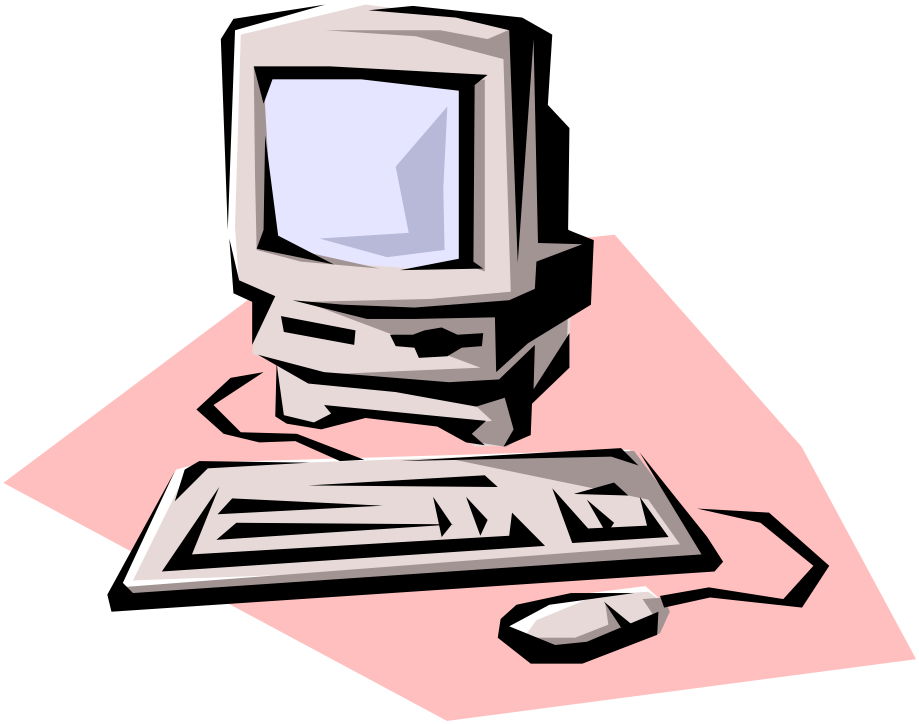
В.З. АБДУЛЛИНА

БАЗЫ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Учебник



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени К. И. САТПАЕВА

В. З. Абдуллина

БАЗЫ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Рекомендован Министерством образования и науки
Республики Казахстан в качестве учебника

Алматы 2015

УДК 004.6 (075.8)
ББК 32.973.26-018.2 м 73

Рецензенты:

Айтхожаева Е.Ж., профессор кафедры информационной безопасности КазНТУ им. К.И. Сатпаева, канд. техн. наук

Тукеев У.А., профессор кафедры информационных систем КазНУ им. Аль-Фараби, д-р техн. наук, проф.,

Самигулина Г.А., зав.лаб. «Интеллектуальные системы Управления и сети» ИПИУ МОН РК, д-р техн. наук,

Имангалиев Ш.И., зав. кафедрой информационных систем АУЭС, канд. техн. наук

А 55 **Абдуллина В.З.** Базы данных в информационных системах: Учебник. – Алматы: КазНТУ, 2015. – 288 с. Ил. 134. Табл. 36. Библиогр. – 30 назв.

ISBN 978-601-228-798-1

В учебнике рассматриваются основные понятия из области систем баз данных, описываются функции банка данных и СУБД, свойства данных, поддерживаемые в базе, используемые языки, уровни представления данных. Представлен ряд моделей данных: реляционная, иерархическая, сетевая и др. Вводятся определения процесса нормализации отношений для реляционной модели, реляционная алгебра, реляционное исчисление, язык запросов SQL.

Описываются возможности языка программирования Microsoft Visual Basic и СУБД Microsoft Access, поддерживающих мощные средства визуального проектирования баз данных и информационных приложений. Представлен процесс проектирования базы данных, приводятся примеры создания информационных приложений и информационных систем на основе баз данных для различных предметных областей.

Учебник предназначен для студентов, обучающихся в бакалавриате по специальности 5В070300 «Информационные системы», а также для студентов других специальностей, изучающих дисциплины, связанные с проектированием и эксплуатацией систем баз данных и СУБД.

УДК 004.6 (075.8)
ББК 32.973.26-018.2 м 73

Печатается по плану издания Министерства образования и науки Республики Казахстан на 2015 г.

ISBN 978-601-228-798-1

Посвящается 80-летию

ВВЕДЕНИЕ

Учебник «Базы данных в информационных системах» по дисциплине «Базы данных в информационных системах» предназначен для студентов, обучающихся на бакалавров по специальности 5В070300 – «Информационные системы». Учебник способствует формированию знаний и навыков в области теории баз данных, исследовании и использовании различных моделей представления данных, языковых средств для представления и обработки данных в базах, а также проектирования информационных систем (ИС) и информационных приложений на основе баз данных в среде СУБД Microsoft Access и в среде языка визуального программирования Microsoft Visual Basic.

Целью преподавания дисциплины «Базы данных в ИС» является овладение знаниями:

- по принципам и методам теории баз данных,
- по принципам построения систем баз данных и банков данных как информационных систем по обработке данных,
- по существующим моделям представления данных,
- по методам проектирования баз данных в информационных системах и способам обработки данных в них,
- по существующим СУБД и применяемым в них способам создания, поддержки и обработки баз данных.

Основные идеи современных информационных технологий базируются на концепции баз данных. Основой любой информационной технологии и информационной системы являются данные, которые должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем выдвинуло требование создания удобных общесистемных средств интеграции хранимых данных и управления ими – систем управления базами данных, предназначенных для организации и ведения баз данных. Большинство современных СУБД – это системы реляционного типа или системы, оперирующие базами данных с реляционной структурой представления данных.

Теория баз данных [1-11] является одним из разделов информатики, оказывающим сильное воздействие на развитие архитектуры вычислительных сетей, современные информационные технологии, методологию проектирования систем обработки данных, в том числе и информационных систем. Исследования и разработки в области систем баз данных были чрезвычайно успешными на протяжении их 50-летней истории. Они привели к появлению мощной индустрии, обладающей сформировавшимся фундаментом и фактически затрагивающей каждую более или менее важную компанию или

фирму в мире. Было бы немыслимо управлять большими объемами ценной информации, которую государства и корпорации имеют и продолжают накапливать, без поддержки этого средствами систем управления базами данных.

Теория баз данных развивается по таким направлениям как теория моделей данных и методы их эквивалентных преобразований, теория функциональных зависимостей, методы эквивалентных представлений баз данных и их схем, проблемы полноты и эквивалентности в реляционной алгебре и реляционном исчислении, методы обеспечения целостности базы данных в условиях потока конкурентных произвольных запросов, теория баз данных с неполной информацией.

В современных информационных системах и компьютерных технологиях базы данных стали не только источником получения информации, отвечающей различным информационным потребностям пользователей, но и платформой, на базе которой можно создавать интегрированные данные, как для принятия управленческих решений, так и для прогнозирования и моделирования поведения исследуемой системы.

В главе 1 рассматриваются основные понятия методологии баз данных: база данных, система баз данных, банк данных как ИС, процесс обработки запросов, функции и возможности СУБД, используемые языки описания и обработки данных, схема и подсхема, уровни представления данных, основные операции над данными.

В главе 2 представлены три основные модели данных: реляционная, иерархическая, сетевая. Для реляционной модели определяется процесс нормализации отношений, описана реляционная алгебра и реляционное исчисление как средства обработки данных, языки запросов SQL и QBE. Представлены результаты научной работы в области реляционных баз данных с временной динамикой [21 - 26]. Введены понятия по описанию объектов для иерархической и сетевой модели. Представлены такие модели данных, как постреляционная, многомерная, объектно-ориентированная.

Глава 3 посвящена вопросам создания и обработки систем баз данных в среде языка Microsoft Visual Basic, имеющего развитые средства для работы с базами данных, обладающего мощными возможностями визуального проектирования баз данных и приложений, осуществляющего многие функции по управлению данными с помощью встроенного языка SQL. Здесь рассмотрены методы создания и обработки таблиц базы данных, представлены программы по реализации операций обработки данных в БД для информационных систем, описаны возможности языка и его инструментарий, приведены примеры проектирования и реализации различных функций по обработке данных из БД для информационных приложений.

В главе 4 описаны возможности работы с СУБД Microsoft Access, поддерживающей мощные средства проектирования и использования баз данных, являющейся сегодня одной из самых популярных настольных СУБД. СУБД Access имеет богатый набор средств визуального проектирования и развитые инструментальные средства для работы с базами данных. Здесь

рассмотрены способы создания таблиц базы данных, форм, отчетов и различных запросов, а также представлены способы и методы обработки данных в БД, примеры разработки информационных приложений на основе БД.

В главе 5 изложены вопросы, связанные с проектированием баз данных, описаны все этапы проектирования: формулировка и анализ требований, концептуальное, логическое и физическое проектирование, представлены методы и средства ведения проектных работ. Для каждого этапа описаны как исходные данные, так и конечный результат проектирования.

В главе 6 представлены конкретные шаги и этапы проектирования информационных систем и информационных приложений на основе баз данных, приведены примеры реализации различных приложений для конкретных предметных областей таких, как «Учеба в семестре», «Ипотечное кредитование жилья», «Сотовая связь», «Библиотека», «Продажа автомобилей». Представлены технология создания реляционной базы данных с временной динамикой для природного очага чумы, созданная структура данных и сформулированные правила для поддержки ограничений целостности данных в БД, описаны функции информационной системы с временной динамикой и реализованный проект в среде языка Visual Basic. Описаны этапы нового системного проектирования бизнес-процессов в информационных системах, приведены примеры тем для разработки различных информационных приложений на основе базы данных в конкретных предметных областях.

В конце учебника приводится глоссарий, позволяющий студентам освоить и понять основные термины в области баз данных и СУБД.

Все изложение сопровождается хорошим иллюстративным материалом, множеством примеров, заданиями и контрольными вопросами, приведенными в конце каждой главы. Ряд разделов предназначен для самостоятельного изучения (2.8, 2.11 – 2.13, 3.5, 3.9 – 3.11, глава 4, глава 6) как студентами, так и магистрантами и докторантами специальности «Информационные системы».

Кроме того, этот учебник может использоваться студентами других специальностей и инженерами, специализирующимися и работающими в области проектирования и эксплуатации баз данных в информационных системах и СУБД.

Глава 1. СИСТЕМЫ БАЗ ДАННЫХ И СУБД

1.1. Информация и данные

Каждой цивилизации приходилось иметь дело с обработкой информации. С развитием экономики и ростом численности населения возрастает и объем взаимосвязанных данных, необходимых для решения экономических, административных и управленческих задач. Выделим 3 области, о которых можно говорить при обсуждении понятия «информация» (рис. 1.1).

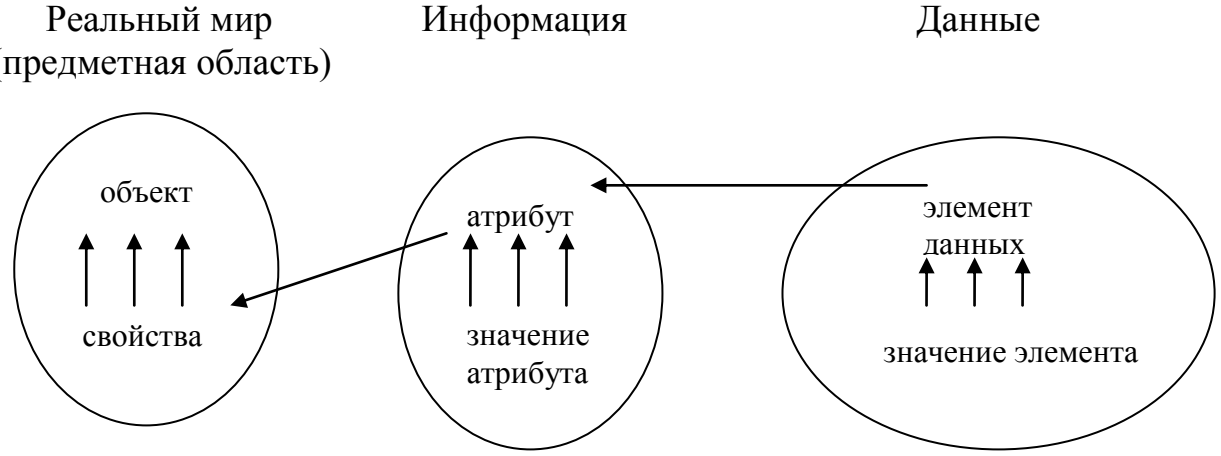


Рис. 1.1. Информация и данные

Первая область – это реальный мир, в котором существует множество объектов, обладающих определенными свойствами. В реальном мире выделяют некоторую предметную область, которая является частью реального мира. Примерами предметной области могут служить наука, животные, книга, автомашина, компьютер, язык программирования, изучаемая дисциплина, техническая литература, сессия, живопись, кино, спорт, флора, фауна, транспорт, связь, больница, аэропорт, институт, театр и т.д.

В предметной области можно выделить один или несколько объектов. Объектом может быть человек, предмет, событие, место, явление, понятие. Каждый объект должен характеризоваться несколькими свойствами.

Пример 1. Для предметной области «Больница» можно выделить такие объекты как «Больница», «Больной», «Врач», «Палата», «Отделение», «Младший медперсонал», «Лаборатория», «Лекарства» и т.д. Например, объект «Больница» может описываться с помощью таких свойств как «Номер больницы», «Наименование больницы», «Фамилия главного врача», «Количество койко-мест», «Количество персонала», «Фонд заработной платы» и т.д. Объект «Больной» может быть описан с помощью таких свойств как «Фамилия и инициалы больного», «Номер лечебной карточки», «Номер палаты», «Диагноз», «Лечение», «Фамилия лечащего врача» и т.д.

Пример 2. Для предметной области «Библиотека» можно выделить такие объекты как «Читатель», «Книга», «Журнал», «Газета», «Каталог», «Персонал», «Формуляр», «Читальный зал» и т.д. Например, объект «Читатель» может описываться с помощью таких свойств как «Номер читательского билета», «Фамилия», «Имя», «Отчество», «Место работы», «Домашний адрес» и т.д. Объект «Книга» может быть описан с помощью таких свойств как «Инвентарный номер», «Автор», «Название», «Жанр», «Год выпуска», «Стоимость» и т.д.

Точность, полнота и объемность описания как предметной области, так и отдельных объектов зависит от решаемой задачи, функций информационной системы и потребностей пользователей в информации.

Вторая область – область информации, существующей в представлении людей. Практически информация отображает все то, что имеется в реальном мире. Информация стала ресурсом, ее можно добывать, перерабатывать, использовать, распространять. Информация является одним из ценнейших ресурсов общества наряду с такими традиционными материальными видами ресурсов, как нефть, газ, полезные ископаемые и др. В области информации говорят об *атрибутах* объектов и обозначают атрибуты символически на естественном языке или на языке программирования или в какой-либо другой форме. Атрибутам приписывают значения. Например, для атрибута «Фамилия и инициалы больного» объекта «Больной» можно ввести такие обозначения: « ФИО», «ФИО больного», «FIO», «bolnoy». Приведем примеры значений этого атрибута: «Каримов А.К.», «Иванов И.А.» и т. д.

Третья область – это область *данных*, хранящихся в компьютере. Элемент данных представляет собой действительные данные, соответствующие определенному атрибуту, связанному с конкретным объектом из предметной области. Значения элементов данных в памяти компьютера представляют собой строки символов или битов. В зависимости от того, как элементы данных описывают объект, они могут быть количественными, качественными или описательными. Значения элементов данных могут существовать независимо от информации, которая запоминается с их помощью. Но смысл они приобретут только тогда, когда будут привязаны к определенному элементу. Например, в компьютере постоянно хранятся такие значения: «красный», «синий», «зеленый», «желтый». Затем их можно связать с конкретными элементами данных: «красный мак», «синее небо», «зеленая трава», «желтый светофор». В табл. 1.1 приведено описание объекта «Больница».

Таблица 1.1

Описание объекта «Больница»

Объект	Атрибут	Значение атрибута
больница	Номер больницы	5
	Наименование больницы	ЦГКБ
	Фамилия главного врача	Каримов
	Количество койко-мест	230
	Количество персонала	150
	Фонд заработной платы	400000

Соответственно двум понятиям «информация» и «данные» в базах данных различают два аспекта рассмотрения вопросов:

1) инфологический аспект – употребляется при рассмотрении вопросов, связанных со смысловым содержанием данных независимо от способов их представления в памяти компьютера;
2) датологический аспект – употребляется при рассмотрении вопросов представления данных в памяти компьютера.

1.2. Информационные отношения и взаимосвязи данных

Между различными объектами, свойствами объектов и объектами и их свойствами существуют объективные, определяемые предметной областью отношения. При информационном отображении объектов и присущих им свойств эти отношения переносятся и на элементы информации. Отношения между элементами информации информационны по своей природе, отображая реально существующие взаимосвязи, и многообразны по своему характеру, внутреннему механизму, математической интерпретации.

Даже для простейшего случая $r(z, s)$, где z и s – простые переменные, r – отношение между ними, это отношение является одним из бесконечного множества отношений, определяемых как типом переменных z и s, так и многими другими обстоятельствами. Так что для данного случая точнее говорить об отношении $r_i(z, s)$, выбранном из множества возможных отношений: $R(r_1, r_2, ..., r_i, ...)$.

Рассмотрим *пример*: пусть $z = 3$, а $s = 6$. Тогда можно определить такие отношения, существующие между z и s: 1) z и s – числа; 2) z и s – целые числа; 3) z и s – натуральные числа; 4) z и s – положительные числа; 5) $z < s$; 6) $z = s - 3$; 7) $s = z*2$; 8) $s = z/2$. Можно определить и другие отношения.

Среди множества отношений могут быть выделены отношения следующих видов:

- теоретико-множественного характера – принадлежность какому-то множеству, иерархическое отношение, отношение эквивалентности элементов и др.;
- логической природы – предикатные отношения, отношения операндов логического выражения или аргументов логической функции и др.;
- арифметического типа – сравнение чисел, упорядоченность чисел, функциональная зависимость и др.;
- лексикографической упорядоченности – упорядоченность информации символьного типа в соответствии с алфавитом используемого языка (например, список фамилий студентов группы).

Имеются и другие отношения самого разнообразного типа, природа которых подчас не формализуема или требует для формализации слишком сложного аппарата. Для рассмотренного выше примера отношения 1 – 4 носят теоретико-множественный характер и указывают принадлежность переменных z и s одному из множеств. Отношения 5 – 8 относятся к арифметическому типу

и указывают сравнение переменных *z* и *s* и функциональные зависимости между ними. Отношение 5 имеет логическую природу, поскольку его результатом является логическая переменная («истина» или «ложь»).

Приведем примеры бинарных отношений: 1) Астана – столица Казахстана; 2) Канат Оспанов имеет профессию инженера-системотехника; 3) институт информационных технологий является одним из институтов КазНТУ.

Взаимосвязь выражает отображение или связь между двумя множествами данных. Различают взаимосвязи следующих типов:

1) “один к одному” или (1:1) или 

2) “один ко многим“ или (1:M или 1: ∞) или

3) “многие ко многим” или (M : M) или

Пример 3. Рассмотрим взаимосвязи между атрибутами одного объекта. Пусть имеется объект СТУДЕНТ. При его описании используем атрибуты ФАМИЛИЯ, ИМЯ, ГОД РОЖДЕНИЯ, НОМЕР ЗАЧЕТНОЙ КНИЖКИ, НОМЕР ГРУППЫ. Для этого примера считаем, что фамилии у студентов не повторяются, т.е. нет однофамильцев. Между атрибутами имеются следующие взаимосвязи:

1. “Один к одному”

ФАМИЛИЯ  НОМЕР ЗАЧЕТКИ

2. “Один ко многим”

НОМЕР ГРУППЫ  НОМЕР ЗАЧЕТКИ

3. “Многие ко многим”

ГОД РОЖДЕНИЯ НОМЕР ГРУППЫ

Аналогично такие взаимосвязи могут быть установлены и между объектами.

Пример 4. Рассмотрим взаимосвязи между двумя объектами. Пусть имеются объект СТУДЕНТ с указанными выше атрибутами и объект ГРУППА с атрибутами НОМЕР ГРУППЫ, КОЛИЧЕСТВО СТУДЕНТОВ, ФАМИЛИЯ СТАРОСТЫ. Между указанными объектами ГРУППА и СТУДЕНТ существует взаимосвязь “Один ко многим”:

ГРУППА СТУДЕНТ ,

а между объектами СТУДЕНТ и ГРУППА существует взаимосвязь “Один к одному”:

СТУДЕНТ ГРУППА

Полем связи является поле «НОМЕР ГРУППЫ» из таблиц СТУДЕНТ и ГРУППА.

Пример 5. Рассмотрим взаимосвязи между двумя объектами ЗАВОД и АВТОМОБИЛЬ. Пусть объект ЗАВОД имеет такие атрибуты как НАИМЕНОВАНИЕ, ФАМИЛИЯ ДИРЕКТОРА, СТРАНА, АДРЕС, ТЕЛЕФОН. Объект АВТОМОБИЛЬ имеет атрибуты НОМЕР АВТО, МАРКА, ЦВЕТ,

НАИМЕНОВАНИЕ ЗАВОДА, ФАМИЛИЯ ВЛАДЕЛЬЦА, СТОИМОСТЬ. Между указанными объектами ЗАВОД и АВТОМОБИЛЬ существует взаимосвязь “Один ко многим”:

ЗАВОД $\longrightarrow\!\!\!\!\!\rightarrow$ АВТОМОБИЛЬ ,

а между объектами АВТОМОБИЛЬ и ЗАВОД существует взаимосвязь “Один к одному”:

АВТОМОБИЛЬ $\longrightarrow$ ЗАВОД

Полем связи является поле «НАИМЕНОВАНИЕ» из таблицы ЗАВОД и поле «НАИМЕНОВАНИЕ ЗАВОДА» из таблицы АВТОМОБИЛЬ.

Пример 6. Рассмотрим взаимосвязи между двумя объектами ЗАВОД и ЭКОНОМИКА ЗАВОДА. Пусть объект ЗАВОД имеет такие атрибуты как НАИМЕНОВАНИЕ, ФАМИЛИЯ ДИРЕКТОРА, СТРАНА, АДРЕС, ТЕЛЕФОН. Объект ЭКОНОМИКА ЗАВОДА имеет атрибуты НАИМЕНОВАНИЕ, УСТАВНОЙ КАПИТАЛ, ПРИБЫЛЬ, РЕНТАБЕЛЬНОСТЬ, ОБЪЕМ ВЫПУСКА ПРОДУКЦИИ В ГОД, СТОИМОСТЬ ЕДИНИЦЫ ПРОДУКЦИИ. Между объектами ЗАВОД и ЭКОНОМИКА ЗАВОДА существует взаимосвязь “Один к одному”:

ЗАВОД $\longrightarrow$ ЭКОНОМИКА ЗАВОДА ,

а между объектами ЭКОНОМИКА ЗАВОДА и ЗАВОД также существует взаимосвязь “Один к одному”:

ЭКОНОМИКА ЗАВОДА $\longrightarrow$ ЗАВОД

Полем связи является поле «НАИМЕНОВАНИЕ» из таблицы ЗАВОД и таблицы ЭКОНОМИКА ЗАВОДА.

1.3. Системы баз данных

База данных (БД) представляет собой поименованную совокупность данных, отображающую состояние множества объектов из рассматриваемой предметной области, их атрибутов (свойств) и взаимоотношений. Практически база данных является информационной моделью предметной области, от обоснованности, точности и достоверности которой зависит эффективность информационной системы. *Запись* данных представляет собой совокупность значений атрибутов, описывающих конкретный объект реального мира. *Первичный ключ* (ключевое поле) – это атрибут (группа атрибутов), идентифицирующий запись (объект) уникальным образом. *Вторичный ключ* – это атрибут (группа атрибутов), который идентифицирует группу записей.

В современных СУБД, построенных на основе реляционной модели представления данных, база данных состоит из таблиц, каждая из которых содержит описание одного объекта предметной области. Свойства объекта представляются в таблице в виде полей. Первичный ключ и вторичные ключи организованы в виде индексов (индексных файлов).

Система баз данных (СБД) – это компьютеризированная система хранения записей. СБД удобно рассматривать как простую структуру, состоящую из сервера (собственно СУБД) и набора клиентов (приложений). Упрощенная схема системы баз данных приведена на рис.1.2.

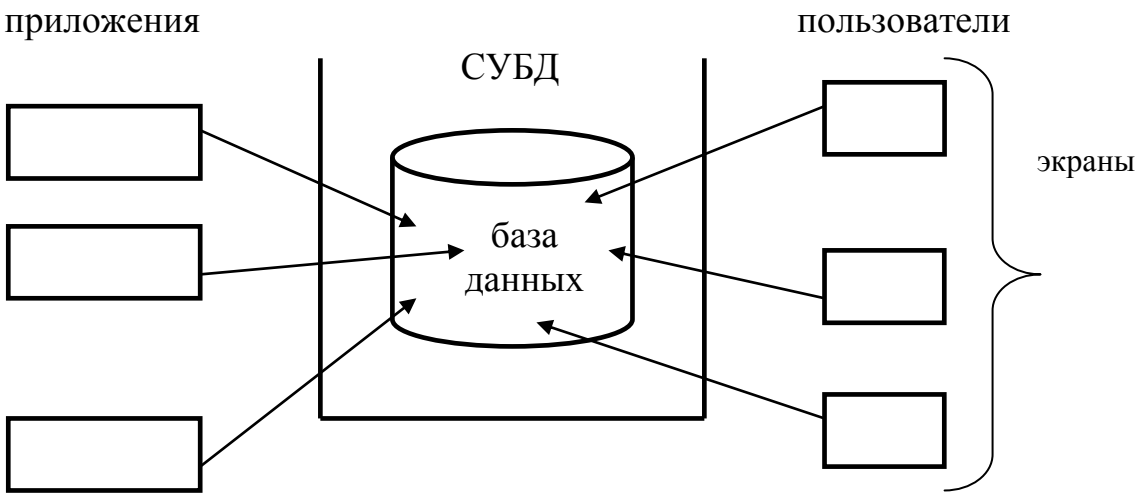


Рис. 1.2. Упрощенная схема СБД

Под *банком данных* (БнД) понимается организационно-техническая система, представляющая собой совокупность баз данных, технических и программных средств формирования и ведения этих баз и коллектива специалистов, обеспечивающих функционирование этой системы. Фактически банк данных есть совокупность базы данных и системы управления базой данных (СУБД). Такая структура банка данных представлена на рис. 1.3.

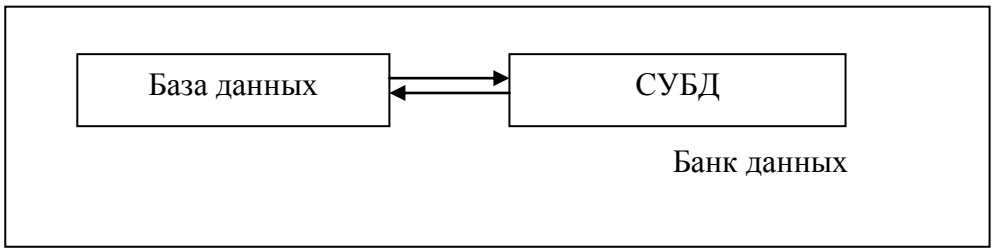


Рис. 1.3. Структура банка данных

Отличительными чертами банка данных являются:

- удовлетворение актуальных информационных потребностей множества пользователей, проведение поиска информации по произвольным запросам, выдача информации пользователю в различных формах;
- обеспечение возможности хранения и модификации больших объемов многоаспектной информации, обеспечение доступа к данным только пользователей с соответствующими полномочиями;
- обеспечение требуемого уровня достоверности хранимой информации и ее непротиворечивости;
- возможность реорганизации и расширения баз данных при изменении границ предметной области;

- обеспечение заданных требований эффективности функционирования.

Практически банк данных представляет собой *информационную систему* (ИС), отвечающую за работу с данными на основе базы данных и реализующую такие операции, как добавление новых данных в БД, удаление ненужных данных в БД, корректировка ряда данных в БД, поиск данных в БД по критериям.

Преимущества банка данных: сокращение избыточности данных, устранение противоречивости хранимых данных, многоаспектное использование данных, комплексная оптимизация при проектировании структур баз данных, обеспечение возможности санкционированного доступа к данным. К недостаткам банка данных можно отнести сложность создаваемых систем, их чувствительность к последствиям сбоев и аварий. Структура банка данных с учетом всех его компонентов представлена на рис. 1.4.

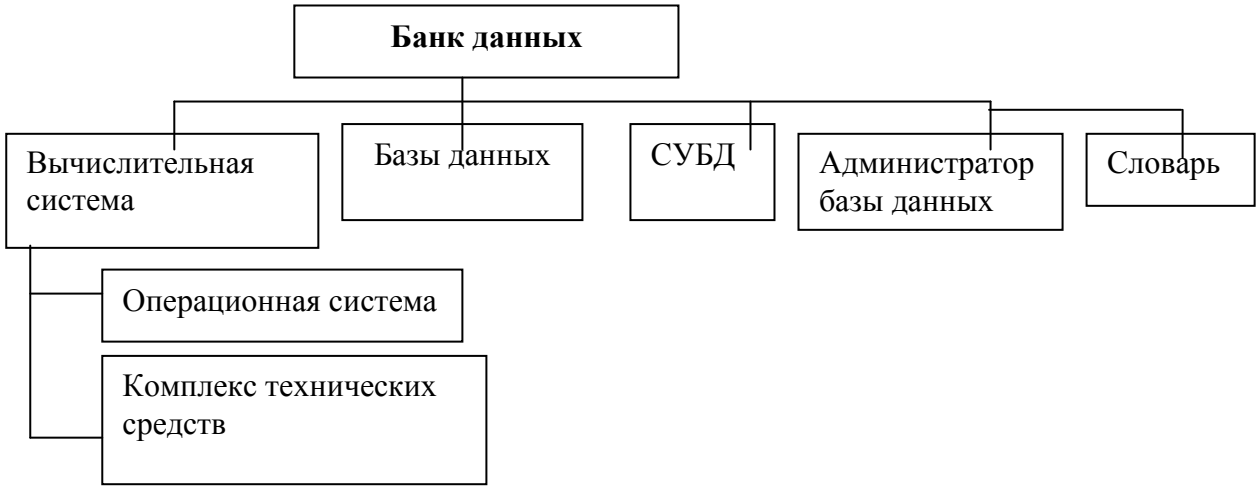


Рис. 1.4. Компоненты банка данных

Операционная система (ОС) служит программной средой, позволяющей организовать работу с СУБД. Комплекс технических средств (КТС) представляет собой техническую среду - компьютер и его устройства, компьютерные сети. Администратор базы данных (АБД) отвечает за создание и ведение БД. *Словарь* хранит информацию обо всех ресурсах: об объектах, их свойствах и отношениях для каждой предметной области, о возможных значениях и форматах представления данных, об источниках возникновения данных, о кодах защиты и разграничениях доступа, о наименовании данных, смысловом описании, структуре, связях и др.

1.4. СУБД

Система управления базой данных (СУБД) – представляет собой совокупность языковых и программных средств, предназначенных для создания и ведения баз данных. Централизованное управление базами данных посредством СУБД обеспечивает:

- сокращение избыточности и устранение несовместимости в хранимых данных;
- совместное использование данных множеством пользователей и приложений, что достигается необходимой интеграцией данных;
- стандартизация представления данных, упрощающая эксплуатацию банка данных;
- разграничение доступа к данным;
- целостность данных, которая достигается за счет процедур, предотвращающих включение в базу неверных данных, и восстановление базы данных после аварий и сбоев.

Структура СУБД приведена на рис. 1.5. Данные представляют собой БД. Метаданные (схема) – это информация о структуре данных: имена отношений, имена атрибутов, их типы (для реляционной СУБД). Индекс – это структура, позволяющая быстро находить данные.

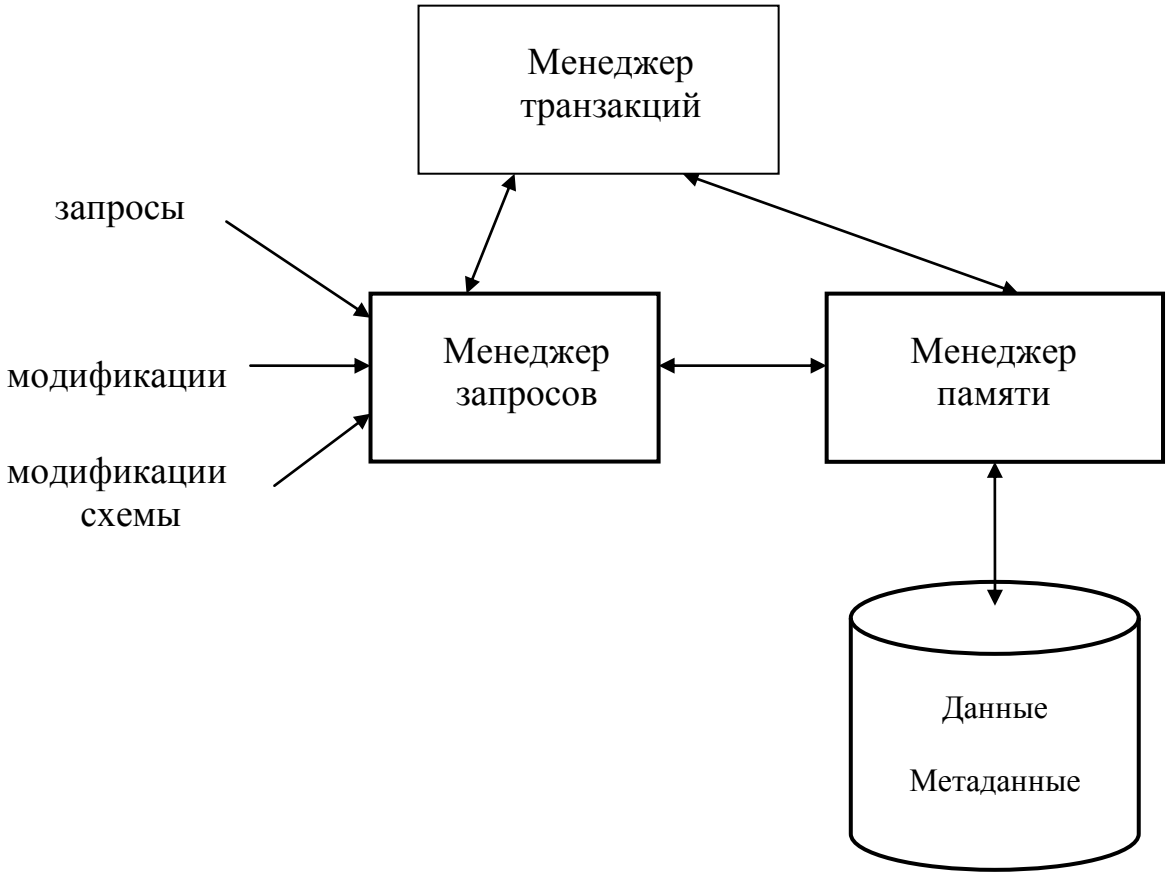


Рис. 1.5. Структура СУБД

Различают три типа запросов к СУБД:

1) запросы – это запросы на получение данных через общий интерфейс запросов или через интерфейс приложений (прикладных программ);
2) модификации – операции по изменению данных в БД;
3) модификации схемы – команды, задаваемые АБД с целью изменить схему БД или создать новую БД.

Менеджер памяти позволяет получать требуемую информацию из БД и изменять в БД информацию. Состоит из двух компонентов:

1 – *менеджер файлов* – контролирует расположение файлов во внешней памяти и получает по запросу менеджера буфера блок (блоки), содержащий нужные файлы (блок – единица дискового пространства во внешней памяти);

2 - *менеджер буфера* - управляет основной памятью, получает блок (блоки) данных с диска и выбирает страницу основной памяти для его хранения.

Менеджер (процессор) запросов – определяет лучший способ выполнения операции и дает соответствующие команды менеджеру памяти, практически строит последовательность простейших действий с БД, позволяющих выполнить запрос.

Менеджер транзакций – отвечает за целостность системы, должен обеспечить одновременную обработку множества запросов и защиту данных в случае выхода системы из строя.

Транзакция – это группа операций, которые необходимо выполнить последовательно как единое целое. *Пример транзакции*: получение денег через банкомат. При этом должны быть выполнены следующие действия:

- ввод карточки в банкомат;
- ввод пароля;
- идентификация пользователя;
- вопрос о действиях пользователя – получение наличных или получение информации о доступной сумме;
- если пользователь выбрал получение наличных, то идет запрос требуемой суммы;
- предложение забрать карточку;
- предложение забрать деньги.

В ходе обработки этой транзакции пользователь взаимодействует с сервером базы данных, который находится в центральном офисе банка, возможно, в другом городе или даже в другой стране.

Правильное выполнение транзакций требует обеспечения 4-х свойств (ACID-свойства):

1. *атомарность* (atomicity) – требуется, чтобы были выполнены все транзакции или ни одна; например, изъятие денег из банкомата и изменение счета клиента (вычитание из него полученной суммы) должны произойти *одновременно*;
2. *непротиворечивость* (consistency) – нельзя выполнять противоречивые действия; например, нельзя продать одно и то же место на рейс двум клиентам; число проданных мест, в конечном итоге, обязательно должно быть равно числу обслуженных клиентов, купивших билет;
3. *изоляция* (isolation) – при параллельном выполнении двух или более транзакций их результаты должны быть изолированы друг от друга; например, недопустимо одно место на рейс продать дважды, особенно, если это последнее свободное место;

4. *долговременность* (durability) – если транзакция завершена, то ее результат не должен быть утрачен (результаты транзакции обязательно должны быть сохранены в базе данных).

Перечислим основные *функции* СУБД.

1) *Определение данных*. СУБД должна предоставлять средства определения данных (внешних схем, концептуальной схемы, внутренней схемы, а также всех необходимых отображений) в виде исходной формы и преобразования этих определений в соответствующую объектную форму. Иначе говоря, СУБД должна включать в себя компоненты процессора языка описания данных ЯОД или компилятора ЯОД для каждого из поддерживаемых ею ЯОД.
2) *Обработка данных*. СУБД должна уметь обрабатывать запросы пользователя на выборку – изменение – удаление – добавление данных. Другими словами СУБД должна включать в себя компонент процессора языка манипулирования данными ЯМД или компилятора ЯМД, обеспечивающего поддержку ЯМД. Планируемый запрос определен заранее, а непланируемый – это произвольный запрос.
3) *Оптимизация и выполнение*. Запросы ЯМД должны быть обработаны оптимизатором, который должен производить поиск достаточно эффективных способов выполнения каждого из запросов. Оптимизированные запросы затем выполняются под управлением менеджера времени выполнения (run-time manager).
4) *Защита и сохранение целостности данных.* СУБД должна контролировать пользовательские запросы и пресекать любые попытки нарушения ограничений целостности данных, определенные СУБД. Этот контроль должен осуществляться во время компиляции, во время выполнения и на обоих этих этапах обработки запроса.
5) *Восстановление данных и поддержка параллельности.* СУБД или другой связанный с ней программный компонент, называемый менеджером транзакций или диспетчером выполнения транзакций, должен обеспечить необходимый контроль над восстановлением данных и управлением параллельностью обработки.
6) *Словарь данных.* СУБД должна поддерживать функцию ведения словаря данных. Словарь содержит «данные о данных» (метаданные), т.е. определения других объектов системы.
7) *Производительность.* СУБД должна выполнять все указанные функции с максимально возможной производительностью, повышая тем самым эффективность функционирования информационной системы, построенной на основе базы данных.

Структура банка данных с указанием программ, входящих в СУБД, представлена на рис. 1.6.

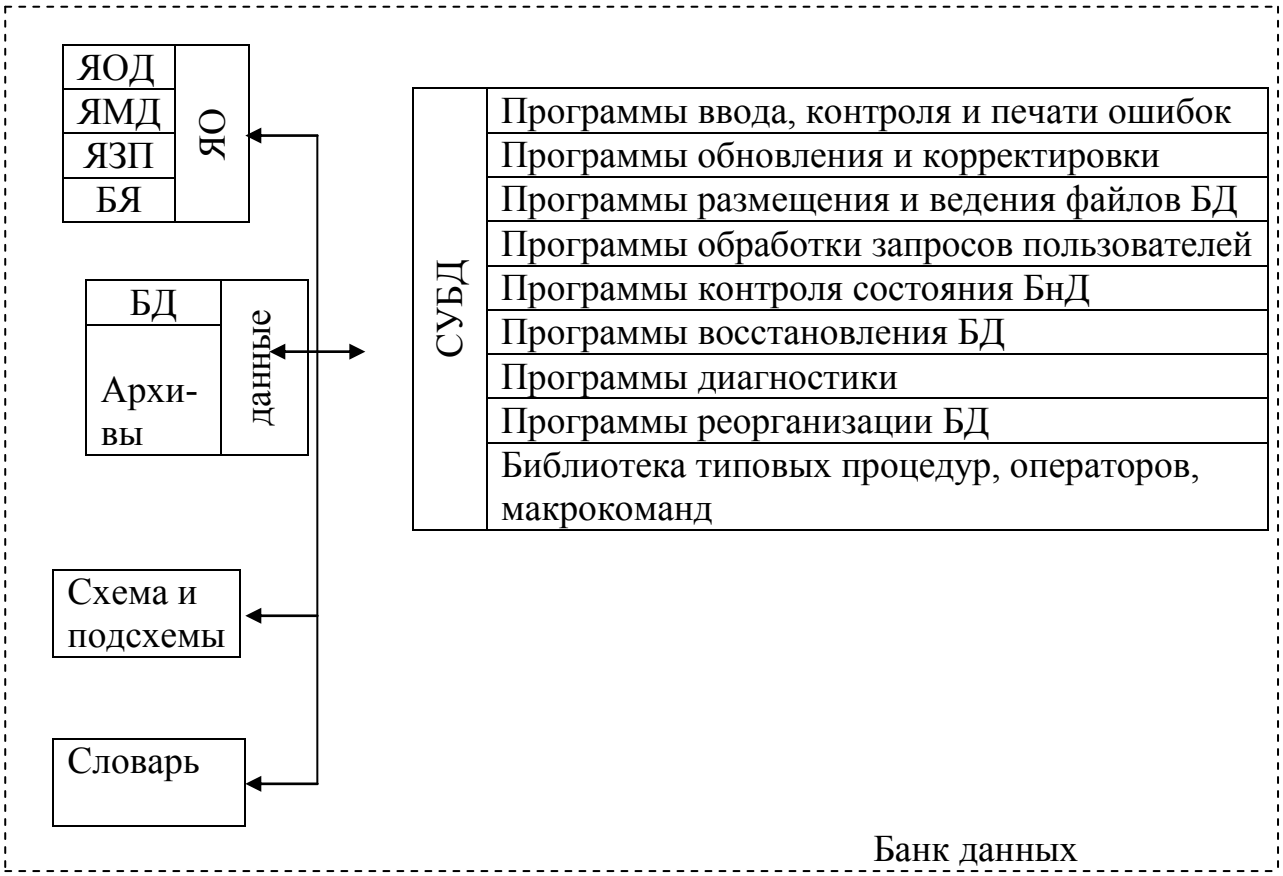


Рис. 1.6. Структура банка данных с указанием программ СУБД

1.5. Обработка запросов в банке данных

Процесс обработки запроса на данные с помощью СУБД представлен на рис. 1.7.

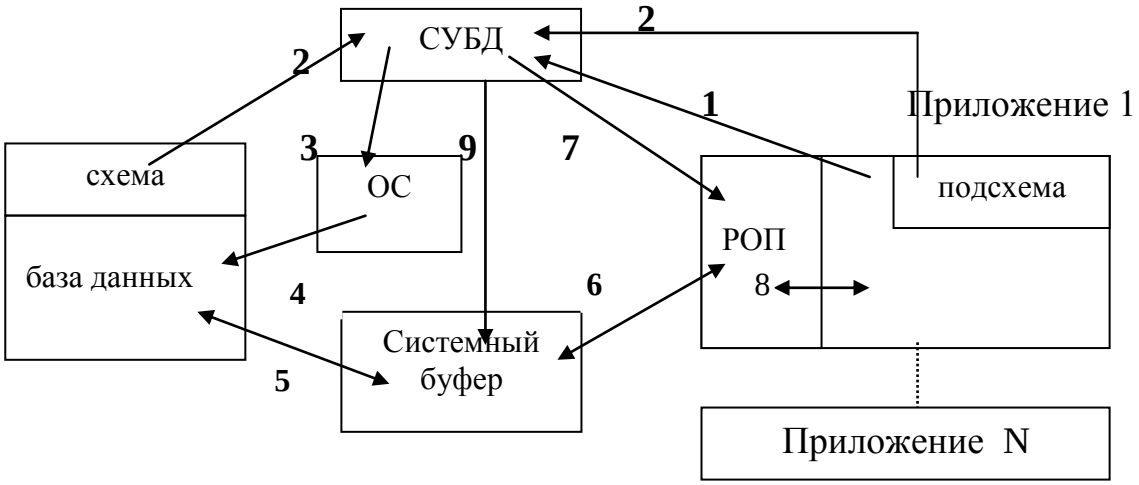


Рис. 1.7. Обработка запроса от приложения

Рабочая область приложения (РОП) представляет собой область памяти, в которой находятся данные, предназначенные для взаимодействия между

приложением и СУБД. Фактически РОП является зоной загрузки и разгрузки данных, совместно используемых приложением и СУБД. Системный буфер также является областью памяти, совместно используемой операционной системой и СУБД. Как правило, СУБД имеет несколько системных буферов.

Рассмотрим этот процесс, когда приложение (ПП – прикладная программа) обращается к СУБД с запросом на получение данных из БД. Опишем назначение связей:

1 – приложение обращается к СУБД за данными, хранящимися в БД, с помощью операторов ЯМД;

2 – СУБД анализирует запрос на данные и дополняет его информацией из схемы и подсхемы, определяя, где хранятся требуемые данные и в каком виде они должны быть предоставлены приложению;

3 – СУБД вызывает выполнение операционной системой (ОС) физических операций ввода / вывода, необходимых для выполнения запроса;

4 – ОС взаимодействует с внешней памятью, в которой хранится база данных;

5 – данные передаются из БД в системный буфер;

6 – СУБД передает данные из системного буфера в РОП приложения, при этом СУБД выполняет все необходимые преобразования данных из их представления в БД, определяемого схемой, в их представление для приложения, определяемое подсхемой;

7 – в ходе обработки запроса СУБД передает в приложение информацию о найденных ошибках или о нормальном ходе работы;

8 – данные из РОП может использовать приложение;

9 – СУБД управляет системными буферами, которые используются при обработке запросов от всех приложений.

Аналогично обрабатываются запросы от всех приложений, которые обращаются к СУБД за данными, хранящимися в базе данных. Если необходимо поместить в БД данные, то обработка запроса идет аналогично, но связи 8, 6 и 5 срабатывают в обратном направлении.

1.6. Свойства данных, поддерживаемые в базе

В БД поддерживаются такие свойства данных как интеграция, независимость, отсутствие дублирования, защита и целостность.

База данных позволяет осуществить *интеграцию* данных. Это означает, что БД содержит данные для множества приложений и пользователей, каждый из которых работает лишь с небольшой частью данных, хранящихся в БД. Отдельные элементы данных могут обрабатываться и использоваться в нескольких приложениях.

Пример 1. В информационной системе для предприятия с полями «Фамилия сотрудника» и «Должность сотрудника» в сведениях о сотруднике работает и инспектор отдела кадров, и бухгалтер, и начальник подразделения, где работает сотрудник. Это и есть пример интеграции данных в БД. Но каждый из них имеет дело и с другой информацией о сотруднике. Например,

бухгалтер обрабатывает табель для сотрудника, начисляет ему зарплату, вычисляет налоги и т.д.

В базе данных необходимо обеспечить независимость приложений от данных по двум причинам:

1) в разных приложениях одни и те же данные необходимо представлять по-разному;
2) в БД должна иметься возможность изменять структуру хранения или стратегию доступа к данным так, чтобы в случае изменения требований не надо было модифицировать приложения.

В связи с этим говорят о логической и физической независимости данных. *Логическая независимость* данных означает, что общая логическая структура – схема данных – может быть изменена без изменения всех приложений. *Физическая независимость* данных означает, что физическое расположение и организация данных во внешней памяти компьютера могут изменяться, не вызывая при этом изменений ни схемы данных, ни приложений.

Пример 2. В информационной системе «Предприятие» в сведениях о сотруднике появился такой новый элемент данных как ИИН (индивидуальный идентификационный номер). Этот элемент необходимо ввести в схему данных и в подсхемы для тех приложений, которые будут использовать ИИН при решении ряда задач (например, подсистема ИС «Кадры»). Все остальные приложения будут работать по-прежнему, поскольку данное изменение их не касается. Это и есть пример логической независимости данных.

Пример 3. Если для какого-то приложения необходимо использовать в сведениях о сотруднике в качестве первичного ключа поле «ИИН» или использовать в качестве вторичного ключа поле «Должность», то изменения во внешней памяти компьютера, где хранится база данных, коснутся только этого приложения. Все остальные приложения будут работать по-прежнему, поскольку данное изменение их не касается. Это и есть пример физической независимости данных.

В БД должно обеспечиваться отсутствие *дублирования* или избыточности данных. Фактически БД можно определить как неизбыточную совокупность данных. Однако при обработке для уменьшения времени доступа или упрощения адресации и поиска данных во многих БД избыточность присутствует в некоторой степени. Приходится идти на компромисс для обеспечения более быстрой обработки или восстановления данных в случае сбоев. Поэтому говорят об управляемом или минимальном уровне дублирования данных.

Пример 4. В примере 6 п.1.2 описаны таблицы ЗАВОД и ЭКОНОМИКА ЗАВОДА. Общим полем (полем связи) в них является поле «НАИМЕНОВАНИЕ». Наличие общего поля позволяет организовать связь между таблицами и сделать обработку данных в этих таблицах более эффективной. Это и есть пример дублирования или избыточности данных.

Защита данных означает, во-первых, предупреждение несанкционированного или случайного доступа к данным, их изменения или разрушения со стороны пользователей, во-вторых, предупреждение изменения

или разрушения данных при сбоях технических или программных средств и ошибках в работе администратора БД. Зашита данных обеспечивает их безопасность и секретность. Под *функцией безопасности* понимается защита данных от непреднамеренного доступа к данным и от возможности их искажения со стороны пользователей или лиц, выполняющих эксплуатацию банка данных, а также при сбоях в технических и программных средствах. Обеспечение безопасности – внутренняя задача банка данных.

Под *функцией секретности* понимается защита данных от преднамеренного доступа к данным пользователей или лиц, выполняющих эксплуатацию банка данных, или посторонних лиц с целью получения секретной информации или преднамеренного изменения и искажения данных. Как правило, в БД данные делятся на общедоступные и секретные. Обеспечение секретности – внешняя задача банка данных, решение вопросов секретности находится в компетенции юридических и административных органов банка данных.

Целостность данных означает безошибочность, точность и достоверность данных в БД в каждый момент времени. Целостность обеспечивается набором специальных правил, устанавливающих допустимость данных и связей между ними, называемых ограничениями целостности. Ограничения целостности могут относиться к различным объектам БД: атрибутам, записям, отношениям, связям между ними и т.п. Например, для атрибутов могут быть установлены следующие виды ограничений целостности:

1) заданный тип и формат атрибута автоматически допускают ввод данных только указанного типа; например, если атрибут имеет тип «Дата» в формате ДД.ММ.ГГ (день – месяц – год), то число дней не может превышать 31, а число месяцев – 12;
2) задание диапазона значений, как правило, используется для числовых полей; например, *стоимость товара должна быть > 200* или *стоимость товара должна быть >100 и <500*;
3) недопустимость пустого значения для обязательных атрибутов; например, обязательно должен быть задан номер автомобиля или фамилия сотрудника;
4) задание списка значений используется, как правило, для текстовых (символьных) атрибутов, например, наименование должности сотрудника лучше выбирать из заранее созданного списка.

Пример 5. Приведем пример задания ограничений целостности (ограничения вида 2 и 3) в СУБД Access для таблицы АВТО, содержащей сведения об автомобиле. Для поля «Мощность» этой таблицы на закладке «Общие» в Конструкторе заданы такие свойства, как «Подпись», «Значение по умолчанию», «Условие на значение», «Сообщение об ошибке» и «Обязательное поле» (рис. 1.8). Эти свойства и задают ограничения целостности. Сколько бы времени не использовалась данная таблица, всегда при вводе или изменении значения в поле «Мощность» будет идти проверка на принадлежность значения указанному диапазону (свойство «Условие на значение»: *>= 5 and <=30*) и проверка на наличие значения (свойство «Обязательное поле»: *да*). Если,

например, пользователь введет значение *40* в это поле, то сработает свойство «Сообщение об ошибке» и появится сообщение «Мощность авто от 5 до 30 лошадиных сил».

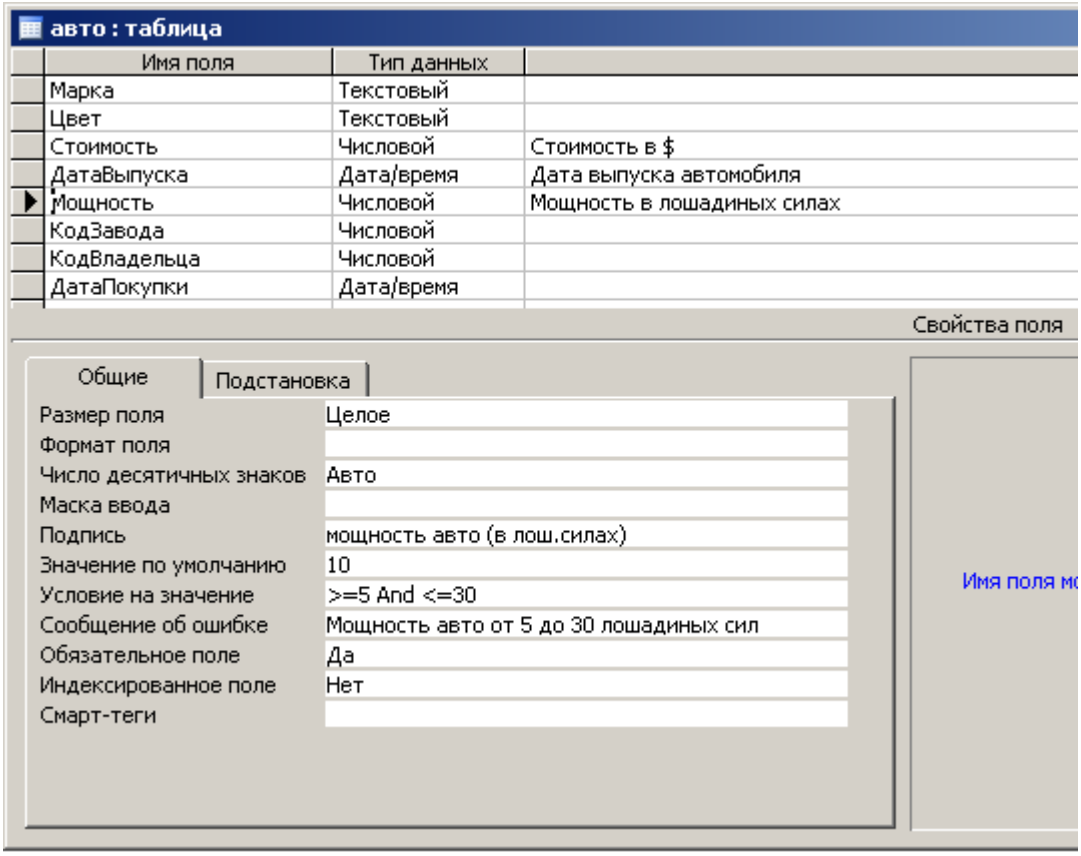


Рис. 1.8. Свойства для поля «Мощность» таблицы АВТО

Пример 6. Приведем пример задания ограничений целостности (ограничение вида 4) в СУБД Access для поля «Марка» таблицы АВТО (рис. 1.9). При работе с данными этой таблицы будет открываться список марок автомобилей, созданный на основе справочной таблицы «Марка» (рис. 1.10).

Для реализации ограничений целостности, имеющих отношение к записям, таблицам или связям между ними, в СУБД используются триггеры. *Триггер* – это предварительно определенное действие или последовательность действий, автоматически осуществляемых при выполнении операций обновления, добавления или удаления данных. Триггер является мощным инструментом контроля за изменением данных в БД и помогает программисту автоматизировать операции, которые должны выполняться в этом случае. Триггер включает в себя следующие компоненты:

1) ограничения, для реализации которых и создается триггер;
2) событие, которое характеризует возникновение ситуации, требующей проверки ограничений целостности;
3) действие, выполняемое за счет одной или нескольких процедур, в которых заложена логика реализации ограничений целостности.

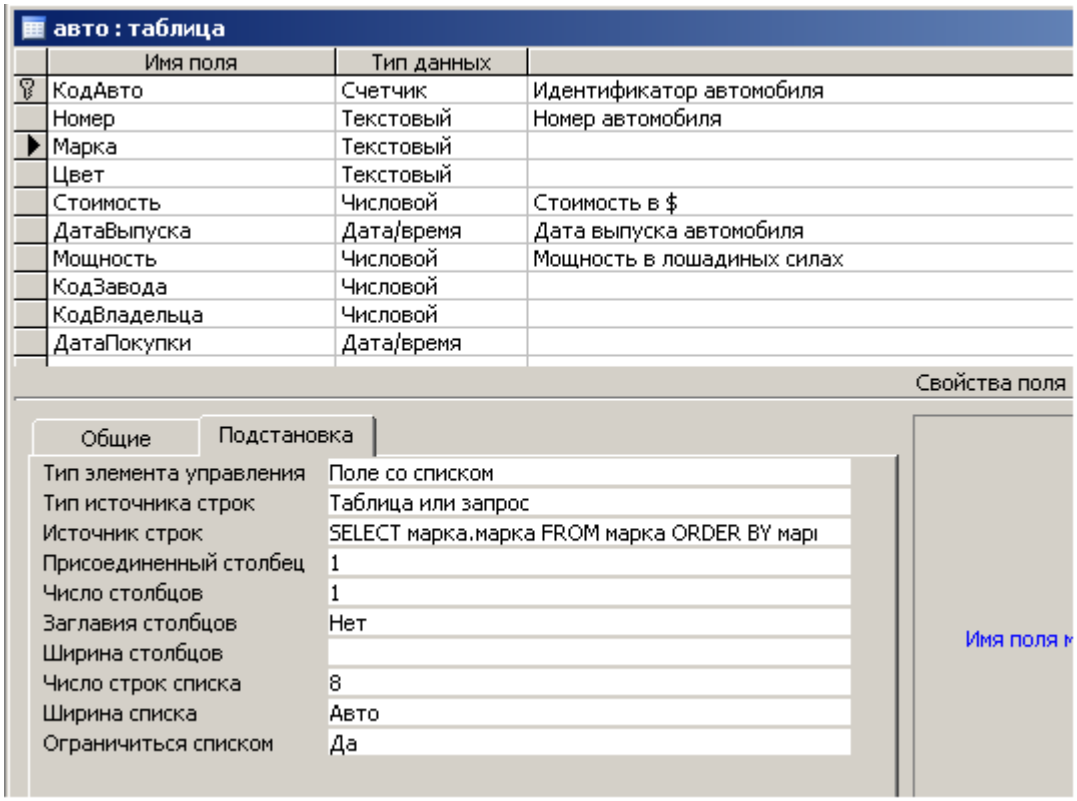


Рис. 1.9. Пример задания списка значений для поля «Марка»

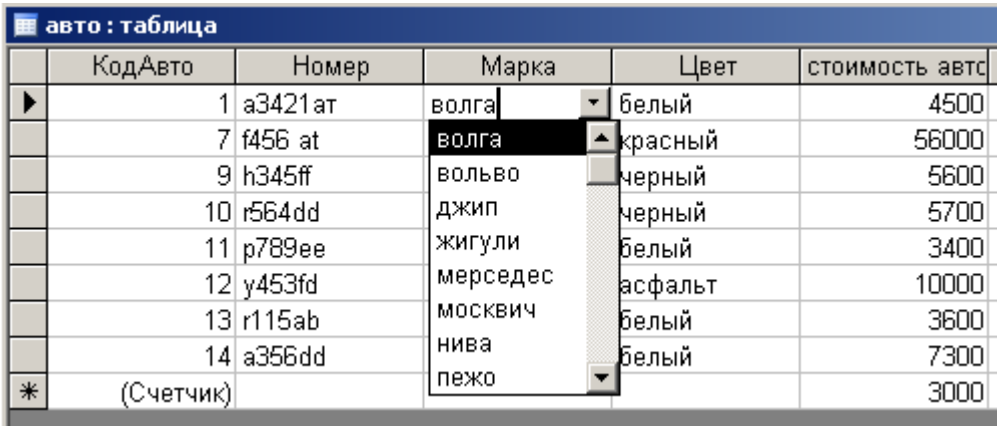


Рис. 1.10. Пример использования списка значений для поля «Марка»

Обеспечение ссылочной целостности данных при установлении взаимосвязей (вида 1:1 или 1:М) между таблицами БД означает выполнение для взаимосвязанных таблиц следующих условий корректировки:

- в подчинённую таблицу не может быть добавлена запись с несуществующим в главной таблице значением ключа связи;
- в главной таблице нельзя удалить запись, если не удалены связанные с ней записи в подчинённой таблице;
- изменение значений ключа связи главной таблицы должно приводить к изменению соответствующих значений в записях подчинённой таблицы.

Для проведения последних двух действий в СУБД могут иметься специальные режимы. Так в режиме каскадного обновления связанных записей при изменении значения в поле связи главной таблицы СУБД автоматически изменит значения в соответствующем поле в подчинённых записях. В режиме каскадного удаления связанных записей при удалении записей из главной таблицы СУБД выполняет каскадное удаление подчинённых записей на всех уровнях.

1.7. Схема и подсхема

Схема представляет собой *логическое описание* всех хранимых данных. *Схема* содержит имена объектов и их атрибутов, а также взаимосвязи между объектами. В современных СУБД схема данных задает структуру базы данных и является не только графическим образом БД, но используется СУБД в процессе работы с БД. В схеме определяются и заполняются связи между таблицами. Это позволяет СУБД автоматически использовать связи данных при конструировании форм, запросов, отчетов на основе взаимосвязанных таблиц.

Схема БД отображается графически в виде таблиц, представленных списками полей, а связи представлены линиями между взаимосвязанными полями различных таблиц. Взаимосвязь 1:1 устанавливается между ключевыми полями главной и подчиненной таблиц. Взаимосвязь 1:М устанавливается между ключевым полем главной таблицы и неключевым полем подчиненной таблицы. Пример схемы данных для информационной системы «Автомобили», включающей три таблицы «Авто», «Завод», «ВладелецАвто», приведён на рис. 1.11.

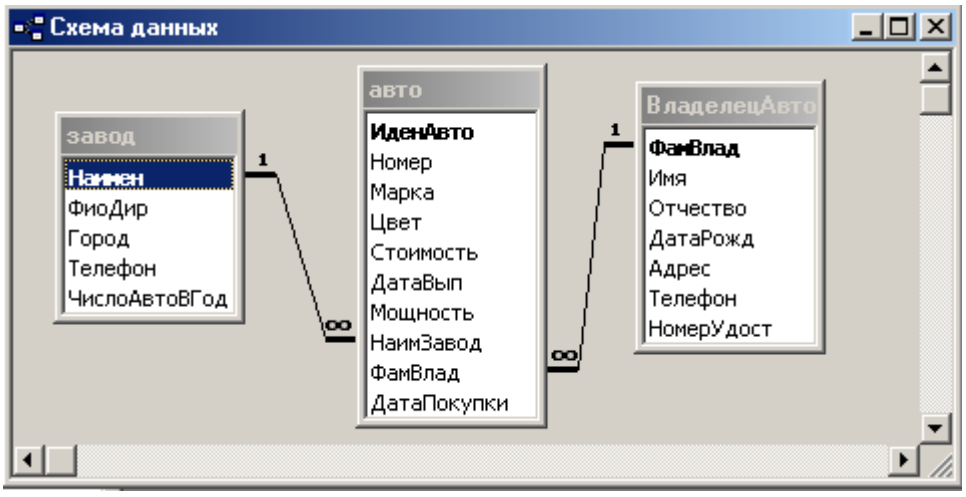


Рис. 1.11. Схема данных

Между таблицами «Завод» и «Авто» определена взаимосвязь 1:М. Аналогичная взаимосвязь установлена между таблицами «ВладелецАвто» и «Авто». Тип связи определяется при использовании флажка «Обеспечение целостности данных». На рис. 1.12 представлено окно связи для таблиц

«ВладелецАвто» и «Авто», где реализована ссылочная целостность путем использования флажков «Каскадное обновление связанных полей» и «Каскадное удаление связанных записей».

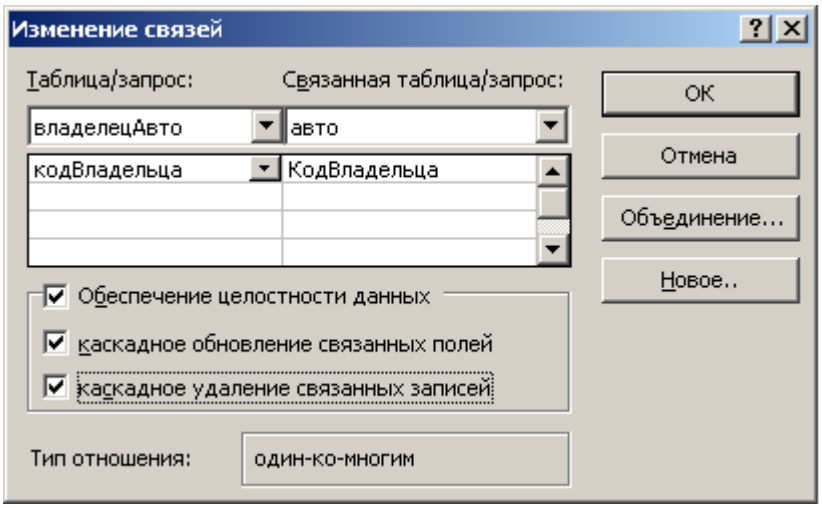


Рис. 1.12. Окно связи для таблиц «ВладелецАвто» и «Авто»

Создание схемы БД позволяет упростить конструирование многотабличных форм, запросов и отчетов в информационной системе, а также обеспечить целостность взаимосвязанных данных при корректировке таблиц.

При выборе в качестве поля связи в главной и подчиненной таблицах неключевого поля возможно установление только *связи-объединения*. Эти связи необходимы при формировании форм и отчетов, содержащих поля как из главной, так и из подчиненной таблиц. Такие связи обеспечивают объединение записей, имеющих одинаковые значения в поле связи, одним из 3-х способов:

1) объединение только тех записей, в которых связанные поля обеих таблиц совпадают;

2) объединение тех записей, в которых связанные поля обеих таблиц совпадают, а также объединение всех записей из 1-й таблицы, для которых нет связанных во 2-й, с пустой записью 2-й таблицы;

3) объединение тех записей, в которых связанные поля обеих таблиц совпадают, а также объединение всех записей из 2-й таблицы, для которых нет связанных в 1-й, с пустой записью 1-й таблицы.

Пример схемы, где кроме связей 1:М имеется и связь-объединение, приведен на рис. 1.13. Связь-объединение реализуется между неключевыми полями «номерГруппы» таблиц СТУДЕНТ и ЭКЗАМЕН.

Подсхема определяет собой описание данных, которые используются каким-либо приложением или пользователем. На основе одной схемы можно построить множество подсхем. Примеры подсхем для представленной на рис. 1.11 схемы:

1) подсхема 1 – Автомобиль (Марка, Стоимость, Цвет) – сформирована на основе таблицы «Авто»;

2) подсхема 2 – Автомобили (Номер, Марка, Стоимость, Цвет, Фамилия Владельца, Телефон, Адрес) – сформирована на основе двух таблиц «Авто» и «ВладелецАвто».

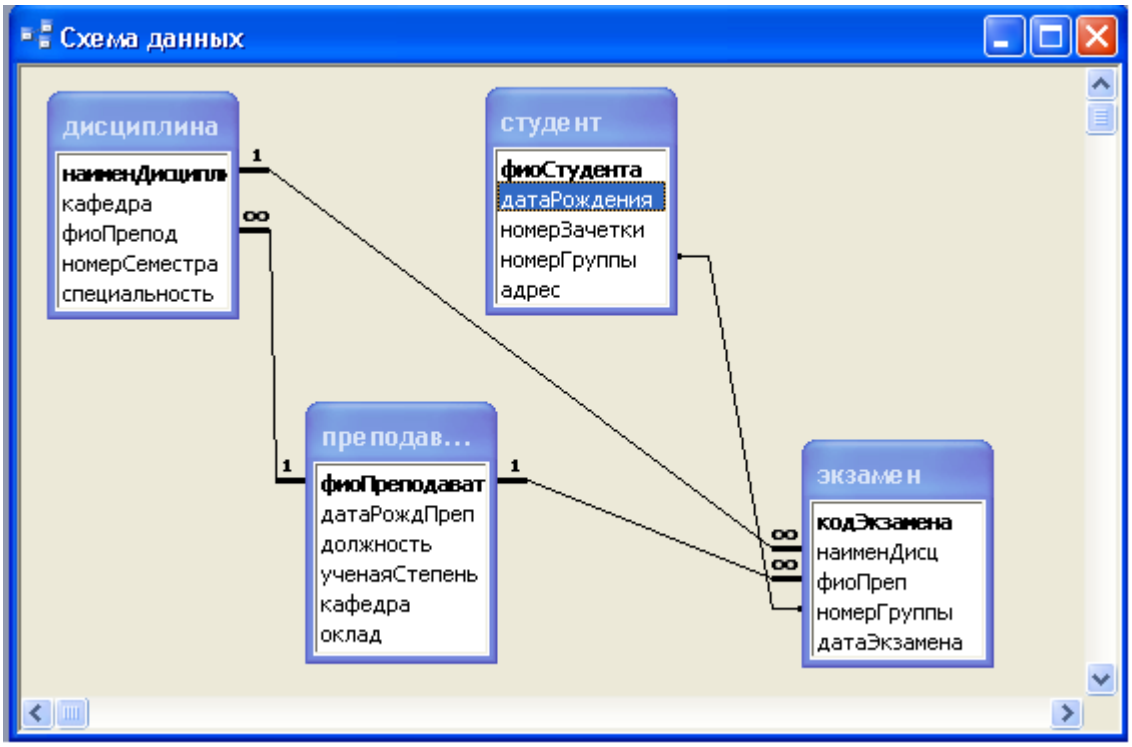


Рис. 1.13. Пример схемы со связью-объединение

1.8. Пользователи банка данных

С базой данных взаимодействуют несколько типов пользователей: проектировщик БД, прикладной программист, конечный пользователь (непрограммист), администратор базы данных (АБД).

Проектировщик БД решает все вопросы, связанные с созданием БД, выбором и использованием СУБД для проектируемой информационной системы. Кроме того, в ходе эксплуатации БД проектировщик может принимать участие в решении вопросов по дальнейшему совершенствованию базы данных.

Прикладной программист разрабатывает приложения (прикладные программы, подсистемы ИС), использующие данные из БД и предназначенные для пользователей БД. Прикладной программист создает подсхему для разрабатываемого приложения и пишет запросы к СУБД на языке манипулирования данными в соответствии с принятыми правилами и нормами. При этом он работает в тесном контакте с администратором БД.

Конечный пользователь обращается в банк данных (в информационную систему) с запросами на поиск требуемых данных. Эти пользователи имеют разный уровень профессиональной подготовки и, как правило, не являются профессионалами в области информационных технологий. Они используют язык запросов пользователей, который является языком, близким к

естественному узко профессиональному языку для рассматриваемой предметной области.

Администратор базы данных отвечает за создание и ведение БД. АБД включает в свой состав группу сопровождения, группу контроля функционирования банка данных, эксперта по языкам запросов, эксперта по разработке приложений, эксперта по системным вопросам, эксперта по вопросам эксплуатации. В функции администратора БД входят:

- координация всех действий по проектированию, реализации и ведению БД, учет перспективных и текущих требований всех пользователей с целью удовлетворения их актуальных информационных потребностей;
- разработка и реализация мер по обеспечению защиты данных от некомпетентного их использования, от сбоев технических и программных средств;
- разграничение доступа к данным и обеспечение их секретности;
- контроль за избыточностью и противоречивостью данных, обеспечение их достоверности;
- проведение при необходимости реорганизации и реструктуризации данных;
- координация работы системных и прикладных программистов.

Структура АБД представлена на рис. 1.14.

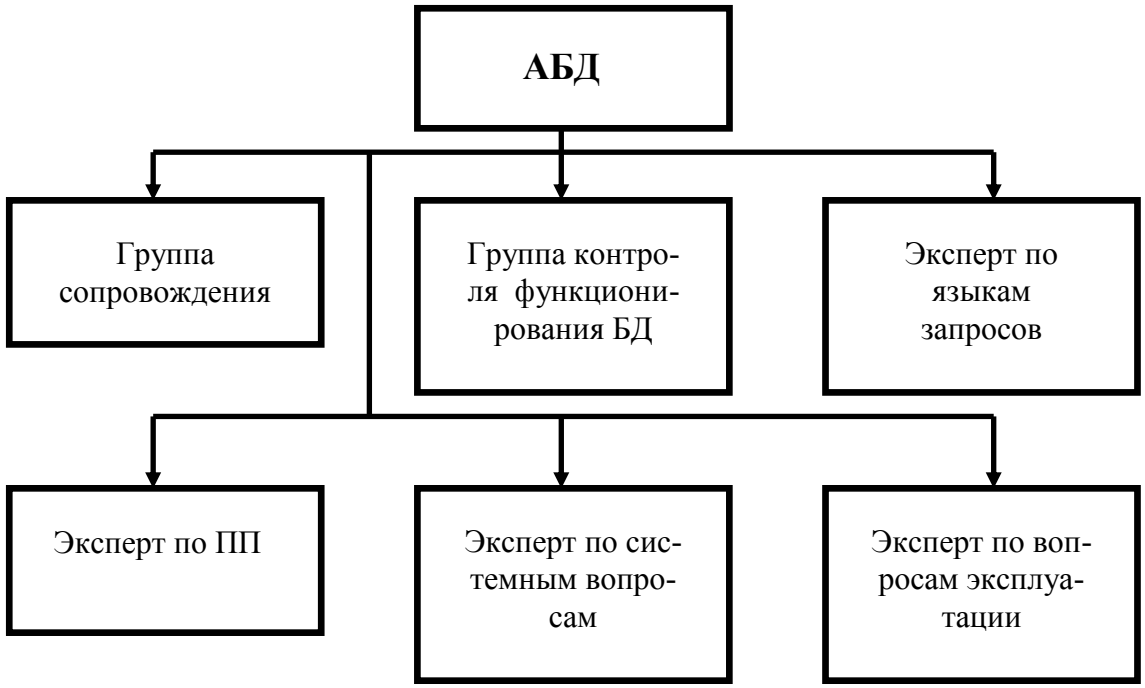


Рис. 1.14. Структура АБД

1.9. Языки, используемые в банке данных

В банке данных используется 4 типа языков: язык описания данных (ЯОД), язык манипулирования данными (ЯМД), язык запросов пользователя (ЯЗП) и базовый язык (БЯ).

Язык описания данных является языком декларативного типа, используется для логического описания данных и поддерживает функции, позволяющие присваивать уникальные имена каждому элементу данных, идентифицировать типы элементов данных (поле, сегмент, запись, набор, отношение, атрибут, таблица и др.), специфицировать порядок вхождения одних элементов данных в другие, устанавливать существующие взаимосвязи между данными.

Язык манипулирования данными позволяет реализовать интерфейс между приложением и СУБД. ЯМД поддерживает такие функции, как открыть / закрыть файл базы данных, найти требуемые элементы данных, изменить / добавить / удалить некоторые данные и т.д. Фактически ЯМД выполняет все функции по обработке данных: загрузка, корректировка, поиск.

Язык запросов пользователя позволяет выбирать из базы все требуемые пользователю данные в соответствии с задаваемыми им критериями. Этот язык должен быть близок к естественному языку для рассматриваемой предметной области.

Базовый язык представляет собой универсальный язык программирования, в среде которого реализована СУБД (например, язык С++).

Схема обработки запросов с использованием различных языков в банке данных представлена на рис. 1.15. При обработке запросов задействованы процессор ЯОД, процессор ЯМД и процессор языка запросов пользователя.

1.10. Уровни представления данных

Предметную область реального мира принято рассматривать в виде трех представлений:

1 – представление предметной области в том виде, в каком оно реально существует;

2 – как его воспринимает человек;

3 – как оно может быть описано в компьютере.

В связи с этим говорят о трех уровнях представления данных: внешнем, концептуальном и внутреннем.

Внешнее представление является совокупностью требований к данным некоторого конкретного приложения или пользователя. Целесообразно разделить понятие внешнего представления на *пользовательское внешнее представление*, которое отображает конкретные функциональные потребности пользователей и предполагаемое их обеспечение, и *локальное внешнее представление*, которое отображает элементы данных и их взаимосвязи, подлежащие хранению в БД.

Например, какому-то приложению требуются данные о возрасте сотрудников – это пользовательское внешнее представление. А в БД лучше хранить дату рождения сотрудников – это локальное внешнее представление. На основе даты рождения сотрудников и текущей даты всегда можно вычислить их возраст.

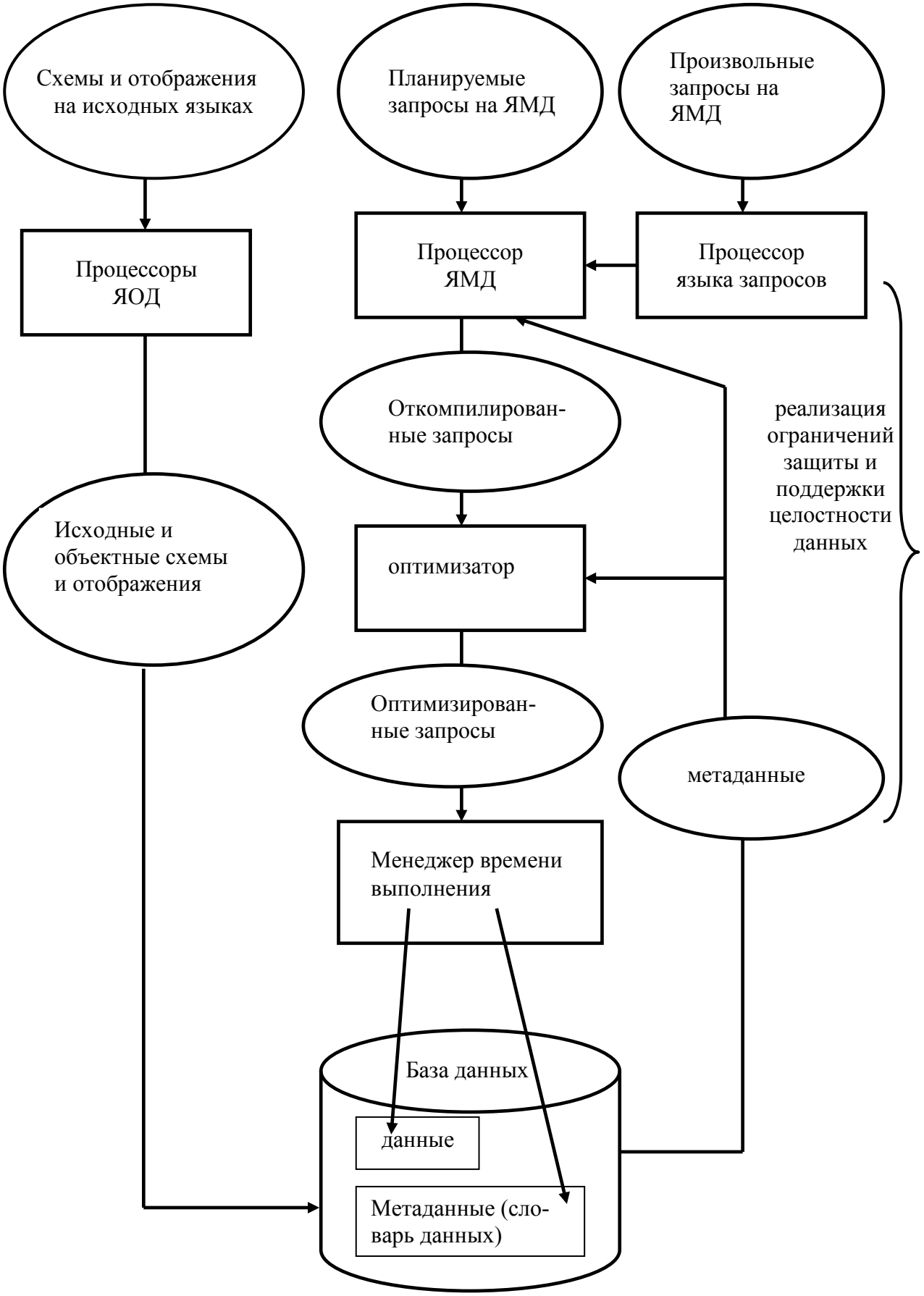


Рис. 1.15. Схема обработки запросов

Каждому внешнему представлению соответствует *внешняя модель* или подсхема. Существует множество внешних представлений – для каждого пользователя или приложения (ПП, подсистемы ИС) формируются свои требования на необходимые данные.

Концептуальное представление – это полная совокупность всех требований к данным, полученным из пользовательских представлений о предметной области реального мира, необходимых для реализации всех функций информационной системы. Концептуальное представление практически есть модель предметной области, абстракция реального мира. Оно служит основой для проектирования БД, обеспечивающей реализацию пользовательских представлений данных. Концептуальное представление реализуется в виде *концептуальной модели*, в которой содержатся объекты, их атрибуты и взаимосвязи между объектами. Концептуальную модель также называют СУБД-независимой схемой данных.

Внутреннее представление – это сама реальная база данных, реализованная в памяти компьютера средствами выбранной СУБД. Внутреннему представлению соответствует *внутренняя модель*, которая включает логическую модель и физическую модель. *Логической моделью* является концептуальная модель, реализованная средствами выбранной СУБД. Фактически это схема данных, описанная операторами ЯОД. Поэтому логическую модель называют СУБД-ориентированной схемой данных. На основе логической модели формируется *физическая модель*, в которой специфицировано размещение файлов БД, методы доступа и техника индексирования данных.

На рис. 1.16 представлена структура банка данных с уровнями представления данных.

1.11. Основные операции над данными в базе

Различают следующие основные операции, выполняемые над данными в БД: загрузка, корректировка, поиск, реорганизация, реструктуризация.

Под *загрузкой* понимается первоначальное заполнение БД данными. Особенностью этой операции являются большие объемы вводимых данных и преобладание операций запоминания данных. Загрузка обычно занимает много времени и выполняется один раз, до начала эксплуатации БД. Контроль данных должен осуществляться на всех стадиях загрузки:

- при первичной подготовке данных необходимо проводить поиск ошибок в документах;
- при проведении загрузки должны быть выявлены дублирующие данные, некорректные значения, несогласованность вводимых данных с уже хранящимися в БД данными;
- необходимо выявлять дублирующие значения среди первичных ключей;
- после загрузки очередной порции данных необходимо проверять БД на полноту и достоверность.

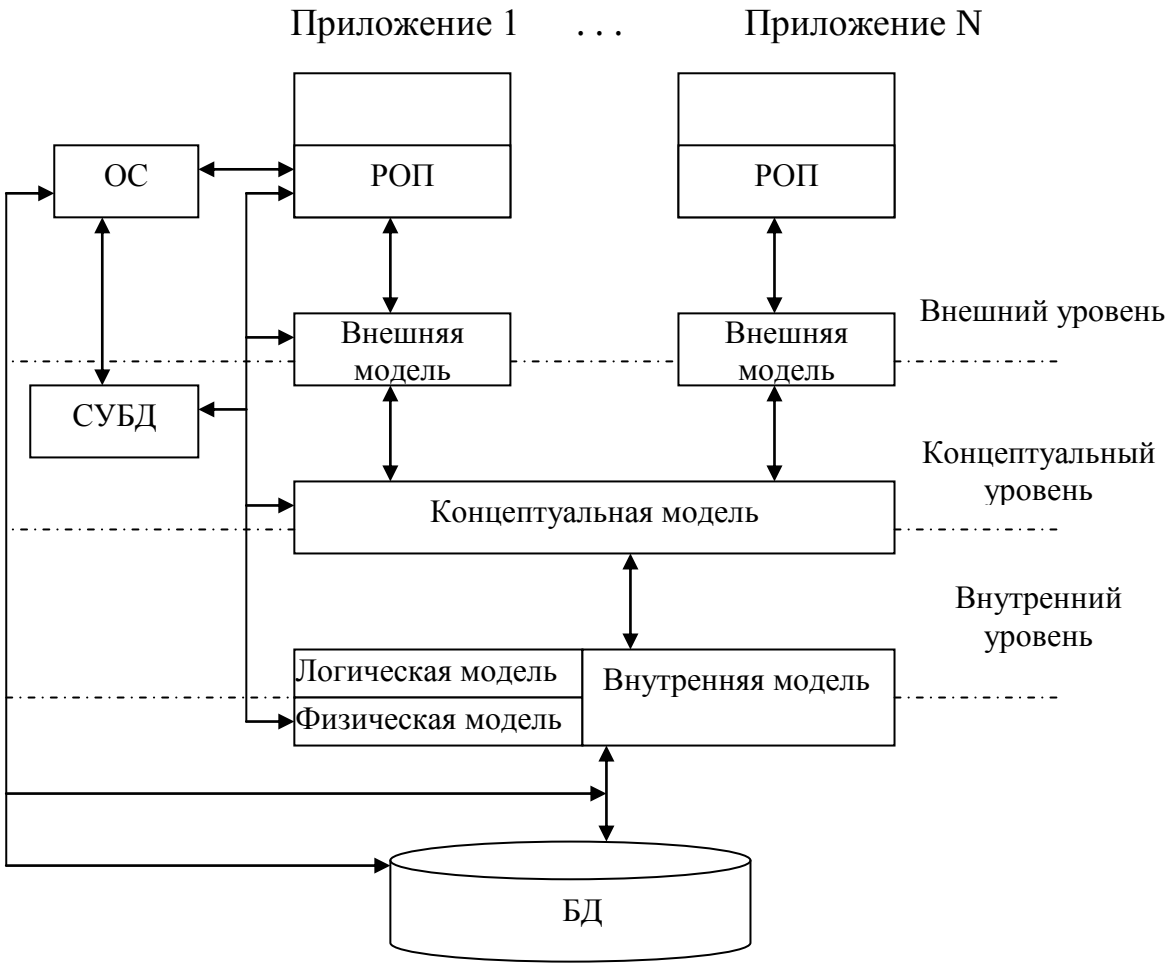


Рис. 1.16. Уровни представления данных

В ходе выполнения загрузки СУБД должна выводить на печать сообщения о выявленных ошибках или сообщать о нормальном ходе загрузки.

Корректировка позволяет изменять содержимое БД в ходе эксплуатации информационной системы. Корректировка вызывается несколькими причинами:

- необходимо исправлять ошибки, обнаруженные в данных в БД;
- необходимо проводить текущие изменения в данных и добавлять в БД новые данные в соответствии с изменениями в предметной области;
- необходимо уничтожать устаревшие или ненужные данные в БД.

В запросе на корректировку следует указать критерий отбора данных из БД, подлежащих корректировке, и указать новые значения, которые необходимо ввести в эти данные. При этой операции можно изменять как часть атрибутов, так и описание объекта целиком. Особенностью этой операции является регулярность ее выполнения.

Пример 1. Рассмотрим пример запросов на корректировку для информационной системы «Автомобили». Пусть имеется таблица «авто», включающая такие поля как «ИденАвто», «Номер», «Марка», «Цвет», «Стоимость» и другие (см. рис. 1.11). Составим следующие запросы на корректировку:

1) изменить стоимости всех автомобилей белого цвета, список новых стоимостей прилагается;
2) изменить мощность и дату выпуска для всех автомобилей черного цвета и марки «Ауди», список новых значений прилагается;
3) удалить сведения об автомобилях марки «Нива»;
4) изменить поля «Номер», «Марка» и «НаимЗавод» для автомобилей стоимостью > 10000 или красного цвета, список новых значений прилагается;
5) изменить марку и цвет автомобилей с идентификационными номерами от 31 до 35, список новых значений прилагается.

Поиск позволяет выбирать данные из БД по определенным критериям и представлять их в требуемом пользователю ИС виде. При реализации этой операции определить виды запросов на поиск, методы и алгоритмы поиска, способы задания критериев отбора данных и формы представления результатов поиска. После поступления соответствующего запроса СУБД осуществляет поиск данных в БД, выдает требуемые данные пользователю или печатает сообщение об отсутствии таких данных в БД. В ходе поиска СУБД может печатать сообщения об обнаруженных в запросе ошибках.

Пример 2. Рассмотрим примеры запросов на поиск по заданному критерию для информационной системы «Автомобили» на основе представленной выше таблицы «авто»:

1) найти сведения обо всех автомобилях марки «Волга»;
2) найти значения полей «ИденАвто», «Номер», «Марка» для всех автомобилей черного цвета;
3) найти сведения обо всех автомобилях белого или черного цвета;
4) найти значения полей «ФамВлад» и «ДатаПокупки» для автомобилей стоимостью < 12000 и красного цвета;
5) найти значение поля «Номер» для всех автомобилей стоимостью от 3000 до 10000.

В ходе эксплуатации время от времени базу данных необходимо приводить в порядок, так как в результате удаления некоторых данных между действительными записями образуется неиспользуемая память. Кроме того, может иметь место так называемая фрагментация памяти, когда внешняя память становится разбитой на отдельные небольшие кусочки. Практически *реорганизация* позволяет получить копию БД, в которой все данные размещены строго последовательно без свободных промежутков памяти. Решение о необходимости реорганизации принимает администратор базы данных в зависимости от конкретных условий эксплуатации. Основными причинами выполнения этой операции являются повышенная активность включений и удалений записей в БД, ошибки со стороны СУБД при определении свободной памяти, неверный выбор способа организации данных или метода доступа к ним.

После определенного периода эксплуатации ИС и БД некоторые представления о предметной области в динамически изменяющемся реальном мире могут измениться по сравнению с принятыми на стадии проектирования

БД. В этом случае администратор базы данных принимает решение о проведении *реструктуризации* БД, которая может включать изменение логической и внутренней моделей, изменение методов доступа к данным и процедур работы с БД. Различают реструктуризационные изменения трех типов:

1) процедурные изменения, когда появляется новая версия СУБД и могут измениться различные процедуры обработки данных (процедуры управления доступом к данным, процедуры восстановления данных и т.д.);

2) изменения на физическом уровне, касающиеся изменений в составе технических средств, изменений размеров используемой памяти и размеров системных буферов, изменения методов поиска и методов доступа к данным;

3) изменения на логическом уровне, когда может быть изменена логическая структура хранимых данных (добавление / удаление элементов данных или описание новых объектов, изменение взаимосвязей между данными).

Реструктуризация по своим затратам аналогична загрузке, так как требует перепроектирования базы данных, перепрограммирования и отладки части или всех приложений в информационной системе.

1.12. Задания и контрольные вопросы

Для закрепления рассмотренного теоретического материала и получения практических навыков по его применению необходимо выполнить задания, представленные ниже, и ответить на контрольные вопросы.

Задание 1.

1. Выберите предметную область.
2. Перечислите объекты из этой предметной области.
3. Определите свойства для каждого объекта.
4. Определите, какими являются эти свойства: качественными, количественными, описательными.

Задание 2.

1. Определите информационную систему, которую необходимо создать.
2. Определите предметную область из реального мира для создания базы данных, которая будет служить основой информационной системы.
3. Выберите ряд важных для информационной системы объектов из этой предметной области.
4. Определите свойства для каждого объекта.
5. Определите функции информационной системы, которые должны обрабатывать данные по выбранным объектам, например, такие как добавление новых значений, изменение некоторых значений в соответствии с изменениями в реальном мире и т.д.

Задание 3.

1. Определите несколько объектов из одной предметной области.

2. Определите, какие взаимосвязи существуют между этими объектами.
3. Приведите пример каждого вида взаимосвязи между объектами.
4. Приведите пример для каждого вида взаимосвязи между атрибутами (свойствами) одного объекта.

Задание 4.

1. Изучите схему, отображающую процесс обработки запроса от приложения в банке данных (рис. 1.7).
2. Рассмотрите весь процесс обработки.
3. Объясните роль системного буфера.
4. Объясните, почему происходит обращение к операционной системе.
5. Объясните, почему стрелки 5, 6, 8 двойные.

Задание 5.

1. Задайте описание объекта и его свойств.
2. Определите ограничения целостности для каждого свойства объекта.
3. Определите условия на значения (диапазон значений) для ряда свойств.
4. Определите, для каких свойств следует использовать список значений.

Задание 6.

1. Задайте описание нескольких объектов и их свойств для информационной системы «Университет», например, объекты «Студент», «Факультет», «Преподаватель» и т.д.
2. Определите взаимосвязи между объектами.
3. Постройте схему данных для этой ИС.
4. Постройте несколько подсхем, используя элементы данных из нескольких таблиц.

Задание 7.

1. Изучите структуру банка данных с уровнями представления данных.
2. Объясните, какому компоненту из рис. 1.16 соответствует внешняя модель из рассматриваемой структуры.
3. Объясните, какому компоненту из рис. 1.16 соответствует логическая модель из рассматриваемой структуры.

Задание 8.

1. Задайте описание объекта и его свойств.
2. Определите конкретные значения свойств для 15 – 20 реальных объектов.
3. Определите несколько запросов на корректировку с указанием критерия отбора данных, подлежащих корректировке, и заданием новых значений для них. Покажите, какими станут исходные данные после проведения корректировки.
4. Определите несколько запросов на поиск с указанием критерия поиска данных в базе. Составьте ответы на эти запросы.

Контрольные вопросы

1. Какие три области можно выделить при обсуждении понятия «информация»?
2. Что такое предметная область?
3. Назовите примеры предметных областей, объектов и их свойств?
4. Что может являться объектом?
5. Существуют ли отношения между объектами?
6. Какие виды отношений между объектами можете назвать?
7. Какие примеры отношений между объектами можете назвать?
8. Какие типы взаимосвязей можете назвать?
9. Что такое база данных?
10. Что такое система баз данных?
11. Что такое банк данных?
12. Каковы отличительные черты банка данных?
13. Какие компоненты входят в банк данных?
14. Как соотносятся друг с другом банк данных и информационная система?
15. Что такое СУБД?
16. Какие функции обеспечивает СУБД?
17. Что такое схема?
18. Что такое подсхема?
19. Какие языки входят в состав языкового обеспечения банка данных?
20. Какие группы пользователей знаете?
21. Какие функции выполняет администратор базы данных?
22. Какие свойства данных поддерживаются в базе?
23. Как поддерживается целостность данных?
24. Какие уровни представления данных можно выделить?
25. Какие основные операции выполняются над данными?

Глава 2. МОДЕЛИ ДАННЫХ

2.1. Абстрактная модель данных

Как правило, данные (факты, значения) и их интерпретация (семантика) фиксируются совместно, так как естественный язык достаточно гибок для представления того и другого. *Пример*: «рост 180 см», «180» – данное, а его семантика – «рост в см». При компьютерной обработке данные отделяются от их интерпретации, что и приводит к понятию модели данных.

Модель данных – это абстрактное логическое определение объектов, операторов и других элементов, в совокупности составляющих абстрактную машину, с которой взаимодействуют пользователи. Объекты позволяют моделировать структуру данных, а операторы – моделировать поведение данных. Модель данных можно рассматривать как сочетание трех компонентов:

- структурная часть – набор правил, по которым может быть построена база данных;
- управляющая часть – определяет типы допустимых операций с данными, таких как обновление и извлечение данных, а также модификация структуры данных;
- набор ограничений поддержки целостности данных, гарантирующих корректность используемых данных.

Модель данных – это средство абстракции, которое дает возможность увидеть информационное содержание данных («лес»), а не конкретные значения данных («отдельные деревья»).

Рассмотрим модель реального мира и свойства, которые она отображает. Эти свойства делятся на два класса:

1) статические – свойства инвариантны ко времени, всегда справедливы и неизменны;
2) динамические – свойства соответствуют эволюционной природе мира.

Определим модель данных $\boldsymbol{M}$ как множество правил порождения $\boldsymbol{G}$ и множество операций $\boldsymbol{OP}$:

$$\boldsymbol{M = < G, OP >} \quad .$$

Множество правил порождения выражает статические свойства модели данных и соотносится с языком описания данных ЯОД. Средствами ЯОД определяются допустимые структуры данных – объекты и связи, а также допустимые реализации данных.

В некоторых моделях правила порождения $\boldsymbol{G}$ разделяются на два вида:

G_s – правила порождения структур,

G_c – правила порождения ограничений.

Пример: в СУБД Access для объекта «автомобиль» атрибут «гос. номер» является ключом. Тогда G_s (правило порождения структур) имеет вид: атрибут «гос. номер» принадлежит объекту «автомобиль». А G_c (правило порождения

ограничений) имеет вид: значения атрибута «гос. номер» не должны повторяться.

Динамические свойства выражаются множеством операций ***OP***, которые соотносятся с языком манипулирования данными ЯМД. Множество операций определяет допустимые действия над реализацией БД D_i для преобразования ее в другую реализацию БД D_j:

$$\{D_i, OP\} \rightarrow D_j \ .$$

При проектировании баз данных для различных ИС используются три основные модели представления данных: реляционная модель, иерархическая модель, сетевая модель.

2.2. Реляционная модель представления данных

Реляционная модель представления данных построена на системе отношений. Название идет от английского слова *relation* – отношение. Автором модели является *E. Codd* (Е. Кодд). В реляционной модели используются понятия **отношение – атрибут**. Объект представляется в виде отношения, а его свойства – в виде совокупности атрибутов. Базу данных можно представить в следующем виде:

$$R_1(A_{11}, A_{12},..., A_{1N1})$$
$$R_2(A_{21}, A_{22},..., A_{2N2})$$
$$\cdots$$
$$R_m(A_{m1}, A_{m2},..., A_{mNm})$$

где R_i, $i = \overline{1,m}$ – множество отношений, A_{ij}, $i = \overline{1,m}$, $j = \overline{1,Ni}$ – множество атрибутов. Экземпляр каждого отношения представляется в виде таблицы. Совокупность всех значений одного атрибута называется *доменом*. Строка в таблице называется *кортежем.* Практически это описание всех используемых свойств реального объекта.

Основной особенностью реляционной модели является то, что связи между кортежами представлены исключительно значениями данных в столбцах, полученных из общего домена. Обработка отношений осуществляется с помощью операций реляционной алгебры, предложенных Е. Коддом. Кроме того, можно использовать традиционные операции над множествами.

В качестве примера представления данных в виде реляционной модели рассмотрим описание объекта «студент». Пусть R – отношение «студент», которое включает следующие атрибуты: A_1 – номер зачетки, A_2 – фамилия студента, A_3 – дата рождения, A_4 – номер группы. В табл. 2.1 представлен экземпляр рассматриваемого отношения.

К *достоинствам* реляционной модели относятся простота и наглядность представления данных; наличие теоретического обоснования, так как модель основана на хорошо проработанной теории отношений; наличие

математического аппарата – реляционной алгебры. К *недостаткам* модели относится сложность реализации связей 1 : М и М : М, которая ведет к дублированию данных.

Таблица 2.1

Отношение *R*

A_1	A_2	A_3	A_4
112356	Ахметов	11.03.93	ИСб-11-1р
213478	Каримов	25.01.94	ВПб-12-3к
146732	Петров	03.05.94	ИСб-11-1р
…	…	…	…
120176	Ли	07.11.93	ИСб-12-2р

Примеры СУБД для реляционной модели: dBase, DB2, R:Base, FoxBase, FoxPro, Visual Foxpro, Clarion, Paradox, Access, Ingress, Oracle, ПАЛЬМА (ИК АН УССР), HyTech (МИФИ) и др.

2.3. Нормализация отношений

Нормализация отношений представляет собой процесс построения оптимальной структуры отношений и связей в реляционной базе данных. Теория нормализации основана на том, что определенное множество отношений обладает лучшими свойствами при добавлении, корректировке и удалении данных, чем другие множества отношений, с помощью которых могут быть представлены те же данные. Е. Кодд первоначально определил три уровня нормализации, которые назвал 1-ой, 2-ой и 3-ей нормальными формами (1НФ, 2НФ, 3НФ). В дальнейшем были определены еще три нормальные формы (НФБК, 4НФ, 5НФ). Наличие различных зависимостей в отношении приводит к сложности выполнения операций по обработке данных.

Определим эти нормальные формы. *Нормализованным* называется отношение, экземпляр которого представлен в виде таблицы и обладает следующими свойствами:

1) каждый элемент таблицы представляет собой один элемент данных (повторяющиеся группы отсутствуют), т.е. каждое значение в отношении является атомарным;
2) все столбцы – однородные, т. е. имеют одинаковую природу; столбцам однозначно присвоены имена (имена атрибутов);
3) нет двух одинаковых строк;
4) в операциях с такой таблицей строки и столбцы могут просматриваться в любом порядке и в любой последовательности безотносительно к их информационному содержанию и смыслу.

В нормализованном отношении каждый домен содержит только атомарные (неделимые) значения, и поэтому каждое значение в отношении в свою очередь

является атомарным. Ненормализованные отношения, встречающиеся на практике, должны быть преобразованы в отношения, находящиеся в одной из нормальных форм. Все нормализованные отношения находятся в 1НФ, некоторые отношения 1НФ находятся в 2НФ, некоторые отношения 2НФ находятся в 3НФ.

Отношение *R* находится в **1-ой нормальной форме**, если ни один из атрибутов, входящих в него, не является множеством (атрибут имеет только атомарные значения), и между атрибутами нет никакой нежелательной зависимости.

Введем понятие *функциональной зависимости*. Атрибут *B* отношения *R* функционально зависит от атрибута *A*, принадлежащего *R*, тогда и только тогда, когда каждое значение *A* в отношении *R* в каждый момент времени связано точно с одним значением *B*:

$$R.A \rightarrow R.B .$$

Если между атрибутами отношения выявлена функциональная зависимость, то избавиться от нее можно путем разбиения исходного отношения на два и более отношений, которые будут находиться во **2-ой нормальной форме**. Отношение *R* находится в 2НФ, если оно находится в 1НФ, и каждый неключевой атрибут функционально полно зависит от первичного ключа.

Пример 1. Пусть имеется отношение *R*, включающее в свой состав следующие атрибуты: A_1 – фамилия студента, A_2 – выпускающая кафедра, A_3 – местоположение кафедры. В табл. 2.2 представлен экземпляр этого отношения.

Таблица 2.2

Экземпляр отношения *R*

A_1	A_2	A_3
Ахметов	Информационные технологии	319 ГМК
Каримов	Информатика	808 ГУК
Петров	Информационные технологии	319 ГМК
Оспанов	Информатика	808 ГУК
Ли	Информационные технологии	319 ГМК
Скаков	Информационные технологии	319 ГМК

Как видно из табл. 2.2, между атрибутами A_2 и A_3 отношения *R* существует функциональная зависимость: $R.A_2 \rightarrow R.A_3$. Избавиться от нее можно, разбив исходное отношение на следующие два отношения: $R_1(A_1, A_2)$ и $R_2(A_2, A_3)$, которые перейдут в 2НФ. Экземпляры этих отношений приведены в табл. 2.3 и 2.4. Из табл. 2.4 видно, что если изменится местоположение кафедры, то провести корректировку хранимых данных проще в отношении R_2, чем в исходном отношении *R*.

Таблица 2.3

Отношение R_1

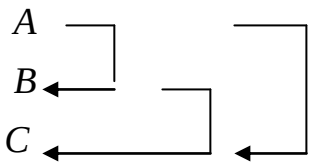
A_1	A_2
Ахметов	Информационные технологии
Каримов	Информатика
Петров	Информационные технологии
Оспанов	Информатика
Ли	Информационные технологии
Скаков	Информационные технологии

Таблица 2.4

Отношение R_2

A_2	A_3
Информационные технологии	319 ГМК
Информатика	808 ГУК

Введем понятие *транзитивной зависимости*. Пусть *A, B, C* – атрибуты отношения *R*. Если *C* зависит от *B*, а *B* зависит от *A* и при этом обратное соответствие неоднозначно (т.е. *A* не зависит от *B* или *B* не зависит от *C*), то говорят, что *C* транзитивно зависит от *A*:



Если между атрибутами отношения выявлена транзитивная зависимость, то избавиться от нее можно путем разбиения исходного отношения на два и более отношений, которые будут находиться в **3-ей нормальной форме.** Отношение *R* находится в 3НФ, если оно находится в 2НФ, и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Пример 2. Пусть имеется отношение *S*, включающее в свой состав следующие атрибуты: A_1 – номер зачетки (уникальный ключ), A_2 – фамилия студента, A_3 – дата рождения, A_4 – номер группы, A_5 – фамилия старосты, A_6 – адрес старосты. В табл. 2.5 представлен экземпляр этого отношения.

Как видно из табл. 2.5, атрибуты A_2, A_3, A_4 отношения *S* зависят от ключевого атрибута A_1, атрибут A_5 зависит от A_4, а атрибут A_6 зависит от A_5, т.е. между атрибутами A_4, A_5, A_6 существует транзитивная зависимость. Избавиться от нее можно, разбив исходное отношение на следующие два отношения: $S_1(A_1, A_2, A_3, A_4)$ и $S_2(A_4, A_5, A_6)$, которые перейдут в 3НФ. Экземпляры этих отношений приведены в табл. 2.6 и 2.7.

В настоящее время в теории нормализации БД введены еще три нормальные формы: нормальная форма Бойса – Кодда, 4-я и 5-я нормальные формы (НФБК, 4НФ, 5НФ).

Таблица 2.5

Отношение *S*

A_1	A_2	A_3	A_4	A_5	A_6
112356	Ахметов	11.03.93	ИСб-11-1р	Аманов	Абая 22-15
213478	Каримов	25.01.94	ВПб-12-3к	Иванов	Сатпаева 3-20
146732	Петров	03.05.94	ИСб-11-1р	Аманов	Абая 22-15
123155	Оспанов	09.06.93	ВПб-12-3к	Иванов	Сатпаева 3-20
120176	Ли	07.11.94	ИСб-11-1р	Аманов	Абая 22-15

Таблица 2.6

Отношение S_1

A_1	A_2	A_3	A_4
112356	Ахметов	11.03.85	ИСб-11-1р
213478	Каримов	25.01.84	ВПб-12-3к
146732	Петров	03.05.84	ИСб-11-1р
123155	Оспанов	09.06.85	ВПб-12-3к
120176	Ли	07.11.84	ИСб-11-1р

Таблица 2.7

Отношение S_2

A_4	A_5	A_6
ИСб-11-1р	Аманов	Абая 22-15
ВПб-12-3к	Иванов	Сатпаева 3-20

Отношение *R* находится в *нормальной форме Бойса – Кодда* только в том случае, если любая функциональная зависимость между его атрибутами сводится к полной функциональной зависимости от возможного ключа.

Пример 3. Пусть имеется отношение следующего вида, содержащее оценки студента: ***R(A1****, A2, A3, A4, A5, A6, A7)*, где ***A1*** – код оценки (ключевой атрибут), *A2* – номер зачетки студента, *A3* – дисциплина, по которой студент сдает экзамен, *A4* – номер группы студента, *A5* – фамилия преподавателя, *A6* – дата сдачи экзамена, *A7* – оценка, полученная студентом на экзамене. Атрибуты отношения *R* зависят только от ключевого атрибута ***A1***.

4-ая нормальная форма вводится в случае наличия в отношении многозначной зависимости. В отношении *R(A, B, C)* существует многозначная зависимость (*MVD-multi valid dependence*) $\boldsymbol{R.A \rightarrow\!\!\!\rightarrow R.B}$ 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*.

Теорема Фейджина. Отношение *R(A, B, C)* можно спроецировать без потерь в отношения *R1(A, B)* и *R2(A, C)* в том и только в том случае, когда существует *MVD* $\boldsymbol{A \rightarrow\!\!\!\rightarrow B \,/\, C}$ (что равнозначно наличию двух зависимостей $\boldsymbol{A \rightarrow\!\!\!\rightarrow B}$ и $\boldsymbol{A \rightarrow\!\!\!\rightarrow C}$).

Отношение *R* находится в 4НФ в том и только в том случае, если при существовании многозначной зависимости $\boldsymbol{A \rightarrow\rightarrow B}$ все остальные атрибуты в *R* функционально зависят от *A*.

Пример 4. Имеется отношение *R(A1, A2, A3)* – читаемые дисциплины, *A1* – наименование курса, *A2* – фамилия преподавателя, *A3* – название учебника. В табл. 2.8 представлен экземпляр этого отношения.

Таблица 2.8

Отношение *R*

A1	*A2*	*A3*
Физика	Каримов	Основы механики
Физика	Каримов	Основы оптики
Физика	Иванов	Основы механики
Физика	Иванов	Основы оптики

В отношении *R* имеет место многозначная зависимость, затрудняющая обработку: $\boldsymbol{A1 \rightarrow\rightarrow A2}$ и $\boldsymbol{A1 \rightarrow\rightarrow A3}$. Избавиться от нее можно, разбив исходное отношение на следующие два отношения: *R1(A1, A2)* и *R2(A1, A3)*, которые перейдут в 4НФ. Экземпляры этих отношений приведены в табл. 2.9 и 2.10.

Таблица 2.9

Отношение *R1*

A1	*A2*
Физика	Каримов
Физика	Иванов

Таблица 2.10

Отношение *R2*

A1	*A3*
Физика	Основы механики
Физика	Основы оптики

Отношении *R* находится в **5-ой нормальной форме** (*НФ проекция – соединение PJNF*) в том и только в том случае, когда любая зависимость соединения в *R* следует из существования некоторого возможного ключа в *R*.

5НФ редко используется на практике. В большей степени эта нормальная форма является теоретическим исследованием. Очень тяжело определить само наличие зависимостей «проекция – соединение», потому что утверждение о наличии такой зависимости делается для всех возможных состояний базы данных, а не только для текущего экземпляра отношения *R*.

Процесс нормализации включает следующие шаги для одного отношения:

- представление объекта в виде отношения в 1-й нормальной форме;
- определение наличия функциональной зависимости в отношении и устранение ее путем разбиения отношения на два и более отношений, которые перейдут в 2НФ;
- определение наличия транзитивной зависимости в отношении и устранение ее путем разбиения отношения на два и более отношений, которые перейдут в 3НФ;
- устранение других зависимостей в отношении путем перехода в НФБК, 4НФ, 5НФ.

На практике достижение 3НФ считается достаточным. Нормальные формы высших порядков связаны не с функциональными зависимостями между атрибутами отношения, а отражают более тонкие вопросы семантики предметной области и связаны с другими видами зависимостей.

2.4. Реляционная алгебра

Систему операций, используемую для манипулирования отношениями, называют *реляционной алгеброй* (РА). Операция в РА имеет одно или два отношения в качестве операндов и образует новое отношение по определенному правилу. Реляционная алгебра включает 6 операций: перестановка, проекция, соединение, ограничение, композиция, деление. Операции «перестановка» и «проекция» являются одноместными операциями, поскольку выполняются над одним отношением. Остальные операции являются двухместными операциями, поскольку выполняются над двумя отношениями. Рассмотрим эти операции.

1) **Перестановка** – позволяет изменить порядок следования атрибутов в отношении. Пусть R – отношение, включающее n атрибутов. $L = \{i_1, i_2, ..., i_n\}$ – новый порядок следования атрибутов, r – строка из R, i_j, $j = \overline{1,n}$ – номер атрибута из R.

$$S = R[L] = \{r[L] \mid r \in R\} - \text{новое отношение.}$$

2) **Проекция** – позволяет выбрать из отношения атрибуты, указанные в списке L. Пусть R – отношение, включающее n атрибутов. $L = \{i_1, i_2, ..., i_k\}$ – новый порядок следования атрибутов, i_j, $j = \overline{1,k}$, $\boldsymbol{k < n}$ – номер атрибута из R.

$$S = R[L] = \{r[L] \mid r \in R\} - \text{новое отношение.}$$

3) **Соединение** – позволяет получить новое отношение из 2-х исходных отношений (R и S), имеющих одинаковые атрибуты ($A \in R$ и $B \in S$), по которым и происходит соединение строк отношений.

$$Q = R[A \Theta B]S = \{(r,s) \mid r \in R \wedge s \in S \wedge (r[A] \Theta s[B])\} - \text{новое отношение,}$$

где $\Theta = \{=, \neq, >, \geq, <, \leq\}$ – знаки сравнения, *r* – строка из *R*, *s* – строка из *S*, *r[A]* – значение атрибута *A* в отношении *R*, *s[B]* – значение атрибута *B* в отношении *S*, $\wedge$ – логическое «И».

4) **Ограничение** – позволяет выбрать из исходного отношения *R* только те строки, для которых выполняются условия, указанные в отношении *S*.

$Q = R[A\Theta B]S = \{r \mid r \in R \wedge (r[A]\Theta s[B])\}$ – новое отношение.

5) **Композиция** – аналогична «соединению», но в новое отношение не входят атрибуты, по которым проходило соединение.

$Q = R[A\Theta B]S = \{(r^1, s^1) \mid r^1 \in R \wedge s^1 \in S \wedge (r[A]\Theta s[B], r[A] \notin r^1, s[B] \notin s^1)\}$ – новое отношение.

6) **Деление** – выполняется над двумя исходными отношениями (*R* и *S*), новое отношение формируется по следующей формуле:

$$Q = R[A \div B]S = \{r[\overline{A}] \mid r \in R \wedge y \mid \{(r[\overline{A}], y \in R)\Theta s[B]\}.$$

Рассмотрим примеры выполнения некоторых операций.

Пример 1. Пусть имеется отношение *R*, имеющее атрибуты A_1, A_2, A_3 и представленное в табл. 2.11. Для операции «*перестановка*» задан новый порядок следования атрибутов *L={3,1,2}*. Новое отношение *Q*, полученное как результат операции «*перестановка*», представлено в табл. 2.12.

Таблица 2.11

Отношение *R*

A_1	A_2	A_3
2	11	103
3	15	112
7	16	120

Таблица 2.12

Отношение *Q*

A_3	A_1	A_2
103	2	11
112	3	15
120	7	16

Пример 2. Исходное отношение *R* представлено в табл. 2.11. Для операции «*проекция*» задан новый порядок следования атрибутов *L={3,1}*. Новое отношение *Q1*, полученное как результат операции «*проекция*», представлено в табл. 2.13.

Таблица 2.13

Отношение *Q1*

A_3	A_1
103	2
112	3
120	7

Пример 3. Рассмотрим выполнение операции «*соединение*». Исходными являются отношение *R* (табл. 2.11) и отношение *S* (табл. 2.14). Сравнение будет

идти по знаку $\Theta = \{\geq\}$ (больше или равно) для атрибута A_3, который является одинаковым для отношений *R* и *S*. По этому атрибуту и будет идти соединение исходных отношений. Новое отношение *Q* представлено в табл. 2.15.

Таблица 2.14

Отношение *S*

A_3	A_5
105	552
120	563

Таблица 2.15

Отношение *Q*

A_1	A_2	A_3	A_3	A_5
3	15	112	105	552
7	16	120	105	552
7	16	120	120	563

Пример 4. Рассмотрим еще один пример выполнения операции «*соединение*». Исходными являются те же отношения: отношение *R* (табл. 2.11) и отношение *S* (табл. 2.14). Сравнение будет идти по знаку $\Theta = \{<\}$ (меньше) для атрибута A_3. Новое отношение *Q11* представлено в табл. 2.16.

Таблица 2.16

Отношение *Q11*

A_1	A_2	A_3	A_3	A_5
2	11	103	105	552
2	11	103	120	563
3	15	112	120	563

Алгоритм выполнения операции «*соединение*»:

1) организуется цикл по обработке всех строк 1-го отношения;
2) организуется цикл по обработке всех строк 2-го отношения;
3) значение атрибута *A* текущей строки отношения *R* сравнивается со значением атрибута *B* текущей строки отношения *S*;
4) если сравнение истинно, то текущая строка *r* из *R* и текущая строка *s* из *S* заносятся в новое отношение *Q*;
5) если же сравнение ложно, то переходят к обработке следующей строки в отношении *S*;
6) если все строки из отношения *S* обработаны, то переходят к обработке следующей строки из отношения *R* и повторяют шаги 2 – 5;
7) если все строки из отношения *R* обработаны, то 1-ый цикл окончен, конец алгоритма.

Пример 5. Рассмотрим пример выполнения операции «*ограничение*». Исходными являются отношение *R* (табл. 2.11) и отношение *S* (табл. 2.14). Сравнение будет идти по знаку *Θ={=}* (равно) для атрибута A_3. Новое отношение *Q1* представлено в табл. 2.17.

Таблица 2.17

Отношение *Q1*

A_1	A_2	A_3
7	16	120

Алгоритм выполнения операции «*ограничение*»:

1) организуется цикл по обработке всех строк 1-го отношения;
2) организуется цикл по обработке всех строк 2-го отношения;
3) значение атрибута *A* текущей строки отношения *R* сравнивается со значением атрибута *B* текущей строки отношения *S*;
4) если сравнение истинно, то текущая строка *r* из *R* заносится в новое отношение *Q*;
5) если же сравнение ложно, то переходят к обработке следующей строки в отношении *S*;
6) если все строки из отношения *S* обработаны, то переходят к обработке следующей строки из отношения *R* и повторяют шаги 2 – 5;
7) если все строки из отношения *R* обработаны, то 1-ый цикл окончен, конец алгоритма.

Пример 6. Рассмотрим выполнение операции «*композиция*». Исходными являются отношение *R* (табл. 2.11) и отношение *S* (табл. 2.14). Сравнение будет идти по знаку $\Theta = \{\geq\}$ (больше или равно) для атрибута A_3. Новое отношение *Q2* представлено в табл. 2.18.

Таблица 2.18

Отношение *Q2*

A_1	A_2	A_5
3	15	552
7	16	552
7	16	563

Алгоритм выполнения операции «*композиция*»:

1) организуется цикл по обработке всех строк 1-го отношения;
2) организуется цикл по обработке всех строк 2-го отношения;
3) значение атрибута *A* текущей строки отношения *R* сравнивается со значением атрибута *B* текущей строки отношения *S*;
4) если сравнение истинно, то текущая строка *r* из *R*, кроме элемента *r[A]*, и текущая строка *s* из *S*, кроме элемента *s[B]*, заносятся в новое отношение *Q*;
5) если же сравнение ложно, то переходят к обработке следующей строки в отношении *S*;
6) если все строки из отношения *S* обработаны, то переходят к обработке следующей строки из отношения *R* и повторяют шаги 2 – 5;
7) если все строки из отношения *R* обработаны, то 1-ый цикл окончен, конец алгоритма.

Как видно из алгоритмов, и операцию «*ограничение*», и операцию «*композиция*» можно выполнить с помощью двух операций «*соединение - проекция*».

Результатом операции «*деление*» являются те атрибуты 1-го отношения, для которых выполнены поставленные условия, но которые сами в проверке не участвовали. Приведем примеры выполнения этой операции.

Пример 7. Найти значение A_2, выполнив условие: $A_1 \geq 2 \wedge A_3 > 115$. Исходным является отношение *R* (табл. 2.11), а отношение *S* составлено по указанному условию (табл. 2.19). Полученный результат представлен в табл. 2.20.

Таблица 2.19

Отношение *S*

A_1	A_3
2	115

Таблица 2.20

Отношение *S1*

A_2
16

Пример 8. Пусть имеется отношение $R(A_1, A_2)$, содержащее фамилии программистов и языки программирования, которые они знают. Необходимо найти программистов, которые знают Visual Basic и Delphi – это отношение $S(A_2)$. Исходные отношения представлены в табл. 2.21 и 2.22. Результат представлен в табл. 2.23.

Таблица 2.21

Отношение *R*

A_1	A_2
Иванов	Visual Basic
Иванов	Lisp
Иванов	Fortran
Ахметов	Visual Basic
Ахметов	Lisp
Ахметов	Delphi
Ли	Visual Basic
Ли	Delphi

Таблица 2.22

Отношение *S*

A_2
Visual Basic
Delphi

Таблица 2.23

Отношение *S1*

A_1
Ахметов
Ли

2.5. Реляционное исчисление

Реляционное исчисление (РИ) позволяет описать отношение и операции над ним в виде аналитического выражения или формулы путем использования ограниченного исчисления предикатов, кванторов и переменных выборки. Пользователь просто указывает, что он хочет получить из БД. Для построения выражений используются следующие символы:

1) $x.y$ – множество значений атрибута *y* из отношения *x*;
2) $A(x_1.y_1, x_2.y_2, \ldots)$ – отношение *A*, заданное с указанными атрибутами;

3) : – «Такой, что», выражение слева от двоеточия означает то, что должно быть найдено, выражение справа означает условие поиска данных;
4) $\exists$ – квантор существования («существует»);
5) $\forall$ – квантор всеобщности («для любого»);
6) $\wedge, \vee, \neg$ – логические операции «И», «ИЛИ», «НЕ»;
7) $\Theta = \{=, \neq, >, \geq, <, \leq\}$ – знаки сравнения.

Приведем примеры использования реляционного исчисления. Пусть имеется отношение *R*, включающее в свой состав следующие атрибуты: A_1 – номер служащего (уникальный ключ), A_2 – фамилия служащего, A_3 – номер отдела, в котором он работает, A_4 – зарплата служащего.

Пример 1: создать новое отношение, содержащее фамилии всех служащих, работающих в отделе с номером 3:

$$Q\,(R.\,A_2) : R.A_3 = 3\ .$$

Пример 2 : создать новое отношение, содержащее номера и фамилии всех служащих, работающих в отделе номер 3 с зарплатой больше 8000:

$$Z\,(R.A_1, R.A_2) : R.A_3 = 3 \wedge R.A_4 > 8000\ .$$

Пример 3. Имеются три отношения:

1) $R_1(A_1, A_2, A_3)$ – отношение «студент», A_1 - номер зачетной книжки, A_2 - фамилия студента, A_3 - адрес студента;

2) $R_2(A_4, A_5, A_6)$ – отношение «преподаватель», A_4 – табельный номер преподавателя, A_5 – фамилия преподавателя, A_6 – должность преподавателя;

3) $R_3(A_1, A_4)$ – отношение «студент – преподаватель», указывающее, какие студенты учатся у каких преподавателей.

Необходимо образовать новое отношение, содержащее сведения о студентах, которые обучаются у всех преподавателей:

$$Q(R_1.A_1, R_2.A_4) : \forall R_2 \exists \wedge R_3 (R_3.A_1 = R_1.A_1 \wedge R_3.A_4 = R_2.A_4)$$

Отметим некоторые преимущества реляционного исчисления по сравнению с реляционной алгеброй:

1) пользователя не интересует, каким образом компьютер получит результат, компьютер же имеет некоторую свободу для выбора наилучшего способа получения результата;
2) процедуры, обеспечивающие секретность данных, выглядят гораздо проще, так как они связаны с явно выраженными в запросе требованиями к данным, а не с процедурами пользователя;
3) для пользователя-непрограммиста естественнее требовать данные по их названиям, чем вводить собственный набор операций,
4) пользователь может задавать операции поиска в удобной для него форме шагов запланированного диалога, которые компьютер интерпретирует в формальный язык реляционного исчисления.

Недостатком РИ является сложность его разработки. Реляционное исчисление требует более высокого уровня автоматизации.

Проведем сравнение реляционной алгебры и реляционного исчисления. Множество функций отношений, выразимых в РА, являются в точности тем же самым, что и множество функций, выразимых формулами РИ. Языки, основанные на РИ, представляют собой языки более высокого уровня по сравнению с алгебраическими языками, поскольку алгебра специфицирует порядок операций в то время, как РИ оставляет определение более эффективного порядка вычислений программному пакету. Выигрыш получают, если выражение РИ оптимизировано, что является весьма сложной проблемой. РИ требует более высокого уровня автоматизации, реализовать его сложнее. Для пользователя, естественно, более удобно указать, что надо найти, чем как это искать, т.е. пользоваться реляционным исчислением проще.

2.6. Язык SQL

Язык **SQL** – это язык структурированных запросов (Structured Query Language), позволяющий получать необходимую информацию из БД и являющийся в настоящее время стандартом в области взаимодействия с базами данных.

Существует два языка SQL:

1) интерактивный – применяется для выполнения действий непосредственно в БД с целью получить результат, который непосредственно используется пользователем; при этом вводится команда, она выполняется, после чего можно немедленно увидеть выходные данные;
2) встроенный – состоит из команд SQL, включенных в программы, которые написаны на каком-либо языке (Delphi, Visual Basic и др.).

Рассмотрим интерактивный SQL. Этот язык определен ANSI (American National Standards Institute – Американский национальный институт стандартов). В SQL есть множество секций или подразделов:

- **ЯОД** (Data Definition Language – DDL) – язык описания данных или язык определения схемы (Scheme Definition Language) – состоит из трех команд, которые создают объекты (таблицы, индексы, представления) в БД;
- **ЯМД** (Data Manipulation Language – DML) – язык манипулирования данными – множество команд, определяющих, какие данные представлены в таблицах в любой момент времени;
- **Язык управления данными** (Data Control Language – DCL) – состоит из предложений, определяющих, может ли пользователь выполнить определенное действие (это часть DDL).

Рассмотрим некоторые операторы этого языка.

Команда **SELECT (**команда выборки или поиска данных) позволяет выбрать необходимые записи по указанному критерию и имеет следующую конструкцию:

SELECT * | {[**DISTINCT** | **ALL**] <список полей>…}
FROM {<имя таблицы> [<алиас>}…
[**WHERE** <предикат>]
[**GROUP BY** {<имя атрибута>| <целое>},…]
[**HAVING** <предикат>]
[**ORDER BY** { {<имя атрибута>| <целое>},…]
[{ **UNION [ALL]**
SELECT …
FROM …
[**WHERE** …]
[**GROUP BY** …]
[**HAVING** …]
{ **ORDER BY** …]
}] …

Опишем элементы структуры этой команды.
<алиас> – временный синоним имени таблицы, определенный здесь и используемый только в этой таблице.
<предикат> – условие.
<целое> – номер столбца в таблице.
* – все поля.
[ ] – конструкция в квадратных скобках может отсутствовать.
| – разделитель «вертикальная черта» означает, что можно использовать одну из двух конструкций.
Операторы сравнения: =, >, >=, < , <= , < > (не равно).
Логические операторы: *AND* (логическое «И»), *OR* (логическое «ИЛИ»), *NOT* (логическое «НЕ»).
DISTINCT – различные значения, подсчитывается количество различных значений для указанного поля.
ALL – включает дубликаты, но не подсчитывает *NULL*-значения.
В конструкции *WHERE* указываются условия отбора данных. При этом можно использовать следующие виды выражений для задания условий отбора данных:

1) *IN (<список значений>)* – позволяет отбирать записи БД, значения указанного поля для которых соответствует представленному списку;
2) *BETWEEN <нач.значение> AND <кон.значение>* – позволяет отбирать записи БД, значения указанного поля для которых соответствует заданному диапазону (между начальным и конечным значением);
3) *LIKE* – применяется для полей типа *CHAR* (текстовых), знак _ (подчеркивание) означает любой один символ, знак % означает любые несколько символов;
4) *NULL* – значение не задано.

Приведем примеры использования этих выражений.
1) Выбрать товары с ценой, соответствующей списку значений:

Cena **IN** (30, 500, 1200)

2) Выбрать товары с ценой, соответствующей диапазону от 30 до 120:

Cena **BETWEEN** 30 AND 1200

3) Выбрать сведения о фирмах, размещенных в городах с названием на букву «К»:

Gorod **LIKE** "К%"

4) Выбрать данные из текста с городом с названием «КАРАГАНД» плюс любая буква (КАРАГАНДУ, КАРАГАНДЕ):

Gorod **LIKE** "КАРАГАНД_"

GROUP BY – позволяет формировать группы и упорядочивать их по какому-либо полю, при этом агрегатная функция применяется к каждой серии групп.

HAVING – определяет критерий, согласно которому определенные группы исключаются из числа выходных данных (как это делает *WHERE* для записей).

ORDER BY – упорядочение по столбцу (*ASC* – по возрастанию, *DESC* – по убыванию).

UNION – запросы выполняются независимо, а результаты объединяются в одно отношение.

Команда **INSERT** позволяет добавить новую запись и имеет следующую конструкцию:

INSERT INTO <имя таблицы> [(<имя атрибута> …)]
VALUES (<значение1 >, < значение2> …)

где *VALUES* – список значений.

Команда **DELETE** позволяет удалить записи из БД и имеет следующую конструкцию:

DELETE FROM <имя таблицы> [**WHERE** <предикат>
| **WHERE CURRENT**]

где *CURRENT* – удалить текущую запись, *WHERE* – указываются условия отбора данных.

Команда **UPDATE** позволяет изменить необходимые значения в записях и имеет следующую конструкцию:

UPDATE <имя таблицы> **SET** <список значений для атрибутов>
[**WHERE** <предикат>]

где *SET* – список значений для атрибутов, где указывается имя атрибута и его новое значение, *WHERE* – указываются условия отбора данных.

Приведем примеры использования рассмотренных операторов языка SQL. *Пример1.* Пусть имеется исходное отношение $R\ (A_1, A_2, A_3, A_4, A_5)$, где R – сведения о факультетах, A_1 – номер факультета, A_2 – наименование факультета,

A_3 – фамилия и инициалы декана, A_4 – количество групп, A_5 – количество студентов. Примеры запросов:

1) получить все сведения о факультетах:

SELECT * FROM R

где * – означает поиск значений всех атрибутов;

2) получить номера всех факультетов, где число студентов больше 400:

SELECT A_1 FROM R WHERE $A_5 > 400$

3) получить наименования факультетов и фамилии и инициалы их деканов для всех факультетов, где число студентов больше 400 и число групп меньше или равно 22:

SELECT A_2 , A_3 FROM R WHERE $A_5 > 400$ AND $A_4 <= 22$

4) откорректировать число групп для факультета с номером 3, новое число групп 28:

UPDATE R SET $A_4 = 28$ WHERE $A_1 = 3$

5) удалить сведения о факультете с наименованием «горный»:

DELETE FROM R WHERE A_2 = "горный"

6) ввести сведения о новом факультете, список новых значений прилагается:

INSERT INTO R VALUES (7, "гидро", "Каримов А.К. ", 5, 130) .

Пример 2. Пусть имеется отношение *firma (n, name, gorod, rat)* – сведения о фирмах, *n* – номер фирмы, *name* – наименование фирмы, *gorod* – город, где размещена фирма, *rat* – рейтинг фирмы.

1) Найти сведения о фирмах в городе Алматы:

SELECT * FROM firma WHERE gorod="Алматы"

2) Найти сведения о фирмах в городе Алматы или с рейтингом >100:

SELECT * FROM firma WHERE gorod="Алматы" OR rat>100

3) Найти наименования фирм с номерами > 30 и < 50:

SELECT name FROM firma WHERE (n BETWEEN 30 AND 50) AND
NOT n IN (30, 50)

4) Найти номера и наименования фирм в городах, название которых начинается на «К», и упорядочить их по наименованию:

SELECT firma.n, firma.name FROM firma WHERE gorod LIKE "К%"
GROUP BY firma.name

5) Кроме того, имеется отношение *tovar (nt, firm, cena, kol)* – сведения о товарах, *nt* – наименование товара, *firm* – фирма-производитель товара, *cena* – цена за единицу товара, *kol* – количество товара. Создать новое отношение, содержащее наименование товара, количество товара, наименование фирмы, город, где размещена фирма, для всех фирм с одинаковыми именами в 1-м и 2-м отношении:

SELECT tovar.nt, tovar.kol, firma.name, firma.gorod FROM tovar, firma
WHERE tovar.firm= firma.name

6) Включить сведения о новой фирме:

INSERT INTO firma VALUES(4, "алси", "Алматы", 135)

7) Включить сведения о наименовании и городе для новой фирмы

INSERT INTO firma (name, gorod) VALUES("сименс", "Астана")

8) Удалить все записи в таблице «фирма»:

DELETE FROM firma

9) Удалить записи в таблице «фирма», где город Тараз:

DELETE FROM firma WHERE gorod="Тараз"

10) Удалить текущую запись в таблице «фирма»

DELETE FROM firma WHERE CURRENT

11) Для всех записей в таблице «фирма» изменить рейтинг на 200:

UPDATE firma SET rat=200

12) Изменить наименование фирмы на новое значение "айна" и название города на новое значение "Астана" для всех фирм с рейтингом 50:

UPDATE firma SET name="айна", gorod="Астана" WHERE rat=50

Рассмотрим команды создания объектов БД. Команда создания таблицы БД имеет вид:

CREATE TABLE <имя таблицы> (<имя поля1> <тип>
[,<имя поля2> <тип> …])

Здесь указываются имя таблицы и имена полей, входящих в нее, со своими типами.

Команда добавления поля в таблицу БД имеет вид:

ALTER TABLE <имя таблицы> **ADD COLUMN** <имя поля> <тип>

В таблицу добавляется только одно поле. Если необходимо добавить несколько полей, то команда выполняется несколько раз.

Команда создания индекса для таблицы БД имеет вид:

CREATE [**UNIQUE**] **INDEX** <индекс> **ON** <таблица> [<поле>
ASC | **DESC**] [,<поле> [**ASC** | **DESC**],…]) [**WITH**
PRIMARY | **DISALLOW NULL** | **IGNORE NULL**]

где *UNIQUE* – уникальный индекс, *INDEX* – имя индексного файла, *ASC* – по возрастанию, *DESC* – по убыванию, *PRIMARY* – первичный ключ, *DISALLOW NULL* – запрещает наличие пустых значений в индексированных полях, *IGNORE NULL* – запрещает включение в индекс записей, имеющих значение *NULL* в индексированных полях. При создании индексного файла для первичного ключа используются конструкции *UNIQUE* и *PRIMARY*.

Команда удаления таблицы в БД имеет вид:

DROP TABLE <таблица>

Команда удаления индекса для указанной таблицы БД имеет вид:

DROP INDEX <индекс> **ON** <таблица>

Команда удаления поля из таблицы БД имеет вид:

ALTER TABLE <таблица> **DROP COLUMN** <поле>

При использовании этой команды может быть удалено только одно поле из структуры таблицы.

Пример 3.

1) Создать новую таблицу, содержащую информацию об автомобиле, с полями «марка», «цвет» и «цена»:

CREATE TABLE avto (marka TEXT (10), cvet TEXT (10), cena LONG)

2) Добавить поле «гос. номер» в таблицу *avto*:

```
ALTER TABLE avto ADD COLUMN gocnomer TEXT (7)
```

3) Создать индекс (первичный ключ) по полю «гос.номер» для таблицы *avto*:

```
CREATE UNIQUE INDEX indNomer ON avto gocnomer
WITH PRIMARY
```

4) Удалить поле «цвет» из таблицы *avto*:

```
ALTER TABLE avto DROP COLUMN cvet
```

5) Удалить индекс *indNomer* для таблицы *avto*:

```
DROP INDEX indNomer ON avto
```

6) Удалить таблицу *avto*:

```
DROP TABLE avto
```

Практически язык SQL является гибридом реляционной алгебры и реляционного исчисления, обладает реляционной полнотой.

2.7. Язык QBE

Для реляционной модели еще имеется язык **QBE** (Query By Example – запрос посредством примера), первая версия которого была разработана фирмой IBM. Этот язык позволяет использовать специальный редактор для подготовки запросов на получение данных из базы, в соответствии с критерием, который задает пользователь.

Теоретической основой языка QBE является реляционное исчисление с переменными-доменами. Этот язык позволяет сложные запросы к БД путем заполнения предлагаемого СУБД бланка запроса. Такой способ задания запросов обеспечивает высокую наглядность и не требует задания алгоритма выполнения операции – достаточно описать образец ожидаемого результата. В каждой современной СУБД имеется свой вариант языка QBE.

С помощью запросов на языке QBE можно выполнять такие операции над данными, как выборка (поиск), добавление новых записей (вставка), удаление записей, корректировку данных, вычисления над данными. Результатом запроса является новая таблица или исходная таблица, содержащая измененные в соответствии с запросом данные. Запросы для реализации выборки, вставки, удаления и корректировки могут выполняться над всеми данными или на основе условий, задаваемых пользователем – критерий отбора данных с использованием логических выражений. Если запрос содержит вычисления, то

в новой таблице появляются новые поля, значения которых вычислены в соответствии с указанным в запросе выражением.

Рассмотрим пример использования этого языка в СУБД Access. Бланк запроса представлен на рис. 2.1. Как видно из рисунка, запрос позволяет получить информацию об автомобиле и заводе на основе двух таблиц БД Критерием отбора записей из таблицы БД служит задаваемая при обработке запроса марка автомобиля. Кроме того, в запросе есть вычисляемое поле – «*ндс*», значение которого вычисляется на основе значения поля «Стоимость» таблицы «авто».

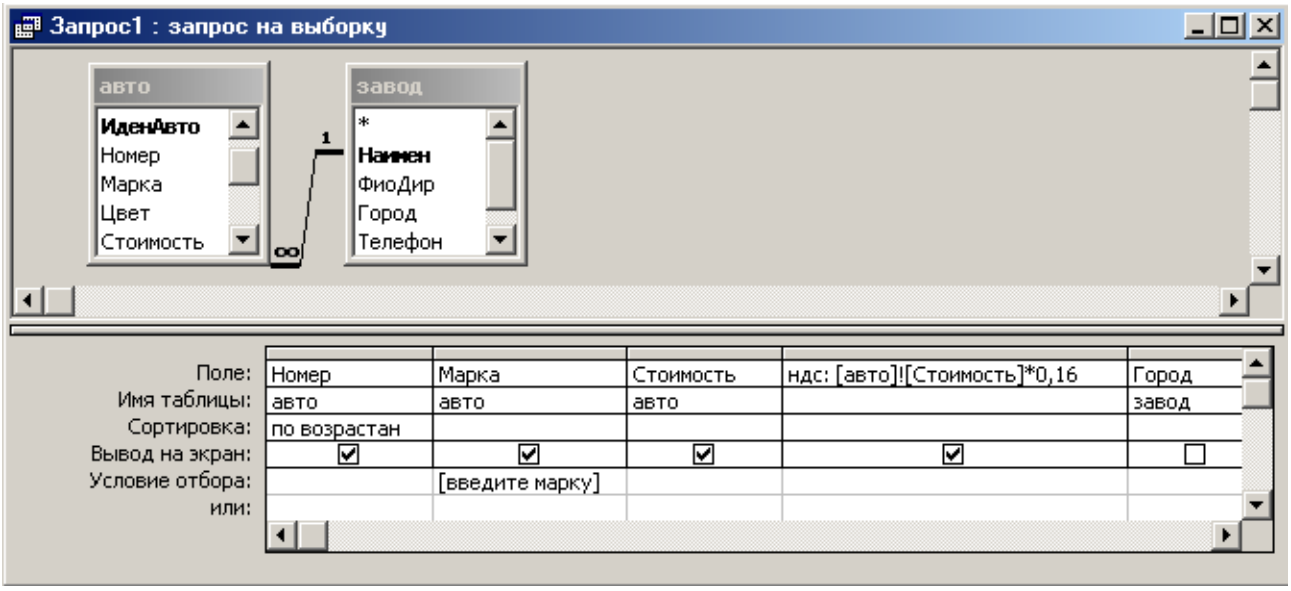


Рис. 2.1. Бланк запроса

Анализ современных СУБД позволяет предположить следующие направления развития языка QBE:

- повышение наглядности представления данных и удобства формирования запроса для пользователя;
- появление средств, реализующих новые возможности СУБД по формированию запросов – обработка неточных или нечетких запросов, обработка больших объемов разнотипных данных, обработка мультимедиа данных (аудио-данных, видео-данных, графических данных, пространственных данных) и т.д.;
- использование при формулировке запроса узко профессионального естественного языка;
- использование речевого ввода запросов в перспективе.

2.8. Реляционная модель данных с временной динамикой

Одно из интенсивно разрабатываемых и исследуемых направлений в теории баз данных – это создание математических основ баз данных,

построенных на моделях представления данных, учитывающих фактор времени [20 – 22].

В этом разделе представлены полученные теоретические результаты по моделям и методам создания и обработки реляционных баз данных, включающие формализацию реляционной модели данных с временной динамикой и созданное для работы с ней в динамике ANU-исчисление, сформулирован и доказан ряд теорем и лемм по семантике ANU-исчисления, а также сформулированы правила поддержки целостности реляционной базы данных с временной динамикой [21, 25, 26].

Базы данных систем реального времени и информационных систем, в которых важен временной фактор, обладают такими особенностями как требование хранения динамики данных и требование высокой скорости обработки запросов к базе данных. Указанные особенности ставят задачу разработки нового подхода к созданию формальных основ и технологии проектирования баз данных систем реального времени и информационных систем с временным фактором.

Реляционная модель данных с временной динамикой (РМДВД) позволяет решить поставленные задачи. РМДВД предоставляет возможность выделять и описывать объекты, хранимые в базе данных, на основе динамических свойств элементов данных, по их совпадению. Время вводится как один из атрибутов отношения. РМДВД определяется тремя компонентами:

$$M =< G_s, G_c, O >,$$

где G_s – правила построения структур данных, G_c – правила ограничения целостности, O – операции обработки данных.

Операции обработки данных определяются следующим образом:

$$O =< O_V, O_K, O_{VB}, O_U, O_{FI}, O_P, O_F, O_{\min}, O_{\max} >,$$

где O_V – включение новых данных за определенное время, O_K – корректировка данных за определенное время, O_{VB} – выборка данных за определенный период времени, O_U – удаление данных за определенное время, O_{FI} – формирование интегральных характеристик данных путем суммирования временных интервалов за определенный период времени, O_P – просмотр данных на заданное время назад (в прошлое), O_F – формирование данных на заданное время вперед (в будущее), $O_{\min}$ - поиск минимального значения элемента данных на заданном временном интервале, $O_{\max}$ - поиск максимального значения элемента данных на заданном временном интервале.

Правила построения структур данных G_s включают два типа правил:

$$G_s =< G_s^A, G_s^R > \quad ,$$

где G_s^A – правила формирования характеристик элементов данных, G_s^R – правила построения схем отношений данных.

Правила формирования характеристик элементов данных G_s^A формализуются следующим образом: каждый элемент данных определяется своим именем, характеризуется описанием назначения, связанным с процессами системы управления, набором параметров, значения которых определяют его динамические свойства. Для отношений данных РМДВД правила построения G_s^R заключаются в объединении таких элементов данных, у которых ограничения целостности по области определения атрибута времени T совпадают.

Правила ограничений целостности G_c для РМДВД включают:

$$G_c =< G_c^A, G_c^R, G_c^F > \quad ,$$

где G_c^A – ограничения целостности на элементах данных, G_c^R – структурные ограничения целостности на элементах данных, G_c^F – функциональные ограничения целостности на элементах данных. Ограничения целостности G_c^A элементов данных включают тип значения и множество значений элемента.

Структурные ограничения целостности G_c^R означают, что в качестве первичного ключа отношения определяется атрибут времени T. Между атрибутом времени T и невременными атрибутами A_j отношения существуют функциональные зависимости $T \to A_j$. Функциональные ограничения целостности G_c^F на отношениях данных РМДВД определяются следующим образом: внутри кортежа отношения имеют место вычислительные зависимости (*C*-зависимости) между элементами данных. В качестве операций манипулирования данными в РМДВД используются такие операции как выборочная запись, выборочное чтение, чтение на n элементов ранее относительно текущего момента, удаление записей.

Для описания динамики данных в РМДВД формализуется *ANU*-исчисление, аксиомы которого имеют вид:

$$B1.\ T(qTr) \supset pTq \tag{2.1}$$

$$B2.\ \sim(pT(q\ \&\sim\ q)) \tag{2.2}$$

$$B3.\ \sim((q\ \&\sim q)Tp) \tag{2.3}$$

$$B4.\ pTq \supset p \tag{2.4}$$

$$B5.\ p \supset pT\ \Delta \tag{2.5}$$

$$B6.\ (p \cup q)T(r \cup s) \supset pTr \cup pTs \cup qTr \cup qTs \tag{2.6}$$

$$B7.\ pTq \,\&\, pTs \supset pT(q \,\&\, s) \tag{2.7}$$

$$B8.\ pTr \,\&\, qTr \supset (p \,\&q)Tr \tag{2.8}$$

$$B9.\ (p \,\&\, q)T(r \,\&\, s) \supset pTr \,\&\, pTs \,\&\, qTr \,\&\, qTs \tag{2.9}$$

$$A1.\ Aa \sim p \equiv \sim Aap \tag{2.10}$$

$$A2.\ Aa\,(p \supset q) \equiv Aap \supset Aaq \tag{2.11}$$

$$A3.\ AaFnp \equiv A\,(a+n)\,p \tag{2.12}$$

$$A4.\ AaPnp \equiv A\,(a-n)\,p \tag{2.13}$$

$$A5.\ Aa\,(\exists n)\alpha \equiv (\exists n)Aa\alpha \tag{2.14}$$

$$A6.\ UabSmn \supset (\exists c)\,(Uacm \,\&\, Ucbn) \tag{2.15}$$

В аксиомах используются следующие обозначения. Пусть *pTq* означает «имеет место случай, что *p* и в следующий момент *q*», Δ – тавтология, $\sim$, &, $\cup, \supset$ – пропозициональные связки, *Aap* – «в момент *a* имеет место случай, что *p*», *Pnp* – «*n* единиц времени назад имел место случай, что *p* », *Fnp* – «*n* единиц времени вперед будет иметь место случай, что *p* », $(\exists n)P\alpha$ – «существует *n* интервалов времени, для которых в прошлом было α », *Uabn* – «момент *a* ранее момента *b* на интервал *n* », *Smn* – «сумма интервалов *m* и *n*».

Аксиомы имеют следующую интерпретацию:

- *B1* определяет строгое предшествование событий;
- *B2, B3* задают закон непротиворечия;
- *B4, B5* указывают, что единственный момент отнесения расположен в настоящем;
- *B6* определяет выбор события;
- *B7, B8* описывают следование, предшествование одновременности (параллельности) событий;
- *B9* определяет «комбинаторное» следование событий;
- *A1* означает, что в момент *a* имел место случай, что $\sim p$, следовательно, нет момента *a*, в который имел место случай, что *p*;
- *A2* означает, что в момент *a* имел место случай, что $p \supset q$, следовательно, в момент *a*, в который имел место случай, что *p*, следует, что в момент *a* имел место случай, что *q*;
- *A3* означает просмотр на *n* моментов времени в будущее: в момент *a* имеет место случай, что *n* единиц времени вперед будет иметь место *p*, что эквивалентно тому, что *p* будет иметь место в момент *(a + n)*;

- *A4* означает просмотр на *n* моментов времени в прошлое: в момент *a* имеет место случай, что *n* единиц времени назад имел место *p*, что эквивалентно тому, что *p* имел место в момент *(a - n)*;
- *A5* означает использование квантора существования $\exists$ – в момент *a* существует *n* единиц времени, для которых в момент *a* имел место α, следовательно, существует *n* единиц времени, для которых в момент *a* имел место α;
- *A6* означает суммирование интервалов времени: момент *a* был ранее момента *b* на *m+n* интервалов, следовательно, между ними был момент *c*, отстоящий от момента *a* на *m* интервалов, за которым через *n* интервалов следовал момент *b*.

Правила вывода:

1) подстановка (формула вместо переменной),
2) отделение (если доказуема α и $(\alpha \subset \beta)$, то доказуема β),
3) экстенсиональность (взаимозаменимость доказуемо эквивалентных формул).

Аксиомы (2.1) – (2.9) позволяют описывать формирование отношений данных во времени, а аксиомы (2.10) – (2.15) позволяют описать процесс обработки данных во времени.

Далее представлена реляционная семантика *ANU*-исчисления [25, 26]. Пусть имеется модельная структура $\sigma = < W, R, G >$, где *W* – непустое множество, *R* – бинарное отношение со свойствами рефлексивности и транзитивности на *W* , *G* – подмножество множества *W*. Модельная структура может иметь следующую интерпретацию: *W* – множество всех возможных миров, *G* – реальный мир. Для двух миров *H1* и *H2*, $H1 \in G$, $H2 \in G$, если *H2* возможен относительно *H1*, то *H2 R H1*.

Введем понятие модели. Моделью правильно построенной формулы *A* в модельной структуре σ назовем двухместную функцию $\varphi(v, g)$, где v пробегает множество пропозициональных переменных *var* формулы *A*, g – множество элементов *G*: $var \to G$. Областью значений этой функции является $\{T, F\}$, где *T* (*true*) – истина, *F* (*false*) – ложь.

Правильно построенная формула (ППФ) в *ANU*-исчислении определяется следующим образом:

- отдельная переменная есть *ANU*-выражение;
- если *C* и *B* есть *ANU*-выражение, то *ANU*-выражениями являются $\sim C$, $C \cup B$, $C \,\&\, B$, $C \supset B$, $C \equiv B$, *CTB*, *PC*.

Таким образом, модель $\varphi(v, g)$ задает истинностные значения каждой пропозициональной переменной *A* в каждом мире *G*. Модель $\varphi(v, g)$ связывает каждую переменную с множеством миров из *G*, в которых она истинна, и каждый мир ассоциируется с множеством переменных, которые принимают в нем значение «истина».

Пусть $\varphi^*(B, g)$ - расширение функции φ, где *B* – любая подформула *A*. Определим $\varphi^*(B, g)$ следующим образом:

- если B – атомарная формула (т.е. переменная), то

$$\varphi^*(B,g) = \varphi(B,g) \ ;$$

- $\varphi^*(\sim B,g) = T \ <= => \ \varphi(B,g) = F$;

- пусть B имеет вид $C \supset D$, тогда:

$$\varphi^*(\ C \supset D,g) = T \ <= => \ \varphi^*(\ C,g) = F \ \cup \ \varphi^*(\ D,g) = T;$$

- пусть B имеет вид CTD, где T – «*и следующий*», тогда:

$$\varphi^*(CTD,g) = T \ <= => \ (\exists u[gR^r u \wedge \varphi^*(\ C,g) = T \wedge \phi^*(D,u) = T]) \ ,$$

где R^r – редукция (отношение непосредственного следования);

- пусть B имеет вид $C \cup D$, тогда:

$$\varphi^*(\ C \cup D,g) = T \ <= => \ \varphi^*(\ C,g) = T \ \cup \ \varphi^*(\ D,g) = T \ ;$$

- пусть B имеет вид $C \ \& \ D$, тогда:

$$\varphi^*(\ C \& D,g) = T \ <= => \ [\varphi^*(\ C,g) = T \ \wedge \ \varphi^*(\ D,g) = T \ ] \ ;$$

- пусть B имеет вид Pp, тогда:

$$\varphi^*(\ Pp,g) = T \ <= => \ (\exists u)[uR^r g \wedge \varphi^*(\ P,u) = T] \ .$$

Введем ряд определений.

Определение 1. Будем говорить, что A **истинна** в модели $\varphi(v,g)$ в мире $w \in W$ в модельной структуре σ, если

$$\varphi^*(\ A,w) = T \ .$$

Определение 2. Будем говорить, что A **общезначима** в $< W, R, G >$, если A истинна во всех моделях $\varphi(v,g)$ во всех $w \in W$ в модельной структуре $< W, R, G>$.

Определение 3. Будем говорить, что A ***ANU*-общезначима**, если A истинна во всех модельных структурах *ANU*-исчисления.

Определим алгебраическую семантику *ANU*-исчисления. Пусть задана модельная структура $\sigma =< W,R,G >$. Определим алгебру σ^+ на σ как:

$$\sigma^+ =< G,\cup,\cap,\neg,\tau > \ ,$$

где G – непустое множество всех подмножеств W ; $\cap$, $\cup$, $\neg$ – теоретико-множественные операции объединения, пересечения и дополнения до G ; для формулы $A \in G$:

$$\tau A = \{x | (\exists y)((y \in A \wedge xR^r y) \cup x \in G)\} .$$

При этом справедливы следующие теоремы.

Теорема 1. Если σ – модельная структура, то σ^+ – модальная алгебра.

Доказательство. Алгебра $< G, \cup, \cap, \neg >$ – булева алгебра по определению. Для *A, B* $\in$ *G* имеем:

$$\tau(A \cup B) = \{x | (\exists y)((y \in A \cup B \wedge xR^r y) \wedge x \in G)\} =$$

$$= \{x | (\exists y)((y \in A \wedge xR^r y) \vee (\exists y)(y \in B \wedge xR^r y) \vee (x \in G)\} =$$

$$= \{x | (\exists y)(y \in A \wedge xR^r y) \vee x \in G\} \cup \{x | (\exists y)(y \in B \wedge xR^r y) \vee x \in G\} =$$

$$= \tau A \cap \tau B \quad ,$$

так что σ^+ является модальной алгеброй.

Теорема 2. $\mid\!\underline{\quad\;}_{ANU} A$ тогда и только тогда, когда *A* удовлетворяется σ^+ для всех модельных структур σ.

Доказательство строится на основе теоремы 1 и известных теорем по семантике модальных логик.

Свяжем реляционный и алгебраический подходы к представлению семантики *ANU*-исчисления. Для каждой ППФ *A* и модели $\varphi(v, g)$ для *A* в модельной структуре σ можно определить приписывание значений *H(φ)* в терминах переменных $v_1,..., v_n$ формулы *A* из алгебры σ^+, полагая, что

$$V(v_i) = \{x | x \in W \wedge \varphi(v_i, x) = t\}, \; i = \overline{1, n}$$

$$\text{и} \qquad H(\varphi) = \langle V(v_1),..., V(v_n)\rangle \quad .$$

И наоборот, если дано приписывание $H = \langle A_1,..., A_n\rangle$ $(A_i \subseteq W)$ для *n* переменных из *A*, можно определить модель $\varphi(H)(v, g)$ для *A* в σ, полагая $\varphi(H)(v, g) = T$ тогда и только тогда, когда $x \in A_i$. Для любого приписывания *H* значений из σ^+ для переменных из *A* и подформулы *B* формулы *A* обозначим через $v_H(B)$ значение из σ^+, принимаемое *B* при приписывании *H*.

Определим и докажем леммы 1 и 2.

Лемма 1. Пусть *A* – правильно построенная формула и $\varphi(v, g)$ – модель для *A* в $\sigma = < W, R, G >$. Тогда для всех $x \in W$ $\varphi^*(A, x) = T$, если и только если $x \in V_{H(\varphi)}(A)$.

Доказательство. Лемма 1 доказывается с помощью индукции по длине *A*. Если *A* является одной из переменных $v_1,...,v_n$, то лемма 1 выполняется по определению *H(φ)*. Для доказательства шага индукции предположим, что результат доказан для *B* и *C*. Заметим, что по кванторной логике:

$$\forall y(xR^r y \to \varphi^*(B,y) = T \Leftrightarrow \forall y(xR^r y \to y \in V_{H(\varphi)}(B)) .$$

Теперь предположим, что *A* имеет вид $B \supset C$. Тогда для всех $x \in W$:

$$\varphi^*(B \supset C, x) = T \Leftrightarrow \varphi^*(B,x) = F \vee \varphi^*(C,x) = Y \Leftrightarrow$$

$$\Leftrightarrow x \notin V_{H(\varphi)}(B) \vee x \in V_{H(\varphi)}(c) \Leftrightarrow x \in (\neg V_{H(\varphi)}(B) \vee V_{H(\varphi)}(C) \Leftrightarrow$$

$$\Leftrightarrow x \in V_{H(\varphi)}(B \supset C) .$$

Аналогичное доказательство может быть приведено, когда *A* принимает другой вид (например, *~B, BTC, PB, B ∪ C, B & C, B≡C*).

Лемма 2. Пусть *A* – правильно построенная формула и *H* – задание значений из σ^+ для переменных из *A* в некоторой структуре $\sigma = < W, R, G >$. Тогда для всех $x \in W$ $\varphi^*(H)(A,x) = T$, если и только если $x \in V_H(A)$.

Аналогично доказательству леммы 1 проводится доказательство леммы 2: утверждение справедливо в случае, когда *A* является одной из переменных $v_1,...,v_n$ согласно определению *H(φ)*.

Сформулируем основные теоремы, связанные с реляционным и алгебраическим подходом.

Теорема 3. Пусть $\sigma = < W, R, G >$ – модельная структура, а *A* – правильно построенная формула. Тогда *A* удовлетворяется алгеброй σ^+, если и только если *A* истинна в σ.

Доказательство. Пусть *A* есть ППФ, которая удовлетворяется в σ^+. Рассмотрим модель $\varphi(v,g)$ для *A* в σ. Тогда $V_{H(\varphi)}(A) = W$, откуда по лемме 1 $\varphi^*(A,x) = T$ для всех $x \in W$. Таким образом, *A* истинна в σ . И наоборот, предположим, что *A* истинна в σ, и рассмотрим значения переменных формулы *A*. Тогда для всех $x \in W$ $\varphi^*(H)(A,x) = T$, откуда по лемме 2 для всех $x \in W$ $x \in V_H(A)$, так что $V_H(A) = W$ и *A* удовлетворяется в алгебре σ^+.

Теорема 4. $\underset{ANU}{|\!\!-\!\!-} A$ тогда и только тогда, когда *A* *ANU* – общезначима.

Доказательство этой теоремы следует из теоремы 2, теоремы 3, леммы 1, леммы 2 и различных определений истинности. Приведем его.

Поскольку $\underset{ANU}{|\!\!-\!\!-} A$, то в силу теоремы 2 *A* удовлетворяется алгебрами σ^+ для всех модельных структур $\sigma = < W, R, G >$. По лемме 1 когда *A* – правильно построенная формула и $\varphi(v,g)$ – модель для *A* в $\sigma = < W, R, G >$, то для всех $x \in W$

$\varphi^*(A,x)=T$, если $x \in V_{H(\varphi)}(A)$. Например, если A имеет вид $B \supset C$, то для всех $x \in W$:

$$\varphi^*(B \supset C,x)=T \Leftrightarrow \varphi^*(B,x)=F \vee \varphi^*(C,x)=Y \Leftrightarrow$$

$$\Leftrightarrow x \notin V_{H(\varphi)}(B) \vee x \in V_{H(\varphi)}(c) \Leftrightarrow x \in (\neg V_{H(\varphi)}(B) \vee V_{H(\varphi)}(C) \Leftrightarrow$$

$$\Leftrightarrow x \in V_{H(\varphi)}(B \supset C) .$$

Аналогичное доказательство можно получить, когда *A* представляется в виде какой-либо другой ППФ: *~B, BTC, PB, B ∪ C, B & C, B≡C.*

По лемме 2 поскольку *A* – правильно построенная формула и *H* – задание значений из σ^+ для переменных из *A* в некоторой структуре $\sigma = <W,R,G>$, то для всех $x \in W$ $\varphi^*(H)(A,x)=T$, если $x \in V_H(A)$. В силу теоремы 3 поскольку *A* – правильно построенная формула, то *A* удовлетворяется алгеброй σ^+, когда *A* истинна в структуре *σ*. Следовательно, $\mid\!\underset{ANU}{\underline{\quad\quad}} A$ *ANU* – общезначима.

Сравнительная оценка предлагаемого подхода к исследованию вопроса динамики данных для информационных систем, использующих время, заключается в следующем. В РМДВД использована структура данных, представляющая собой множество состояний абстрактного объекта, приписанных к моментам времени. Правила формирования абстрактных объектов основаны на динамике свойств отдельных элементов данных. Для представления семантики *ANU*-исчисления использован как реляционный, так и алгебраический подходы. Доказана теорема полноты *ANU*-исчисления.

Рассмотрим, как формализованы правила поддержки целостности в реляционной базе данных с временной динамикой. Под **целостностью** данных понимается безошибочность, точность и достоверность данных, хранимых в БД, в каждый момент времени. Целостность поддерживается с помощью ограничений целостности, представляющих собой набор специальных правил, которые определяют допустимость данных и связей между ними в каждый момент времени. Представим ограничения целостности $\boldsymbol{G}$ в следующем виде:

$$G = < G_s, G_z >,$$

где G_s – структурные ограничения целостности, G_z – ограничения целостности на значения данных.

Для реляционной модели данных структурные ограничения целостности G_s означают, что имеется множество типов структур и согласованное с ними множество операций над данными. Ограничения целостности на значения данных G_z определяются в базе данных с помощью специальных конструкций языка описания данных или языка манипулирования данными. Такие ограничения требуют, чтобы значение атрибута принадлежало заданному

диапазону или эти ограничения выражают некоторое арифметическое соотношение между различными атрибутами.

Для реляционной модели данных с временной динамикой формализованы следующие правила ограничений целостности. Ограничения целостности G_c^A атрибутов данных включают тип значения и множество значений атрибута:

$$G_c^A = < TIP, D > ,$$

где *TIP* – множество используемых типов данных, а *D* – множество значений атрибута. Для каждого типа данных применяется свое множество операций.

В большинстве современных СУБД используются следующие типы данных: *TIP = < Integer, Long, Single, Double, Text, Date >*, где *Integer* – данные целого типа, *Long* – длинное целое, *Single* – вещественный тип, *Double* – вещественный тип удвоенной точности, *Text* – текстовый или символьный тип, *Date* – данные типа дата (день-месяц-год). Для каждого типа данных определен свой набор операций. Для данных типа *Integer, Long, Single, Double* в качестве операций вычисления используются все арифметические операции («сложение», «вычитание», «умножение», «деление», «возведение в степень») $OV_C = \{+,-,*,/,\wedge\}$ и все операции сравнения («равно», «не равно», «больше», «больше или равно», «меньше», «меньше или равно») $OSR_C = \{=, \neq, >, >=, <, <=\}$.

Для данных типа *Date* в качестве операций вычисления используются арифметические операции «сложение» и «вычитание» $OV_D = \{+,-\}$ и все операции сравнения $OSR_D = \{=, \neq, >, >=, <, <=\}$. Для данных типа *Text* в качестве операций вычисления используется операция конкатенация $OV_T = \{+ | \&\}$ и операции сравнения $OSR_T = \{=, \neq\}$.

Структурные ограничения целостности на элементах данных G_c^R означают, что в одно отношение объединяют атрибут времени *T* и невременные атрибуты A_i, $i = \overline{1,n}$, характеризующие описываемый объект:

$$R^t = R(T, A_1, A_2, ..., A_n).$$

Естественно, имеются функциональные зависимости между атрибутом времени T и невременными атрибутами A_i, $i = \overline{1,n}$, отношения: $T \to A_i$.

Функциональные ограничения целостности на элементах данных G_c^F означают, что между атрибутами данных в одном или нескольких отношениях существуют вычислительные зависимости различного вида. Введем три основных вида этих зависимостей.

Зависимость 1. Равенство значений атрибутов в разных отношениях: $R^t.A_i = S^t.B_j$. Эта зависимость означает, что для двух отношений R и S, рассматриваемых в заданный момент времени t, имеются связанные между собой атрибуты A_i и B_j, связь реализуется по равенству значений.

Зависимость 2. Сумма значений атрибутов в разных отношениях:

$R^t.A_i = \sum_{n=1}^{N} S^t.B_{jn}$. Эта зависимость означает, что атрибут A_i из отношения R формируется как сумма значений атрибута B_j из отношения S, N – количество значений атрибута B_j, участвующих в суммировании. Отношения R и S рассматриваются в заданный момент времени *t*.

Зависимость 3. Вычисление значения атрибута по формуле: $R^t.A_i = \Phi(S_{l_1}.B_{k_1}, S_{l_2}.B_{k_2}, ..., S_{l_n}.B_{k_n})$. Эта зависимость означает, что атрибут A_i из отношения R вычисляется по заданной формуле на основе атрибутов B_{k_1}, B_{k_2}, …, B_{k_n} из нескольких отношений S_{l_1}, S_{l_2}, …, S_{l_n}, рассматриваемых в заданный момент времени *t*.

При проектировании реляционной базы данных с временной динамикой для каждого отношения и его атрибутов, исходя из особенностей рассматриваемой предметной области и семантики описываемых объектов, устанавливаются свои ограничения целостности, позволяющие поддерживать целостность базы данных и достоверность хранимых данных в течение всего периода эксплуатации информационной системы.

Разработанная реляционная модель данных с временной динамикой и аксиоматика построенного *ANU*-исчисления позволяют описывать и отслеживать временную динамику данных и служат основой для проектирования реляционных баз данных с временной динамикой для сложных природно-биологических объектов, каким является, например, природный очаг чумы [22-24]. Полученные теоретические результаты были использованы при разработке базы данных для Прибалхашского природного очага чумы в рамках двух международных проектов:

- проект К-159-98 по линии Международного научно-технического центра «Мониторинг количественных и качественных параметров особо опасных инфекций на природных очагах чумы в Республике Казахстан»

- грант Inco-Copernicus «STEPICA» ICA2-CT-2000-10048 «Чума в Центральной Азии – эпидемиологическое исследование, основанное на пространственно-временной динамике».

2.9. Иерархическая модель представления данных

Рассмотрим, как можно представить данные, подлежащие хранению в БД, с помощью иерархической модели (рис.2.2).

В иерархической модели связь между данными является иерархической с четко установленными уровнями вложенности. Данные представляются в виде **поле – сегмент – запись**. С помощью *сегментов* описываются объекты предметной области. Сегмент состоит из множества *полей*, которые описывают свойства объектов. *Запись* представляет собой полную совокупность иерархически упорядоченных сегментов на всем иерархическом пути. Всегда имеется один корневой сегмент.

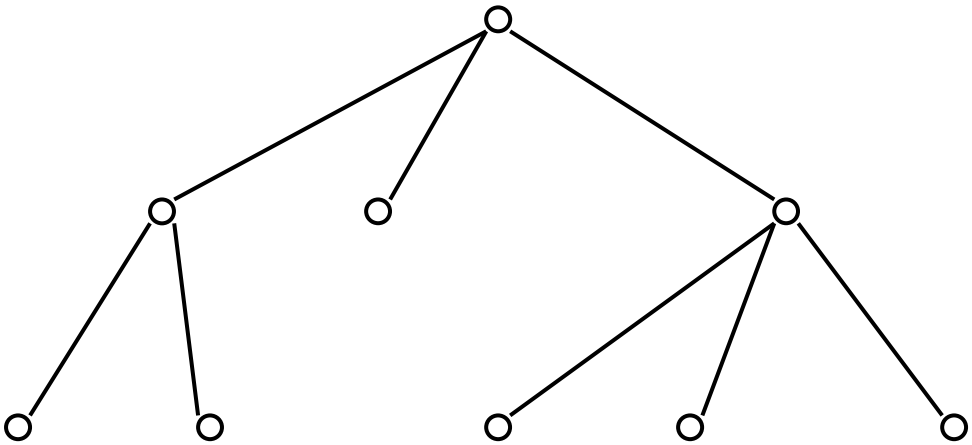


Рис. 2.2. Представление данных в виде иерархической модели

Между сегментами существует связь «исходный - порожденный», которая может быть взаимосвязью 1 : 1 или 1: М. В иерархической модели представление данных подчиняется следующим правилам:

1) каждая запись содержит только один корневой сегмент;
2) корневой сегмент может иметь произвольное число порожденных сегментов;
3) каждый порожденный сегмент имеет только один исходный сегмент и множество порожденных сегментов (на практике до 15 уровней вложенности и до 255 типов сегментов в одной записи).

В качестве примера рассмотрим базу данных «Обучение», в которой будет представлено 5 сегментов: «курс», «предварительный курс», «цикл занятий», «преподаватель», «студент». Структура БД для иерархической модели представлена на рис. 2.3. Экземпляр одной записи для этой БД представлен на рис. 2.4. Практически запись представляется в виде дерева.

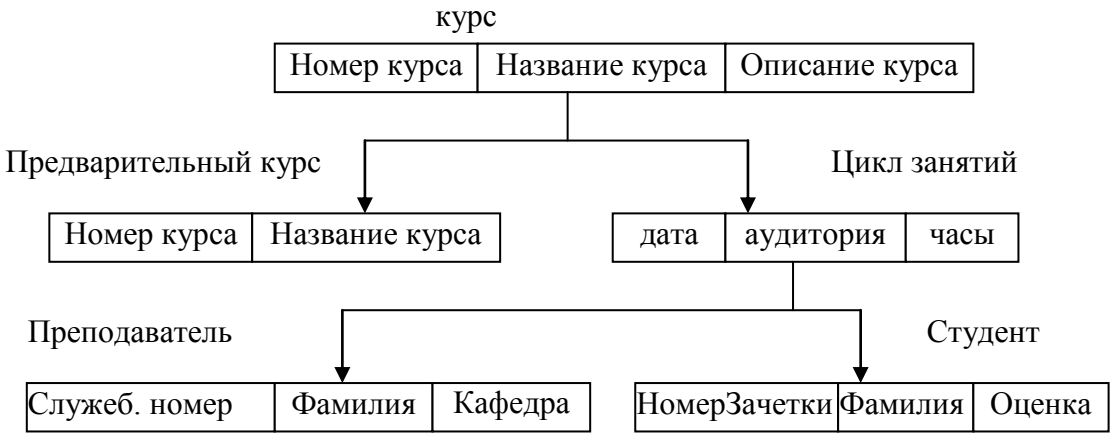


Рис. 2.3. Структура БД «Обучение» для иерархической модели

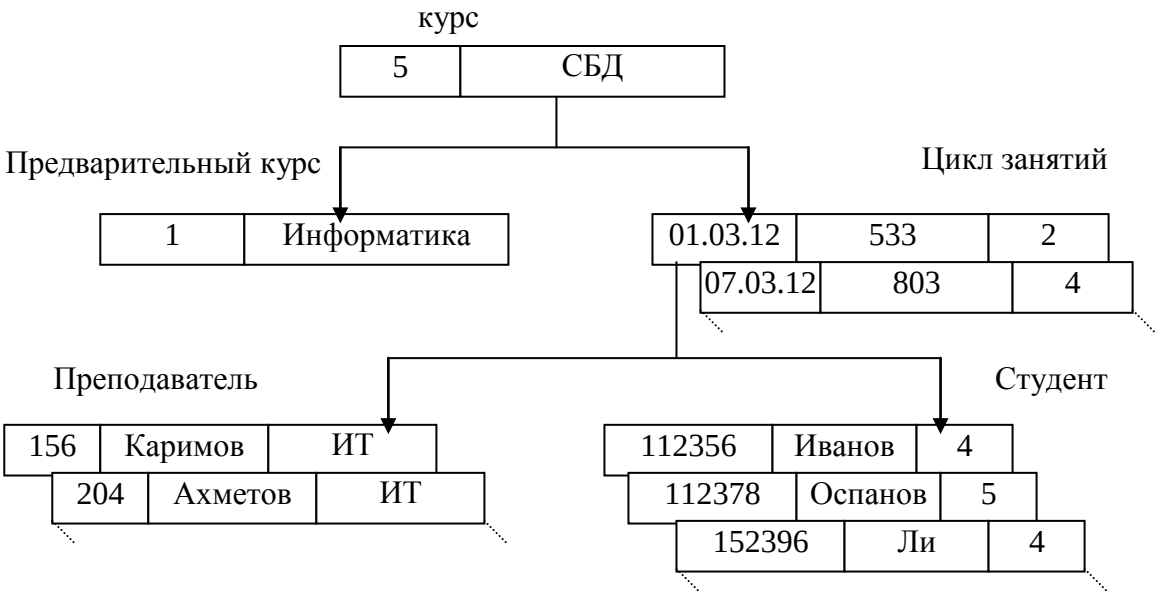


Рис. 2.4. Экземпляр записи для иерархической модели

Иерархическая база данных представляет собой упорядоченную совокупность экземпляров данных типа «дерево», содержащих экземпляры типа «запись». Часто отношения родства между типами переносятся на отношения между самими записями. Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо. Команды ЯОД для иерархической модели включают операторы по описанию базы данных, сегментов и полей. Команды ЯМД для иерархической модели содержат операторы, организующие поиск (например, «дать уникальный» - специфицирует путь от экземпляра корневого сегмента до искомого сегмента, «дать следующий», «дать следующий под исходным»), включение нового сегмента, удаление сегмента, замену сегмента и др.

При выполнении ряда операций, таких как включение и удаление, возникают трудности. При удалении, например, будут удалены экземпляры всех подчиненных сегментов. К *достоинствам* иерархической модели относятся простота понимания и использования, наличие хорошо зарекомендовавших себя СУБД, простота оценки операционных характеристик благодаря заранее заданным взаимосвязям. К *недостаткам* модели относится трудность организации выполнения таких операций как включение, добавление и удаление данных, возможность доступа к любому сегменту только через все предшествующие сегменты включая обязательно корневой сегмент и другие проблемы, связанные со спецификой модели. Примеры СУБД для иерархической модели: IMS, PC/FOCUS, Team-Up, Data Edge, Adabas, ОКА, ИНЭС, МИРИС и др.

2.10. Сетевая модель представления данных

Рассмотрим, как можно представить данные, подлежащие хранению в БД, с помощью сетевой модели (рис.2.5).

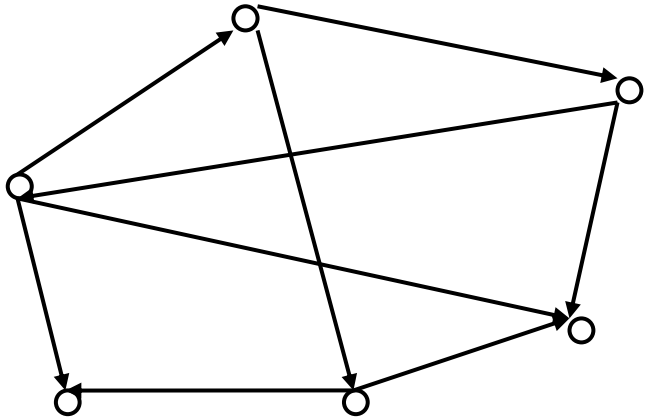


Рис.2.5. Представление данных в виде сетевой модели

В сетевой модели используются понятия **элемент – запись – набор** и успешно реализуется взаимосвязь М : М. Объект описывается в виде *записи*, а его свойства – в виде совокупности *элементов*, составляющих запись. *Набор* служит для организации связей между записями. Каждая порожденная запись может иметь более одной исходной записи. Сетевая структура бывает простой или сложной в зависимости от того, какова связь «исходный – порожденный», может содержать циклы. Сеть по сравнению с иерархией является более общей структурой.

Введем понятие *набора*. Если m – отображение вида М : 1 записей типа R в записи типа S, то с каждой записью r типа R можно ассоциировать множество S_r записей s типа S таким образом, что $m(s)=r$. Тогда каждое множество S_r вместе с записью r называется экземпляром набора. Запись r называется владельцем набора, а каждая запись s такая, что $m(s)=r$ – членом этого набора.

Приведем пример набора (рис. 2.6). Между записью типа R и записью типа S имеется взаимосвязь 1: М. В экземпляр набора для записи-владельца r_1 как члены набора входят записи-члены s_1 и s_2 (множество S_{r1}). В экземпляр набора для записи-владельца r_2 как члены набора входят записи-члены s_1 , s_5 и s_{11} (множество S_{r2}).

Рассмотрим в качестве примера БД «Поставляемые изделия». Имеются три записи: 1) запись *P* «поставщик» с элементами «Номер поставщика», «Фамилия», «Город»; 2) запись *IZ* «Изделие» с элементами «Номер изделия», «Название», «Цвет», «Завод»; 3) запись *PC* «Поставка» с элементами «Номер поставщика», «Номер изделия», «Количество». Для реализации взаимосвязей создано два набора: набор *P-PC* – «Поставщик - Поставка»; набор *IZ-PC* – «Изделие - Поставка». Структура этой базы данных в виде сетевой модели представлена на рис. 2.7.

К *достоинствам* сетевой модели относится простота реализации взаимосвязи М : М, разделение данных и связей, легкость выполнения операций включения и удаления.

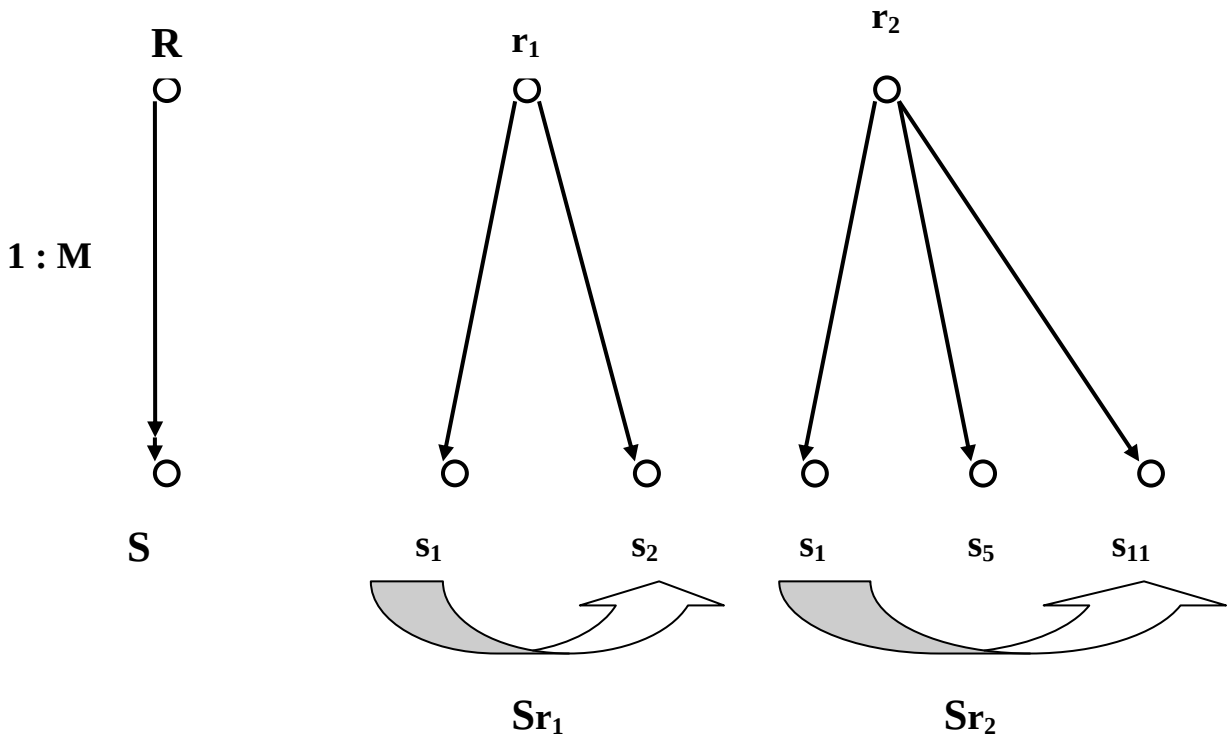


Рис.2.6. Пример набора

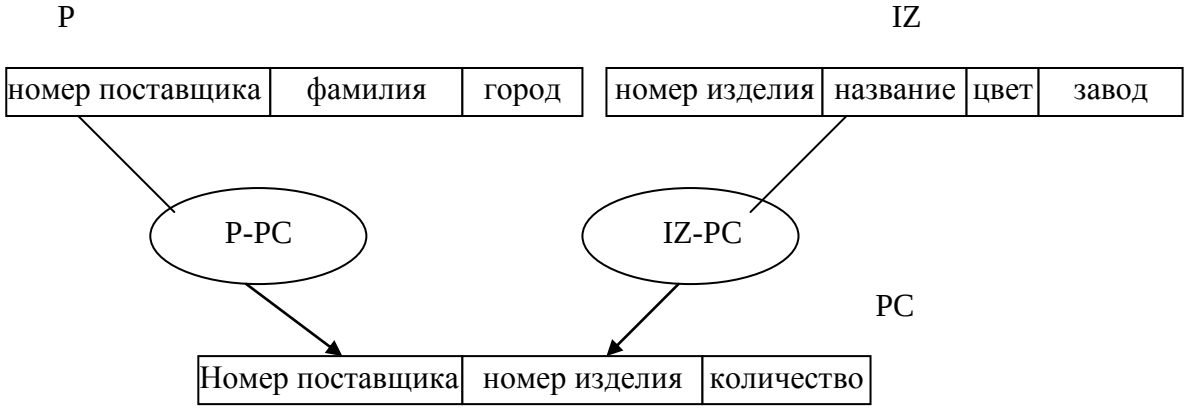


Рис. 2.7. Структура БД «Поставляемые изделия»

К *недостаткам* относится сложность представления данных как на логическом, так и на физическом уровне, сложность программ при реализации СУБД для сетевой модели. Примеры СУБД для сетевой модели: IDMS, СЕТЬ, db-Vista, СЕТОР, КОМПАС и др.

2.11. Постреляционная модель представления данных

Постреляционная модель представления данных является расширенной реляционной моделью, в которой снято ограничение неделимости данных, хранящихся в записях таблиц БД. В постреляционной модели допускаются

многозначные поля, значения в которых являются множеством. Набор значений многозначных полей является самостоятельной таблицей, встроенной в основную таблицу. В табл. 2.24 приведен пример представления данных в виде постреляционной модели.

Таблице 2.24

Пример данных в виде постреляционной модели

Код заказа	Фамилия клиента	Наименование товара	Стоимость за единицу товара	Количество товара
112	Каримов	холодильник телевизор	45000 38000	1 2
126	Ли	видеокамера	55000	4
131	Ахметов	холодильник видеокамера газплита	45000 55000 26000	2 1 2

В постреляционной модели [5] данные хранятся более эффективно по сравнению с реляционной моделью. При обработке не требуется выполнять операцию «соединение» для двух таблиц. Кроме того, в постреляционной модели поддерживаются ассоциированные многозначные поля (множественные группы). Совокупность ассоциированных полей называется ассоциацией. В строке таблицы первое значение одного столбца ассоциации соответствует первым значениям всех других столбцов ассоциации. В постреляционной модели на длину и количество полей в записях таблицы не накладывается требование постоянства, что делает структуру данных и таблиц более гибкой.

В постреляционной модели допускается хранение в таблицах БД ненормализованных данных, вследствие чего возникает проблема обеспечения целостности и непротиворечивости данных. Эта проблема решается путем включения в СУБД на основе постреляционной модели инструментов, подобных хранимым процедурам в клиент-серверных системах. Для поддержки функций контроля значений в полях таблиц БД создаются специальные процедуры – коды конверсии и коды корреляции, которые автоматически запускаются на обработку как до, так и после обращения к данным. При этом коды корреляции выполняются сразу после чтения данных, перед их обработкой, а коды конверсии выполняются сразу после обработки данных.

Достоинства постреляционной модели: возможность представления совокупности связанных реляционных таблиц в виде одной постреляционной таблицы; высокая наглядность представления информации; повышение эффективности обработки данных в БД.

Недостатки постреляционной модели: сложность решения проблемы обеспечения целостности и непротиворечивости данных.

Примеры СУБД на основе постреляционной модели: uniVers, Bubba, Dasdb и др.

2.12. Многомерная модель представления данных

Многомерная модель представления данных появилась почти одновременно с реляционной моделью, но интерес к этой модели возрос только с середины 90-ых годов прошлого столетия. Codd E. опубликовал статью (1993 г.), в которой сформулировал 12 основных принципов к системам класса OLAP (OnLine Analytical Processing – оперативная аналитическая обработка), важнейшие из которых связаны с возможностями концептуального представления и обработки многомерных данных. Системы с использованием многомерной модели позволяют оперативно обрабатывать информацию для проведения анализа и принятия решений.

Многомерность модели данных означает не многомерность визуализации цифровых данных, а многомерное логическое представление структуры информации при описании и в операциях манипулирования данными. По сравнению с реляционной моделью многомерная модель обладает более высокой наглядностью и информативностью. В таблицах 2.25 и 2.26 приведены соответственно реляционное и многомерное представления одних и тех же данных.

Таблица 2.26

Данные в виде реляционной модели

Модель телефона	*Месяц*	*Объем продаж*
Nokia	январь	70
Nokia	февраль	60
Nokia	март	50
Samsung	январь	20
Samsung	февраль	35
Ericson	февраль	45

Таблица 2.27

Данные в виде многомерной модели

Модель телефона	*январь*	*февраль*	*март*
Nokia	70	60	50
Samsung	20	35	-
Ericson	-	45	-

В развитии концепции информационных систем можно выделить два направления:

- системы оперативной (транзакционной) обработки – такие ИС создаются на основе баз данных, построенных на реляционной модели;
- системы аналитической обработки (системы поддержки принятия решений) – такие ИС создаются на основе баз данных, построенных на

многомерной модели, которая обеспечивает более эффективную обработку данных.

СУБД на основе многомерной модели данных [5] предназначены для интерактивной аналитической обработки информации. Здесь используют такие основные понятия, как агрегируемость, историчность и прогнозируемость данных.

Агрегируемость данных означает рассмотрение информации на различных уровнях ее обобщения. В ИС степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь-оператор, менеджер, руководитель.

Историчность данных предполагает обеспечение высокого уровня статичности (неизменности) собственно данных и их взаимосвязей, а также обязательную привязку данных к времени. Статичность данных позволяет использовать при их обработке специализированные методы хранения, загрузки, индексации и выборки. Привязка данных к времени необходима, поскольку запросы, как правило, содержат в критерии на выборку данных время и дату.

Прогнозируемость данных подразумевает задание функций прогнозирования и применение их к различным временным интервалам.

Основными понятиями *многомерной модели данных* являются:

- ***показатель*** – это величина (обычно числового типа), которая собственно и является предметом анализа. Это, например, объём продаж некоторого товара, или выручка от продаж товара. Один OLAP-куб может обладать одним или несколькими показателями;
- ***измерение*** (dimension) – это множество объектов одного или нескольких типов, организованных в виде иерархической структуры и обеспечивающих информационный контекст числового показателя. Измерение принято визуализировать в виде ребра многомерного куба;
- ***члены измерений*** (members) – объекты, совокупность которых и образует измерение. Члены измерений визуализируют как точки или участки, откладываемые на осях гиперкуба. Например, временное измерение: день, месяц, квартал, год – наиболее часто используются в анализе, может содержать следующие члены: 8 мая 2012 года, май 2012 года, 2-ой квартал 2012 года и 2012 год. Объекты в измерениях могут быть различного типа, например «производители – марки автомобиля» или «годы – кварталы»;
- ***ячейка*** (cell) – атомарная структура куба, соответствующая конкретному значению некоторого показателя. Ячейки при визуализации располагаются внутри куба. Здесь же принято отображать соответствующее значение показателя.

Пример многомерной модели данных представлен на рис. 2.8. Для многомерной модели с мерностью больше двух необязательно информацию представлять в виде многомерных объектов (трех-, четырех- и более мерных гиперкубов). Пользователю зачастую более удобно иметь дело с двухмерными таблицами, которые представляют собой «срезы» многомерных данных, выполненные с разной степенью детализации.

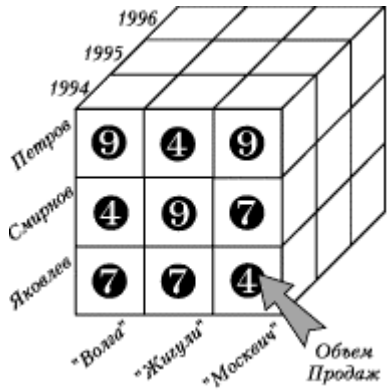


Рис. 2.8. OLAP-куб с тремя измерениями

В многомерных СУБД применяют два основных варианта организации данных (схемы):

1) поликубическую схему, когда в БД может быть определено несколько гиперкубов с различной размерностью и с различными измерениями в качестве граней;

2) гиперкубическую схему, когда в БД все показатели определяются одним и тем же набором измерений; при наличии нескольких гиперкубов все они имеют одинаковую размерность и совпадающие измерения.

Для многомерной модели применяют ряд специальных операций.

1) «*Срез*» (Slice) –представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений. Формирование срезов выполняется для ограничения используемых пользователем значений, поскольку все значения гиперкуба практически никогда одновременно не используются. *Пример*: если ограничить значения измерения «модель авто» в гиперкубе (рис. 2.8) маркой «Жигули», то получится двухмерная таблица продаж этой марки различными менеджерами по годам (табл. 2.28).

2) «*Вращение*» (Rotate) – применяется при двухмерном представлении данных. Суть операции заключается в изменении порядка следования измерений при визуальном представлении данных. *Пример*: «вращение» (табл. 2.27) приведет к изменению вида таблицы таким образом, что по оси X будет марка телефона, а по оси Y – месяцы. Эту операцию можно обобщить и на многомерный случай, если под ней понимать процедуру изменения порядка следования измерений.

3) «*Агрегация*» (Drill Up) – переход к более общему порядку представления информации пользователю из гиперкуба. *Пример*: пусть имеется гиперкуб с измерениями: *Год, Менеджер, Модель авто, Подразделение, Регион, Фирма, Страна*. В этом случае в гиперкубе существует иерархия (снизу вверх) отношений между измерениями:

Менеджер
Подразделение
Регион
Фирма
Страна

Пусть в гиперкубе определено, насколько успешно менеджер Петров в 1995 году продавал автомобили марки «Жигули» и «Волга». Тогда, поднимаясь на уровень выше по иерархии, с помощью операции «агрегация» можно выяснить, как выглядит соотношение продаж этих же марок на уровне подразделения, где работает Петров.

4) «*Детализация*» (Drill Down) – переход к более детальному представлению информации пользователю из гиперкуба. Эта операция противоположна «агрегации».

Таблица 2.28

Продажи автомобилей «Жигули»

год / Продавец	1994	1995	1996
Петров	4	10	2
Смирнов	9	3	8
Яковлев	7	5	6

Достоинства многомерной модели: удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. При организации обработки аналогичных данных в реляционной модели происходит нелинейный рост трудоемкости операций в зависимости от размера базы данных и существенное увеличение затрат оперативной памяти на хранение индексных файлов.

Недостатки многомерной модели: громоздкость модели для простейших задач обычной оперативной обработки информации.

Примеры СУБД, поддерживающих многомерную модель: Essbase (Arbor Software), Media Multi-matrix (Speedware), Oracle Express Server (Oracle), Cache (InterSystems) и др. Некоторые системы, например, Media Multi-matrix, позволяют одновременно работать с многомерными и реляционными БД. С СУБД Cache, где внутренней моделью является многомерная модель, реализованы три способа доступа к данным: прямой (на уровне узлов многомерных массивов), объектный и реляционный.

2.13. Объектно-ориентированная модель представления данных

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи БД. Между записями БД и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, аналогичных подобным средствам в объектно-ориентированных языках программирования.

Стандартизированная объектно-ориентированная модель описана в рекомендациях стандарта ODMG-93 (Object Database Management Group – группа управления объектно-ориентированными БД). Рассмотрим несколько

упрощенную объектно-ориентированную модель, поскольку в полном объеме рекомендации ODMG-93 пока не реализованы.

Структура объектно-ориентированной БД [5] графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом (например, *string*) или типом, создаваемым пользователем (определяется как *class*). Значением свойства типа *string* является строка символов. Значением свойства типа *class* является объект, представляющий собой экземпляр соответствующего класса. Каждый объект- экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект- экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в объектно-ориентированной БД образуют связную иерархию объектов.

Пример логической структуры объектно-ориентированной БД для предметной области «Обслуживание в библиотеке» представлен на рис. 2.9. Объект «Библиотека» является родителем для объектов- экземпляров классов «Абонент», «Каталог» и «Выдача». Различные объекты типа «Книга» могут иметь одного или разных родителей. Объекты типа «Книга», имеющие одного и того же родителя, должны различать по крайней мере *номером* (уникален для каждого экземпляра книги), но могут иметь одинаковые значения свойств *isbn*, *удк*, *название* и *автор*.

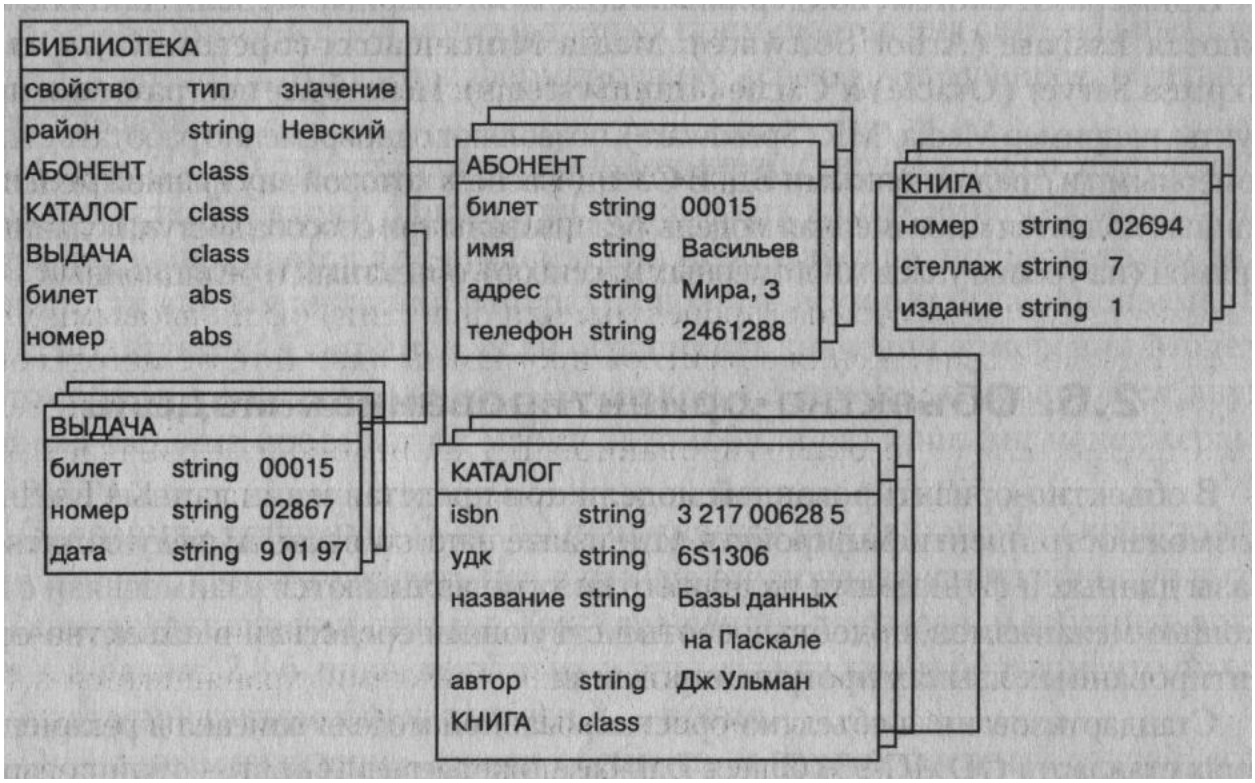


Рис. 2.9. Логическая структура объектно-ориентированной БД

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними заключается в методах манипулирования данными. Для выполнения действий над данными в

объектно-ориентированной БД применяются логические операции, усиленные объектно-ориентированными механизмами инкапсуляции, наследования и полиморфизма. Создание и модификация БД сопровождается автоматическим формированием и последующей корректировкой индексов, содержащих информацию для быстрого поиска данных.

Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено. Так, если в объект типа «Каталог» добавить свойство *телефон* для автора книги, то мы получим одноименные свойства у объектов «Каталог» и «Абонент». Смысл такого свойства будет определяться тем объектом, в котором оно инкапсулировано.

Наследование, наоборот, распространяет область видимости свойства на всех потомков объекта. Так, всем объектам типа «Книга», являющимся потомками объекта типа «Каталог», можно приписать свойства объекта-родителя: *isbn, удк, название* и *автор*. Если необходимо расширить действие механизма наследования на объекты, не являющиеся непосредственными родственниками (например, между двумя потомками одного родителя), то в их общем предке определяется абстрактное свойство *abs*. Так, определение абстрактных свойств *билет* и *номер* в объекте «Библиотека» приводит к наследованию этих свойств всеми дочерними объектами «Абонент», «Каталог» и «Выдача». Именно поэтому значения свойства *билет* классов «Абонент» и «Выдача» одинаковые (00015 – см. рис. 2.9).

Полиморфизм в объектно-ориентированных языках программирования означает способность одного и того же программного кода работать с разнотипными данными. Практически это означает, что в объектах разного типа можно иметь методы (процедуры, функции) с одинаковыми именами. Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента. Применительно к объектно-ориентированной БД полиморфизм означает, что объекты класса «Книга», имеющие разных родителей из класса «Каталог», могут иметь разный набор свойств. Следовательно, программы работы с объектами класса «Книга» могут содержать полиморфный код.

Поиск в объектно-ориентированной БД заключается в выяснении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в БД. Определяемый пользователем объект, называемый объектом-целью (свойство объекта имеет тип *goal*), в общем случае может представлять собой подмножество всей хранимой в БД иерархии объектов. Как объект-цель, так и результат выполнения запроса, могут храниться в БД. Пример запроса о номерах читательских билетов и именах читателей, получивших в библиотеке хотя бы одну книгу, приведен на рис. 2.10.

Достоинства объектно-ориентированной модели: возможность отображения информации о сложных взаимосвязях объектов, возможность идентифицировать отдельную запись БД и определить для нее функции обработки. *Недостатки* объектно-ориентированной модели: высокая понятийная сложность, неудобство обработки данных, низкая скорость выполнения запросов.

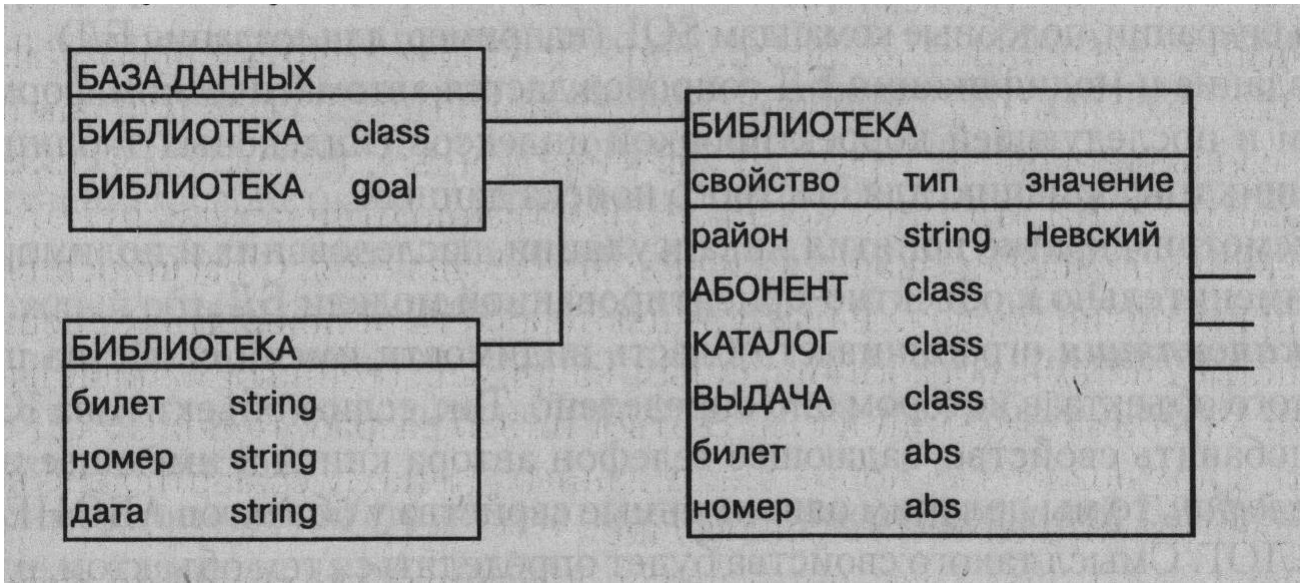


Рис. 2.10. Фрагмент БД с объектом-целью

СУБД на основе объектно-ориентированной модели: G-Base (Grapael), GemStone (Servio-Logic совместно с OGI), Statice (Symbolics), ObjectStore (Object Design), Objectivity/DB (Objectivity), Versant (Versant Technologies), O2 (Ardent Software), ODB-Jupiter (НПЦ «Интелтек Плюс»), Iris, Orion, Postgres и др.

2.14. Задания и контрольные вопросы

Для закрепления рассмотренного теоретического материала и получения практических навыков по представлению данных в виде одной из существующих моделей необходимо выполнить задания, представленные ниже, и ответить на контрольные вопросы.

Задание 1.

5. Выберите предметную область, выберите несколько связанных между собой объектов из этой предметной области и определите их свойства.
6. Сформируйте представление этих объектов в виде отношений для реляционной модели.
7. Изучите имеющиеся нормальные формы.
8. Определите, какие нежелательные зависимости имеются в отношениях и проведите процесс нормализации отношений.

Задание 2.

1. Изучите операции реляционной алгебры и алгоритмы их выполнения.
2. Составьте примеры и выполните каждую из операций реляционной алгебры.

Задание 3.

1. Изучите аппарат реляционного исчисления.

2. Используя отношения, построенные в задании 1, сформулируйте запросы на поиск данных и постройте выражения для реляционного исчисления.
3. Определите, как можно оформить эти же запросы с помощью операций реляционной алгебры.

Задание 4.

1. Изучите операторы языка SQL.
2. Используя отношения, построенные в задании 1, сформулируйте запросы на выполнение таких операций над данными как выборка, корректировка, удаление, добавление.
3. Оформите эти запросы с помощью операторов на языке SQL.

Задание 5.

1. Изучите принципы представления данных в виде иерархической модели.
2. Составьте пример на представление данных в виде иерархической модели.
3. Определите, какие элементы являются сегментами, какие полями, какие образуют запись.

Задание 6.

1. Изучите принципы представления данных в виде сетевой модели.
2. Составьте пример на представление данных в виде сетевой модели.
3. Определите, какие элементы являются записями, какие элементами, какие образуют набор.

Задание 7.

1. Изучите принципы представления данных в виде постреляционной модели.
2. Составьте пример на представление данных в виде постреляционной модели.

Задание 8.

1. Изучите принципы представления данных в виде многомерной модели.
2. Составьте пример на представление данных в виде многомерной модели.

Контрольные вопросы

1. Какие понятия используются в реляционной модели?
2. Как представляется объект и его свойства в реляционной модели данных?
3. Что такое функциональная зависимость?
4. Что такое транзитивная зависимость?
5. Какие шаги включает алгоритм нормализации отношений?
6. Какие операции есть в реляционной алгебре?
7. Как выполняется операция «проекция»?
8. Как выполняется операция «соединение»?
9. Какие символы используются в выражении реляционного исчисления?
10. Как записывается выражение в реляционном исчислении?

11. Для чего предназначен язык SQL?
12. Какие операторы имеются в языке SQL?
13. Как записывается оператор SELECT в языке SQL?
14. Как записывается оператор DELETE в языке SQL?
15. Как записывается оператор INSERT в языке SQL?
16. Как записывается оператор UPDATE в языке SQL?
17. Какие операторы языка SQL позволяют создавать таблицу БД с полями?
18. Какой вид имеет бланк запроса в языке QBE?
19. Какие термины используются в иерархической модели представления данных?
20. Как описывается объект в терминах иерархической модели?
21. Каковы особенности выполнения операций включения и удаления в иерархической модели?
22. Какие термины используются в сетевой модели представления данных?
23. Для чего используется набор в сетевой модели?
24. В чем особенности постреляционной модели?
25. Какие понятия используются в многомерной модели?
26. В чем особенности объектно-ориентированной модели?
27. Какие понятия используются в объектно-ориентированной модели?

ЗАКЛЮЧЕНИЕ

Теория баз данных и практика их применения активно развиваются. Огромные объемы накопленных и перерабатываемых информационными системами данных становятся доступными пользователям в любое время в любом месте земного шара. Быстрое развитие технологии хранения данных, коммуникаций и обработки позволяет переместить всю информацию в **киберпространство**, которое пересекает национальные границы. Программное обеспечение для определения, поиска и визуализации оперативно доступной информации – ключ к созданию, доступу и обработке данных. Сегодня наряду с графическими и текстовыми данными в базах хранятся и более богатые формы данных: графические, звуковые, видеообразы, географические карты, фотоснимки. **Мультимедийные базы** данных и средства доступа к ним являются краеугольным камнем на пути движения человечества к киберпространству.

Достижения в области разработки компьютерной техники сделали возможным переход от ручной бумажной обработки к современным средствам поиска и обработки информации. Ожидается, что этот прогресс в мире компьютерной техники будет продолжаться и в будущем. Программные пакеты управления данными развивались параллельно с развитием компьютерной техники. Системы, ориентированные на записи и наборы данных, открыли дорогу реляционным системам, которые теперь переросли в **объектно-реляционные СУБД**. Благодаря своей мощности и гибкости объектно-реляционные СУБД способны удовлетворять постоянно растущие требования к размерам и сложности данных. Обеспечивая хранение больших и сложных объектов и отслеживая их поведение, объектно-реляционные СУБД позволяют существенно повысить полезность и ценность баз данных с точки зрения их смыслового содержания. Реляционная модель, параллельные системы баз данных, активные базы данных и объектно-реляционные базы данных появились на свет в академических и промышленных исследовательских лабораториях. Развитие теории баз данных является хрестоматийным примером успешного сотрудничества академии и индустрии.

Применение баз данных в информационных системах стало неотъемлемой составляющей как функционирования большинства предприятий и фирм, так и деловой деятельности современного человека. В связи с этим все большую актуальность приобретает освоение новых принципов построения и эффективного применения современных информационных технологий и программного инструментария: СУБД, Case-средств автоматизации проектирования, средств администрирования и защиты баз данных. От правильного выбора инструментальных и программных средств проектирования информационных систем, определения рациональной модели представления данных, разработки оптимальной схемы данных в среде выбранной СУБД, оптимальной организации запросов к базе данных на поиск и

обработку и ряда других моментов во многом зависит эффективность функционирования информационных систем и информационных приложений.

Однако остается много нерешенных задач в управлении данными, как технических, так и социальных. В ближайшем будущем должны быть решены следующие задачи:

- определение моделей представления данных для новых типов данных (пространственных, темпоральных, графических) и их интеграция с традиционными системами баз данных;
- масштабирование баз данных по размеру (до петабайт), пространственному размещению (распределенные) и многообразию (неоднородные);
- автоматическое обнаружение тенденций изменения данных, их структур и аномалий;
- автоматизация проектирования и администрирования баз данных.

Оценку модели представления данных можно производить по следующим критериям:

1) структурная достоверность – соответствие способу определения и организации информации на предприятии;
2) простота – легкость понимания модели;
3) выразительность – способность представлять различия между разными типами данных, различия связей между данными и ограничений;
4) отсутствие избыточности;
5) способность к совместному использованию во многих информационных приложениях;
6) расширяемость – способность эволюционировать с целью включения новых требований с минимальным влиянием на уже существующие информационные приложения;
7) целостность – согласованность со способом использования и управления информацией внутри предприятия;
8) представление в виде диаграмм через понятные обозначения.

Распределенные системы имеют дело с проблемами, возникающими в связи с наличием данных, распределенных на множестве компьютеров в сети. При создании таких систем должны решаться вопросы, связанные с обработкой запросов, обнаружением тупиковых ситуаций и интеграцией неоднородных данных. *Масштабируемые системы* баз данных должны настраиваться на эффективное и надежное управление такими объемами данных, которые превышают размер физической памяти на несколько порядков.

Создание и использование *динамических* баз данных позволяет естественным образом представить временной фактор, обеспечить в любой момент времени адекватность хранимой информации отображаемым реальным объектам и отношениям, в том числе, на основе прогнозирования ее изменений, расширить возможности использования баз данных для указанного класса информационных систем, включая выдачу необходимой информации в виде временных рядов для периодов времени в прошлом, настоящем и сравнительно недалеком будущем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Дейт К. Введение в системы баз данных. М.: Издательский дом «Вильямс», 2001. 815 с.
2 Абдуллина В.З. Системы баз данных: Учебник. Алматы: КазНТУ, 2009. 222 с.
3 Четвериков В.Н. и др. Базы данных и знаний: Учебник. М.: Высшая школа, 1987. 248 с.
4 Ульман Д., Уидом Дж. Введение в системы баз данных. М.: Издательство «Лори», 2000. 508 с.
5 Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник. М.: Бином-пресс, 2007. 635 с.
6 Горев А. И. и др. Эффективная работа с СУБД. СПб.: Питер, 1997. 704 с.
7 Абдуллина В.З. Базы и банки данных: Уч.пос. Алматы: КазНТУ, 2003. 109 с.
8 Карпова Т.С. Базы данных: модели, разработка, реализация. СПб.: Питер, 2001. 472 с.
9 Григорьев Ю.А., Ревунков Г.И. Банки данных: Учебник для вузов. М.: Изд-во МГТУ им.Н.Э. Баумана, 2002. 320 с.
10 Кузин А.В. Базы данных: учеб. пос. для вузов. М.: Акад., 2010. 343 с.
11 Диго С.М. Базы данных: проектирование и использование: Учебник. М.: «Финансы и статистика», 2005. 378 с.
12 Райтингер М., Муч Г. Visual Basic 6: полное руководство. Киев: Издательская группа BHV, 1999. 718 с.
13 Коннэл Дж. Visual Basic 6. Введение в программирование баз данных. М.: ДМК, 2000. 514 с.
14 Андерсен В. Базы данных Microsoft Access. Проблемы и решения. Практ. пособие. М.: Издательство ЭКОМ, 2001. 384 с.
15 Абдуллина В.З., Балафанов Е.К., Бурибаев Б. Работа с Access: Лабораторный практикум. Уч.пос. Алматы, ИНТ, 2005. 103 с.
16 Абдуллина В.З. , Балапанов Е.Қ., Бөрібаев Б. Access жүйесімен жұмыс істеу: Лабораториялық практикум. Оқу құралы. Алматы, ЖТИ, 2005. 109 с.
17 Абдуллина В.З. «Информационные технологии» - Учебно-методический комплекс дисциплины (для спец-ти 050703 – Информ. системы). Уч. пос.. Алматы: КазНТУ, 2011. 114 с.
18 Абдуллина В.З., Балгабаева Л.Ш. Системы баз данных. Программа курса (SYLLABUS) и учебно-методический комплекс дисциплины (для спец-ти 5В070300 – «Информационные системы»). Уч. пос. Алматы: КазНТУ, 2011. 140 с.
19 Абдуллина В.З., Балгабаева Л.Ш. «Мәліметтер база жүйелері». Курс бағдарламасы (SYLLABUS), студенттің пәндік оқу-әдістемелік кешені 5В070300 – «Ақпараттық жүйелер» мамандығы үшін. Оқу құралы. Алматы, ҚазҰТУ, 2011. 117 с.

20 Тукеев У.А. Автоматизированное проектирование баз данных и программных комплексов АСУТП. Алматы: Гылым,1993. 183 с.
21 Казиев Г.З., Абдуллина В.З. Поддержка целостности в темпоральной реляционной базе данных. // Вестник КазНТУ, 3/4, 2003. С. 235 – 242.
22 Sokolova S.P., Abdullina V.Z., et. al. Artificial Immune System for the gerbil natural plague focus. Almaty, 2002. 180 p.
23 Sokolova S.P., Abdullina V.Z., Sokolova L.U. Monitoring of the natural plague foci using immunocomputing. // Proceedings of the 15th International conference on System Since. Vol.3. Knowledge Engineering and Intelligent systems. Information systems. Technical applications. Non - technical applications. Wroclaw, Poland, 2004. P. 509 – 516.
24 Абдуллина В.З. Проектирование информационных систем с временной динамикой. // Труды междун. конференции «Инженерное образование и наука в XXI веке», т.3. Алматы, КазНТУ, 2004. С. 78 –83.
25 Абдуллина В.З. Реляционная модель данных с временной динамикой. // Вестник КазНТУ, 1 (45), 2005. С. 147 – 153.
26 Абдуллина В.З. Исследование реляционной и алгебраической семантики *ANU*-исчисления в реляционной модели данных с временной динамикой. // Вестник КазНТУ, 5, 2007. С. 166 – 170.
27 Абдуллина В.З. Проектирование и отладка пользовательских приложений. Ч. 1. и ч. 2. Методические указания к курсовой работе и самостоятельной работе студентов по курсу «Базы, банки данных и экспертные системы». Алматы: КазНТУ, 2001. 76 с.
28 Абдуллина В.З. Обработка баз данных в Access и Visual Basic. Методические указания к лабораторным работам по дисциплине «Базы данных». Алматы: КазНТУ, 2004. 44 с.
29 Абдуллина В.З. Реализация бизнес-процессов как информационной технологии. Метод. указания к лаб. работам и СРС по дис-не «Информационные технологии» для спец-ти спец-ти 050703 – Информационные системы. Алматы: КазНТУ им. К.И. Сатпаева, 2012. 46 с.
30 Абдуллина В.З. Особенности проектирования информационных систем для бизнеса. Материалы III международной научно-практической конференции «Бизнес и образование: вектор развития». Алматы, 2003. С. 10-14.

ГЛОССАРИЙ

Предметная область – часть реального мира, в которой можно выделить один или несколько объектов.

Объект – человек, предмет, событие, место, явление, понятие.

Информация – отображение предметной области, существующее в представлении людей.

Данные – отображение предметной области, хранящееся в компьютере.

Взаимосвязь – выражает отображение или связь между двумя множествами данных, бывает трех типов: «один к одному» (1:1), «один ко многим» (1:M или 1: ∞), «многие ко многим» (M : M).

База данных – представляет собой поименованную совокупность данных, отображающую состояние множества объектов из предметной области, их атрибутов (свойств) и взаимоотношений.

Первичный ключ – это атрибут, идентифицирующий объект (запись) уникальным образом.

Вторичный ключ – идентифицирует группу объектов (записей).

Банк данных – организационно-техническая система, представляющая собой совокупность баз данных, технических и программных средств формирования и ведения этих баз и коллектива специалистов, обеспечивающих функционирование этой системы.

Система управления базой данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания и ведения баз данных.

Свойства данных в БД – интеграция, независимость, отсутствие дублирования, защита и целостность.

Целостность данных – безошибочность, точность и достоверность данных в БД в каждый момент времени.

Схема – логическое описание всех хранимых данных.

Подсхема – описание данных, которые используются каким-либо приложением или пользователем.

Администратор базы данных – коллектив специалистов, который отвечает за создание и ведение БД.

ЯОД – язык описания данных, используется для логического описания данных.

ЯМД – язык манипулирования данными, используется для обработки данных, позволяет реализовать интерфейс между приложением и СУБД.

ЯЗП – язык запросов пользователя, позволяет выбирать из БД все требуемые данные в соответствии с задаваемыми критериями.

Внешнее представление – совокупность требований к данным некоторого конкретного приложения или пользователя.

Концептуальное представление – это полная совокупность всех требований к данным, полученных из пользовательских представлений о предметной области.

Внутреннее представление – это сама реальная база данных, реализованная в памяти компьютера средствами выбранной СУБД.

Внешняя модель (подсхема) – описание каждого внешнего представления.

Концептуальная модель – реализация концептуального представления, в ней содержится описание объектов, их атрибутов и взаимосвязей (СУБД-независимая схема).

Внутренняя модель – реализует внутреннее представление, включает логическую модель и физическую модель.

Логическая модель – СУБД-ориентированная схема данных.

Физическая модель – специфицирует размещение файлов БД во внешней памяти, методы доступа и технику индексирования данных.

Загрузка – первоначальное заполнение БД данными.

Корректировка – изменяет содержимое БД в ходе ее эксплуатации.

Поиск – позволяет выбирать данные из БД по определенным критериям.

Модель данных – это абстрактное логическое определение объектов, операторов и других элементов.

Реляционная модель – представляет данные в виде отношение – атрибут, объект представляется в виде отношения, а его свойства – в виде совокупности атрибутов.

Нормализация отношений – процесс построения оптимальной структуры отношений и связей в реляционной базе данных путем избавления от нежелательных зависимостей.

Реляционная алгебра – система операций, используемая для манипулирования отношениями.

Реляционное исчисление – позволяет описать отношение и операции над ним в виде аналитического выражения.

Язык SQL (Structured Query Language) – язык структурированных запросов, позволяющий получать необходимую информацию из БД и являющийся стандартом в области взаимодействия с базами данных.

Язык QBE (Query By Example) – использует специальный редактор для подготовки запросов на получение данных из базы.

Иерархическая модель – представляет данные в виде поле – сегмент – запись, связь между данными является иерархической с четко установленными уровнями вложенности.

Сетевая модель – используется понятия элемент – запись – набор для представления данных, успешно реализует взаимосвязь M : M.

Постреляционная модель – является расширенной реляционной моделью, в которой снято ограничение неделимости данных, хранящихся в записях таблиц БД.

Многомерная модель – реализует многомерное логическое представление структуры информации при описании и в операциях манипулирования данными.

Объектно-ориентированная модель – устанавливает между записями БД и функциями их обработки взаимосвязи с помощью механизмов,

аналогичных подобным средствам в объектно-ориентированных языках программирования.

Окно проекта (Project1) в VB – содержит все формы проекта.

Окно свойств (Properties) в VB – содержит свойства активного объекта.

Элемент Data – устанавливает связь с таблицей БД.

Свойство DatabaseName для элемента Data – указывает имя файла, содержащего базу данных (полный путь), с которой устанавливается связь.

Свойство Record Source для элемента Data – указывает имя таблицы из базы данных, с которой устанавливается связь.

Свойство DataSource – задает имя элемента Data для элемента управления, отображающего поле таблицы БД.

Свойство DataField – задает имя поля из таблицы БД для элемента управления, отображающего поле этой таблицы.

Метод MoveFirst для элемента Data – переход на первую запись.

Метод MoveLast для элемента Data – переход на последнюю запись.

Метод MoveNext для элемента Data – переход на следующую запись.

Метод MovePrevious для элемента Data – переход на предыдущую запись.

Метод FindFirst для элемента Data – поиск первой записи, удовлетворяющей указанному критерию.

Метод FindLast для элемента Data – поиск последней записи, удовлетворяющей указанному критерию.

Метод Find Next для элемента Data – поиск следующей записи, удовлетворяющей указанному критерию.

Метод FindPrevious для элемента Data – поиск предыдущей записи, удовлетворяющей указанному критерию.

Метод AddNew для элемента Data – добавление новой записи в набор.

Метод Delete для элемента Data – удаление текущей записи из набора.

Метод Update для элемента Data – корректировка текущей записи в наборе.

Команда Utility / Data Form Designer в главном меню – служит для создания формы в автоматическом режиме в окне *Database Window*.

Команда Project / Add Form в главном меню – служит для создания формы в автоматическом режиме на закладке «New» при выборе Мастера VB Data Form Wizard.

Элемент AdoDC – организует связь с таблицей БД на новом этапе развития технологии доступа к данным.

Жизненный цикл БД – включает временной период от момента принятия решения о создании БД до момента принятия решения об ее уничтожении с последующим физическим удалением БД.

Концептуальное проектирование – результатом этапа является концептуальная модель или СУБД-независимая схема базы данных.

Логическое проектирование – результатом этапа является логическая модель или СУБД-ориентированная схема базы данных.

Физическое проектирование – результатом этапа является физическая модель, которая отражает решения по проектированию физических записей и их размещению в памяти, выбору методов доступа к данным в БД.

СОДЕРЖАНИЕ

Учебное издание

Абдуллина Валентина Заманбековна

БАЗЫ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Учебник

Нач. РО НТИЦ **З.А. Губайдулина**
Редактор Г.М. Дюсенбаева
Компьютерная верстка А.Б. Аришова

Подписано в печать 19.03.2015
Тираж 300 экз. Формат 60 х 84 1/16 . Бумага типографская № 1.
Уч.-изд. л. 23. Усл.п.л. 21,3. Заказ № 317. Цена договорная.

Издание Казахского национального технического университета
имени К.И. Сатпаева
Учебно- издательский центр КазНТУ,
г. Алматы, ул. Сатпаева, 22