмет, событие, место или понятие, о котором записаны данные. Так, в банковском деле примерами объектов могут служить клиенты, банковские счета, ссуды по закладам и т.д.

Предметная область – часть реального мира, отражаемая в БД, может относиться к любому типу организации (например, банк, университет, завод, больница).

Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Программы, с помощью которых пользователи работают с базой данных называются **приложениями.**

Значение данных – действительные данные, содержащиеся в каждом поле данных.

Хранимое поле – это наименьшая единица хранения данных. Каждое поле имеет свой тип информации. Например, БД на рис. 1, содержащая информацию о служащих, содержит 4 поля. Каждое

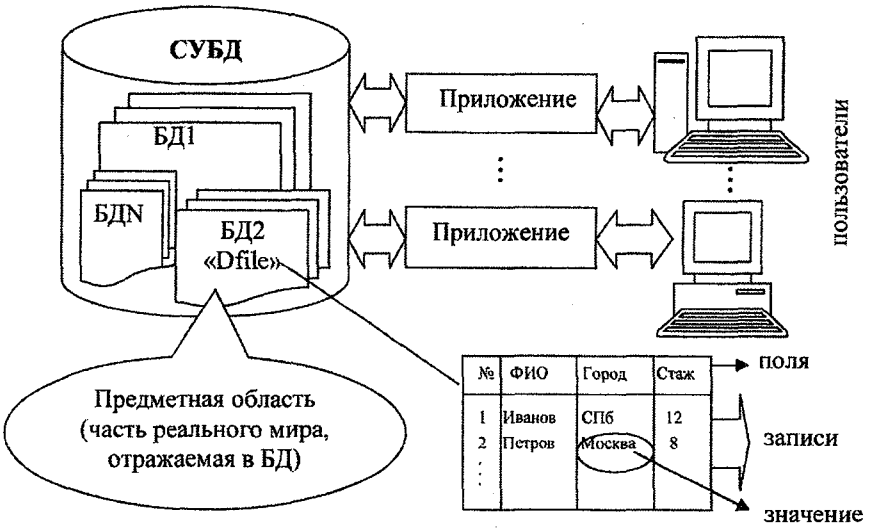


Рис. 1. Основные понятия и определения

поле имеет свой тип: №, ФИО, город, адрес.

Хранимая запись – это набор связанных хранимых полей, относящихся к одному объекту. В примере на рис. 1 служащий №1 имеет запись: «1 Иванов СПб 12».

Хранимый файл – это набор всех записей. Например, это файл Dfile со своим содержанием.

2. Модели СУБД

Система управления базами данных (СУБД) основывается на использовании определенной модели данных. Модель данных отражает взаимосвязи между объектами. Современная классификация СУБД предусматривает реализацию иерархических, сетевых и реляционных моделей данных.

Иерархическая модель данных строится по принципу иерархии типов объектов, т.е. один объект является главным, а остальные, находящиеся на низших уровнях иерархии, – подчиненными. Каждому узлу структуры соответствует один сегмент (S), представляющий собой элементы данных, характеризующих объект определенного уровня. Между главным и подчиненными типами объекта устанавливается взаимосвязь «один ко многим» (1:N): каждому сегменту (кроме корневого S1) всегда соответствует один входной

сегмент и несколько выходных. Каждый сегмент структуры лежит на единственном иерархическом пути, начинающемся от корневого сегмента (рис.2).

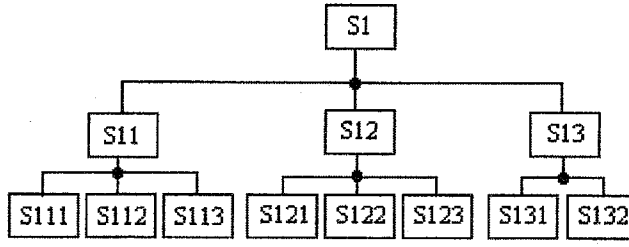


Рис. 2. Структура иерархической модели СУБД

Пример иерархической БД, в которой хранятся сведения об успеваемости студентов РГГМУ, приведен на рис. 3: в университете имеются 4 факультета, на каждом факультете – несколько кафедр, на кафедре – несколько групп, в группе – студенты, каждый студент имеет результаты сдачи сессии по нескольким дисциплинам. Везде связи между уровнями соответствуют «один – ко – многим».

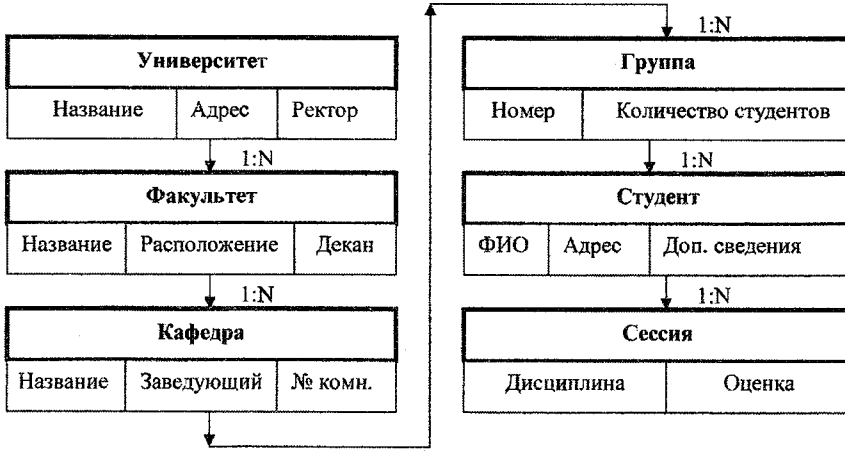


Рис. 3. Пример иерархической модели БД «Сведения об успеваемости студентов РГГМУ»

В сетевой модели данных понятия главного и подчиненных объектов несколько расширены: для любого сегмента допускается несколько входных сегментов наряду с возможностью наличия сег-

ментов без входов (взаимосвязь «многие-ко-многим»). Любой объект может быть и главным, и подчиненным. Это означает, что каждый объект может участвовать в любом числе взаимосвязей. Графическое представление структуры связей сегментов в такого типа моделях представляет собой сеть (рис.4).

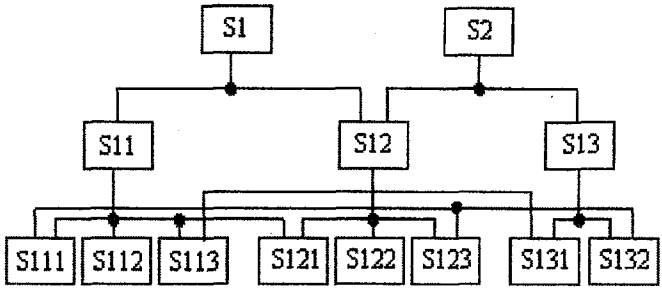


Рис. 4. Структура сетевой модели СУБД

Пример фрагмента сетевой БД, хранящей информацию о расписании занятий на кафедре приведен на рис.5: есть несколько преподавателей, несколько групп, несколько параллельно идущих занятий, и все это связано в сетевую модель по принципу «многие – ко – многим».

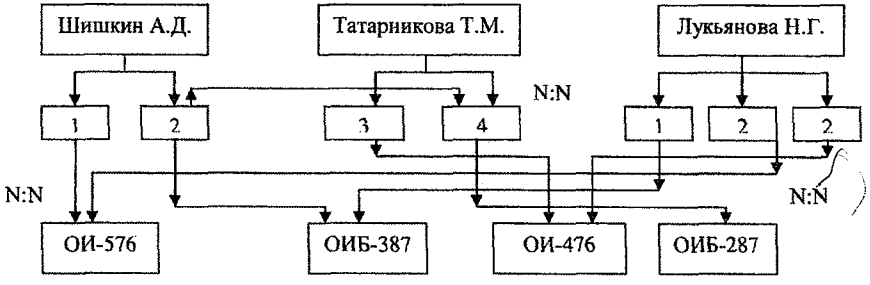


Рис. 5. Пример сетевой модели БД «Фрагмент расписания занятий на кафедре»

Реляционная модель СУБД основывается на математическом понятии отношения, а общая структура данных представляется в виде таблицы, в которой каждая строка значений соответствует логической записи, а заголовки столбцов являются характеристиками объектов, информацию о которых необходимо хранить в БД.

Американский математик Э.Ф. Кодд в 1970 г. впервые сформулировал основные понятия и ограничения реляционной модели. Предложив реляционную модель данных, Э.Ф. Кодд создал и инструмент для удобной работы с таблицами (отношениями) – реляционную алгебру. Каждая операция этой алгебры использует одну или несколько таблиц (отношений) в качестве ее операндов и продуцирует в результате новую таблицу, т.е. позволяет «разрезать» или «склеивать» таблицы. Появление реляционной алгебры, простота и наглядность для пользователей-непрограммистов определили большую популярность реляционной модели на современном рынке информационных технологий. Поэтому в дальнейшем будем говорить о проектировании БД, основанных только на реляционной модели.

3. Создание модели
3.1. Реляционная структура данных

Рассмотренные выше понятия и определения являются общепринятыми, не привязанными к конкретной модели СУБД. В реляционных БД существует своя (табличная) терминология (рис.6).

Отношение – соответствует тому, что мы до сих пор называли таблицей.

Кортеж – соответствует строке этой таблицы.

Рис. 6. Реляционная структура данных

Атрибут – столбцу.

Количество кортежей называется кардинальным числом, а количество атрибутов степенью.

Первичный ключ – уникальный идентификатор для таблицы, т.е. столбец или такая комбинация столбцов, что в любой момент времени не существует двух строк, содержащих одинаковое значение в этом столбце или комбинации столбцов.

Домен – общая совокупность значений, из которых берутся настоящие значения для определенных атрибутов определенного отношения.

3.2. Реляционная алгебра

Реляционная алгебра – набор математических операций, позволяющих манипулировать табличными данными. Реляционная алгебра, определенная Коддом состоит из 8-ми операций, состоящих из 2-х групп по 4 операции в каждой:

1. *Традиционные операции:* объединение, пересечение, вычитание, декартово произведение.
2. *Специальные операции:* выборка, проекция, соединение, деление.

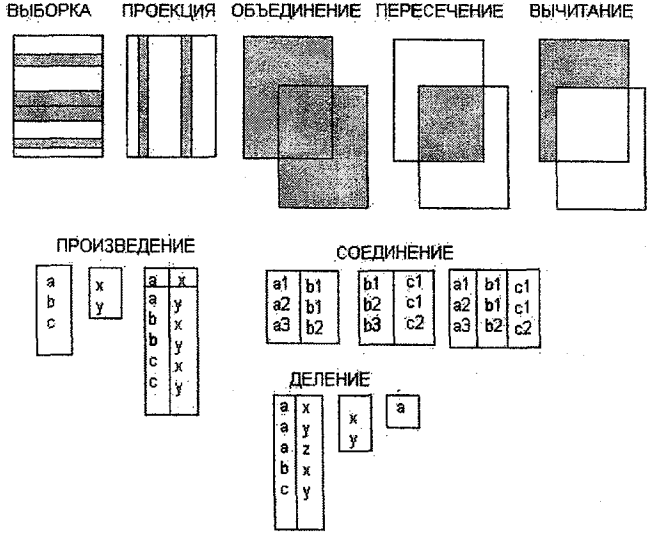


Рис. 7. Некоторые операции реляционной алгебры

Рассмотрим примеры выполнения этих операций, которые в теории множеств имеют обозначения, представленные на рис.7.

Выборка. Результатом выполнения этой операции являются все кортежи из определенного отношения, которые удовлетворяют определенным условиям.

Дано отношение:

№	Имя	Возраст	Адрес
1	Иванов	30	Москва
2	Петров	35	СПб
3	Сидоров	21	СПб
4	Сидоров	20	Москва

Выбрать лиц, моложе 30-ти лет (Условие: возраст<30)

Результат:

№	Имя	Возраст	Адрес
3	Сидоров	21	СПб
4	Сидоров	20	Москва

Проекция. Результатом будут все кортежи определенного отношения после исключения из него некоторых повторяющихся атрибутов.

Дано отношение:

ФИО спортсмена	Вид спорта	Город
Иванов И.Д.	Плавание	Москва
Петров С.М.	Легкая атлетика	СПб
Сидоров А.А.	Тяжелая атлетика	Красноярск
Смирнов С.А.	Фигурное катание	СПб
Архипов Л.И.	Плавание	Москва
Алексеев А.С.	Теннис	Томск
Викторов М.В	Фигурное катание	Пермь

Проекция по городам:

Город
Москва
СПб
Красноярск
Томск
Пермь

Проекция по видам спорта и городам:

Вид спорта	Город
Плавание	Москва
Легкая атлетика	СПб
Тяжелая атлетика	Красноярск
Фигурное катание	СПб
Теннис	Томск
Фигурное катание	Пермь

Произведение. Результат операции – всевозможные картежи, которые являются сочетанием 2-х картежей, принадлежащих соответственно двум определенным отношениям.

Пусть имеются два отношения А и В:

А

Код поставки
а
в
с

В

Код изделия
х
у

Произведение **АВ**:

Код поставки	Код изделия
а	х
а	у
в	х
в	у
с	х
с	у

Пересечение. Результатом является отношение, которое содержит все кортежи, которые принадлежат одновременно двум отношениям.

Дано: отношение А и отношение В:

А

№	ФИО	Возраст	Город
1	Иванов И.О.	20	Москва
2	Петров С.Т.	20	Москва

В

№	ФИО	Возраст	Город
1	Иванов И.О.	20	Москва
2	Сидоров В.Т.	16	СПб

Результат пересечения А и В:

№	Фамилия	Возраст	Город
1	Иванов И.О.	20	Москва

Объединение – результатом будет отношение, содержащее все кортежи, которые принадлежат одному из двух определенных отношений, или обоим.

Даны отношение А и отношение В:

А

№	ФИО	Возраст	Город
1	Иванов И.О.	20	Москва
2	Петров С.Т.	20	Москва-

В

№	ФИО	Возраст	Город
1	Иванов И.О.	20	Москва
3	Сидоров В.Т.	16	СПб

Результат объединения А и В:

№	Фамилия	Возраст	Город
1	Иванов И.О.	20	Москва
2	Петров С.Т.	20	Москва
3	Сидоров В.Т.	16	СПб

Вычитание – результатом будет отношение, содержащие все кортежи, которые принадлежат первому из 2-х отношений и не принадлежащих второму.

Даны отношение A и отношение B:

A

№	ФИО	Возраст	Город
1	Иванов И.О.	20	Москва
2	Петров С.Т.	20	Москва

B

№	ФИО	Возраст	Город
1	Иванов И.О.	20	Москва
3	Сидоров В.Т.	16	СПб

Результат вычитания A-B:

№	Фамилия	Возраст	Город
2	Петров С.Т.	20	Москва

Соединение – результирующее отношение, которое содержит кортежи 1-го и 2-го отношения, имеющих общее значение 1-го или нескольких полей. И такие общие значения в результирующем кортеже появляются только один раз, а не дважды.

Даны два отношения A и B:

A

Код поставщика	Поставщик	Стаж	Город поставки
S1	Иванов И.О.	20	Москва
S2	Петров С.Т.	10	СПб
S3	Сидоров В.Т.	30	СПб
S4	Архипов Л.И.	20	Москва
S5	Антонов А.А.	30	Пермь

B

Код детали	Наименование детали	Количество	Город поставки
P1	Болт	100000	Москва
P2	Винт	200000	СПб
P3	Гайка	150000	Воронеж
P4	Маркер	100000	Москва
P5	Подшипник	1000000	СПб
P6	Шуруп	50000	Москва

Результат соединения A и B:

Код поставщика	Поставщик	Стаж	Город поставки	Код детали	Наименование детали	Количество
S1	Иванов И.О.	20	Москва	P1	Болт	100000
S1	Иванов И.О.	20	Москва	P4	Маркер	100000
S1	Иванов И.О.	20	Москва	P6	шуруп	50000
S2	Петров С.Т.	10	Париж	P2	винт	200000
S2	Петров С.Т.	10	Париж	P5	маркер	100000
S3	Сидоров В.Т.	30	Париж	P2	винт	200000
S3	Сидоров В.Т.	30	Париж	P5	маркер	100000
S4	Архипов Л.И.	20	Москва	P4	маркер	100000
S4	Архипов Л.И.	20	Москва	P6	шуруп	50000
S4	Архипов Л.И.	20	Москва	P1	болт	100000

Деление. Деление выполнятся для двух отношений, в первом отношении должно быть задействовано два поля, а во втором – одно. Результатом будет отношение, содержащее все значение одного атрибута отношения с двумя полями, которым соответствуют все значениям в отношении с одним полем.

4. Проектирование реляционных БД

Этапы жизненного цикла БД изображены на рис.8.

Процесс проектирования БД представляет собой последовательность перехода от неформального словесного описания информационной структуры предметной области к формальному описанию объектов предметной области в терминах некоторой модели. В общем случае можно выделить следующие этапы проектирования:

1. Системный анализ и словесное описание информационных объектов предметной области.

2. Проектирование инфологической модели предметной области - частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах Е-модели.

3. Даталогическое или логическое проектирование БД, т. е. описание БД в терминах принятой даталогической модели данных.

4. Физическое проектирование БД, т. е. выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения.

Рис. 8. Этапы жизненного цикла БД

4.1. Системный анализ предметной области

С точки зрения проектирования БД в рамках системного анализа, необходимо осуществить первый этап, т. е. провести подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами.

В общем случае существуют два подхода к выбору состава и структуры предметной области:

• Функциональный подход – он реализует принцип движения «от задач» и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД. В этом случае можно четко определить минимальный необходимый набор объектов предметной области, которые должны быть описаны.

• Предметный подход – когда информационные потребности будущих пользователей БД жестко не фиксируются. Они могут быть многоаспектными и весьма динамичными. Не возможно точно выделить минимальный набор объектов предметной области, которые необходимо описывать. В описание предметной области в этом случае включаются такие объекты и взаимосвязи, которые наиболее характерны и наиболее существенны для нее. БД, конструируемая при этом, называется предметной, т. е. она может быть использована при решении множества разнообразных, заранее не определенных задач. Конструирование предметной БД в некотором смысле кажется гораздо боле заманчивым, однако трудность всеобщего охвата предметной области с невозможностью конкретизации потребностей пользователей может привести к избыточно сложной схеме БД, которая для конкретных задач будет неэффективной.

Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в БД, формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны гене-

рироваться в системе, описанием входных документов, которые служат основанием для заполнения данными БД.

Приведем пример описания предметной области.

Пусть требуется разработать информационную систему для автоматизации учета получения и выдачи книг в библиотеке. Система должна предусматривать режимы ведения системного каталога, отражающего перечень областей знаний, по которым имеются книги в библиотеке. Внутри библиотеки области знаний в систематическом каталоге могут иметь уникальный внутренний номер и полное наименование. Каждая книга может содержать сведения из нескольких областей знаний. Каждая книга в библиотеке может присутствовать в нескольких экземплярах. Каждая книга, хранящаяся в библиотеке, характеризуется следующими параметрами:

- уникальный шифр;
- название;
- фамилии авторов (могут отсутствовать);
- место издания (город);
- издательство;
- год издания;
- количество страниц;
- стоимость книги;
- количество экземпляров книги в библиотеке.

Книги могут иметь одинаковые названия, но они различаются по своему уникальному шифру (ISBN).

В библиотеке ведется картотека читателей.

На каждого читателя в картотеку заносятся следующие сведения:

- фамилия, имя, отчество;
- домашний адрес;
- телефон;
- дата рождения.

Каждому читателю присваивается уникальный номер читательского билета.

Каждый читатель может одновременно держать на руках не более 5 книг. Читатель не должен одновременно держать более одного экземпляра книги одного названия.

Каждая книга в библиотеке может присутствовать в нескольких экземплярах. Каждый экземпляр имеет следующие характеристики:

- уникальный инвентарный номер;
- шифр книги, который совпадает с уникальным шифром из описания книг;

- место размещения в библиотеке.

В случае выдачи экземпляра книги читателю в библиотеке хранится специальный вкладыш, в котором должны быть записаны следующие сведения:

- номер билета читателя;
- дата выдачи книги;
- дата возврата.

Предусмотреть следующие ограничения на информацию в системе:

1. Книга может не иметь ни одного автора.
2. В библиотеке должны быть записаны читатели не моложе 17 лет.
3. В библиотеке присутствуют книги, изданные начиная с 1960 по текущий год.
4. Каждый читатель может держать на руках не более пяти книг.
5. Каждый читатель при регистрации в библиотеке должен дать телефон для связи: он должен быть рабочим или домашним.
6. Каждая область знаний может содержать ссылки на множество книг, но каждая книга может относиться к различным областям знаний.

С данной информационной системой должны работать следующие группы пользователей:

- библиотекари;
- читатели;
- администрация библиотеки.

При работе с системой библиотекарь должен иметь возможность решать следующие задачи:

1. Принимать новые книги и регистрировать их в библиотеке.
2. Относить книги к одной или к нескольким областям знаний.
3. Проводить каталогизацию книг.
4. Проводить списание старых и не пользующихся спросом книг.
5. Вести учет выданных книг читателям.
6. Проводить списание утерянных книг по специальному акту списания или замены.
7. Проводить закрытие абонемента читателя.

Читатель должен иметь возможность решать следующие задачи:

1. Просматривать системный каталог, т. е. перечень всех областей знаний, книги по которым есть в библиотеке.

2. По выбранной области знаний получить полный перечень книг, которые числятся в библиотеке.
3. Для выбранной книги получить инвентарный номер свободного экземпляра книги или получить сообщение о том, что свободных экземпляров книги нет.
4. Для выбранного автора получить полный список книг, которые числятся в библиотеке.

Администрация библиотеки должна иметь возможность получать сведения о должниках-читателях библиотеки, которые не вернули вовремя взятые книги; сведения о книгах, которые не являются популярными, т.е. ни один экземпляр, которых не находится на руках у читателей; сведения о стоимости конкретной книги, для того чтобы установить возможность возмещения стоимости утерянной книги или возможность замены ее другой книгой; сведения о наиболее популярных книгах, т.е. таких, все экземпляры которых находятся на руках читателей.

Этот пример показывает, что перед началом разработки необходимо иметь точное представление о том, что же должно выполняться в нашей системе, какие пользователи в ней будут работать, какие задачи будет решать каждый пользователь.

4.2. Инфологическое проектирование

Инфологическое проектирование применяется на втором этапе проектирования БД, т. е. после словесного описания предметной области.

Инфологическое проектирование, прежде всего, связано с попыткой представления семантики предметной области в модели БД. В настоящий момент модель Чена «сущность-связь» (ER-модель) стала фактическим стандартом при инфологическом проектировании БД.

В основе ER-модели лежат следующие базовые понятия:

• ***Сущность***. С помощью сущности моделируется класс однотипных объектов. Сущность имеет имя, уникальное в пределах моделируемой системы. Так как сущность соответствует некоторому классу однотипных объектов, то предполагается, что в системе существует множество экземпляров данной сущности. Объект, которому соответствует понятие сущности, имеет свой набор атрибутов – характеристик, определяющих свойства данного представителя класса. При этом набор атрибутов должен быть таким, чтобы можно было различать конкретные экземпляры сущности. Например, у сущ-

ности «Студент» может быть следующий набор атрибутов: Номер зачетной книжки, Фамилия, Имя, Отчество, Группа. Набор атрибутов, однозначно идентифицирующий конкретный экземпляр сущности, называется ключевым. Для сущности «Студент» ключевым будет атрибут Номер зачетной книжки, поскольку для всех студентов данного вуза номера зачетных книжек будут различны. Экземпляром сущности «Студент» будет описание конкретного студента вуза (рис.9).

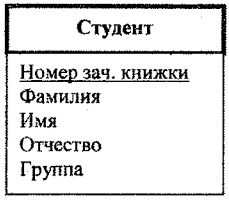


Рис. 9. Пример определения сущности в модели ER

• ***Связь***. Между сущностями могут быть установлены связи – бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют между собой. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). Она показывает, как связаны экземпляры сущностей между собой. Например, если у нас имеется связь между сущностью «Студент» и сущностью «Преподаватель» и эта связь - руководство дипломным проектированием, то каждый студент имеет только одного руководителя, но один и тот же преподаватель может руководить множеством студентов-дипломников. Поэтому это будет связь «один–ко–многим» (1:M), один со стороны «Преподаватель» и многие со стороны «Студент» (рис.10)

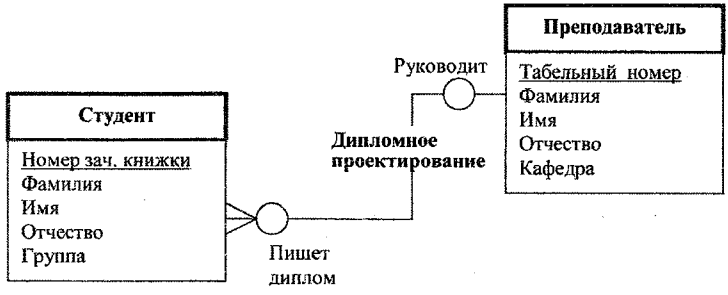


Рис. 10. Пример отношения «один-ко-многим» при связывании сущностей «Студент» и «преподаватель»

Связи делятся на три типа по множественности: «один-к-одному» (1:1), «один-ко-многим» (1:М), «многие-ко-многим» (М:М).

Связь 1:1 означает, что экземпляр одной сущности связан только с одним экземпляром другой сущности. Связь 1:М означает, что один экземпляр сущности, расположенный слева по связи, может быть связан с несколькими экземплярами сущности, расположенной справа по связи. Связь М:М означает, что один экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и наоборот, один экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

На рис. 11 представлена полная инфологическая модель базы данных «Библиотека», системный анализ которой рассматривался в разделе 4.1.

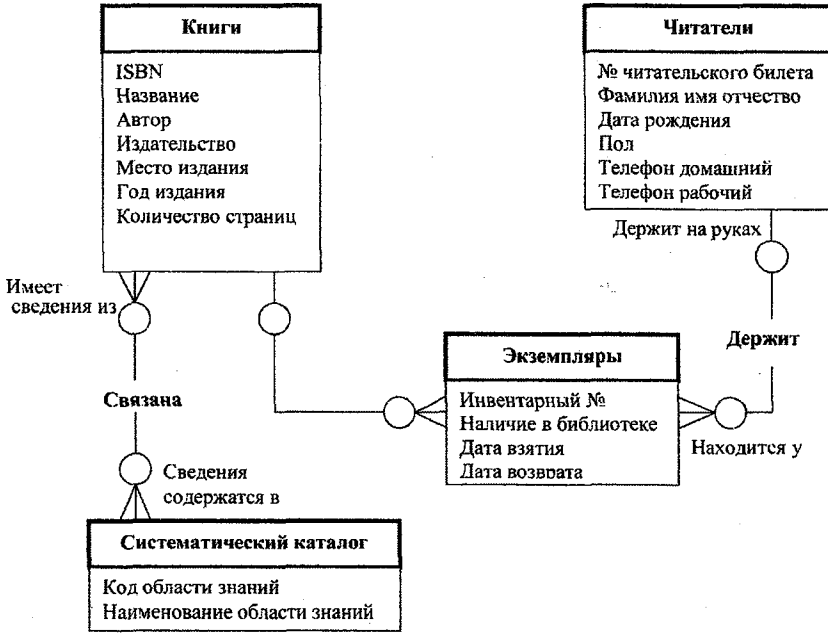


Рис. 11. Инфологическая модель «Библиотека»

4.3. Даталогическое проектирование

В реляционных БД даталогическое или логическое проектирование приводит к разработке схемы БД, т.е. совокупности схем отношений, которые адекватно моделируют абстрактные объекты предметной области и семантические связи между этими объектами. В общем случае в результате выполнения этого этапа должны быть получены следующие результирующие документы:

- описание концептуальной схемы БД в терминах выбранной СУБД;
- описание внешних моделей в терминах выбранной СУБД;
- описание декларативных правил поддержки целостности БД;
- разработка процедур поддержки семантической целостности БД.

Разработанная на этом этапе схема БД должна быть корректной, поэтому назовем логическое проектирование процессом разработки корректной схемы реляционной БД. Корректной назовем схему БД, в которой отсутствуют нежелательные зависимости между атрибутами отношений.

Проектирование схемы БД может быть выполнено двумя путями:

- путем декомпозиции (разбиения), когда исходное множество отношений, входящих в схему БД, заменяется другим множеством отношений (число их при этом возрастает), являющихся проекциями исходных отношений;
- путем синтеза, т.е. путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД.

Классическая технология проектирования реляционных БД связана с теорией нормализации, основанной на анализе функциональных зависимостей между атрибутами отношений.

Процесс проектирования с использованием декомпозиции представляет собой процесс последовательной нормализации схем отношений, при этом каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.

Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной, если удовлетворяет свойственному ей набору ограничений.

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2 NF);
- третья нормальная форма (3 NF);
- нормальная форма Бойса-Кодда (BC NF);
- четвертая нормальная форма (4 NF);
- пятая нормальная форма, или форма проекции-соединения (5 NF или PJNF).

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле улучшает свойства предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Отношение находится в первой нормальной форме тогда и только тогда, когда на пересечении каждого столбца и каждой строки находятся только элементарные значения атрибутов.

Отношения, находящиеся в первой нормальной форме, часто называют просто нормализованными отношениями. Соответственно, ненормализованные отношения могут интерпретироваться как таблицы с неравномерным заполнением, например, табл. 1 «Расписание»:

Таблица 1

Расписание

Преподаватель	День недели	Номер пары	Название дисциплины	Тип занятий	Группа
Бескид П.П.	Понедельник	2	Схемотехника	Лекция	ОИ, ОИБ-III
	Пятница	2	Микропроцессоры	Лекция	ОИ, ОИБ-IV
Шишкин А.Д.	Среда	1	Информатика	Лекция	ОИ, ОИБ-I
	Четверг	2	Информатика	Лаб. раб.	ОИ-191
Татарникова Т.М.	Вторник	1	Моделирование систем	Лекция	ОИ-V
	Четверг	1	СУБД	Лаб. раб.	ОИ-480
	Пятница	3	СУБД	Лаб. раб.	ОИ-481

Здесь на пересечении одной строки и одного столбца находится целый набор элементарных значений, соответствующих набору дней, перечню пар, набору дисциплин, по которым проводит занятия один преподаватель. Для приведения отношения «Расписание» к первой нормальной форме необходимо дополнить каждую строку фамилией преподавателя (см. табл. 2).

Таблица 2

Нормализованное расписание

Преподаватель	День недели	Номер пары	Название дисциплины	Тип занятий	Группа
Бескид П.П.	Понедельник	2	Схемотехника	Лекция	ОИ, ОИБ-III
Бескид П.П.	Пятница	2	Микропроцессоры	Лекция	ОИ, ОИБ-IV
Шишкин А.Д.	Среда	1	Информатика	Лекция	ОИ, ОИБ-I
Шишкин А.Д.	Четверг	2	Информатика	Лаб. раб.	ОИ-191
Татарникова Т.М.	Вторник	1	Моделирование систем	Лекция	ОИ-V
Татарникова Т.М.	Четверг	1	СУБД	Лаб. раб.	ОИ-480
Татарникова Т.М.	Пятница	3	СУБД	Лаб. раб.	ОИ-481

Отношение находится во второй нормальной форме тогда и только тогда, когда оно находится в первой нормальной форме и не содержит неполных функциональных зависимостей непервичных атрибутов первичного ключа.

Рассмотрим отношение, моделирующее сдачу студентами текущей сессии. Структура этого отношения определяется следующим набором атрибутов: (ФИО, № зач. Книжки, Группа, Дисциплина, Оценка).

Так как каждый студент сдает целый набор дисциплин в процессе сессии, то первичным ключом отношения может быть (№ зач. Книжки, Дисциплина), который однозначно определяет каждую строку отношения. Атрибуты «ФИО» и «Группа» зависят только от части первичного ключа – от значения атрибута «№ зач. Книжки», поэтому мы должны констатировать наличие неполных функциональных зависимостей в данном отношении. Для приведения данного отношения ко второй нормальной форме следует разбить его на проекции, при этом должно быть соблюдено условие восстановления исходного отношения без потерь. Такими проекциями могут быть два отношения:

- (ФИО, № зач. Книжки, Группа);
- (№ зач. Книжки, Дисциплина, Оценка).

Этот набор отношений не содержит неполных функциональных зависимостей, поэтому эти отношения находятся во второй нормальной форме.

А почему надо приводить отношения ко второй нормальной форме? Иначе говоря, какие аномалии могут возникнуть, если оставить исходное отношение и не разбивать его на два? Предположим,что студент переведен из одной группы в другую. Тогда в первом случае (если не разбивать исходное отношение на два) мы должны найти все записи с данным студентом и в них изменить значе-

ние атрибута «Группа» на новое. Во втором же случае меняется только один кортеж в первом отношении. И конечно, опасность нарушения корректности БД в первом случае выше. Может получиться так, что часть кортежей поменяет значения атрибута «Группа», а часть по причине сбоя в работе аппаратуры останется в старом состоянии. Тогда наша БД будет содержать записи, которые относят одного студента одновременно к разным группам. Чтобы этого не произошло, необходимо принимать дополнительные непростые меры, например, организовать процесс согласованного изменения с использованием сложного механизма транзакций. Если же БД приведена ко второй нормальной форме, то достаточно изменить только один кортеж. Кроме того, если есть студенты, которые еще не сдавали экзамены, то в исходном отношении вообще нет возможности хранить о них информацию, а во второй схеме информация о студентах и их принадлежности к конкретной группы хранится отдельно от информации, которая связана со сдачей экзаменов, и поэтому можно отдельно работать со студентами и отдельно хранить и обрабатывать информацию об успеваемости и сдаче экзаменов, что в действительности и происходит.

Отношение находится в третьей нормальной форме тогда и только тогда, когда оно находится во второй нормальной форме и не содержит транзитивных зависимостей.

Рассмотрим отношение, связывающее студентов с группами, факультетами и специальностями, на которых он учится: (ФИО, № зач. Книжки, Группа, Факультет, Специальность, Выпускающая кафедра).

Первичным ключом отношения является «№ зач. Книжки», однако рассмотрим остальные функциональные зависимости. Группа, в которой учится студент, однозначно определяет факультет, на котором он учится, а также специальность и выпускающую кафедру. Кроме того, выпускающая кафедра однозначно определяет факультет, на котором обучаются студенты, выпускаемые по данной кафедре. Но если предположим, что одну специальность могут выпускать несколько кафедр, то специальность не определяет выпускающую кафедру. В этом случае есть следующие функциональные зависимости:

- № зач. Книжки -- ФИО;
- № зач. Книжки -- Группа;
- № зач. Книжки -- Факультет;
- № зач. Книжки -- Специальность;
- № зач. Книжки -- Выпускающая кафедра;

- Группа -- Факультет;
- Группа -- Специальность;
- Группа -- Выпускающая кафедра;
- Выпускающая кафедра -- Факультет.

Эти зависимости образуют транзитивные группы. Для того чтобы избежать этого, можно предложить следующий набор отношений:

- (**№ зач. Книжки**, ФИО, Специальность, Группа);
- (**Группа**, Выпускающая кафедра);
- (**Выпускающая кафедра**, факультет).

Первичные ключи отношений выделены. Теперь необходимо удостовериться, что при естественном соединении не теряется ни одна строка и не получается лишних кортежей. Данное упражнение предлагается выполнить самостоятельно. Полученный набор отношений находится в третей нормальной форме.

Отношение находится в нормальной форме Бойса-Кодда, если оно находится в третей нормальной форме и каждый детерминант отношения является возможным ключом отношения.

Рассмотрим отношение, моделирующее сдачу студентом текущих экзаменов. Предположим, что студент может сдавать экзамен по одной дисциплине несколько раз, если он получил неудовлетворительную оценку. Отношение имеет следующую структуру: (№ зач. Книжки, Идентификатор студента, Дисциплина, Дата, Оценка). Возможными ключами отношения являются (№ зач. Книжки, Дисциплина, Дата) и (Идентификатор Студента, Дисциплина, Дата).

Определим имеющиеся функциональные зависимости:

- № зач. Книжки, Дисциплина, Дата -- Оценка;
- Идентификатор студента, Дисциплина, Дата -- Оценка;
- № зач. Книжки -- Идентификатор студента;
- Идентификатор студента -- № зач. Книжки.

Это отношение находится в третьей нормальной форме, потому что неполных функциональных зависимостей непервичных атрибутов от атрибутов возможного ключа здесь не присутствует и нет транзитивных зависимостей. Но под четвертую нормальную форму данное отношение не подходит, так как есть два детерминанта – «№ зач. Книжки» и «Идентификатор студента», которые не являются возможными ключами отношения. Для приведения отношения к нормальной форме Бойса-Кодда надо разделить отношение, например, на два со следующими схемами:

- (Идентификатор студента, Дисциплина, Дата, Оценка);
- (№ зач. Книжки, Идентификатор студента);

или наоборот

- (№ зач. Книжки, Дисциплина, Дата, Оценка);
- (№ зач. Книжки, Идентификатор студента).

В большинстве случаев достижение третей нормальной формы или даже формы Бойса-Кодда считается достаточным для реальных проектов БД, однако в теории нормализации существуют нормальные формы высших порядков, которые уже связаны не с функциональными зависимостями между атрибутами отношений, а отражают более тонкие вопросы семантики предметной области и связаны с другими видами зависимостей.

Перед тем, как рассмотреть нормальные формы высших порядков, остановимся на некоторых понятиях, которые необходимы для определения этих форм.

В отношении R(A,B,C) существует многозначная зависимость R.A -->> R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

Пусть дано отношение, которое моделирует предстоящую сдачу экзаменов на сессии. Допустим, оно имеет вид: (№ зач. Книжки, Группа, Дисциплина).

Перечень дисциплин, которые должен сдавать студент, однозначно определяется не его фамилией, а номером группы (т.е. специальностью, на которой он учится).

В данном отношении существуют следующие две многозначные зависимости:

- Группа ->> Дисциплина;
- Группа ->> № зач. Книжки.

Это означает, что каждой группе однозначно соответствует перечень дисциплин по учебному плану и номер группы определяет список студентов, которые в этой группе учатся. Если мы будем работать с исходным отношением, то мы не сможем хранить информацию о новой группе и ее учебном плане – перечне дисциплин, которые должна пройти группа до тех пор, пока в нее не будут зачислены студенты. С одной стороны, при изменении перечня дисциплин по учебному плану, например, при добавлении новой дисциплины, внести эти изменения в отношение для всех студентов, занимающихся в данной группе, весьма затруднительно. С другой стороны, если добавлять студентов в уже существующую группу, то необходимо добавить множество кортежей, соответствующих перечню дисциплин для данной группы. Эти аномалии модификации отношения как раз и связаны с наличием двух многозначных

зависимостей.

В теории реляционных БД доказывается, что в общем случае в отношении R(A,B,C) существует многозначная зависимость R.A>>R.B в том и только в том случае, когда существует многозначная зависимость R.A->>R.C.

Дальнейшая нормализация отношений основывается на теореме Фейджина.

Теорема Фейджина

Отношение R(A,B,C) можно спроецировать без потерь в отношение R1(A,B) и R2(A,C) в том и только в том случае, когда существует многозначная зависимость A->>B и A->>C.

Под проецированием без потерь понимается такой способ декомпозиции отношения путем применения операции проекции, при котором исходное отношение полностью и без избыточности восстанавливается путем естественного соединения полученных отношений. Практически теорема доказывает наличие эквивалентной схемы для отношения, в котором существует несколько многозначных зависимостей.

Отношение R находится в четвертой нормальной форме (4NF) в том и только в том случае, если в случае существования многозначной зависимости A->>B все остальные атрибуты R функционально зависят от A.

В рассматриваемом примере можно произвести декомпозицию исходного отношения в два отношения:

- (№ зач. Книжки, Группа);
- (Группа, Дисциплина).

Оба эти отношения находятся в четвертой нормальной форме и свободны от отмеченных аномалий. Действительно, обе операции модификации теперь упрощаются: добавление нового студента связано с добавлением всего одного кортежа в первое отношение, а добавление новой дисциплины – с добавлением одного кортежа во второе отношение; кроме того, во втором отношении можно хранить любое количество групп с определенным перечнем дисциплин, в которые пока еще не зачислены студенты.

Пятая нормальная форма связана с анализом нового вида зависимостей – зависимостей «проекции-соединения». Этот вид зависимостей является в некотором роде обобщением многозначных зависимостей.

Отношение R(X,Y,...,Z) удовлетворяет зависимости соединения (X,Y,...,Z) в том и только в том случае, когда R восстанавливается без потерь путем соединения своих проекций на X,Y,...,Z. Здесь

X,Y,...,Z – наборы атрибутов отношения R.

Наличие зависимости «проекция-соединение» в отношении делает его в некотором роде избыточным и затрудняет операции модификации.

Отношение R находится в пятой нормальной форме (нормальной форме проекции-соединения) в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R..

Рассмотрим отношение R1:

R1(Преподаватель, Кафедра, Дисциплина).

Предположим, что каждый преподаватель может работать на нескольких кафедрах и на каждой кафедре может вести несколько дисциплин. В этом случае ключом отношения является полный набор из трех атрибутов. В отношении отсутствуют многозначные зависимости, и поэтому отношение находится в 4NF.

Введем следующие обозначения наборов атрибутов:

- ПК(Преподаватель, Кафедра);
- ПД(Преподаватель, Дисциплина);
- КД(Кафедра, Дисциплина).

Допустим, что отношение R1 удовлетворяет зависимости проекции соединения (ПК, ПД,КД). Тогда отношение R1 не находится в нормальной форме проекции соединения, потому что единственным ключом его является полный набор атрибутов, а наличие зависимости проекции соединения связано с набором атрибутов, которые не составляют возможные ключи отношения R1. Для того чтобы привести это отношение к нормальной форме проекции соединения, его надо представить в виде трех отношений:

- R2(Преподаватель, Кафедра);
- R3(Преподаватель, Дисциплина);
- R4(Кафедра, Дисциплина).

Пятая нормальная форма редко используется на практике. В большей степени она является теоретическим исследованием.

4.4. Выбор СУБД

4.4.1. Архитектура MS Access

Microsoft Access называет объектами все, что может иметь имя. В СУБД Access основными объектами являются таблицы, запросы, формы, отчеты, макросы и модули. Назначения этих объектов приведено в табл. 3 «Объекты СУБД Access»

Таблица 3

Объекты СУБД Access

Объект	Назначение объекта
Таблица	Объект, который определяется и используется для хранения данных. Любая таблица содержит информацию о субъектах (предметах) одного типа (например, клиентах). Поля (столбцы) таблицы служат для хранения различных характеристик субъектов (например, фамилии, имена, адреса клиентов), а любая запись (которая называется также строкой) содержит сведения о конкретном субъекте (например, данные о клиенте по имена Иван). Для любой таблицы можно определить первичный ключ (одно или несколько полей, имеющих уникальные для любой записи значение) и один или несколько индексов, ускоряющих доступ к данным.
Запрос	Объект, позволяющий пользователю получить нужные данные из одной или нескольких таблиц. Для определения запроса можно использовать бланк QBE или написать инструкцию SQL. Можно создать запросы на выборку, обновление, удаление, добавление данных. С помощью запросов можно также создавать новые таблицы, использующие данные из одной или нескольких существующих таблиц.
Форма	Объект, предназначенный в основном для ввода данных, отображения их на экран или управления работой приложения. Формы можно использовать для того, чтобы реализовывать требования пользователей к представления данных таблиц или наборов записей запросов. Формы можно также распечатывать.
Отчет	Объект, предназначенный для форматирования, вычисления итогов и печати выбранных данных. Прежде чем выводить отчет на принтер, можно предварительно просмотреть его на экране.
Макрос	Объект, представляющий собой структурированное описание одного или нескольких действий, которые должен выполнить Access в ответ на определенное событие. Например, можно определить макрос, который при выборе некоторого элемента в основной форме открывает другую форму. Макросы можно использовать для открытия таблиц, выполнения запросов, просмотра или печати отчетов. Из одного макроса можно также запускать другой макрос.
Модуль	Объект, содержащий программы на языке Visual Basic.

Взаимосвязь объектов СУБД Access можно представить в виде графа (рис.12).

Рис. 12. Взаимосвязь объектов СУБД Access

В табл.3 хранятся данные, которые можно извлекать с помощью запросов. С помощью вводятся данные на экран или изменяются. Формы и отчеты получают данные как непосредственно из таблиц, так и через запросы. С помощью макросов и модулей можно изменять ход выполнения приложения: открывать, фильтровать и изменять данные в формах и отчетах, выполнять запросы и создавать новые таблицы.

4.4.2. Создание таблиц

Таблица представляет собой набор данных по определенной части предметной области. Таблицы, лежащие в основе реляционной модели данных, обладают следующими свойствами:

- каждая строка представляет собой все сведения об одном объекте;
- данные одного столбца (атрибута) имеют одинаковый тип;
- каждый атрибут имеет свое имя;
- порядок следования строк не имеет значения.

Атрибуту может быть назначен один из следующих типов, поддерживаемых Access:

- Текстовые данные. Это текст или числа, не требующие вы-

полнения расчетов. Размер текстового поля ограничен – не более 255 символов.

- Данные типа Числовой. Позволяет вводить числовые данные, используемые для выполнения расчетов.
- Данные типа Дата/Время. Могут содержать значения даты, времени, либо и то, и другое.
- Данные типа Счетчик. Хранят целое значение, которое Access увеличивает автоматически при добавлении новых записей.
- Данные типа Логический. Содержат данные, которые могут иметь одно из двух значений: «1» - Да (True) и «0» - Нет (False).
- Данные типа Поле объекта OLE. Обеспечивают доступ к данным, которые можно связать с OLE–сервером. Этот тип данных включает битовые образы: например, рисунки-файлы из Paint Windows, файлы со звуком, деловой графикой и видеоизображением.
- Данные типа Мастер подстановки. Образуют поле со списком, который позволяет выбрать значения из другой таблицы или из набора постоянных значений.

Создать таблицу можно с помощью мастера или конструктора. Мастер таблиц представляет собой программу, написанную на языке Access Visual Basic, позволяющую быстро сконструировать таблицу на основе копирования подходящих полей из нескольких десятков типов полей, предоставляемых разработчиками Access. При работе мастера все свойства полей заполняются автоматически. Создание таблицы в режиме конструктора сводится к заполнению строк конструктора, в том числе обязательных полей «Имя поля», «Тип данных» и необязательных «Описание». Каждая строка в таблице конструктора соответствует столбцу создаваемой таблицы. Для создания табл. 4 «Сотрудники» необходимо заполнить конструктор таблицы следующим образом (рис.13). Затем сохранить данную конструкцию и открыть таблицу для заполнения ее записями.

Таблица 4

Сотрудники

Табельный номер	Фамилия	Имя	Должность	Дата рождения
2401	Петров	Денис	Начальник отдела	24.04.66
2403	Васильева	Анна	Секретарь	05.06.77
2405	Антонова	Ольга	Ведущий специалист	15.11.71
2406	Варшавский	Олег	Главный экономист	25.12.73
2407	Смирнов	Сергей	Менеджер	11.05.80

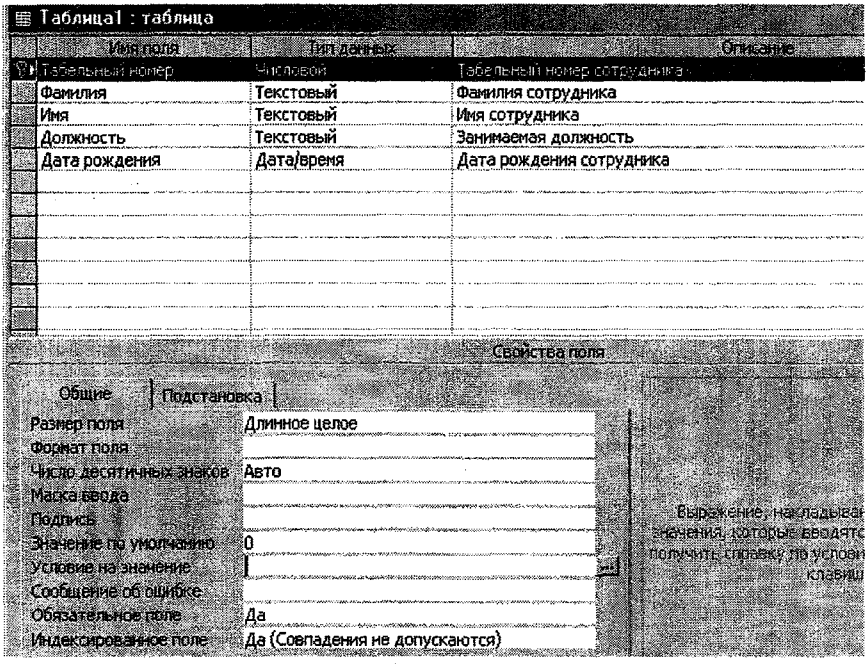


Рис. 13. Конструирование таблицы «Сотрудники»

4.4.3. Создание формы

Формы ввода данных помогают пользователю быстро, легко и без ошибок поместить информацию в таблицы БД. Форму можно сконструировать на базе существующей таблицы или запроса. Access поддерживает создание подчиненных форм (форма внутри формы), которые используются для отображения связей типа «один – ко - многим». В главной форме представлены данные из основной таблицы, а подчиненная содержит несколько строк записей из связной таблицы.

Создать форму можно с помощью мастера и конструктора. Использование мастера форм позволяет облегчить пользователю процесс создания формы. Мастер руководит процессом проектирования, задавая вопросы о структуре и оформлении формы, выбирая их из нескольких вариантов. С помощью конструктора разработчик самостоятельно создает свою индивидуальную форму.

Рассмотрим пример создания формы для таблицы 5 «Абитуриенты».

Таблица 5

Абитуриенты

Фамилия	Имя	Код группы	Код дисциплины	Оценка	Досрочная сдача	Фотография
Андреев	Иван	ОИ-186	1	5	Да	Рисунок Paint
Борисова	Анна	ОИБ-187	1	3	Нет	Рисунок Paint

В данной таблице использованы три словаря: «Группы» (Код группы, Название), «Дисциплины» (Код дисциплины, Название), «Оценки» и бинарный признак досрочной сдачи («Да» «Нет»).

Схема данных, объединяющая эти таблицы представлена на рис. 14.

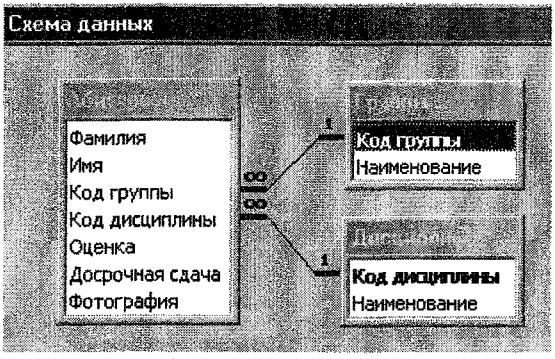


Рис. 14. Связь словарей «Группа», «Дисциплины», «Оценки»

Конструирование формы для таблицы «Абитуриент» состоит из следующих шагов:

1. Создать «Новую форму», в диалоговом окне которой следует выбрать пункт «Конструктор», а в качестве источника данных таблицу «Абитуриенты» (рис. 15).

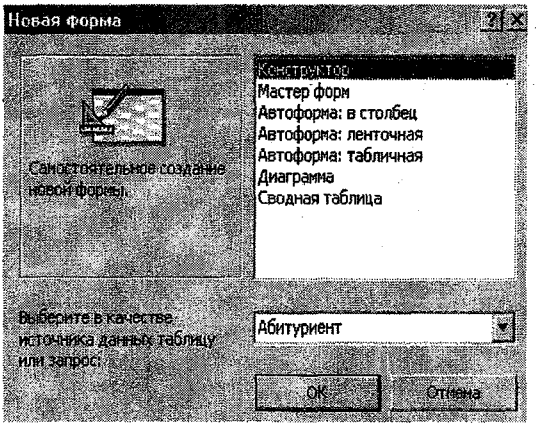


Рис. 15. Диалоговое окно «Новая форма»

2. Для работ по конструированию формы нужна специальная панель с элементами. На рис. 16 показана форма, содержащая только один раздел - «Область данных», который представляет собой пока пустую форму. С помощью панели элементов, собственно, и конструируется форма, которая должна представлять собой удобный пользовательский интерфейс, содержащий все необходимые функции по работе с БД в виде кнопок, переключателей, списков и т.п. для пользователей-непрограммистов.

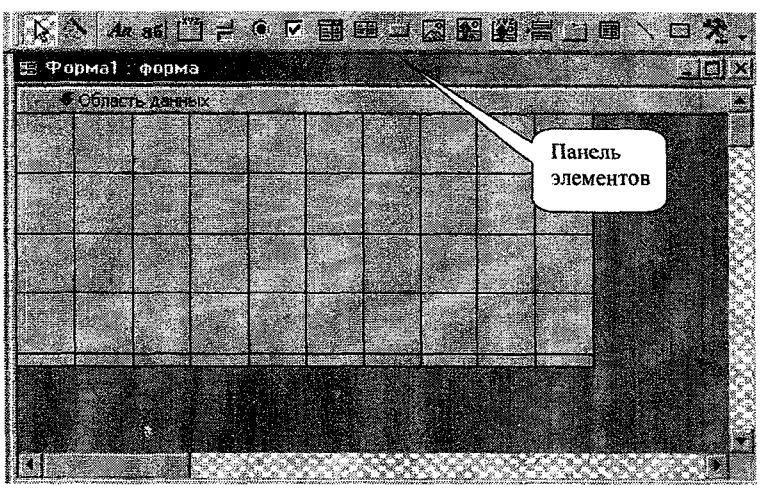
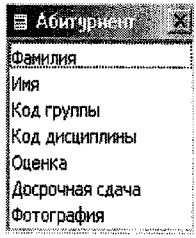


Рис. 16. Окно конструктора форм с панелью элементов

3.  Создадим заголовок формы «Результаты приемных экзаменов» с помощью элемента «Надпись».

4.  В заголовок текста включим поле с текущей датой. В этом поле в свойстве «Данные» необходимо набрать выражение =Date(), где Date - встроенная функция Access, возвращающая текущую дату.

5.  Поместим в области данных поля «Фамилия», «Имя», «Фотография» для ввода соответствующих данных, используя элемент «Список», после активизации которого появляется панель, содержащая список всех полей в таблице «Абитуриенты». Из этого списка выбираем по очереди и переносим в область данных три поля «Фамилия», «Имя» и «Фотография»;



6.  Создадим поле со списком дисциплин, из которых можно будет выбрать только одну из перечисленных в словаре дисциплину. Для создания поля со списком нужно воспользоваться элементом «Мастер», обеспечив тем самым автоматизацию настройки свойств управляющих  элементов в форме. После этого активизируем элемент «Поле со списком» и в качестве источника данных для списка указывается таблица «Дисциплины», из которой берем наименование соответствующей дисциплины;

7. Таким же способом построим список групп из таблицы «Группы»;

8.  Для ввода оценки воспользуемся элементом «Переключатель», сделав подписи и соответствующие числа под возможными состояниями переключателя (Отлично 5, Хорошо - 4, Удовлетворительно – 3, Неудовлетворительно – 2).

9.  Обеспечим ввод бинарного поля «Досрочная сдача» с помощью элемента «Флажок». Этот элемент не поддерживается мастером, поэтому необходимо установить его  свойства: в графе «Данные» указывается поле «Досрочная сдача» из таблицы «Абитуриенты»;

10. Для наглядности и улучшения дизайна можно использовать и другие элементы, например, линии, прямоугольники, раскрашивание объектов и другое. В результате выполнения последовательности вышеперечисленных шагов получилась следующая форма, представленная на рис.17.

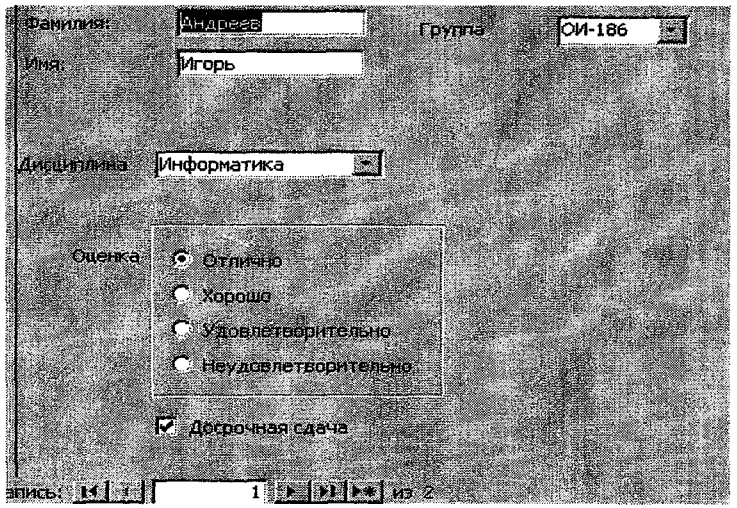


Рис. 17. Форма для ввода данных об абитуриентах

4.4.4. Запросы

Запросы служат для отбора или фильтрации набора данных. Они позволяют выбрать из базы только необходимую информацию, т.е. ту, которая соответствует определенному критерию и нужна для решения конкретных задач. Выбранные записи образуют динамический набор, который может изменяться вместе с данными в таблицах. Запросы, созданные с помощью конструктора запросов, называют QBE-запросами (Query by Example – запросы по образцу). Существуют еще так называемые SQL-запросы (Structured Query Language – структурированный язык запросов), написанные на специальном языке запросов SQL.

Access включает в себя пять основных категорий запросов:

- **Запрос с вычисляемыми полями**. В таком запросе появляется новое поле, которого нет в исходных таблицах, но значение которого вычисляется с помощью некоторого арифметического выражения над известными полями исходных таблиц.
- **Перекрестный запрос**. Отображает результаты статистических расчетов в виде двумерной матрицы.

- **Запрос на выборку (многотабличные запросы).** Извлекает данные из одной или нескольких таблиц на основе заданных условий отбора записей.
- **Итоговый запрос.** Этот тип запросов используется для группировки записей и вычисления сумм, средних значений, максимального значения и т.д.
- **SQL-запрос.** К такому запросу прибегают, когда его невозможно создать в режиме конструктора. Он пишется на специальном языке запросов SQL и обеспечивает очень сложные вычисления и обработку данных.

ЗАПРОС С ВЫЧИСЛЯЕМЫМИ ПОЛЯМИ

Можно задать вычисления над любыми полями таблицы и сделать вычисляемое значение новым полем в наборе значений. В выражениях можно использовать следующие операторы (см. табл.6):

Таблица 6

Оператор	Назначение оператора
+	Складывает два арифметических выражения.
-	Вычитает из первого арифметического выражения второе.
*	Перемножает два арифметических выражения.
/	Делит первое арифметическое выражение на второе.
\	Округляет два арифметических выражения до целого значения и делит первое число на второе, результат округляется до целого.
^	Выводит первое арифметическое выражение в степень, задаваемую вторым арифметическим выражением.
MOD	Округляет оба арифметических выражения до целых значений, делит первое число на второе и возвращает в качестве результата остаток.
&	Создает текстовую строку как результат присоединения второй строки к концу первой. Если один из операндов является числом, MS Access перед проведением сцепления преобразует его в строку символов.

Операторы, используемые для построения выражения

При построении выражения СУБД Access предлагает использовать утилиту, называемую «Построитель выражений». Предположим, необходимо вычислить, на какую сумму не продан товар в БД «Товар» (таблица 7). В выражение для ее вычисления должны входить следующие поля: стоимость товара, количество и сколько товара продано.

Таблица 7

Товар	Стоимость	Количество	Продано
Ручка	10,00руб.	100	100
Линейка	6,00руб.	200	190
Карандаш	9,00руб.	150	150
Блокнот	56,50руб.	50	35
Тетрадь	13,70руб.	500	450

БД «Товар»

Для использования построителя выражения необходимо выполнить следующую последовательность шагов:

1. Щелкнуть мышью по пустому полю бланка QBE.
2. Кнопка «Построить» на панели инструментов СУБД Access откроет окно «Построитель выражений».
3. В верхней части этого окна (рис. 18) расположена пустая область ввода, в которой создается выражение. Выражение можно ввести самостоятельно, но легче использовать различные кнопки операторов, расположения под областью ввода. В нижней части окна расположены 3 списка, которыми можно воспользоваться, чтобы найти необходимые для создания выражения имена полей и функций. В связи с тем, что нужны поля из таблиц «Товар», надо в левом списке выбрать папку «Таблицы», и раскроется список имеющихся имен таблиц. В нем находим «Товар». Во втором поле откроются все поля этой таблицы. Для того чтобы рассчитать полную сумму товаров надо начать со значения «Количество». Выбираем это поле и щелкаем мышью по кнопке: «Добавить», чтобы поместить имя поля в область ввода. Access вставит в область ввода выражение ([Товар]![Количество]).

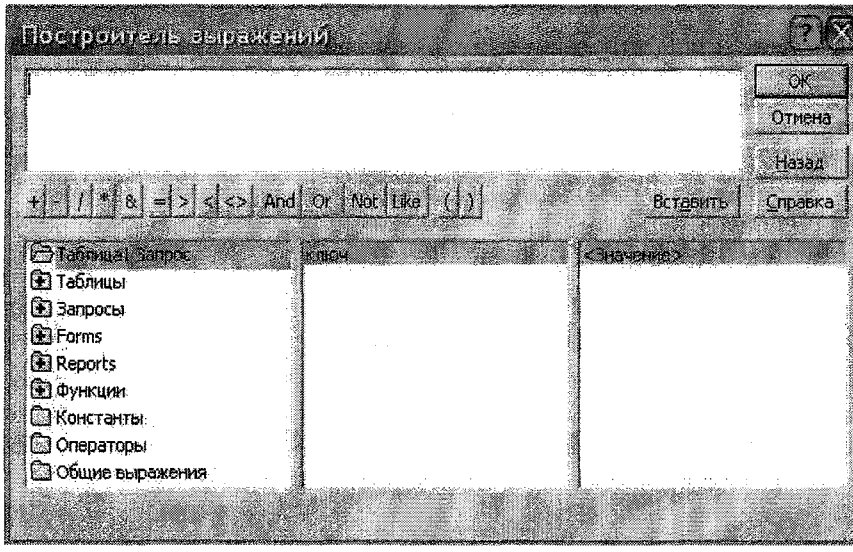


Рис. 18. Вид экрана построителя выражений

При создании выражения необходимо помнить о двух правилах: во-первых, все имена объектов Access должны быть заключены в квадратные скобки. Если имя не содержит пробелов, можно не включать его в квадратные скобки, в этом случае Access сам правильно расставит их. Во-вторых, построитель выражений не знает, будут ли использоваться имена другой таблицы в запросе и не будут ли некоторые из этих таблиц иметь имена полей, совпадающие с именами уже выбранных полей. Во избежание конфликтов следует использовать полное имя поля, помещая перед именем полня имя таблицы. При этом перед именем ставится знак «!», разделяющий имена объектов. Затем необходимо вычесть значение «Продано» товаров и умножить на их стоимость. Это вычисляется следующим образом: [Общее количество товаров – Количество проданных] · Стоимость товара.

В «Построителе выражений» выбираем кнопку со знаком «-«, чтобы вставить его в выражение. Если пользователь ошибся, то, чтобы отменить последнее действие, всегда можно использовать кнопку «Отмена». Затем необходимо выбрать поле «Продано» и заключить разность в скобки. Следующим этапом выбираем действие «,» и знак вставляется в выражение. После этого выбираем поле «Стоимость» и «Добавить» и вставляем его в выражение. Таким образом, получаем выражение, вычисляющее общую стоимость непроданного товара (рис. 19).

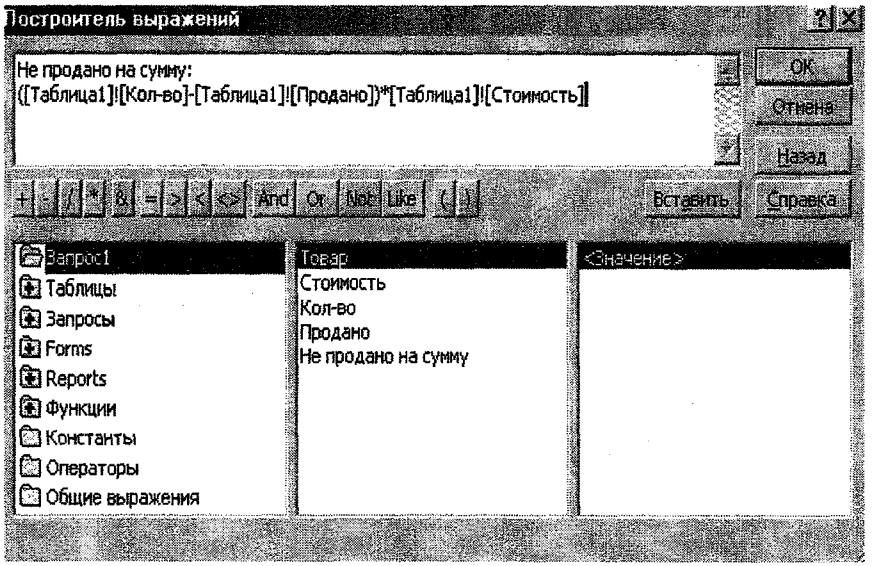


Рис. 19. Построенное выражение

Если есть выражение, которое должно находить сумму или разность раньше, чем произведение или частное, хотя они стоят после этих знаков, можно использовать скобки, которые также присутствуют на арифметической панели инструментов.

После полной записи выражения полученный результат будет перенесен в бланк QBE. Когда откроется запрос в режиме таблицы, он будет выглядеть следующим образом:

Товар	Стоимость	Количество	Продано	Не продано на сумму
Ручка	10,00руб.	100	100	0,00р.
Линейка	6,00руб.	200	190	60,00р.
Карандаш	9,00руб.	150	150	0,00р.
Блокнот	56,50руб.	50	35	847,50р.
Тетрадь	13,70руб.	500	450	685,00р.

Перекрестные запросы

Этот тип запросов разберем на примере БД «Канцелярские товары» (см. табл. 8).

Таблица 8

Товар	Месяц	Количество	Продано	Продавец
Ручка	Май	1000	500	Иванов
Линейка	Апрель	1000	800	Магазин №2
Линейка	Май	500	500	Магазин №10
Тетрадь	Май	1200	700	Магазин №36
Ручка	Февраль	800	750	Петров
Ручка	Март	2000	1800	Магазин №10
Тетрадь	Апрель	1200	1000	Иванов

БД «Канцелярские товары»

Access поддерживает специальный тип итоговых запросов, называемый перекрестным запросом. Такой запрос позволяет увидеть вычисляемые значения в виде перекрестной таблицы, напоминающий электронные таблицы.

Для БД «Канцелярские товары» перекрестная таблица будет выглядеть следующим образом (см. табл. 9):

Таблица 9

Товар	Продано за февраль	Продано за март	Продано за апрель	Продано за май	Сумма
Ручка	750	1800		500	3800
Линейка			800	500	1500
Тетрадь			100	700	2400

Перекрестная таблица к БД «Канцелярские товары»

Для построения перекрестного запроса необходимо:

1. Выбрать в качестве источника нужную таблицу в окне БД, затем активизировать кнопку «Новый запрос» на панели инструментов.
2. В открывшемся диалоговом окне «Создание запроса» выбрать «Запрос», затем выбрать тип запроса «Перекрестный».

Access добавит в бланк QBE новую строку «Перекрестная таблица». В этой строке для любого поля перекрестного запроса может быть выбрана одна из четырех установок:

а) заголовки строк;

б) заголовки столбцов;

в) значение (выводимое в сетке перекрестного запроса);

г) не выводить.

Для перекрестного запроса должно быть определено, по крайне мере, одно поле в качестве заголовка строки, одно поле в качестве заголовка столбца и одно поле значений. Любое поле, являющееся заголовком строки или столбца должно иметь в строке «Групповая операция» установку «Группировка». Для поля, которое в бланке QBE имеет установку «Значение», необходимо выбрать одну из групповых функций (sum, max, и др.).

Бланк запроса для верхнего примера перекрестного запроса выглядит следующим образом (рис.20).

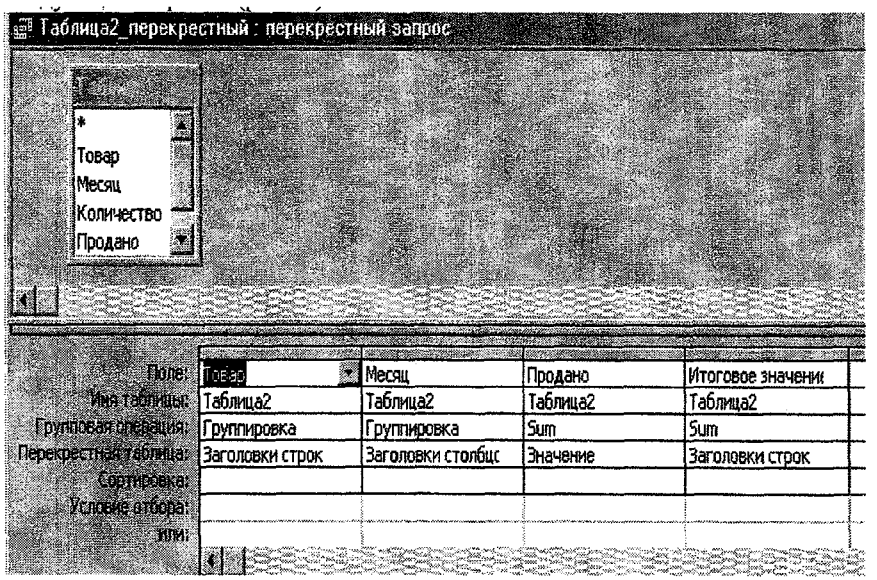


Рис. 20. Бланк перекрестного запроса

Многотабличные запросы

Для построения многотабличного запроса, таблицы, к которым обращается запрос, должны быть связаны между собой. Поэтому вначале следует установить связь между таблицами. Для этого выбираем вкладку «Запрос» а «Создать» а «Новый запрос». Access открывает диалоговое окно «Добавление таблицы». Это окно позволяет выбрать таблицы и запросы, которые будут связаны между собой и являются базовыми для нового запроса.

Пусть даны табл.10 «Изделие» и табл. 11 «Поставщик»:

Таблица 10

Изделие

Изделие	Цена	Цвет
Болт	2 руб.	Синий
Винт	1 руб.	Зеленый
Гайка	0,50коп.	Красный

Таблица 11

Поставщик

Изделие	Поставщик	Количество
Болт	Иванов	500
Винт	Петров	1000
Шуруп	Сидоров	1500
Гайка	Антонов	1000

Если правильно определена связь между таблицами, верхняя панель запроса будет выглядеть следующим образом: две таблицы будут соединены линией «? - 1» Access сам связывает таблицы по одинаковым полям.

В бланке QBE включаются поля из таблицы «Изделие» и поля из таблицы «Поставщик», которые будут нужны пользователю для создания общего запроса. Пусть это будут следующие поля (рис.21):

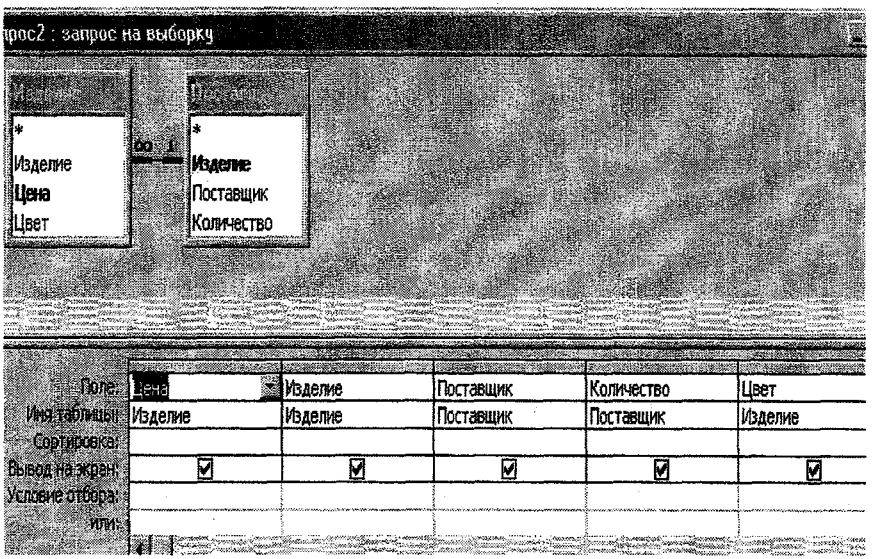


Рис. 21. Конструктор многотабличного запроса

После выполнения запроса результатом будут следующие поля:

Изделие	Поставщик	Количество
Болт	Иванов	500
Винт	Петров	1000
Шуруп	Сидоров	1500
Гайка	Антонов	1000

Такой запрос будет содержать данные для тех изделий, которые входят в таблицу «Изделие» и которые не входят в таблицу «Поставщик»(см табл.11). Такой тип многотабличных запросов называется *симметричным объединением*. Симметричное объединение – объединение нескольких таблиц, которые в общей таблице содержат только те записи, которые пересекаются (являются общими) по какому-либо атрибуту во всех связанных таблицах. В рассмотренном примере, общий атрибут – изделие, а записи, соответствующие общему атрибуту, - это болт, винт и гайка.

Если необходимо видеть те записи, которые не вошли в запрос (в данном примере – шуруп), то для этого строят *внешнее объединение.* Внешнее объединение – объединение нескольких таблиц

не по общему атрибуту, а так, как этого хочет пользователь.

Чтобы создать внешнее объединение, нужно изменить параметры объединения:

1. Открывается уже созданный запрос в режиме конструктора.
2. Дважды щелкается мышью по линии связи между таблицами в верхней панели окна запроса.
3. Открывается диалоговое окно «Параметры объединения». По умолчанию в этом окне параметров объединения устанавливается параметр №1. Для пользователя имеются еще два параметра (рис.22).

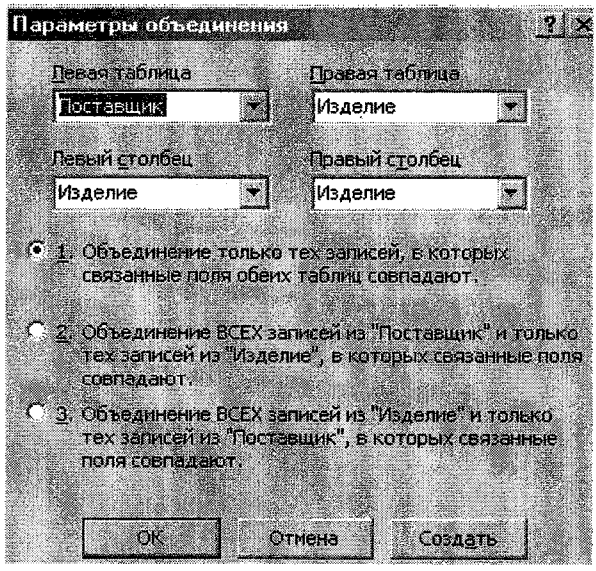


Рис. 22. Параметры объединения

4. Чтобы вывести в запросе дополнительно изделие шуруп, необходимо выбрать параметр №2. Тогда стрелка линии, связывающая две таблицы, будет направлена от «Поставщик» к «Изделие» и будет показывать внешнее объединение (рис. 23), хотя в результирующей таблице не будет цены и цвета у шурупа, т. е. эти поля будут пустыми.

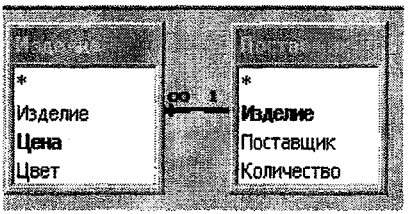


Рис. 23. Внешнее объединение

Результат внешнего объединения будет следующий:

Цена	Изделие	Поставщик	Количество	Цвет
2,00руб.	Болт	Иванов	500	Синий
1,00руб.	Винт	Петров	1000	Зеленый
0,50руб.	Гайка	Антонов	1000	Красный
		Сидоров	1500	

В многотабличном запросе также можно задавать различные условия отбора с помощью операторов <, >, <>, Like, In, Between и т.п., групповых операторов sum, max, min, avg и т.п., строить перекрестные запросы. В этом случае все операторы и групповые операции будут выполняться для всех связанных таблиц, как для одной объединенной таблицы по общему атрибуту или объединенной по внешними связями.

Итоговые запросы

Иногда необходимо значение не любой троки таблицы, а сразу итог: значение по группам данных, например, общая сумма продаж для всех видов товаров. Для вычисления в запросе итоговых значений кнопка а Групповые операции на панели инструментов конструктора запросов отобразит в бланке QBE строку «Групповая операция».

Access предоставляет 8 функций получения итогов, описание функций представлено ниже (см. табл. 12):

Таблица 12

Итоговые функции

Функция	Назначение
SUM	Сумма всех значений заданного поля в любой группе. Операцию можно использовать только для числовых или денежных полей.
AVG	Вычисление среднего арифметического всех данных поля в любой группе. Ее можно использовать только для числовых или денежных полей.
MIN	Возвращает наименьшее значение, найденное в этом поле внутри любой группы. Для числовых полей возвращается наименьшее число, для текстовых – наименьшее из символьных значений. Пустые поля – игнорируются.
MAX	Возвращает наибольшее значение, найденное в этом поле внутри любой группы. Для числовых полей возвращается наибольшее число, для текстовых – наибольшее из символьных значений. Пустые поля – игнорируются.
FIRST	Возвращает первое значение этого поля в группе.
LAST	Возвращает последнее значение этого поля в группе
StDev	Среднеквадратичное отклонение от среднего значения поля.
VAR	Дисперсия значений поля.

Пусть дана таблица 13 «Товары»

Таблица 13

Товары

Товар	Поставщик	Продано	Стоимость	Всего	Налог
Ручка	Поставщик	1000	6,00руб.	1500	0,60руб.
Тетрадь	Магазин № 1	200	2,50руб.	200	0,25руб.
Ручка	Магазин № 2	500	9,00руб.	550	0,90руб.
Тетрадь	Магазин № 10	1200	15,50руб.	1800	1,55руб.
Линейка	Иванов	600	8,70руб.	800	0,87руб.
Ручка	Магазин № 300	400	60,00руб.	550	6,00руб.

Создадим итоговой запрос для таблицы «Товары». Для этого надо выполнить следующую последовательность действий:

1. Щелкнуть по кнопке «Групповые операции» на панели инструментов. В бланке QBE появится новая строка «Групповая операция». Сохранить группировку для «Товар».
2. Выберем групповые функции для полей: стоимость - Avg, налог - Avg, всего – max, продано – sum.
3. В условии отбора удаляем все критерии.
4. Таким образом, получим следующий бланк запроса (рис. 24):

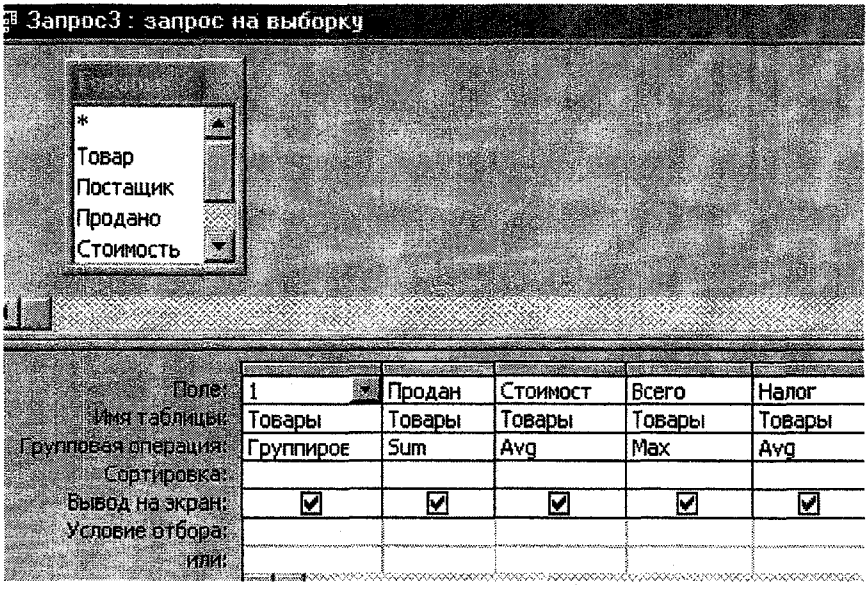


Рис. 24. Конструктор итогового запроса

Результат выполнения группового запроса для таблицы 13 «Товары»:

Товар	Sum-Продано	Avg-Стоимость	Max-Всего	Avg-Налог
Линейка	600	8,70руб.	800	0,87руб.
Ручка	1900	25,00руб.	1500	2,50руб.
Тетрадь	1400	9,00руб.	1800	0,90руб.

В раскрывающемся списке строки «Групповая операция» бланка QBE также имеется установка «Выражение». Ее можно выбрать, если нужно ввести выражение в строке «Поле», в котором имеются одна или несколько групповых функций. Например, Вам нужно вычислить размах значений поля в определенной группе, тогда надо ввести: MAX([Налог])-MIN([Налог]), затем выбрать значения, формирующие группы.

Выбор значений, формирующих группы.

Возможно, пользователю не хочется включать некоторые записи в группы итогового запроса. Чтобы в эти группы включались

только определенные записи, надо добавить в бланк QBE поля или поле, которые будут использоваться в фильтре. Для создания фильтра необходимо выполнить следующую последовательность шагов:

1. Выбрать установку «Условие» и строку «Групповые операции».
2. Удалить флажок «Вывод на экран».
3. Ввести условие отбора для данного поля.

Например, для таблицы 13 можно ограничить множество выбранных записей только продавцами из магазинов. Таким образом, бланк QBE будет выглядеть следующим образом (рис. 25).

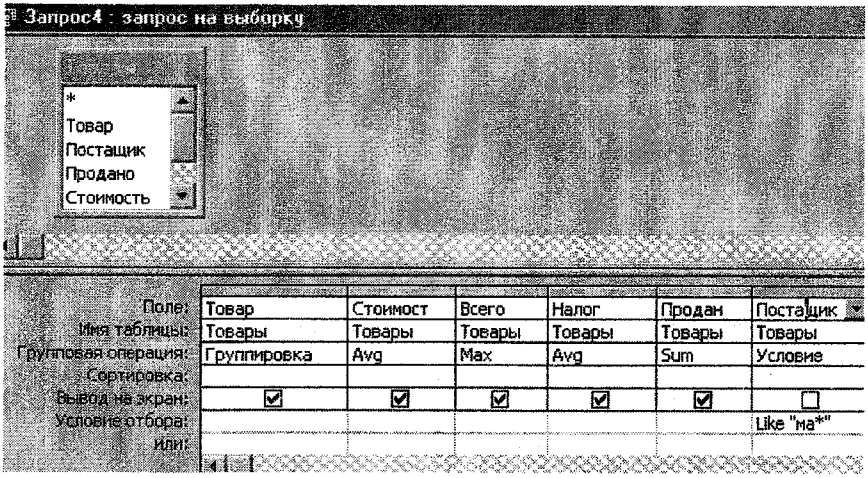


Рис. 25. Конструирование итогового запроса с фильтром

При выполнении такого запроса получится итог, вычисленный только для товаров из магазинов:

Товар	Avg-Стоимость	Max-Всего	Avg-Налог	Sum-Продано
Ручка	34,50руб.	550	3,45руб.	900
Тетрадь	9,00руб.	1800	0,90руб.	1400

SQL- запросы

Язык Structured English Query Language (SEQUEL - SQL) – структурный английский язык запросов.

Запрос SQL – такое название присвоено тем запросам Access, которые нельзя создать с помощью конструктора или мастера. Эти запросы могут быть созданы только с помощью языка SQL. К таким запросам относятся подчиненные запросы, запросы к серверу, запросы объединения и управляющие запросы. Далее проводится выборка данных и таблиц.

Выборка данных и таблиц.

Запросы этого вида не изменяют информации в исходных таблицах, а лишь показывают ее пользователю. Эти запросы конструируются на базе команды SELECT.

Простейшая инструкция поиска информации имеет вид:

SELECT <список полей>

FROM (имена таблиц),где SELECT – ключевое слово, оно сообщает БД, что инструкция SQL является запросом–выборкой;

<список полей> - перечисление полей через «,» которые надо вывести в новой таблице. Символ «*» на этом месте обозначает, какие таблицы или запросы содержит поля, приведенные в инструкции SELECT. Код инструкции языка SQL оканчивается знаком «;», что означает завершение запроса SQL.

Для устранения дублирующих строк после SELECT необходимо исполнить предикат DISTINCT.

Пример.

Дана таблица «Заказов» (табл. 14), надо выяснить: кто из продавцов, что-либо продал, и сколько повторяющихся продавцов.

Таблица 14

Заказы

№	Дата продажи	Сумма	Покупатель	Продавец
301	10.03.02	15874	201	105
302	10.03.02	23654	205	106
303	10.03.02	10002	204	102
304	10.03.02	521466	206	105
305	10.03.02	12369	209	102

Вариант 1:

```
SELECT Продавец
FROM Заказы;
```

На выходе получаем:

Продавец
105
106
102
105
102

Вариант 2:
SELECT DISTINCT Продавец
FROM Заказы;

Результат:

Продавец
105
106
102

Выборка по условию.
Формат команды выборки по условию следующий:
SELECT *
FROM Покупатели
WHERE условие

Пример.
Пусть дана табл. 15.

Таблица 15

Покупатели

Номер	Название	Город	Рейтинг	Продавец
201	Сидоренко	Москва	200	305
202	Власов	СПб	300	306
203	Щукин	Москва	500	407
204	Романов	Киев	100	408
205	Курдюков	Киев	300	502

Используя данные из таблицы «Покупатели», найти всех покупателей, проживающих в Москве. Инструкция SQL такого запроса будет следующей:

```
SELECT Название, Город
FROM Покупатели
WHERE город = «Москва»;
```

Результат:

Название	Город
Сидоренко	Москва
Щукин	Москва

Пример.

Используя данные из табл. 15 «Покупатели», найти всех продавцов с рейтингом больше 200.

```
SELECT *
FROM Покупатели
WHERE Рейтинг > 200;
```

Результат:

Номер	Название	Город	Рейтинг	Продавец
202	Власов	СПб	300	306
203	Щукин	Москва	500	407
205	Курдюков	Киев	300	502

В предложении WHERE можно использовать реляционные и булевы операторы. Реляционные операторы: =, >, <, >=, <=, <>. Булевы операторы: AND, OR, NOT, XOR.

Пример.

Из табл. «Покупатели» найти покупателей из г. Киева с рейтингом >100

```
SELECT *
FROM Покупатели
WHERE Город = «Киев» AND рейтинг > 100";
```

Результат:

№	Название	Город	Рейтинг	Продавец
205	Курдюков	Киев	300	502

Операторы IN, BETWEEN, LIKE.

Пример.

Используя данные табл. «Покупатели», найти всех покупателей приживающих в городах Москва и СПб.

```
SELECT *
FROM Покупатели
WHERE город IN('Москва', 'СПб');
```

Пример.

Найти всех покупателей, рейтинг которых начинается со 100 и заканчивается 300.

```
SELECT *
FROM Покупатели
WHERE рейтинг BETWEEN '100' and '300';
```

Пример.

Предположим, мы точно не знаем фамилию покупателя. Требуется найти список всех покупателей, фамилии которых начинаются с 'Ром'.

```
SELECT *
FROM Покупатели
WHERE название LIKE 'Ром*';
```

Оператор GROUP BY

Оператор GROUP BY позволяет найти в БД подмножество значений отдельного поля в терминах другого поля и применить функцию агрегирования не ко всему множеству, а лишь к некоторому подмножеству записей.

Пример.

Из табл. 14 «Заказы» найти наибольшее заказы из тех, что получил любой продавец.

```
SELECT Продавец, MAX(Сумма) AS [Наибольший заказ]
```

```
FROM Заказы
GROUP BY Продавец;
```

На выходе:

Продавец	Наибольший заказ
106	23654
105	521466
102	12369

Замечание: если имя поля состоит из 2-х отдельных слов, нужно использовать квадратные скобки для идентификации такого поля.

Условие HAVING.

Этот оператор работает только с групповыми запросами, и если WHERE определяет записи, которые должны быть выбраны, то HAVING устанавливает, какие записи, сгруппированные посредствам параметра GROUP BY, должны отображаться на экран.

Пример.

Из табл. «Заказы» найти наибольшие заказы, превышающие 20000 руб., которые сделаны у разных продавцов.

```
SELECT продавец, MAX(Сумма) AS [наибольший заказ]
FROM Заказы
GROUP BY Продавец
HAVING MAX(Сумма) > 20000;
```

Результат:

Продавец	Наибольший заказ
105	521466
106	23654

Упорядочивание значений полей.

Для упорядочивания значений полей используется параметр ORDER BY. По умолчанию осуществляется сортировка по возрастанию. Явно она может задаваться ключевым словом ASC. Для выполнения сортировки по убыванию – DESC.

Пример.

Упорядочить информацию таблицы «Заказы» по продавцам, для любого продавца в порядке убывания суммы заказа.

```
SELECT *
FROM Заказы
ORDER BY Продавец, Сумма DESC;
```

Результат:

№	Дата	Сумма	Покупатель	Продавец
305	10.03.02	12369	209	102
303	10.03.02	10002	204	102
304	10.03.02	521466	206	105
301	10.03.02	15874	201	105
302	10.03.02	23654	205	106

Пример.

Из табл. «Заказы» найти наибольшие заказы, превышающие 20000 руб., которые сделаны у разных продавцов. Упорядочить заказы в порядке убывания сумм.

```
SELECT Продавец, MAX(Сумма) AS [Наибольший заказ]
FROM Заказы
GROUP BY Продавец
HAVING MAX(Сумма) > 20000
ORDER BY MAX(Сумма) ASC
```

Результат:

Продавец	Наибольший заказ
106	23654
105	521466

Соединение таблиц.

Одной из важнейших черт запросов SQL является возможность определить связь между таблицами и работать с ними, как с единым целым.

Операция JOIN – самая мощная операция реляционных БД. Если работа проходит с двумя и более таблицами, то надо использовать полный синтаксис написания полей: <имя таблицы>, <имя поля>

Пример.

Даны табл. 16 «Покупатели» и табл. 17 «Продавцы». Найти пары покупателей и продавцов, живущих в одном городе.

Таблица 16

Покупатели

Покупатель	Город	Телефон
Орлов	Москва	152-12-85
Петров	СПб	365-20-40
Кузьмин	Киев	102-36-67

Таблица 17

Продавцы

Продавец	Город	Рейтинг
Иванов	Москва	300
Вавилов	Киев	100
Кондратьев	Москва	200

Связь между таблицами устанавливается с помощью запятой при их перечислении после параметра FROM.

```
SELECT Покупатель, Продавец, Покупатель.Город
FROM Покупатели, Продавцы
WHERE Покапатель.Город = Продавец.Город;
```

Результат выполнения запроса:

Покупатель	Продавец	Город
Орлов	Иванов	Москва
Орлов	Кондратьев	Москва
Кузьмин	Вавилов	Киев

Перебор осуществляется (n.m) раз, где n – количество записей первой таблицы, а m – количество записей второй таблицы. Воздается множество всех возможных комбинаций строк и для любой проверяется условие: Покупатели.Город = Продавец.Город из предложения WHERE.

Соединение таблиц, основанное на условиях, в предложении WHERE называются «эквисоединение таблиц».

Пример.

Пусть необходимо найти все пары покупателей и продавцов, живущих в одном городе, но рейтинг продавцов должен превышать 100.

```
SELECT Покупатель, Продавец, Покупатели.Город
FROM Покупатели, Продавцы
WHERE Покупатели.Город = Проавец.Город AND Рейтинг > 100
```

Результат:

Покупатель	Продавец	Город
Орлов	Иванов	Москва
Орлов	Кондратьев	Москва

Симметричное объединение таблиц.

Формат команды: FROM табл. 1 INNER JOIN табл. 2 ON табл. 1.Поле 1 = табл. 2.Поле 2;

Здесь табл. 1, табл.а 2 – имена таблиц, записи которых подлежат объединению; поле 1, поле 2 – имена полей, которые должны быть объединены. Если эти поля не являются числовыми, то должны иметь одинаковый тип данных и содержать данные одного рода, однако могут иметь разные имена. Предложение ON описывает условие объединения таблиц.

Пример.

Даны две табл. «Покупатели» и «Продавцы» с соответствующими полями:

«Продавец» «Продавцы»

Покупатель
Город
Телефон

Продавец
Город
Рейтинг

Учитывая сходство данных, необходимо найти все пары покупателей и продавцов, живущих в одном городе, но рейтинг продавцов должен превышать 100.

```
SELECT Покупатель, Продавец, Покупатели.Город
FROM Покупатели INNER JOIN Продавцы
        ON Покупатели.Город = Продавцы.Город
WHERE Рейтинг > 100;
```

Результат симметричного объединения:

Покупатель	Продавец	Город
Орлов	Иванов	Москва
Орлов	Кондратьев	Москва

Надо отметить, что результат тот же, что и в предыдущем примере, но время выполнения меньше, поскольку для любой строки таблицы «Покупатели» просматриваются только те строки таблицы «Продавцы», для которых выполняется условие «Покупатели.Город = Продавцы.Город».

Внешнее объединение.

Формат команды: FROM Табл. 1 LEFT JOIN Табл. 2 ON Табл. 1.Поле 1 = Таблица 2. Поле 2.

Здесь операция LEFT(RIGHT) указывает, из какой таблицы брать все записи. Если LEFT, то нужно брать все записи таблицы, расположенной слева от JOIN, если RIGHT – из таблицы справа.

Пример.

Пусть дана табл. 15, рассмотренная в верхних примерах «Заказы», в ней продавец предусмотрен кодом, поэтому нужен справочник (см. табл. 18), в котором перечислены фамилии всех продавцов фирмы с указанием их кода.

Таблица 18

Справочник «Продавцы»

Код	Продавец	Город	Рейтинг
101	Иванов	Москва	300
102	Петров	СПб	500
103	Сидоров	Москва	100
104	Смирнов	СПб	300
105	Андреев	Киев	100
106	Алексеев	Киев	200

Для того чтобы по полю «Продавец» из таблицы «Заказы», найти строку в таблице «Справочник» нужно связать это поле с полем «Код» в табл. «Продавцы», как показано на рис. 26.

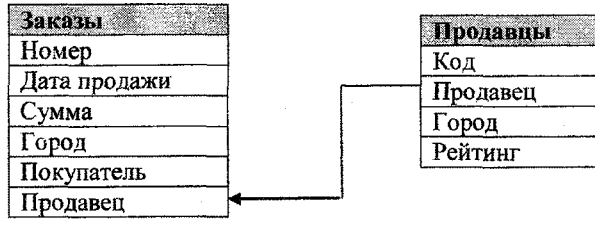


Рис. 26. Схема данных таблиц «Заказы» и «Продавцы»

Используя схему данных найти для любого продавца количество оформленных им заказов и суммарную стоимость этих заказов.

```
SELECT Продавцы.Продавец, Count(Заказы.Покупатель) [Количество заказов], Sum(Заказы.Сумма) AS [Всего на сумму]
FROM Продавцы LEFT JOIN Заказы ON Продавцы.Код = Заказы.Продавец
GROUP BY Продавцы.Продавец;
```

Результат:

Продавец	Количество заказов	Всего на сумму
Петров	2	(12368+1002)
Андреев	2	(521466+158740)
Алексеев	1	23654

Аргумент DISTINGCTROW команды SELECT

Аргумент DISTINGCTROW следует после слова SELECT и используется, если в запросе две или более таблицы. В тех случаях, если надо оставить полностью дублирующие друг друга записи, но не дублирование значений отдельных полей, выводимых на экран, и сделать выходную таблицу динамическим набором данных, используется аргумент DISTINGCROW. Динамический набор отличается от статического той особенностью, что его можно корректировать и все изменения передаются в исходные таблицы. Характерным признаком динамического набора данных является наличие в последней строке признака новой записи, правда не во всех случаях.

Пример

Используя связанные по полю «Покупатель» табл. 19 «Покупатели» и таблю 20 «Продажи», построить динамический набор данных «Покупатели, сделавшие покупки», позволяющий «вести» таблицы «Покупатели» и «Продажи». Слово «вести» означает добавление новой записи в конец таблицы, удаление записи или изменение полей.

Таблица 19

Покупатели

Покупатель	Телефон
Петренко	165-96-40
Драгунов	541-30-85
Николаев	145-63-82
Кудаев	206-35-41

Таблица 20

Продажи

Покупатель	Дата	Сумма
Драгунов	12.05.02	5200
Кудаев	24.05.02	18000
Кудаев	15.07.02	600
Дрогунов	30.08.02	8000
Дрогунов	27.10.02	7.52

SELECT DISTINGCTROW Покупатели.Покупатель, Покупатели.Телефон
FROM Покупатели INNER JOIN Продажи ON
Покупатель.Покупатель = Продажи.Покупатель;

Результат запроса «Покупатели, сделавшие покупку»:

Покупатель	Телефон
Драгунов	541-30-85
Кудаев	206-55-41

Вложение запросов:

Одни запросы могут управлять другими запросами. Это можно сделать, разместив один запрос внутри другого, а именно – внутри предиката, используя выходные данные подчиненного запроса для определения истинности или ложности предиката.

Пример.

Требуется в табл. 21 «Покупатели» найти все записи, относящиеся к продавцу, фамилию которого нужно указать как параметр и найти по коду в табл. 22 «Продавцы».

Таблица 21

Покупатели

Таблица 22

№	Покупатель	Город	Продавец
2001	Иванов	Москва	1001
2002	Петров	СПб	1003
2003	Сидоров	Москва	1002
2004	Андреев	СПб	1001
2005	Смирнов	Гатчина	1003

Продавцы

Код	Продавец	Город
1001	Смирнов	Москва
1002	Алексеев	СПб
1003	Белов	СПб

```
SELECT Покупатель *
FROM Покупатели
WHERE Покупатель.Продавец = (SELECT Код
                             FROM Продавцы
                             WHERE продавец = [Имя продавцы?] );
```

При выполнении этой программы в первую очередь начнет выполняться подзапрос и появится сообщение «Имя продавца?», в ответ, на которое можно ввести имя одного из продавцов, например, Смирнов. Из табл. 22 «Продавцы» будет найден код 1001 и передан во внешний запрос. После определения кода во внешнем запросе из табл. 21 «Покупатели» будут отображены записи, удовлетворяющие условию: Покупатели.Продавец = 1001.

Результат:

№	Покупатель	Город	Продавец
2001	Иванов	Москва	1001
2004	Андреев	СПб	1001

Замечания:

1. В этом варианте запроса подзапрос (команда SELECT, заключенная в ()) должен вернуть только одно значение поля, иначе будет «отказ».
2. Если в результате подзапроса не было одной строки, то предикат WHERE примет значение UNKNOWN (неизвестно) вместо обычных true и false, следовательно, ни одна из записей возвращена на будет.

Оператор EXISTS

Этот оператор (предикат) используется для оценки события, если в подзапросе есть хоть одна строка. Если строка есть, то оператор возвращает true, если нет ни одной строки, то false.

Пример.

Вывести табл. 21 «Покупатели» предыдущего примера, если хотя бы один из покупателей живет в Москве.

```
SELECT №, Покупатель, Город
```

```
FROM Покупатели
WHERE EXISTS (SELECT *
                From Покупатели
                  Where Город = «Москва»);
```

Результат:

№	Покупатель	Город
2001	Иванов	Москва
2002	Петров	СПб
2003	Сидоров	Москва
2004	Андреев	СПб
2005	Смирнов	Гатчина

Поскольку Иванов и Сидоров живут в Москве, то подзапрос возвращает две строки, оператор EXISTS принимает значение true, и все запросы из табл. «Покупатели» переписываются в выходной набор.

Замечания:

1. EXISTS – это булево выражение, следовательно, его можно комбинировать с любыми другими булевыми выражениями с помощью логических операций AND, NOT, OR.
2. В отличие от предыдущих запросов, EXISTS работает не с одним полем, а со всей строкой. Следовательно, в команде SELECT (в подзапросе), как правило, пишут «*»
3. В рассмотренном выше примере оператор EXISTS считается один раз для первой строки внешнего запроса.

Специальные операторы ANY и ALL.

Оператор ANY (иногда, согласно стандарту SQL, пишут some) берет из подзапроса все значения какого либо поля и оценивает результат сравнения, как true, если среди этих возвращенных значений поля есть хотя бы одно, удовлетворяющее условию.

Пример.

Дано: табл. 21 и табл 22. Найти список продавцов, живущих в одном городе хотя бы с одним из покупателей.

```
SELECT *
FROM продавцы
WHERE город = ANY
```

```
(Select Город
    From Покупатели
    Where Продавцы.Код = Покупатели.Продавец);
```

Результат:

Код	Продавец	Город
1001	Смирнов	Москва
1003	Белов	СПб

Замечания:

1. Любой запрос с ANY можно сделать при помощи оператора EXISTS (обратное утверждение неверно)
2. В запросе ANY можно использовать отношения: <, >, >=, <> и т.д.

Предикат ALL принимает значение true, если любое значение, выбранное в подзапросе, удовлетворяет условию, заданному в предикате внешнего запроса.

Пример.

Выбрать все заказы из табл. 14 «Заказы», превосходящие величину любого из заказов, сделанного 10.03.02.

```
SELECT *
FROM Заказы
WHERE Сумма > ALL
                (Select Сумма
                 From Заказы
                 Where Дата = #03/10/02#);
```

Результат.

№	Дата	Стоимость	Покупатель	Продавец
304	10.03.02	521466	206	105

Замечания:

1. Как правило, оператор ALL используется с неравенствами, так как значение *x*= ALL('*x* равняется всем') получает истину в одном единственно случае, когда в результате выполнения под-

запроса найдено одно значение или несколько значений и каждое из которых равно *x*.

Иногда используют неравенства (т.е. *x* не равен хотя бы одному из найденных значений) *x* <> ALL (‘*x* не равно ни одному из найденных значений’).

4.4.5. Отчеты

Как правило, выбранная из БД информация должна быть представлена в виде распечатки-отчета, оформленного соответствующим образом. Отчеты представляют данные в таком виде, в каком они должны выводиться на печать. В отчете можно группировать и сортировать данные в любом порядке, получать итоговые, средние значения и другие статистические величины, а также помещать в него графические диаграммы. Существует два способа создания отчета: Мастер и Конструктор. Для реализации БД в рамках учебной программы вполне достаточно пользоваться Мастером отчетов, что позволяет упростить процесс его создания, так как с помощью ряда вопросов Мастер автоматически создает макет отчета. Кроме того, Мастер поддерживает все необходимые виды отчетов: с группировкой данных, составной, детальный.

Рассмотрим пример создания отчета на основании табл. 23 «Продажи», в котором сгруппируем данные по датам продажи, для каждой даты по фирмам укажем стоимость продаж, подсчитаем суммарный объем продаж за день и за весь указанный отчетный период.

Таблица 23

Продажи

Фирма	Стоимость	Дата продажи
Аякс	16000	12.06.2004
Кей	13300	12.06.2004
Аякс	20540	12.06.2004
Рамек	30000	14.06.2004
Аврора	23000	14.06.2004

Для создания такого отчета воспользуемся «Мастером отчетов» и выполним следующие шаги:

1. Из базовой таблицы «Продажи» в определенном порядке выбрать поля «Дата продажи», «Фирма», «Сумма».

2. Выбрать уровни группировки, как показано на рис.27.

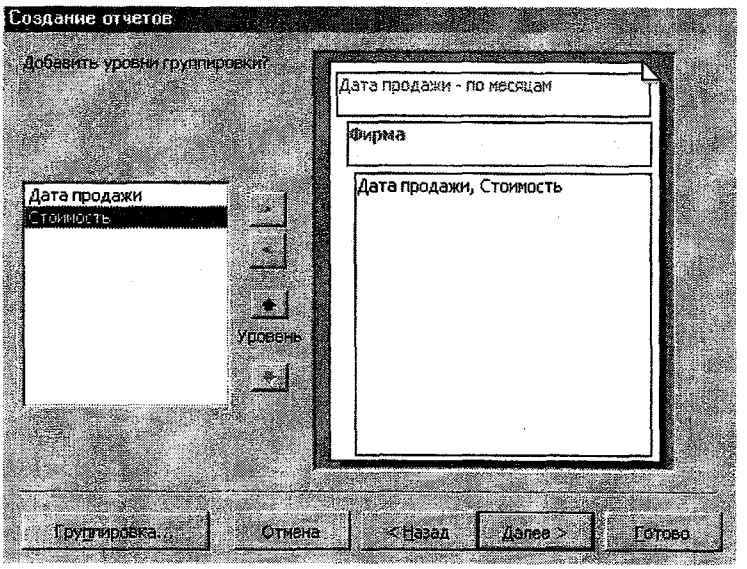


Рис. 27. Задание уровней группировки

3. Задать интервалы группировки с помощью кнопки «Группировка»: для «Даты продажи» – по дням, для «Фирмы» - обычный.
4. Задать итоговую функцию, а именно для поля «Стоимость» - суммирование (SUM), как показано на рис.28.

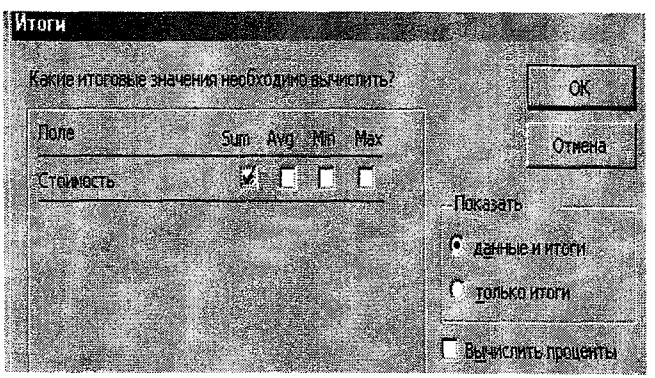


Рис. 28. Окно «Итоги»

5. Определить стиль отчета (оформление) и дать имя отчету, например, «Продажи по дням».

В результате проделанной работы получим отчет, представленный на рис.29.

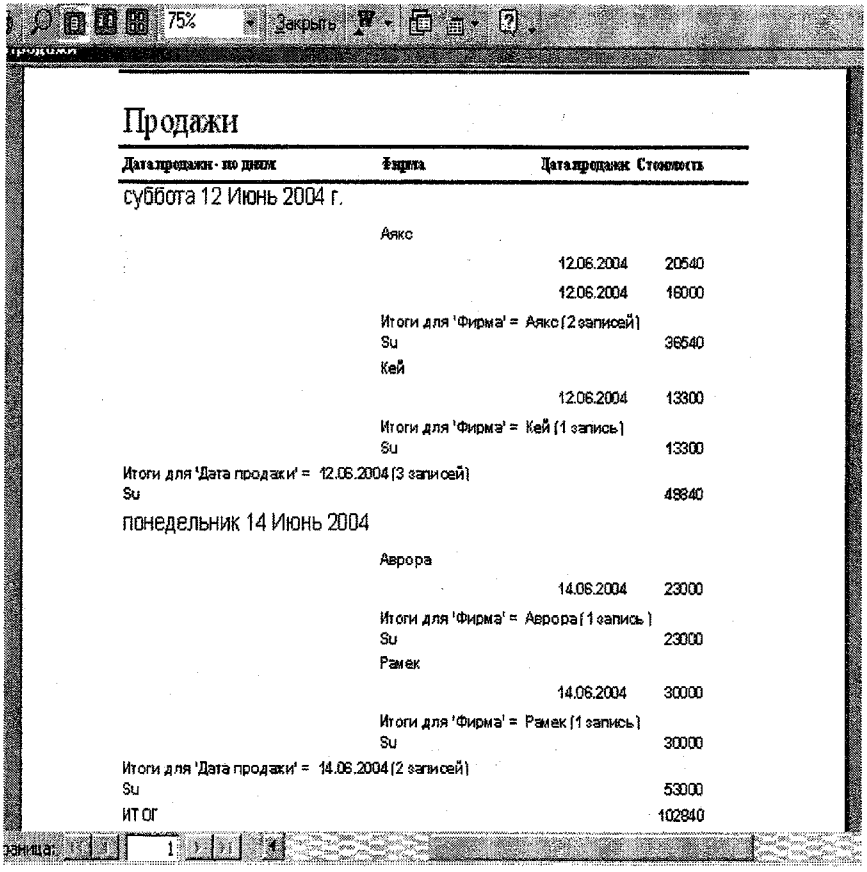


Рис. 29. Отчет с группировкой данных

Для изменения внешнего вида отчета, например, шрифта, цвета, размещения полей, добавления других данных можно перейти в конструктор отчета и с помощью панели элементов доработать отчет до нужного вида (рис.30).

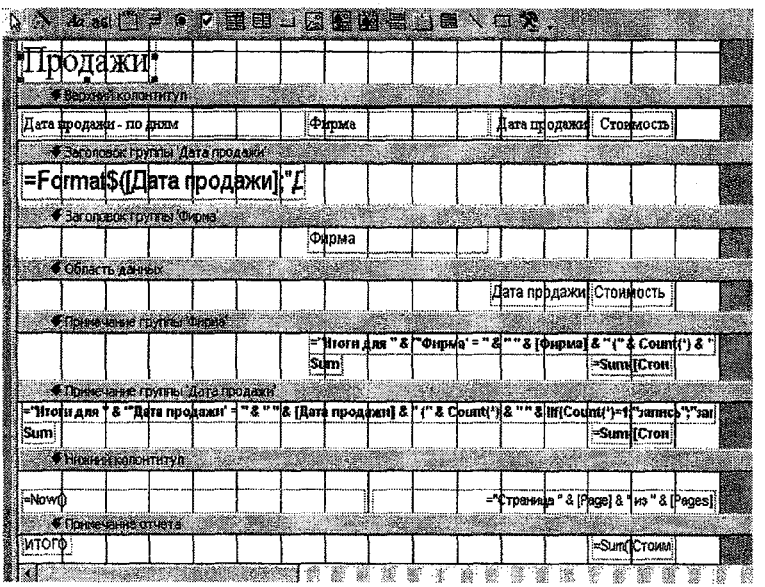


Рис. 30. Конструирование отчета

4.4.6. Построение макросов

Макросы предназначены для автоматизации повторяющихся действий. С помощью макросов можно автоматически реализовать процессы открытия формы, печати отчета, заполнения БД и др. Он представляет собой последовательность операций (макрокоманд), записанных в виде инструкций на специальном языке. С помощью макросов повышается эффективность работы с БД и сокращается время обработки данных, особенно при выполнении часто повторяющихся действий.

В Access предусмотрены специальные средства проектирования и отладки макросов.

Рассмотрим создание макроса «Открыть таблицу» на примере таблицы 23 «Продажи» из предыдущего раздела. Эта команда предназначена для открытия существующей таблицы в одном из режимов (рис. 31):

- таблица;
- конструктор;
- просмотр.

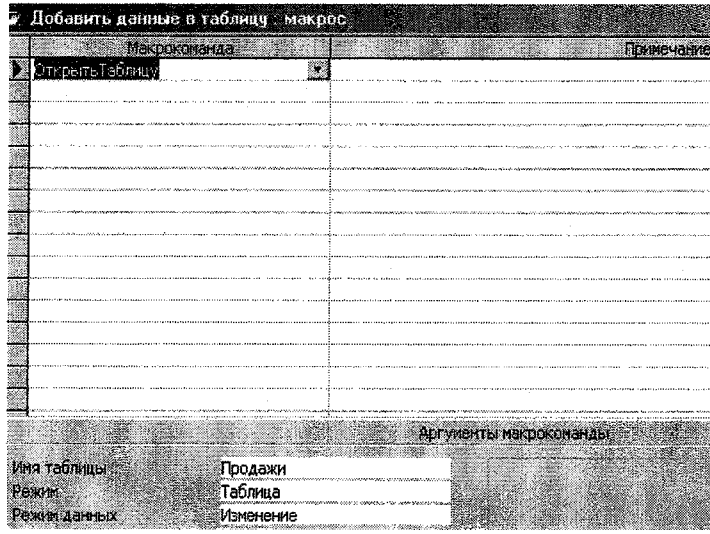


Рис. 31. Окно конструирования макроса

При создании данного макроса необходимо соответствующим образом заполнить аргументы этой макрокоманды:

Имя таблицы – выбираем имя открываемой табл. «Продажи»;

Режим – уточнение того, в каком виде требуется открыть таблицу. Для таблицы предусмотрены три режима: таблица, конструктор, просмотр. Допустим, выберем режим «таблица»;

Режим данных - уточнение того, как можно манипулировать данными, находящимися в таблице: изменять, добавлять, только читать. Допустим, выберем «изменять».

Кроме данной макрокоманды Access поддерживает еще несколько десятков макрокоманд, такие, как открыть форму, просмотреть отчет, закрыть приложение, выход их программы и многие другие. Все они могут быть использованы для создания удобного пользовательского интерфейса, представляющего собой панель с кнопками для работы с БД.

Создадим еще три макроса для того же самого примера из предыдущего раздела: выход из программы «Продажи», открыть отчет «Продажи» с *режимом* – печать и открыть отчет «Продажи» с *режимом* – просмотр. Тогда имея 4 спроектированных макроса можно создать самую простую панель с кнопками для работы с БД «Продажи» (рис.32)

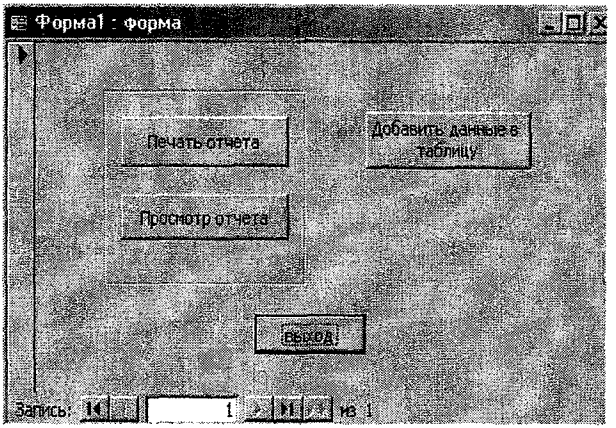


Рис. 32. Панель управления БД «Продажи»

При активизации любой из кнопок, размещенных на панели, будет выполнено соответствующее действие.

Зная, как проектируются объекты СУБД, можно создать базу данных. Варианты заданий для проектирования и администрирования соответствующих баз данных приведены в приложении данного учебного пособия.

Литература

1. Келли Дж. Самоучитель Access 97. – СПб.: Питер, 1999.- 336 с.
2. Робинсон С. Microsoft Access 2000: учебный курс. – СПб.: Питер, 2002.- 576 с.
3. Тихомиров Ю. Microsoft SQL Server 2000: разработка и приложение. – СПб.: BHV, 2003. – 368 с.
4. Берк Пол Д. SQL Server 2000 XML Professional. –М.: Бином, 2002. – 313 с.
5. Кауффман Д. SQL Программирование. – М.: Бином, 2000. – 375 с.

Приложение

Варианты заданий для выполнения лабораторных работ

Задание 1

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога для работников библиотеки. В БД должны храниться сведения об имеющихся в библиотеке книгах, о читателях библиотеки и читальных залах.

Для каждой книги в БД должны храниться сведения: об авторе, названии, годе издания и числе экземпляров, имеющихся в каждом зале библиотеки, а также шифр книги и дата закрепления книги за читателем. Сведения о читателях библиотеки должны включать фамилию, номер телефона и уникальный номер читательского билета. Читатели закрепляются за определенным залом и могут записываться и выписываться из библиотеки. Библиотека имеет несколько читальных залов, которые характеризуются номером, названием и вместимостью. Библиотека может получать новые книги и списывать старые. Шифр книги может изменяться в результате переклассификации.

Библиотекарю могут потребоваться следующие сведения о текущем состоянии библиотеки:

- какие книги закреплены за читателем;
- как называется книга с заданным автором;
- какой шифр у книги с заданным названием;
- когда книга была закреплена за читателем;
- какое число читателей пользуется библиотекой.

Библиотекарь может вносить следующие изменения:

- запись нового читателя в библиотеку;
- списывание старой книги;
- изменение шифра книги.

Необходимо предусмотреть возможность выдачи справки о количестве книг определенного автора в читальном зале и отчета о работе библиотеки в течение месяца (общее количество читателей, количество записавшихся в этот месяц, какие книги и сколько раз были взяты, кто из читателей не брал книг в этот месяц).

Задание 2

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога. В БД должны храниться сведения о комнатных растениях.

Для каждого комнатного растения в БД должны храниться сведения: название растения, к какому кассу оно относится, возраст и благоприятное время пересадки, время подкормки и тип подкормки (для каждого класса), тип почвы для растения, водяной режим для каждого времени года. Возможно введение народного названия растения с последующей заменой на общепринятое. Необходимо предусмотреть возможность замены некоторого типа подкормки более новым. БД должна содержать сведения о помещении: тип помещения и рекомендуемое место для размещения каждого растения. В БД может быть добавлена информация о новом растении.

Могут понадобиться следующие сведения:

- какой рекомендуемый водяной режим у заданного растения летом;
- какие растения можно держать в комнате с северной стороны на окне;
- какие растения, цветущие в мае, неприхотливы к воде летом;
- в какое время необходимо вносить заданную подкормку для указанного растения;
- когда необходимо пересадить указанное растение заданного возраста.

Возможны следующие изменения в БД:

- добавление информации о новом растении;
- отказ от старого типа подкормки;
- замена названия растения.

Необходимо предусмотреть возможность выдачи справки о времени и типе подкормки для заданного растения и отчета о растениях указанного класса (их количество в БД, название каждого растения, водяной режим, время пересадки и цветения).

Задание 3

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для завуча школы.

Для каждого учителя в БД должны храниться сведения о предметах, которые он преподает, номере кабинета, который за ним закреплен, номера классов, в которых он ведет занятия, номере урока и дне, когда он ведет уроки. Существуют учителя, которые не имеют собственного кабинета. Об учениках должны храниться следующие сведения: фамилия и имя, в каком классе учится, какая оценка по каждому предмету получена. Ученик может исправить полученную оценку. Завуч может добавить информацию о новом учителе или ученике, а также удалить - о выбывших.

Завучу могут потребоваться следующие сведения:

- какой предмет будет в заданном классе, например, во вторник на заданном уроке;
- кто из учителей преподает в заданном классе;
- в каком кабинете будет 5-й урок в среду у некоторого класса;
- в каких классах преподает учитель заданный предмет;
- расписание на заданный день недели для класса.

Завуч может вносить следующие изменения:

- вносить информацию о новом учителе;
- удалять запись об ученике;
- изменить оценку ученику.

Необходимо предусмотреть возможность выдачи справки о количестве учеников в данном классе и отчета о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число учащихся на «неудовлетворительно», «удовлетворительно», «хорошо и отлично», «отлично» по классам и по школе).

Задание 4

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для организаторов выставки собак. В БД должны храниться сведения об участниках выставки и экспертах.

Для каждого участника в БД должны храниться сведения о том, из какого клуба участник, кличка, порода и возраст собаки, сведения о родословной (отец и мать собаки). Участник может перейти из одного клуба в другой. На каждый клуб отводится участок номеров, под которыми будут выступать участники выставки. Сведения об экспертах должны включать ФИО; номер ринга, который он обслуживает; клуб, в котором он состоит. Каждый ринг могут обслуживать несколько экспертов. Каждая порода собак выступает на своем ринге. Эксперт может отказаться от судейства, тогда возможно введение нового эксперта. Также должны храниться сведения о медалистах по каждой породе.

Могут потребоваться следующие сведения:

- на каком ринге выступает заданный участник со своей собакой;
- какими породами представлен заданный клуб;
- какие медали и сколько заслужены клубом;
- какие эксперты обслуживают породу;
- какая собака у заданного эксперта.

Администратор БД может вносить следующие изменения:

- переход участника из одного клуба в другой;
- снятие эксперта с судейства на ринге;
- назначение нового эксперта на судейство.

Необходимо предусмотреть возможность выдачи справки о занятии участником призового места на выставках и отчета о выступлении клуба (сколько участников, какие породы, информация о победителях по породам).

Задание 5

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога для работников приемной комиссии. В БД должны храниться сведения об абитуриентах, датах экзаменов и консультаций, номерах аудиторий.

Для каждого абитуриента в БД должны храниться сведения об абитуриенте, номере экзаменационного листа, который он получает; о кафедре и факультете, куда он собирается поступать; о номере группы и потоке, в котором он будет сдавать экзамен (группы объединены в потоки по 3-4 группы на поток); оценка по каждому предмету, которая может быть изменена на апелляции. Абитури-

енты могут подавать и забирать документы, а также перевести их на другую кафедру. Также должны храниться даты консультаций и экзаменов по предметам для каждого потока и номера аудиторий.

Могут потребоваться сведения:

- список абитуриентов на заданный факультет;
- полученные оценки для абитуриента;
- дата консультации и экзамена для абитуриента по данному предмету;
- номера аудиторий, где будут экзамены у заданной группы;
- список групп, которые будут заниматься в заданной аудитории в заданное время.

Администратор БД может вносить следующие изменения:

- ввести информацию о новом абитуриенте;
- изменить оценку абитуриенту;
- удалить запись об абитуриенте.

Необходимо предусмотреть возможность выдачи справки о том, что данный абитуриент поступает в институт на заданный факультет, и отчета о работе приемной комиссии факультета (количество поступающих: на какие кафедры и сколько, количество абитуриентов в каждой группе, в какие дни и где проводятся экзамены, сколько сдало на «неудовлетворительно», «удовлетворительно», «хорошо» и «отлично» по предметам).

Задание 6

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для почтовых работников. В БД должны храниться сведения о газетах, почтовых отделениях, получающих газеты, о типографиях, выпускающих газеты.

Сведения о газетах должны включать в себя: название газеты, шифр издания (может быть изменен) цену экземпляра газеты, ФИО редактора, номера типографий, где печатается эта газета. Возможно появление новых газет. Для типографий: адрес типографии, количество газет данного наименования, печатающихся в этой типографии (в одной типографии может печататься несколько газет). Типография может быть закрыта. Для почтового отделения: адрес отделения, название и количество экземпляров, поступающих на каждое почтовое отделение.

Работникам может потребоваться следующая информация:

- по каким адресам печатается газета данного наименования;
- какая фамилия у редактора газеты, которая печатается в указанной типографии самым большим тиражом;
- на какие почтовые отделения (адреса) поступает газета, имеющая цену больше указанной,
- какие газеты и куда (номер почты) поступают в количестве меньшем, чем заданное;
- на какую почту поступает данная газета, печатающаяся по данному адресу типографии.

Работник может вносить следующие изменения:

- добавить информацию о новой газете;
- изменять цену газеты;
- удалить информацию о типографии.

Необходимо предусмотреть возможность выдачи справки об индексе и цене данной газеты и отчета о работе типографии (общее количество газет, печатающихся в ней, название, индекс и количество экземпляров для каждой газеты, ФИО редактора).

Задание 7

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для директора птицефермы. В БД должны храниться сведения об имеющихся на ферме курах и о работниках фермы.

О каждой курице в БД должна храниться следующая информация: вес, возраст, порода, количество яиц, получаемое от курицы, а также информация о местонахождении курицы. Сведения о породе включают в себя: название, средние производительность и вес, номер рекомендованной диеты. Птицефабрика имеет несколько цехов, и за каждой курицей закреплена отдельная клетка. Код клетки, где находится курица, характеризуется номером цеха, номером ряда в цехе и номером в ряду. О работниках фермы в БД должна храниться следующая информация: ФИО, коды клеток, которые закреплены за работником, зарплата.

При работе с БД могут потребоваться сведения:

- какое количество яиц получают от курицы данного веса, породы, возраста;
- в каком цехе наибольшее количество кур определенной по-

роды;

- в каких клетках сидят двухлетние куры с диетой номер 2;
- сколько яиц в день приносят куры указанной работницы;
- в каком цехе находится клетка, из которой получают больше всего яиц.

Администратор БД может вносить следующие изменения:

- добавление информации о новой курице;
- изменение БД в связи с увольнением работника;
- изменение веса курицы.

Необходимо предусмотреть возможность выдачи справки о номере диеты для курицы указанной породы и отчета о работе птицефабрики (общее количество кур, сколько кур и их средняя производительность для каждой породы, общее количество яиц, получаемое птицефабрикой, число работниц на фабрике и распределение их по цехам).

Задание 8

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для коллекционера марок, собирающего тематическую коллекцию. В БД должны храниться сведения о марках, имеющихся в коллекции, и сведения об их положении в коллекции.

Сведения о марках должны включать в себя номер марки, страну выпуска марки, номер серии, тему серии, год выпуска, цвет марки, размер марки, цену марки, тему марки. Марки расположены в коллекции по темам. Сведения о положении марки в коллекции включают в себя: номер раздела коллекции (разбит на тома по темам и странам), в котором хранится марка, номер тома, номер страницы и уникальное положение марки на странице. Возможно открытие новых или закрытие старых тем. Место расположения марки может изменяться.

Коллекционеру могут потребоваться следующие сведения:

- марки, каких стран содержатся в данном разделе;
- в каком томе коллекции находится марка определенной серии;
- в каких местах коллекции находятся марки указанной темы;
- какие темы у серий, включающих марки определенного размера;

- марка, какой страны находится в данном месте.

Коллекционер может вносить следующие изменения:

- добавление марки новой темы;
- удаление всех марок одной темы;
- изменение места расположения марки в коллекции;

Необходимо предусмотреть возможность выдачи справки о странах, чьи марки находятся в данной теме, и отчета по коллекции (количество и названия тем и стран по разделам, количество марок каждой страны для каждой темы, количество страниц в коллекции).

Задание 9

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для работников управления торговли. В БД хранятся сведения о магазинах города; товарах, имеющихся в магазинах; торговых базах и товарах, хранящихся на базах.

Каждый магазин закреплен за одной торговой базой. Магазин характеризуется классом, номером и имеет несколько отделов. Каждый товар в каждом магазине продается, по крайней мере, в одном отделе. Каждый отдел имеет одного заведующего отделом. Товары, имеющиеся магазине, и хранящиеся на базах, характеризуются ценой, сортом и количеством. Розничные цены в магазине зависят от класса магазина и сорта товара и могут изменяться. Магазин может открыть новый отдел или закрыть старый. В этом случае товар передается в другие отделы. -

При работе с БД могут потребоваться следующие сведения:

- какие товары имеются в магазине (на базе);
- какие отсутствующие товары может заказать магазин на базе;
- какие товары, и в каком количестве имеются в отделе магазина;
- список заведующих отделами магазина;
- в каких отделах магазина продается одинаковый товар.

Администратор БД, может вносить следующие изменения:

- закупка нового товара;
- закрытие отдела в магазина;.
- изменение цены товара.

Необходимо предусмотреть возможность выдачи справки о наличии товаров в отделе магазина и отчета по магазину (количество и наименование товаров в отделах, ФИО заведующих отделами, номер базы, за которой закреплен магазин).

Задание 10

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для диспетчера автобусного парка. В БД хранятся сведения о водителях, маршрутов автобусов и характеристиках автобусов.

Каждый водитель характеризуется: ФИО, классом, стажем работы и окладом, причем оклад зависит от его класса и стажа работы. Маршрут автобуса характеризуется: номером маршрута, временем начала и конца движения, интервалом движения и протяженностью. Характеристиками автобусов являются: номер автобуса, его тип и вместимость, причем вместимость автобуса зависит от его типа. Каждый водитель закреплен за отдельным автобусом, а каждый автобус закреплен к определенному маршруту. Необходимо предусмотреть возможность корректировки БД в случаях поступления на работу нового водителя, списывания старого автобуса, введения нового или изменения старого маршрута и т.п.

Диспетчеру автопарка могут потребоваться следующие сведения:

- список водителей, работающих на определенном маршруте;
- какие номера автобусов обслуживают данный маршрут
- когда начинается или заканчивается движение автобусов на всех или отдельных маршрутах;
- какова протяженность всех или определенных маршрутов автобусов;
- на каких автобусах работает водитель.

Диспетчер может вносить следующие изменения:

- прием на работу нового водителя;
- списание старого автобуса;
- изменение протяженности маршрута.

Необходимо предусмотреть возможность выдачи справки о протяженности маршрута и отчета по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс).

Задание 11

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для работников ремонтного предприятия. Предприятие ремонтирует изделия, состоящие из конструктивных узлов. Ремонт заключается в изготовлении и замене изношенных деталей в узлах.

В БД должны храниться сведения о деталях: обозначение, наименование, тип заготовки (может быть несколько), из которых деталь может быть сделала, вид материала из которого сделана деталь, расход материала и время ее изготовления (зависят от типа заготовки). Кроме того, в БД хранятся описания узлов: обозначение, список и количество деталей, содержащихся в узле, допустимый процент износа каждой детали и время ее замены. Ремонтируемое изделие характеризуется названием, перечнем изношенных деталей в узлах и процентом их фактического износа.

При работе с БД могут потребоваться следующие сведения:

- какое количество материала потребуется для изготовления заменяемых деталей определенного обозначения, входящих в данный узел и имеющих определенный тип заготовки;
- какой узел имеет наибольшее количество изношенных деталей определенного типа;
- какой тип заготовки обеспечивает минимальный расход материала для деталей заданного обозначения;
- сколько времени потребуется на ремонт изделия;

Администратор БД может вносить следующие изменения:

- добавление информации о новом узле;
- удаление информации о ремонтируемом изделии;
- изменение типа заготовки детали.

Необходимо предусмотреть возможность выдачи справки о количестве указанной детали в узле и отчета о работе предприятия (название и количество ремонтируемых изделий, время ремонта каждого изделия, список замененных деталей и расход материала при ремонте изделия).

Задание 12

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для работника

справочной службы кинотеатров города.

В БД должны храниться сведения о кинотеатрах: название, район, где расположен кинотеатр, категория (может быть изменена), вместимость (определяется категорией), о фильмах: назначение, производство, режиссер, жанр; кроме того, должна храниться информация о цене билета, количестве свободных мест на данный сеанс. На разных сеансах в одном кинотеатре могут идти разные фильмы. Кинотеатр может ввести новый фильм в репертуаре или снять старый с проката. Цена билета определяется прокатной стоимостью (названием) фильма и категорией кинотеатра.

- Справочной службе могут потребоваться следующие сведения о текущем состоянии проката фильмов в городе:
- репертуар кинотеатра (по названию кинотеатра);
- адрес и район кинотеатра (по названию кинотеатра);
- число мест (свободных) на данный сеанс (название кинотеатра и сеанс);
- цена балетов на данный сеанс (название кинотеатра и сеанс);
- жанр, производство и режиссер данного фильма (по названию);
- вместимость данного кинотеатра (по названию кинотеатра).

Администратор БД может вносить следующие изменения:

- открытие нового кинотеатра,
- снятие фильма с проката;
- изменение репертуара кинотеатра.

Необходимо предусмотреть возможность выдачи справки о сеансах фильма в указанном кинотеатре и отчета о прокате фильмов в районах города (названия фильмов, в каких кинотеатрах они демонстрировались, цена билета в каждом кинотеатре на них).

Задание 13

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для администратора футбольной команды. В БД должны храниться сведения о командах, участвующих в первенстве, и об игроках, играющих в данной команде, стадионах, на которых проходят встречи, и цене билета на игры.

Сведения о команде представляют собой название команды, город, где она базируется, ФИО тренера, даты встреч команды, счет

встреч, противников команды, стадион, на котором играет команда, место в таблице прошлого сезона. Сведения об игроках включают в себя ФИО игроков, их номера, результативность данного игрока в данной встрече. В один день команда может играть только в одном матче. Сведения о стадионе содержат: название, город, вместимость. Цена билета на матч зависит от вместимости стадиона и положения команды в прошлом роду (наибольшая - при игре тройке призеров, наименьшая - при игре тройки аутсайдеров). Игрока могут переходить из одной команды в другую. Некоторые встречи могут быть перенесены.

Администратору могут потребоваться следующие сведения:

- даты встреч команды, ее противник и счет;
- ФИО и номера игроков, участвовавших во встрече (по названию команды, городу и дате встречи);
- результативность данного игрока в данной встрече (по названию команды, городу, дате встречи и ФИО игрока);
- цена билета на матч указанных команд.

Администратор БД может вносить следующие изменения:

- переход игрока из одной команды в другую;
- отмена встречи;
- назначение нового тренера.

Необходимо предусмотреть возможность выдачи справки об играх на указанном стадионе и отчет о проведенных играх (количество проведенных встреч, число побед хозяев и гостей, ФИО игроков, забивавших мячи в каждой команде, названия стадионов, где проводились встречи).

Задание 14

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога для работников регистратуры поликлиники.

В БД должны храниться сведения о больных: ФИО, адрес, диагноз (может быть уточнен), дата заболевания; сведения о врачах: ФИО, номер кабинета, номер участка, дни и часы приема; описание болезней: название (диагноз), симптомы, лекарство. Возможно появление новых больных. Врач может уволиться из поликлиники.

Работникам регистратуры могут потребоваться следующие сведения:

- адрес, дата заболевания, диагноз данного больного;
- ФИО лечащего врача данного больного;
- номер кабинета, дни и часы приема данного врача;
- больные, находящиеся на лечении у данного врача;
- симптомы данного заболевания и рекомендуемое лекарство.

Администратор БД может вносить следующие изменения:

- появление нового больного;
- увольнение врача;
- изменение диагноза.

Необходимо предусмотреть возможность выдачи оправки о болезни некоторого больного и отчета о работе поликлиники (количество больных, ФИО каждого врача и число лечащихся у него больных, количество заболеваний по каждому виду болезни, расписание работы врачей поликлиники).

Задание 15

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для диспетчера станции техобслуживания.

В БД должны храниться сведения о владельцах автомобилей: ФИО, адрес, марка автомобиля, номер госрегистрации; характеристики автомобилей: год выпуска, изготовитель, перечень устраненных неисправностей, ФИО работника станции и время устранения каждой неисправности. Возможно введение в БД сведений о новых владельцах и новых неисправностях.

Диспетчеру могут потребоваться следующие сведения:

- ФИО и адрес владельца автомобиля с данным номером госрегистрации;
- Изготовитель марка и год выпуска автомобиля данного владельца;
- Перечень устраненных неисправностей у автомобиля данного владельца;
- ФИО работника станции, устранявшего данную неисправность автомобиля данного владельца, и время устранения;
- Какие автомобили ремонтировал данный работник станции;
- ФИО владельцев автомобилей с указанным типом неисправности.

Диспетчер может вносить следующие изменения:

- добавить информацию о владельце ремонтируемого автомобиля
- удалять информацию о работнике станции;
- изменить номер госрегистрации автомобиля.

Необходимо предусмотреть возможность выдачи справки о наличии неисправности автомобиля некоторого владельца и отчета о работе станции техобслуживания (количество ремонтируемых автомобилей, время ремонта каждого автомобиля и ФИО работника, который, их ремонтировал, список неисправностей для каждой марки автомобиля).

Задание 16

Спроектировать базу данных, обеспечивающую взаимодействие с ней в режиме диалога для менеджера музыкальных групп.

В БД должны храниться сведения о группах: название, год образования, страна, состав исполнителей, положение в последнем хит-параде (может измениться); о репертуаре каждой группы: названия песен, композитор, автор текста; данные о последних гастролях группы: название гастрольной программы, дата начала и окончания гастролей, цена билета (зависит от места гастролей и положения в хит-параде). Возможно появление новой группы и изменения в составе исполнителей. Каждая песня может быть в репертуаре только одной группы.

Менеджеру могут потребоваться следующие сведения:

- год образования, страна группы данного названия;
- репертуар наиболее популярной группы;
- автор текста, композитор и дата создания песни с данным названием;
- место и продолжительность гастролей группы данного названия;
- цена билета на концерт указанной группы;
- состав исполнителей группы данного названия, их возраст и амплуа.

Администратор может вносить следующие изменения:

- ввод новой группы;
- изменение положения группы в хит-параде;

- удаление информации об исполнителе, покинувшем группу.

Необходимо предусмотреть возможность выдачи справки о лучших группах в хит-параде и отчета о гастролях групп (название группы, место и сроки гастролей, репертуар с указанием авторов песен).

Задание 17

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для работников гостиницы. В БД должны храниться сведения о проживающих клиентах и служащих гостиницы, убирающих в номерах. Имеются номера трех типов: одноместный, двухместный и трёхместный, отличающиеся стоимостью проживания в сутки. Количество номеров в гостинице известно.

О каждом проживающем должна храниться следующая информация: номер паспорта, ФИО, город, из которого он прибыл, дата поселения в гостинице, выделенный гостиничный номер, на сколько дней выделен номер. Каждый номер характеризуется типом, стоимостью проживания, номером телефона, Номера упорядочены по этажам. О служащем гостиницы должна храниться следующая информация: ФИО, номер этажа, где он убирает, день недели, когда он убирает данный этаж. Служащий гостиницы убирает все номера на одном этаже в определенные дни недели

Работа с БД предполагает обслуживание следующих запросов:

- получение списка фамилий клиентов, проживающих в заданном номере,
- вычисление счета за проживание в гостинице;
- определение количества свободных мест и свободных номеров;
- получение списка прибывающих из заданного города;
- установление ФИО служащего, убиравшего номер в заданный день недели у некоторого клиента.

Администратор БД может вносить следующие изменения:

- освобождение номера проживающего;
- изменение расписания уборки для служащего в указанный день недели;
- увольнение служащего гостиницы.

Необходимо предусмотреть возможность выдачи справки о счете за проживание в гостинице определенного клиента и отчета о работе гостиницы за последний квартал (число клиентов, сколько дней был занят и свободен каждый номер, сумма дохода гостиницы)

Задание 18

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для почтовых работников. В БД должны храниться сведения о подписчиках газет (возможно добавление новых подписчиков), обслуживание некоторым отделением связи.

Каждая газета характеризуется индексом, названием и подписной ценой. Данные о подписчиках включают в себя: ФИО, домашний адрес, телефон, индексы получаемых газет, дату, начиная которой оформлена подписка, и срок подписки каждой газеты. Несколько домов объединены в участок, который обслуживается одним почтальоном. Каждый почтальон может обслуживать несколько участков. В БД должны содержаться сведения о том, к каким участкам относятся подписчики газет, и об обслуживающем их почтальоне.

Возможны следующие вопросы к БД:

- определить наименование и количество экземпляров всех газет; получаемых отделением связи;
- для заданного адреса определить фамилию почтальона, обслуживающего подписчика;
- какие газеты выписывает гражданин с заданными ФИО;
- для почтальона с заданной фамилией определить , какие газеты и в каком количестве он доставляет подписчикам.

Администратор БД может вносить следующие изменения:

- изменение почтальона, обслуживающего данный участок;
- добавление информации о новой выписанной газете;
- удаление информации при окончании срока подписки.

Необходимо предусмотреть возможность выдачи справки о почтальоне, обслуживающем данный участок и отчета о газетах, доставляемых почтой (названия газет, их количество, распределение газет по участкам, сроки получения каждой газеты и ФИО почтальонов, обслуживающих каждый участок).

Задание 19

Спроектировать базу данных, построить программу, обеспечивающую взаимодействий с ней в режиме диалога, для работников технического архива предприятия.

Технический архив содержит стеллажи, полки и ячейки, в которых хранится документация. Ячейка архива может быть пустой или хранить все экземпляры одного документа. Каждый экземпляр документации имеет инвентарный номер и название. В БД должна содержаться следующая информация: номер стеллажа, номер полки, номер ячейки, названия документа и темы, к которой он относится, инвентарный номер, количество экземпляров документа, содержащихся в ячейке, даты поступления документов в архив и запросов к ним. За документом могут обратиться абоненты архива, характеризующиеся ФИО, номером и телефоном отдела, где они работают.

При работе с БД могут потребоваться следующие сведения:

- определить название наиболее часто требуемого документа;
- определить общее количество документов на заданную тему;.
- определить тему по названию документа;
- определять название документа, который имеется в максимальном количестве экземпляров;
- определять отдел, работники которого наиболее часто обращаются к архиву;
- установить ФИО абонента, обращавшегося последним к указанному документу.

Администратор БД может вносить следующие изменения:

- добавление нового документа;
- изменение номера телефона указанного отдела;
- удаление экземпляра некоторого документа.

Необходимо предусмотреть возможность выдачи справки об абонентах отдела, пользующихся архивом, и отчета о работе архива (число единиц хранения, названия документов, поступивших в архив за последний месяц, количество экземпляров каждого документа, место его хранения).

ОГЛАВЛЕНИЕ

Учебное издание

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Учебное пособие

Татарникова Татьяна Михайловна

Редактор О.С. Крайнова
ЛР № 020309 от 30.12.96.

Подписано в печать 18.01.05. Формат 60x90 1/16.
Гарнитура Pragmatica. Бумага офсетная. 5,5 Печ.л. Тираж 100 экз.
Заказ №5.
Отпечатано в ООО «КРОМ». СПб, Новочеркасский пр., 1.