Министерство образования и науки Российской Федерации
Волгоградский государственный архитектурно-строительный университет

С. Ю. Катеринина, Ю. И. Усков

УПРАВЛЕНИЕ ДАННЫМИ

Учебное пособие

Волгоград ВолгГАСУ 2015

УДК 004.451.5(075.8)
ББК 32.97я73
К293

Р е ц е н з е н т ы:
доктор технических наук *В. А. Игнатьев*, профессор,
заведующий кафедрой строительной механики
Волгоградского государственного архитектурно-строительного университета;
кандидат педагогических наук *Н. Ф. Жбанова*,
доцент кафедры информационных систем и технологий
Волгоградского государственного аграрного университета

Утверждено редакционно-издательским советом университета
в качестве учебного пособия

Катеринина, С. Ю.

К293 Управление данными [Электронный ресурс] : учебное пособие / С. Ю. Катеринина, Ю. И. Усков ; М-во образования и науки Рос. Федерации, Волгогр. гос. архит.-строит. ун-т. — Электронные текстовые и графические данные (4,8 Мбайт). — Волгоград : ВолгГАСУ, 2015. — Учебное электронное издание сетевого распространения. — Систем. требования: PC 486 DX-33; Microsoft Windows XP; Internet Explorer 6.0; Adobe Reader 6.0. — Официальный сайт Волгоградского государственного архитектурно-строительного университета. Режим доступа: http://www.vgasu.ru/publishing/on-line/ — Загл. с титул. экрана.

ISBN 978-5-98276-744-8

Рассмотрены основные разделы дисциплины «Управление данными»: понятие базы данных, модели данных. Особое внимание уделено вопросам, касающимся реляционной модели данных и баз данных на ее основе. Изложены основы языков запросов QBE и SQL, освещены перспективы развития баз данных и новых технологий работы с данными. Предложены индивидуальные задания по проектированию баз данных в среде Access.

Предназначено для студентов направления «Информационные системы и технологии» всех форм обучения. Также может быть использовано студентами любых специальностей, изучающими вопросы создания и использования баз данных.

Для удобства работы с изданием рекомендуется пользоваться функцией Bookmarks (Закладки) в боковом меню программы Adobe Reader.

УДК 004.451.5(075.8)
ББК 32.97я73

ISBN 978-5-98276-744-8

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

За последние десятилетия стало общепризнанным, что информация является не менее важным ресурсом человеческого общества, чем сырье, энергия и пища. В любом виде человеческой деятельности требуется удовлетворение информационных потребностей в той или иной степени.

Размещение в памяти компьютера больших объемов хорошо структурированной информации, в которой нуждаются представители многих видов профессиональной деятельности, создание быстрого и удобного доступа к ней — это сфера деятельности специалистов индустрии по обработке информационных ресурсов.

Развитие средств вычислительной техники и информационных технологий открыло новые возможности и способы хранения, представления и поиска информации.

Значительная часть IT-проектов направлена на разработку и создание информационных систем, в рамках которых осуществляется обработка данных различной сложности. Целью таких проектов является разработка и создание информационной системы с базами данных некоторого класса.

С возрастанием объема и разнообразия хранимой в базе данных информации возникает ее избыточность. В различных местах базы данных может находиться, например, фамилия одного и того же человека. Если требуется изменить или удалить ее, то не исключено, что после выполнения этой операции где-нибудь эти данные все-таки останутся без изменения. Как следствие, база данных будет содержать недостоверную или противоречивую информацию, что может привести к получению неверных результатов при ее использовании, причем ошибку пользователь может и не заметить.

Понимание этих и других проблем при создании баз данных — очень важная задача не только для разработчика, но и для пользователей, постоянно взаимодействующих с базами данных.

Определение целей создания базы данных, проектирование ее «на бумаге», без непосредственной связи с системой управления базами данных (СУБД) является достаточно сложной интеллектуальной работой, в которой сочетаются, наряду с некоторыми формальными правилами и логическими рассуждениями, интуиция и опыт.

Изучение этих информационных явлений и процессов обработки информации в среде СУБД (в частности, MS Access, MS SQL Server и др.) позволяет понять суть процесса хранения и обработки больших объемов информации, предоставления конкретному пользователю доступа к ней с соответствующими правами, а также способы разработки таких баз данных.

Целями данного учебного пособия являются: изложение основ современного представления об автоматизированных информационных системах; ознакомление с концепцией и технологией создания и развития банков данных, использованием систем управления базами данных, работой с данными, понятиями и методами организации реляционных баз данных, а также формирование общего представления о тенденциях развития баз данных.

Теория баз данных — сравнительно молодая область знаний. Возраст ее составляет чуть более 30 лет. За этот период существенно изменился сам ритм жизни, время уже не бежит, а летит. Практически все системы в той или иной степени связаны с функциями долговременного хранения и обработки информации. Фактически информация становится фактором, определяющим эффективность любой сферы деятельности.

В настоящее время трудно представить себе информационную систему, которая не имела бы в качестве основы или важной составляющей базу данных (БД). Концепции и технологии баз данных всегда были тесно связаны с развитием систем автоматизированной обработки информации. Проектирование баз данных после появления реляционного подхода превратилось из искусства в науку. Базы данных — это вполне сложившаяся дисциплина, являющаяся скорее инженерной, чем чисто научной, основанная на достаточно формализованных подходах.

Базам свойственна перманентность данных. Соответственно назначение систем управления базами данных (СУБД) — обеспечение в течение длительного времени их сохранности, а также возможностей выборки и актуализации.

Широкое использование баз данных различными категориями пользователей привело, с одной стороны, к созданию интерфейсов, требующих минимума времени на освоение средств управления системой, а с другой — к построению мощных, гибких СУБД. Появились средства автоматизации разработки, позволяющие создать БД любому пользователю, даже не владеющему основами теории БД.

Данное учебное пособие может рассматриваться как введение в проблематику теории и практики информационных систем, основанных на базах данных. Материал пособия может стать отправной точкой при освоении специальных дисциплин, посвященных информационным системам, изучаемых в рамках специальности «Информационные системы и технологии».

1. ОСНОВЫ ПОСТРОЕНИЯ БАНКОВ ДАННЫХ И ЗНАНИЙ

1.1. Информация, данные, знания

К базовым понятиям, которые используются в информатике, относятся: данные, информация и знания. Эти понятия часто используются как синонимы, однако между ними существуют принципиальные различия.

Термин «данные» происходит от слова data — факт, а понятие «информация» (informatiq) означает разъяснение, изложение, т. е. сведения или сообщение.

Информация — это результат преобразования и анализа данных. Информация, в отличие от данных, имеет смысл. Отличие информации от данных состоит в том, что данные — это фиксированные сведения о событиях и явлениях, которые хранятся на определенных носителях, а информация появляется в результате обработки данных при решении конкретных задач. Например, в базах данных хранятся различные данные, а по определенному запросу система управления базой данных выдает требуемую информацию.

Существуют и другие определения информации, например, информация — это сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределенности, неполноты знаний.

Данные — это информация, зафиксированная на определенном носителе в форме, пригодной для постоянного хранения, передачи и обработки.

Соответственно двум понятиям — «информация» и «данные» — в банках данных различают два аспекта рассмотрения вопросов: инфологический и датологический.

Инфологический аспект употребляется при рассмотрении вопросов, связанных со смысловым содержанием данных независимо от способов их представления в памяти системы.

На этапе инфологического проектирования информационной системы должны быть решены вопросы:

1. Информацию о каких объектах или явлениях реального мира требуется накапливать и обрабатывать в системе?

2. Какие основные характеристики и взаимосвязи объектов между собой будут учитываться?

3. Уточнения вводимых в информационную систему понятий об объектах и явлениях, их характеристиках и взаимосвязях.

Таким образом, на этапе инфологического проектирования выделяется часть реального мира, определяющая информационные потребности системы, т. е. ее предметную область.

Датологический аспект употребляется при рассмотрении вопросов представления данных в памяти информационной системы.

При датологическом проектировании системы, исходя из возможностей имеющихся средств восприятия, хранения и обработки информации, разрабатываются соответствующие формы представления информации в системе посредством данных, а также приводятся модели и методы их представления и преобразования, формулируются правила смысловой интерпретации данных.

Данные соответствуют зарегистрированным фактам об объектах или явлениях реального мира. Чтобы использовать их в дальнейшем, требуется смысловое содержание — *семантика данных*. Поэтому в информационной системе должны быть сформулированы правила смысловой интерпретации данных.

Работа с семантикой — это работа со знаниями. *Основное средство представления семантики данных* — естественный язык. Но можно использовать формализованные языки, которые позволяют более эффективно организовать обработку данных на вычислительной технике и представить необходимую семантику данных, удовлетворяющую практическим потребностям целого ряда прикладных задач. К этому классу информационных систем относятся и банки данных

Понятия «информация» и «знания» с философской точки зрения являются понятиями более высокого уровня, чем «данные», возникшие относительно недавно.

Понятие «информация» непосредственно связано с сущностью процессов внутри информационной системы, тогда так понятие «знание» скорее ориентировано на качество процессов. Понятие «знание» тесно связано с процессом принятия решений.

Знания — это зафиксированная и проверенная практикой обработанная информация, которая использовалась и может многократно использоваться для принятия решений. С другой стороны, знания — это вид информации, которая хранится в базе знаний и отображает знания специалиста в конкретной предметной области. Знания — это интеллектуальный капитал.

Формальные знания могут быть в виде документов (стандартов, нормативов), регламентирующих принятие решений, или учебников, инструкций с описанием решения задач. Неформальные знания — это компетенция и опыт специалистов в определенной предметной области.

1.2. Основные понятия банков данных и знаний

Сегодня трудно себе представить сколько-нибудь значимую информационную систему, которая не имела бы в качестве основы или важной составляющей базу данных. Концепции и технологии баз данных складывались постепенно и всегда были тесно связанны с развитием систем автоматизированной обработки информации.

Использование баз данных и информационных систем становится неотъемлемой составляющей деятельности современного человека и организаций, шагающих в ногу со временем. В связи с этим большую актуальность приобретает построение и эффективное применение соответствующих технологий и программных продуктов.

Перерабатывать большой объем информации в заданные сроки без специальных средств практически невозможно. К сожалению, большая часть информации еще находится вне ЭВМ, что объясняется отсутствием достаточного количества и номенклатуры технических средств обработки. Но если учесть, что стоимость ЭВМ снижается, то можно предположить, что в перспективе машинная обработка информации будет основной повсеместно. В ЭВМ могут храниться и обрабатываться не только печатные тексты, но и чертежи, фотографии, запись голосов и т. д.

Методы организации процессов обработки информации, реализуемые в концепции банков данных и знаний, позволили по-новому подойти к их реализации в автоматизированных системах. Рассматривая данные как один из ресурсов автоматизированных систем (АС), можно сказать, что банк данных (БнД) централизованно управляет этим ресурсом в интересах всей системы. Наличие централизованного управления данными — главная отличительная черта банка данных.

Таким образом, *банк данных* — это информационная система, реализующая централизованное управление данными в интересах всех пользователей АС, в состав которой она входит.

Предметная область — это область применения конкретного БнД. Различают банки данных, применяемые в сфере управления предприятиями и организациями, транспортом, в медицине, научных исследованиях и т. д.

Банки данных возникли в связи с потребностью в интеграции данных. Дальнейшее продвижение в этом направлении потребовало решения проблемы интеграции и процессов обработки, что привело к появлению банков знаний.

В банках знаний решаемые задачи интегрируются как по данным, так и по их обработке, возрастает интеллектуализация этих систем, цель которой — максимальное удовлетворение потребностей пользователей. Использование формальных методов преобразования и интерпретации данных позволяет автоматизировать процессы обработки накопленных в системе знаний, их получение и синтез.

Одной из отличительных особенностей банков знаний является наличие в них так называемого интеллектуального интерфейса, в состав которого входят база знаний, процессор общения и программа-планировщик. Интеллектуальный интерфейс решает проблему перевода текста, написанного на естественном языке и содержащего условие задачи, в рабочую программу решения этой задачи на ЭВМ. От пользователя требуется ввести в систему корректную постановку задачи, которая его интересует, на том профессиональном языке, на котором он работает в своей предметной области, а интеллектуальный интерфейс должен выполнить всю работу, которую ранее выполнял программист.

Банки данных и знаний являются одним из основных компонентов автоматизированных систем различных уровней и типов. Их создают для многих отраслей и сфер народного хозяйства: планирования, учета, управления предприятиями, статистики, здравоохранения и др.

Концепция банков данных стала определяющим фактором при создании систем автоматизированной обработки информации. Рассматривая общие вопросы, связанные с функционированием баз данных, обсуждаются основные компоненты банков данных и получивший наибольшее распространение трехуровневый подход к построению банков данных, включающий внешний, концептуальный и внутренний уровни представления данных.

Банк данных является современной формой организации хранения и доступа к информации. Существует много определений банка данных. Мы будем использовать следующее определение: *банк данных — это система специальным образом организованных данных (баз данных), программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.*

Массив данных, хранимый в вычислительной системе, называют базой данных. База данных вместе с системой управления ею является составной частью банка данных.

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Банк знаний (БнЗ) — это автоматизированная система, содержащая различные виды знаний (например, концептуальные, понятийные знания) о предметной области. Хранящиеся в БнЗ знания используются для вывода новых знаний на основании специальных механизмов, имеющихся в БнЗ.

С БнД и БнЗ в процессе их создания и эксплуатации взаимодействуют пользователи различных категорий, основными из которых являются конечные пользователи. Ими являются специалисты предметных областей, для удовлетворения информационных потребностей которых и создаются БнД и БнЗ. Конечные пользователи различаются сферой интересов, информационными потребностями, квалификацией и т. п. Конечными пользователями могут быть как физические лица, так и различные вычислительные процессы, задачи, а иногда и целые системы, взаимодействующие с БнД и БнЗ. Во всех случаях результатом взаимодействия является информация, данные, знания.

Банк данных — основной элемент информационной системы, в нем хранится информация по определенной проблеме в виде, допускающем компьютерную обработку. При этом сами данные образуют базу данных, а банк, наряду с базой, содержит программные средства обработки данных и реализации запросов, т. е. систему управления базой данных (СУБД). Как правило, банки данных являются системами коллективного пользования и информация, хранимая в них, доступна по телекоммуникационным сетям. В современном мире существует огромное число банков данных. В них содержатся сведения коммерческого характера, данные по библиотечным фондам, системам здравоохранения, транспорта и т. д.

Иногда в составе банка данных выделяют архивы. Основанием для этого является особый режим использования данных, когда только часть данных находится под оперативным управлением СУБД. Все остальные обычно располагаются на носителях, не управляемых СУБД. Одни и те же данные в разные моменты времени могут входить как в базы данных, так и в архивы. Банки данных могут не иметь архивов, но если они есть, то состав банка данных может входить и система управления архивами.

Эффективное управление внешней памятью являются основной функцией СУБД. Эти специализированные средства настолько важны с точки зрения эффективности, что при их отсутствии система просто не сможет решать некоторые задачи уже потому, что их выполнение будет занимать слишком много времени. При этом ни одна из таких специализированных функций (построение индексов, буферизация данных, организация доступа и оптимизация запросов) не является видимой для пользователя и обеспечивает независимость между логическим и физическим уровнями системы: прикладной программист не должен писать программы индексирования, распределять память на диске и т. д.

Основными требованиями, предъявляемыми к БнД, являются:

адекватность отображения предметной области (полнота, целостность и непротиворечивость данных, актуальность информации (т. е. ее соответствие состоянию объекта на данный момент времени));

возможность взаимодействия пользователей разных категорий и в разных режимах, обеспечение высокой эффективности доступа для разных приложений;

дружелюбность интерфейсов и быстрое освоение системы, особенно для конечных пользователей;

обеспечение секретности и конфиденциальности для некоторой части данных;

определение групп пользователей и их полномочий;

обеспечение взаимной независимости программ и данных;

обеспечение надежности функционирования БнД, защита данных от случайного и преднамеренного разрушения;

возможность быстрого и полного восстановления данных в случае их разрушения;

технологичность обработки данных, приемлемые характеристики функционирования БнД (стоимость обработки, время реакции системы на запросы, требуемые машинные ресурсы и др.).

1.3. Компоненты банка данных

Банк данных является сложной человеко-машинной системой, включающей в свой состав различные взаимосвязанные и взаимозависимые компоненты, (рис. 1.1).

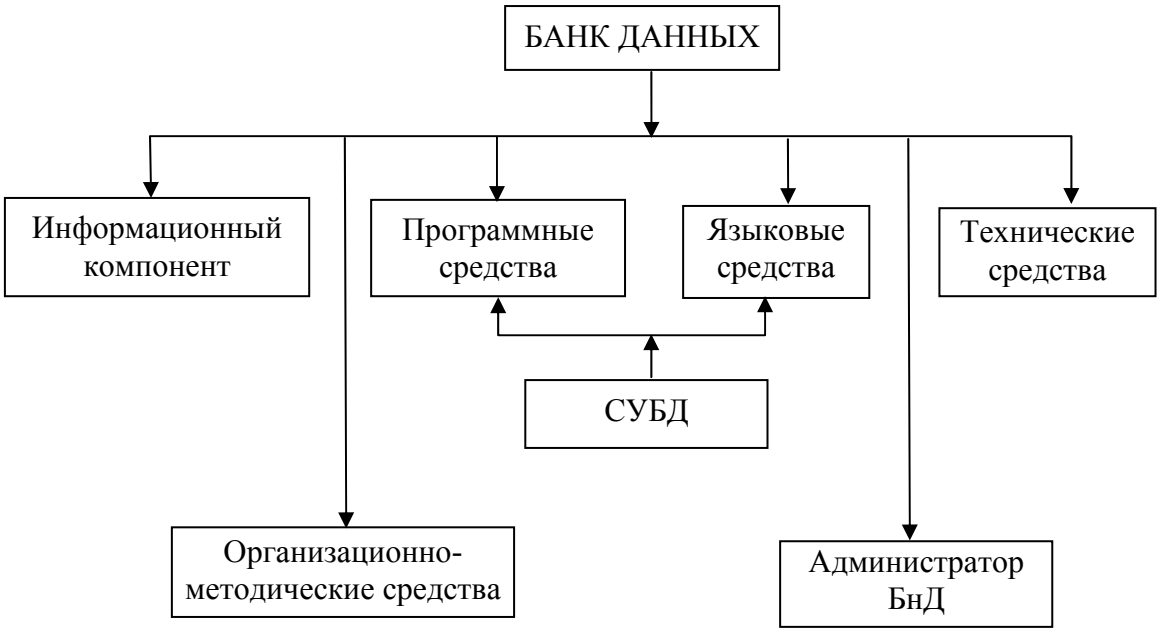

Рис. 1.1

Информационная компонента. Ядром БнД является база данных (БД). База данных — это поименованная совокупность взаимосвязанных данных, находящихся под управлением системы управления базой данных.

Существует множество определений базы данных. Некоторые из них имеют право на существование. Другие устарели и не соответствуют современным представлениям о БД. Так, в ранних определениях базы данных указывалось на их неизбыточность, отсутствие дублирования данных в них. На самом деле это не так. В базах данных может наблюдаться избыточность информации. Она может быть вызвана спецификой используемой модели данных, не позволяющей полностью устранить дублирование, или технологическими причинами (обеспечение большей надежности, сокращение времени реакции системы и др.). Но это должна быть управляемая избыточность, причины и цели возникновения которой известны администратору базы данных и управляются как им, так и системой управления базами данных (СУБД).

Системой управления базой данных называется совокупность языковых и программных средств, облегчающих для пользователей выполнение всех операций, связанных с организацией хранения данных, их корректировкой и доступом к ним.

Программные средства БнД представляют собой сложный комплекс, обеспечивающий взаимодействие всех частей информационной системы при ее функционировании (рис. 1.2).

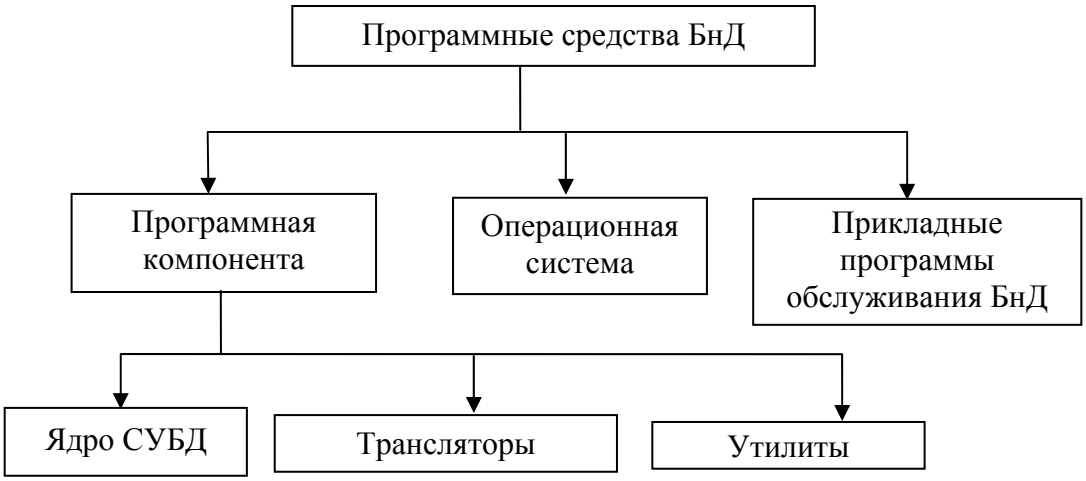

Рис. 1.2

Основу программных средств БнД представляет СУБД. В ней можно выделить ядро СУБД, обеспечивающее организацию ввода, обработки и хранения данных, а также другие компоненты, обеспечивающие настройку системы, средства тестирования, утилиты, обеспечивающие выполнение вспомогательных функций (восстановление баз данных, сбор статистики о функционировании БнД и др.). Важной компонентой СУБД являются трансляторы или компиляторы для используемых ею языковых средств.

Управляет базой данных *администратор базы данных*. Необходимо отметить, что при рассмотрении всего контура управления базой данных, следует учитывать и операционную систему (ОС), поскольку программы управления базой данных выполняются непосредственно под управлением ОС. Подавляющее большинство СУБД работает в среде универсальных операционных систем и взаимодействует с ОС при обработке обращений к БнД.

Для обработки запросов к БД пишутся соответствующие программы, которые представляют прикладное программное обеспечение БнД.

Языковые средства СУБД являются важнейшей компонентой банков данных, так как в конечном счете они обеспечивают интерфейс пользователей разных категорий с банком данных. Языковые средства большинства СУБД относятся к языкам четвертого поколения (к первому поколению языков относят машинные языки, ко второму — символические языки ассемблера, к третьему — алгоритмические языки типа PL, COBOL и т. п., которые в 60-е годы XX в. назывались языками высокого уровня, но уровень которых гораздо ниже, чем у языков четвертого поколения).

В качестве *технических средств для БнД* используется ЭВМ. В технической документации некоторых СУБД, а также в некоторых литературных источниках в состав БД включаются не только собственно хранимые данные о предметной области, но и описания БД. Более правильно описания баз данных считать самостоятельными компонентами БнД, даже если они и хранятся вместе с самими данными.

Организационно-методические средства представляют собой различные инструкции, методические и регламентирующие материалы, предназначенные для пользователей разных категорий, взаимодействующих с БнД.

Функционирование БнД невозможно без администраторов БнД — специалистов, обеспечивающих создание, функционирование и развитие БнД.

Структура банка данных. БнД — совокупность специальным образом организованных (структурированных) данных и связей между ними. Иными словами, БД — это так называемое датологическое (от англ. data — данные) представление информации о предметной области.

Если в состав БнД входит одна БД, банк принято называть локальным, если БД несколько — интегрированным.

ВС — вычислительная система, включающая технические средства и общее программное обеспечение, базы данных, систему управления базами данных, администратора баз данных (АБД), а также обслуживающий персонал и словарь данных.

В состав любой СУБД входят языки двух типов:

1) язык описания данных (с его помощью описываются типы данных, их структура и связи);

2) язык манипулирования данными (его часто называют языком запросов к БД), предназначенный для организации работы сданными в интересах всех типов пользователей.

Словарь данных предназначен для хранения единообразной и централизованной информации обо всех ресурсах данных конкретного банка:

об объектах, их свойствах и отношениях для данной ПО;

данных, хранимых в БД (наименование, смысловое описание, структура, связи и т. п.);

возможных значениях и форматах представления данных;

источниках возникновения данных;

кодах защиты и разграничении доступа пользователей к данным и т. п.

1.4. Администратор базы данных

Администратор — это лицо (группа лиц), реализующее управление БД. В этой связи сам БнД можно рассматривать как автоматизированную систему управления базами данных. Функции администратора базы данных (АБД) являются долгосрочными: он координирует все виды работ на этапах создания и применения БнД. На стадии проектирования администратор БД выступает как идеолог и главный конструктор системы; на стадии эксплуатации — отвечает за нормальное функционирование БнД, управляет режимом его работы и обеспечивает безопасность данных. Администратор баз данных изучает информационные потребности структурных подразделений, рассматривает предлагаемые отделом информации программные средства. Чтобы обеспечить достоверность результатов и уменьшить время расчетов, специалист совершенствует процесс хранения и обработки информации, внедряет новые программные средства, контролирует работу компьютеров, и при возникно-

вении неисправностей сообщает в отдел информатизации. АБД проверяет достоверность и следит за сохранностью данных в информационной системе, систематизирует в архив используемые программные средства и нормативно-справочную информацию, выполняет работы по созданию и обеспечению функционирования БД на протяжении всех этапов жизненного цикла системы. В составе группы администраторов банка данных можно выделить различные подгруппы в зависимости от выполняемых ими функций. Численность группы администрации, выполняемые ими функции будут в значительной степени зависеть от масштаба и типа банка данных, специфики хранимой в нем информации, особенностей используемых программных средств и некоторых других факторов.

В составе администрации базы данных должны быть системные аналитики, проектировщики структур данных и внешнего по отношению к банку данных информационного обеспечения, проектировщики технологических процессов обработки данных, системные и прикладные программисты, операторы, специалисты по техническому обслуживанию. Если речь идет о коммерческом банке данных, то важную роль здесь будут играть специалисты по маркетингу.

Администраторы базы данных выполняют большой круг разнообразных обязанностей:

1. Анализ предметной области: описание предметной области, выявление ограничений целостности, определение статуса информации, определение потребностей пользователей, определение статуса пользователей, определение соответствия «данные — пользователь», определение объемно-временных характеристик обработки данных.

2. Проектирование структуры базы данных: определение состава и структуры информационных единиц, составляющих базу данных, задание связей между ними, выбор методов упорядочения данных и доступа к информации, описание структуры БД на языке обработки данных (ЯОД).

3. Задание ограничений целостности при описании структуры базы данных и процедур обработки БД: задание ограничений целостности, присущих предметной области, определение ограничений целостности, вызванных структурой базы данных, разработка процедур обеспечения целостности БД при вводе и корректировке данных, обеспечение ограничений целостности при параллельной работе пользователей в многопользовательском режиме.

4. Первоначальная загрузка и ведение базы данных: разработка технологии первоначальной загрузки и ведения (изменения, добавления, удаления записей) БД, проектирование форм ввода, создание программных модулей, подготовка исходных данных, ввод и контроль ввода.

5. Защита данных от несанкционированного доступа:

обеспечение парольного входа в систему: регистрация пользователей, назначение и изменение паролей;

обеспечение защиты конкретных данных: определение прав доступа групп пользователей и отдельных пользователей, определение допустимых опера-

ций над данными для отдельных пользователей, выбор/создание программно-технологических средств защиты данных; шифрование информации с целью защиты данных от несанкционированного использования;

тестирование средств защиты данных;

фиксация попыток несанкционированного доступа к информации;

исследование возникающих случаев нарушения защиты данных и проведение мероприятий по их предотвращению.

6. Защита данных от разрушений. Одним из способов защиты от потери данных является резервирование. Используется как при физической порче файла, так и в случае внесения в БД нежелательных необратимых изменений.

7. Обеспечение восстановления БД: разработка программно-технологических средств восстановления БД, организация ведения системных журналов.

8. Анализ обращений пользователей к БД: сбор статистики обращений пользователей к БД, ее хранение и анализ (кто из пользователей к какой информации как часто обращался, какие выполнял операции, время выполнения запросов, анализ причин безуспешных (в том числе и аварийных) обращений к БД.

9. Анализ эффективности функционирования базы данных и развитие системы: анализ показателей функционирования системы (время обработки, объем памяти, стоимостные показатели), реорганизация, реструктуризация и изменение состава баз данных, развитие программных и технических средств.

10. Работа с пользователями: сбор информации об изменениях в предметной области и оценке пользователями работы базы данных, определение регламента работы пользователей с базой данных, обучение и консультирование пользователей.

11. Подготовка и поддержание системных программных средств: сбор и анализ информации о СУБД и других прикладных программ, приобретение, установка и развитие программных средств, проверка работоспособности, поддержание системных библиотек.

12. Организационно-методическая работа: выбор или создание методики проектирования БД, определение целей и направлений развития системы, планирование этапов развития базы данных, разработка и выпуск организационно-методических материалов.

2. АРХИТЕКТУРА БАЗЫ ДАННЫХ

2.1. Трехуровневая архитектура ANSI

Терминология в СУБД, да и сами термины «база данных» и «банк данных», частично заимствованы из финансовой деятельности. Это заимствование не случайно и объясняется тем, что работа с информацией и работа с денежными массами во многом схожи, поскольку и там и там отсутствует персонификация объекта обработки: две банкноты достоинством в сто рублей столь же неотличимы и взаимозаменяемы, как два одинаковых байта (естественно, за исключением серийных номеров). Можно положить деньги на некоторый счет и предоставить возможность родственникам или коллегам использовать их для иных целей, можно поручить банку оплачивать расходы со счета или получить их наличными в другом банке, и это будут уже другие денежные купюры, но их ценность будет эквивалентна той, которая имелась, когда их клали на счет.

Методы доступа к данным развивались на протяжении нескольких последних десятилетий от громоздких, физически ориентированных методов начального периода обработки файлов к различным видам обработки баз данных. Одним из наиболее важных аспектов реляционной революции стала идея отделения логической структуры, как она понимается конечным пользователем, от физического представления, требуемого компьютерным оборудованием. Это суть философии структуры данных, представленной в модели ANSI/SPARC трехуровневой системы организации БД, т. е. трех различных уровней описания элементов данных. Эти уровни формируют трехуровневую архитектуру, состоящую:

из внешнего — на нем представляют данные пользователи;

концептуального, который служит для отображения данных внешнего уровня на внутренний и обеспечивает необходимую независимость данных разного уровня друг от друга;

внутреннего, где данные воспринимаются СУБД и операционной системой (рис. 2.1).

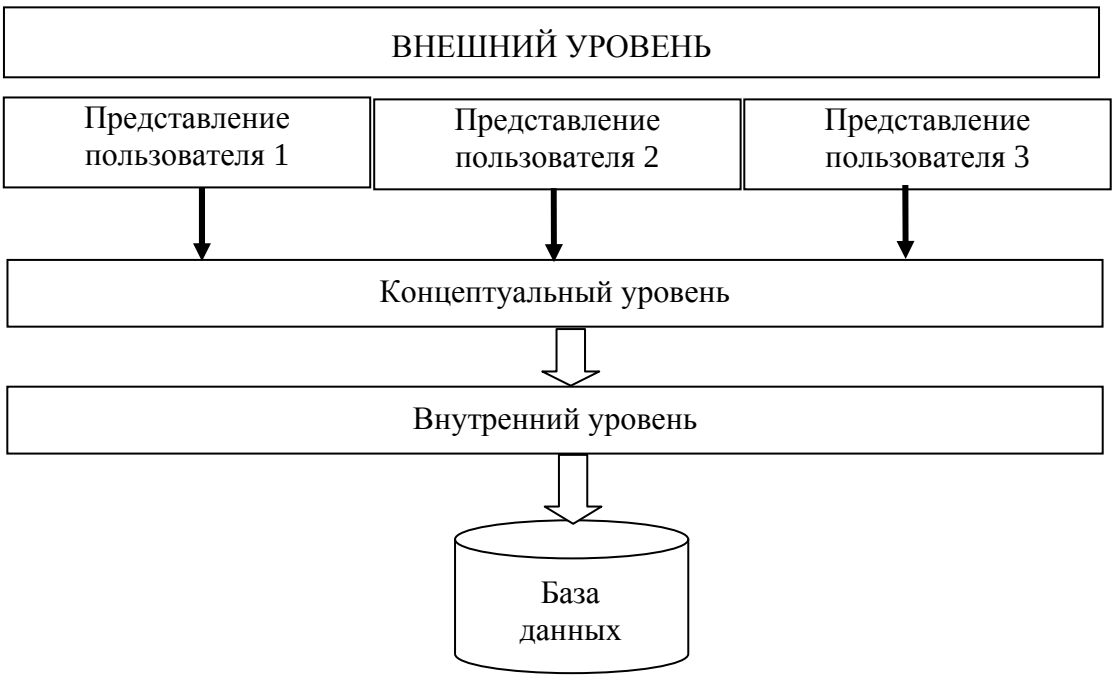

Рис. 2.1

Внешний уровень. Уровень внешних моделей — самый верхний уровень, где модель имеет свое видение данных. Этот уровень определяет точку зрения на БД отдельных приложений или пользователей. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно ему. Например, система распределения работ использует сведения о квалификации сотрудника, но ее не интересуют сведения об окладе, домашнем адресе и телефоне сотрудника, и наоборот, именно эти сведения используются в подсистеме отдела кадров.

Внешний уровень состоит из нескольких различных внешних представлений базы данных. Каждый пользователь имеет дело с представлением реального мира, выраженным в наиболее удобной для него форме.

Внешний уровень — это индивидуальный уровень пользователя, который может быть прикладным программистом или конечным пользователем с любым уровнем профессиональной подготовки. У каждого пользователя есть свой язык общения.

Для прикладного программиста это либо один из распространенных языков программирования (например, семейство C или Java), либо специальный язык (язык 4-го поколения).

Для конечного пользователя — специальный язык запросов, или язык специального назначения, который может быть основан на использовании форм и меню, разработан специально с учетом требований пользователя и интерактивно поддерживаться некоторым оперативным приложением. Внешнее представление для конечного пользователя содержит только те сущности, атрибуты и связи реального мира, которые ему интересны. Другие сущности, атрибуты или связи также могут быть представлены в базе данных, но пользователь может даже не подозревать об их существовании. Помимо этого, различные представления могут по-разному отображать одни и те же данные.

Например, один пользователь может просматривать даты в формате «день, месяц, год», а другой — в формате «год, месяц, день». Некоторые представления могут включать производные или вычисляемые данные, которые не хранятся в базе данных, а создаются по мере надобности.

Промежуточным уровнем в трехуровневой архитектуре является *концептуальный уровень*. Это центральное управляющее звено, здесь БД представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной БД. Фактически концептуальный уровень отражает обобщенную модель предметной области (объектов реального мира), для которой создавалась база данных.

Этот уровень содержит логическую структуру всей базы данных (с точки зрения администратора БД). Фактически это полное представление требований к данным со стороны организации, которое не зависит от любых соображений относительно способа их хранения. На концептуальном уровне представлены следующие компоненты:

все сущности, их атрибуты и связи;

ограничения, накладываемые на данные;

семантическая информация о данных (информация, содержащаяся в высказывании и переводимая через значения единиц языка);

информация о мерах обеспечения безопасности и поддержки целостности данных.

Концептуальный уровень поддерживает каждое внешнее представление, т. е. любые доступные пользователю данные должны содержаться (или могут быть вычислены) на этом уровне. Однако он не содержит никаких сведений о методах хранения данных. Например, описание сущности должно содержать сведения о типах данных атрибутов (целочисленный, действительный или символьный) и их длине (количестве значащих цифр или максимальном количестве символов), но не должно включать сведений об организации хранения данных, например об объеме занятого пространства в байтах. Как любая другая, концептуальная модель отражает только существенные (с точки зрения обработки) особенности объектов реального мира.

Внутренний уровень. Этот уровень описывает, как информация хранится, физическую реализацию базы данных. Этот уровень предназначен для достижения оптимальной производительности и обеспечения экономного использования дискового пространства, он содержит описание структур данных и организации отдельных файлов, используемых для хранения данных на запоминающих устройствах. На этом уровне осуществляется взаимодействие СУБД с методами доступа операционной системы (вспомогательными функциями хранения и извлечения записей данных) с целью размещения информации на запоминающих устройствах, создания индексов, извлечения данных и т. д. На внутреннем уровне хранится:

информация о распределении дискового пространства для хранения данных и индексов;

описание подробностей сохранения записей (с указанием реальных размеров сохраняемых элементов данных);

сведения о размещении записей;

сведения о сжатии данных и выбранных методах их шифрования.

Ниже внутреннего уровня находится физический уровень (physical level), который контролируется операционной системой, но под управлением СУБД. Однако функции СУБД и операционной системы на физическом уровне не вполне четко разделены и могут варьироваться от системы к системе. В одних СУБД используются многие предусмотренные в данной операционной системе методы доступа, тогда как в других применяются только самые основные и реализована собственная файловая организация. Физический уровень доступа к данным ниже СУБД состоит только из известных операционной системе элементов (например, указателей на то, как реализовано последовательное распределение и хранятся ли поля внутренних записей на диске в виде непрерывной последовательности байтов).

Физический уровень — собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации.

Эта архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимость при работе с данными. Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных. Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной базой данных. Это именно то, чего не хватало при использовании файловых систем.

Выделение концептуального уровня позволило разработать аппарат централизованного управления базой данных.

В данном случае наиболее важным моментом в этих и последующих разработках является идентификация трех уровней абстракции. Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления. Такое разделение желательно выполнять по следующим причинам:

1. Каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое собственное представление о них.

2. Каждый пользователь должен иметь возможность изменять свое представление о данных, причем это изменение не должно оказывать влияния на других пользователей.

3. Пользователи не должны непосредственно иметь дело с подробностями физического хранения данных в базе.

4. Администратор базы данных должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательские представления.

5. Внутренняя структура базы данных не должна зависеть от таких изменений физических аспектов хранения информации, как переход на новое устройство хранения.

2.2. Схемы баз данных

Общее описание базы данных называется *схемой базы данных*. Существуют три различных типа схем базы данных, которые определяются в соответствии с уровнями абстракции трехуровневой архитектуры. На самом высоком уровне имеется несколько внешних схем или подсхем, которые соответствуют разным представлениям данных. На концептуальном уровне описание базы данных называют концептуальной схемой, а на самом низком уровне абстракции — внутренней схемой. Концептуальная схема описывает все сущности, атрибуты и связи между ними, с указанием необходимых ограничений поддержки целостности данных. На нижнем уровне находится внутренняя схема, которая является полным описанием внутренней модели данных. Она содержит определения хранимых записей и методов представления, описания полей данных, сведения об индексах и выбранных схемах хеширования. Для каждой базы данных существует только одна концептуальная и одна внутренняя схема.

Важно различать описание базы данных и саму базу данных. *Описанием базы данных* является схема базы данных. Схема создается в процессе проектирования базы данных, причем предполагается, что она изменяется достаточно редко. Однако содержащаяся в базе данных информация может меняться часто — например, всякий раз при вставке сведений о новом сотруднике или новом объекте сдаваемой в аренду недвижимости. Совокупность информации, хранящейся в базе данных в любой определенный момент времени, называется *состоянием базы данных*. Следовательно, одной и той же схеме базы данных может соответствовать множество ее различных состояний. Схема базы данных иногда именуется содержанием базы данных, а ее состояние — детализацией.

На каждом этапе своего существования с банком данных связаны разные категории пользователей.

2.3. Конечные пользователи

Основная категория пользователей, в интересах которых и создается банк данных, — *конечные пользователи*. В зависимости от особенностей создаваемого банка данных круг его конечных пользователей может существенно различаться. Это могут быть случайные пользователи, обращающиеся к БД время от времени за получением некоторой информации, а могут быть регулярные пользователи. В качестве *случайных пользователей* могут рассматриваться, например, возможные клиенты фирмы, просматривающие каталог продукции или услуг с обобщенным или подробным описанием того и другого. *Регулярными пользователями* могут быть сотрудники, работающие со специально разработанными для них программами, которые обеспечивают автоматизацию их деятельности при выполнении своих должностных обязанностей. Например, менеджер, планирующий работу сервисного отдела

компьютерной фирмы, имеет в своем распоряжении программу, которая помогает ему планировать и распределять текущие заказы, контролировать ход их выполнения, заказывать на складе необходимые комплектующие. Главный принцип состоит в том, что от конечных пользователей не должно требоваться каких-либо специальных знаний в области вычислительной техники и языковых средств.

Администраторы базы данных. Это группа пользователей, которая на начальной стадии разработки базы данных отвечает за его оптимальную организацию с точки зрения одновременной работы множества конечных пользователей, на стадии эксплуатации отвечает за корректность работы данной базы в многопользовательском режиме. На стадии развития и реорганизации эта группа пользователей отвечает за возможность корректной реорганизации банка данных без изменения или прекращения его текущей эксплуатации.

Разработчики и администраторы приложений. Это группа пользователей, которая функционирует во время проектирования, создания и реорганизации базы данных. Администраторы приложений координируют работу разработчиков при разработке конкретного приложения или группы приложений, объединенных в функциональную подсистему. Разработчики конкретных приложений работают с той частью информации из базы данных, которая требуется для конкретного приложения.

Не в каждой базе данных могут быть выделены все типы пользователей. Как известно, при разработке информационных систем с использованием настольных СУБД администратор базы данных, администратор приложений и разработчик часто являлись одним лицом. Однако при построении современных сложных корпоративных баз данных, которые используются для автоматизации всех или большей части бизнес-процессов в крупной фирме или корпорации, могут существовать и группы администраторов приложений, и отделы разработчиков.

3. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

3.1. Общие сведения

В прикладной программе, использующей при решении задачи один или несколько отдельных файлов, за сохранность и достоверность данных отвечал программист, работающий с этой задачей. Использование базы данных предполагает работу с ней нескольких прикладных программ, решающих задачи разных пользователей.

Естественно, что за сохранность и достоверность интегрированных данных программист, решающий одну из прикладных задач, отвечать уже не может. Кроме того, расширение круга решаемых с использованием базы данных задач может приводить к появлению новых типов записей и отношений между ними. Такое изменение структуры базы данных не должно вести к изменению множества ранее разработанных и успешно функционирующих прикладных программных систем. С другой стороны, возможное изменение любой из прикладных программ, в свою очередь, не должно приводить к изменению структуры данных. Все вышесказанное обусловливает необходимость отделения данных от прикладных программ.

Роль интерфейса между прикладными программами и базой данных, обеспечивающего их независимость, играет программный комплекс — система управления базами данных (СУБД).

СУБД — программный комплекс поддержки интегрированной совокупности данных, предназначенный для создания, ведения и использования базы данных многими пользователями (прикладными программами) (рис. 3.1).

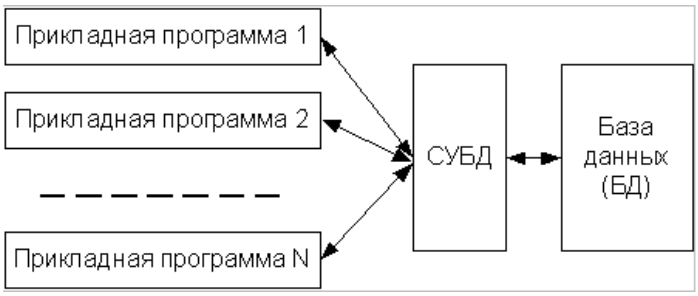

Рис. 3.1

Из этого определения следует, что СУБД можно назвать такую программу, с помощью которой можно создать, ввести или редактировать базу данных. А также имеется возможность использование ее одним или несколькими пользователями.

3.2. Состав СУБД и работа базы данных

СУБД представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнении их данными получается та или иная база данных, в которую входит система программно-технических, организационных и человеческих составляющих (рис. 3.2).

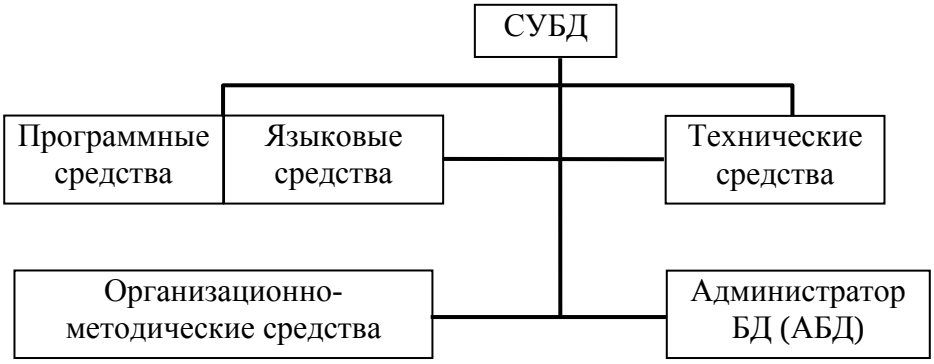

Рис. 3.2

Программные средства включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование БД, трансляторы.

Базовыми внутренними языками программирования являются языки четвертого поколения: C, C++, Pascal, Object Pascal. Следует отметить, что исторически для системы управления базой данных сложились три языка:

1) язык описания данных (ЯОД), называемый также языком описания схем, — для построения структуры (шапки) таблиц БД;

2) язык манипулирования данными (ЯМД) — для заполнения БД данными и операций обновления (запись, удаление, модификация);

3) язык запросов — язык поиска наборов величин в файле в соответствии с заданной совокупностью критериев поиска и выдачи затребованных данных без изменения содержимого файлов и БД (язык преобразования критериев в систему команд).

В настоящее время функции всех трех языков выполняет язык SQL, относящийся к классу языков, базирующихся на исчислении кортежей (кортеж чаще всего является единицей информации), языки СУБД FoxPro, Visual Basic for Application (СУБД Access) и т. д. Вместе с тем сохранились и языки запросов, например язык запросов по образцу Query By Example (QBE) класса исчисления доменов. Необходимо отметить, что эти языки в качестве информационной единицы БД используют отдельную запись. С помощью язы-

ков БД создаются приложения, базы данных и интерфейс пользователя, включающий экранные формы, меню, отчеты. При создании БД на базе СУБД FoxPro эти элементы (объекты) фиксируются в отдельных файлах, которые, в свою очередь, сосредоточиваются в одном файле, называемом проектом. После отработки БД проект преобразуется в приложение. В СУБД Access все созданные объекты размещаются в одном файле.

Для работы с созданной БД пользователю или администратору БД следует иметь перечень файлов-таблиц с описанием состава их данных (структуры, схемы). Для этого создается специальный файл, называемый словарем данных (депозитарием). Описание БД относится к метаинформации.

В качестве технических средств могут выступать супер- или персональные компьютеры с соответствующими периферийными устройствами.

Организационно-методические средства — это совокупность инструкций, методических и регламентирующих материалов, описаний структуры и процедуры работы пользователя с СУБД и БД.

Для крупных централизованных и распределенных баз данных потребность в АБД сохранилась. В широком плане под АБД понимают системных аналитиков, проектировщиков структур данных и информационного обеспечения, проектировщиков технологии процессов обработки, системных и прикладных программистов, операторов, специалистов в предметной области и по техническому обслуживанию.

3.3. Классификация СУБД

По языкам общения СУБД делятся на открытые, замкнутые и смешанные.

Открытые системы — это системы, в которых для обращения к БД используются универсальные языки программирования. *Замкнутые системы имеют* собственные языки общения с пользователями БнД.

По числу уровней в архитектуре различают *одноуровневые*, *двухуровневые*, *трехуровневые системы*.

Под *архитектурным уровнем СУБД* понимают функциональный компонент, механизмы которого служат для поддержки некоторого уровня абстракции данных (логический и физический уровень, а также взгляд пользователя — внешний уровень).

По выполняемым функциям СУБД делятся на информационные и операционные. *Информационные СУБД* позволяют организовать хранение информации и доступ к ней. Для выполнения более сложной обработки необходимо писать специальные программы. *Операционные СУБД* выполняют достаточно сложную обработку, например, автоматически позволяют получать агрегированные показатели, не хранящиеся непосредственно в базе данных, могут изменять алгоритмы обработки и т. д.

По сфере возможного применения различают *универсальные* и *специализированные*, обычно проблемно-ориентированные СУБД.

По мощности СУБД делятся на *настольные* и *корпоративные*. Характерными чертами настольных СУБД являются сравнительно невысокие требования к техническим средствам, ориентация на конечного пользователя, низкая стоимость.

Корпоративные СУБД обеспечивают работу в распределенной среде, высокую производительность, поддержку коллективной работы при проектировании систем, имеют развитые средства администрирования и более широкие возможности поддержания целостности. Наиболее известными из корпоративных СУБД являются Oracle, Informix, Sybase, MS SQL Server, Progress и некоторые другие.

Существует разделение СУБД по поколениям:

1 поколение — основано на иерархической и сетевой модели;

2 поколение — реляционные системы;

3 поколение — СУБД должны поддерживать сложные структуры данных и более развитые средства обеспечения целостности данных, отвечать требованиям, предъявляемым к открытым системам.

3.4. Основные функции СУБД

К числу функций СУБД принято относить следующие:

1. *Непосредственное управление данными во внешней памяти*. Эта функция включает обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным в некоторых случаях (обычно для этого используются индексы). В некоторых реализациях СУБД активно используются возможности существующих файловых систем, в других — работа производится вплоть до уровня устройств внешней памяти. Однако в развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, а если использует, то как организованы файлы. В частности, СУБД поддерживает собственную систему именования объектов БД.

2. *Управление буферами оперативной памяти*. СУБД обычно работают с базами данных значительного размера, которые в большинстве случаев существенно больше доступного объема оперативной памяти. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. При этом даже если операционная система (ОС) производит общесистемную буферизацию (как в случае ОС UNIX), этого недостаточно для целей СУБД, которая располагает ОС гораздо большей информацией о полезности буферизации той или иной части БД. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

Существует отдельное направление СУБД, которое ориентировано на постоянное присутствие в оперативной памяти всей БД. Это направление основывается на предположении, что в будущем объем оперативной памяти компьютеров будет настолько велик, что позволит не беспокоиться о буферизации. Пока эти работы находятся в стадии исследований.

3. Управление транзакциями. *Транзакция — это последовательность операций над БД, рассматриваемых СУБД как единое целое.* Либо транзакция успешно выполняется, и СУБД фиксирует изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД.

Понятие транзакции необходимо для поддержания логической целостности БД, например в информационной системе с файлами СОТРУДНИКИ и ОТДЕЛЫ. Единственным способом не нарушить целостность БД при выполнении операции приема на работу нового сотрудника является объединение элементарных операций над файлами СОТРУДНИКИ и ОТДЕЛЫ в одну транзакцию. Таким образом, поддержание механизма транзакций является обязательным условием даже однопользовательских СУБД (если, конечно, такая система заслуживает названия СУБД). Но понятие транзакции гораздо более важно в многопользовательских СУБД. То свойство, что каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к БД. При соответствующем управлении параллельно выполняющимися транзакциями со стороны СУБД каждый из пользователей может в принципе ощущать себя единственным пользователем СУБД (на самом деле это несколько идеализированное представление, поскольку в некоторых случаях пользователи многопользовательских СУБД могут ощутить присутствие своих коллег).

4. *Журнализация*. Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под *надежностью хранения* понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть: аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя) или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции.

Понятно, что в любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных,

причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

Ж*урнал* — это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД. В разных СУБД изменения БД журнализуются на разных уровнях: иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), иногда — минимальной внутренней операции модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода.

Во всех случаях придерживаются стратегии упреждающей записи в журнал (так называемого протокола Write Ahead Log — WAL). Грубо говоря, эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. Известно, что если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя. Самая простая ситуация восстановления — индивидуальный откат транзакции. Строго говоря, для этого не требуется общесистемный журнал изменений БД. Достаточно для каждой транзакции поддерживать локальный журнал операций модификации БД, выполненных в этой транзакции, и производить откат транзакции путем выполнения обратных операций, следуя от конца локального журнала. В некоторых СУБД так и делают, но в большинстве систем локальные журналы не поддерживают, а индивидуальный откат транзакции выполняют по общесистемному журналу, для чего все записи от одной транзакции связывают обратным списком (от конца к началу).

5. *Поддержка языков БД*. Для работы с базами данных используются специальные языки, в целом называемые языками баз данных. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка:

1. Язык определения схемы БД (SDL — Schema Definition Language).

2. Язык манипулирования данными (DML — Data Manipulation Language).

SDL служил главным образом для определения логической структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык запросов SQL (Structured Query Language).

Язык SQL содержит специальные средства определения ограничений целостности БД. Ограничения целостности хранятся в специальных таблицах-каталогах, а обеспечение контроля целостности БД производится на языковом уровне, т. е. при компиляции операторов модификации БД компилятор SQL на основании имеющихся в БД ограничений целостности генерирует соответствующий программный код.

Специальные операторы языка SQL позволяют определять так называемые представления БД, фактически являющиеся хранимыми в БД запросами (результатом любого запроса к реляционной БД является таблица) с именованными столбцами. Для пользователя представление является такой же таблицей, как любая базовая таблица, хранимая в БД, но с помощью представлений можно ограничить или расширить видимость БД для конкретного пользователя. Поддержание представлений производится также на языковом уровне.

Наконец, авторизация доступа к объектам БД производится также на основе специального набора операторов SQL. Идея состоит в том, что для выполнения операторов SQL разные пользователи должны обладать различными полномочиями. Пользователь, создавший таблицу БД, обладает полным набором полномочий для работы с этой таблицей. В их число входит полномочие на передачу всех или части полномочий другим пользователям, включая полномочие на передачу полномочий. Полномочия пользователей описываются в специальных таблицах-каталогах, контроль полномочий поддерживается на языковом уровне.

4. БАЗА ДАННЫХ КАК ИНФОРМАЦИОННАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

4.1. Понятие «предметная область»

Основным назначением информационных систем является оперативное обеспечение пользователя информацией о внешнем мире путем реализации вопросно-ответного отношения. Вопросно-ответные отношения, получая интерпретацию во внешнем мире (мире вне информационной системы), позволяют выделить для информационной системы определенный его фрагмент — предметную область, который будет воплощен в автоматизированной информационной системе. Информация о внешнем мире представляется в информационной системе (ИС) в форме данных. Это ограничивает возможности смысловой интерпретации информации и конкретизирует семантику ее представления в ИС. Совокупность этих выделенных для ИС данных, связей между ними и операций над ними образует информационную и функциональную модели предметной области, описывающие ее состояние с определенной точностью.

Чтобы понять, как функционирует предметная область, выполняется ее функциональный анализ — определение функционирования по описанию предметной области. В основе функционального анализа лежит принцип декомпозиции действий. Синонимами понятия «действие» являются: процесс, задача, функция, работа. Результатом анализа в этом контексте является функциональная модель, которая дает представление о предметной области в терминах функций и групп данных, сопутствующих выполнению этих функций.

Функциональная модель — это модель инфологического уровня представления, в которой акцентируется функциональный аспект моделирования предметной области.

Функциональная модель в виде иерархии функций способствует пониманию поведения субъекта моделирования.

На основании функционального анализа и выделения локальных представлений предметной области различных категорий потенциальных пользователей базы данных необходимо сгруппировать данные, которые в том или ином качестве сопутствуют реализации функций. Это обстоятельство дает возможность осуществить построение информационной модели предметной области.

Информационная модель — это модель инфологического уровня представления, в которой акцентируется информационный (структурный) аспект моделирования предметной области.

Одним из распространенных средств спецификации модельных представлений этого типа является так называемая модель «сущность-связь» (Entity-Relationship Model).

Важно понимать, что информационная и функциональная модели предметной области создаются на этапе анализа требований к базе данных и не содержат предположений о технологии реализации базы данных. Они строятся независимо от выбираемой модели данных (сетевой, иерархической, реляционной, объектно-ориентированной, многомерной и т. д.), поддерживаемой СУБД, модели вычислений, программно-аппаратной платформы для базы данных. Информационная и функциональная модели предметной области являются входными данными для процесса проектирования базы данных. Поэтому проектировщик должен уметь правильно интерпретировать их в ходе решения своих проектных задач.

Любую предметную область можно рассматривать как динамическое информационное поле, охватывающее свойства объектов, их взаимосвязи, информационные потоки между ними и т. п. Изменения, происходящие в предметной области, приводят к генерации новой информации, новых информационных элементов или их изменению, что и позволяет говорить об информационном поле и, более того, динамическом информационном поле. Понятие предметной области базы данных является одним из базовых понятий информатики и не имеет точного определения. Его использование в контексте ИС предполагает существование устойчивой во времени соотнесенности между именами, понятиями и определенными реалиями внешнего мира, не зависящей от самой ИС и ее круга пользователей. Таким образом, введение в рассмотрение понятия «предметная область» базы данных ограничивает и делает обозримым пространство информационного поиска в ИС и позволяет выполнять запросы за конечное время.

Совокупность реалий внешнего мира — объектов, о которых можно задавать вопросы, образует объектное ядро предметной области, которое имеет онтологический статус (онтология — учение о бытии). Нельзя получить в ИС ответ на вопрос о том, что ей неизвестно. Термин объект является первичным, неопределяемым понятием. Синонимами термина «объект» являются «реалия, сущность, вещь». Однако термин «сущность» понимается нами несколько уже, как компонент модели предметной области, т. е. как уже выде-

ленный на концептуальном уровне объект для базы данных. Таким образом, выделяемые в предметной области объекты превращаются аналитиками (а не проектировщиками базы данных) в сущности. Сущность предметной области является результатом абстрагирования реального объекта путем выделения и фиксации набора его свойств, т. е. в нашем контексте имеет гносеологический статус (гносеология — теория познания). Хотя далее в контексте сущность нередко отождествляется с объектом.

На рис. 4.1 представлен один из подходов к классификации объектов предметной области.

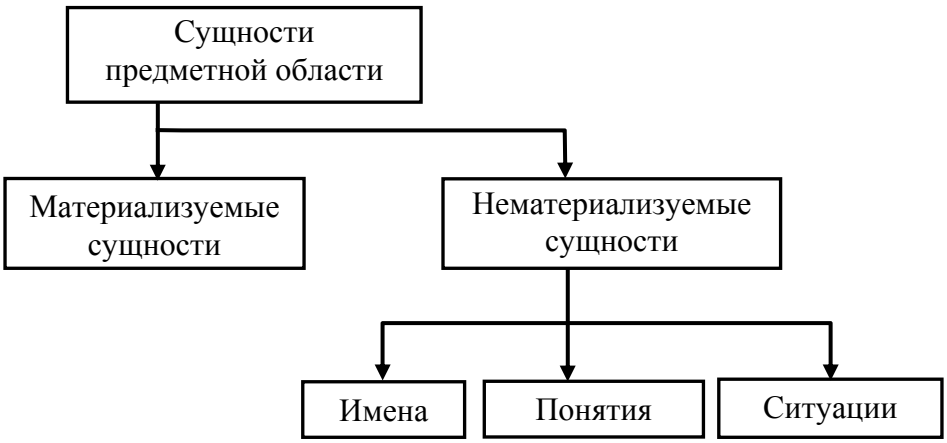

Рис. 4.1

Примерами сущностей (с точки зрения ИС) или объектов (с точки зрения внешнего мира) являются отдельный студент, группа студентов, аудитория, время занятий, слова, числа, символы. Обычно считается, что быть объектом — это значит быть дискретным и различимым. Примеры необъектов — это мир, время, смысл, хотя и такие категории могут сохраняться в базе данных.

С объектами связано две проблемы: идентификация и адекватное описание. Для идентификации используют имя. При этом предполагается, что происходит отказ от его смысла, который присущ естественному языку. Используется только указательная функция имени. Имя — это пря мой способ идентификации объекта. К косвенным способам идентификации объекта относят определение объекта через его свойства (характеристики или признаки).

Объекты взаимодействуют между собой через свои свойства, что порождает ситуации. *Ситуации — это взаимосвязи, выражающие взаимоотношения между объектами.* Ситуации в предметной области описываются посредством высказываний о предметной области с использованием исчисления высказываний и исчисления предикатов, т. е. формальной, математической логики. Например, высказывание «Программист и менеджер есть служащие компании» описывает отношение включения. Таким образом, вся информация об объектах и сущностях предметной области описывается с помощью утверждений на естественном языке.

Методы математической логики позволяют формализовать эти утверждения и представить их в виде, пригодном для анализа.

Пример. Рассмотрим высказывание «Студент Иванов А. А, родился в 1982 году». Оно выражает следующие свойства объекта «Иванов А. А.»:

в явном виде — год рождения;

неявном — принадлежность к студентам.

Первое свойство устанавливает связь между объектами «Иванов А. А.» и «Год рождения», а второе — между объектами «Иванов А. А.» и «Множество студентов». Формализация этого высказывания представляется как результат присваивания значений переменным, входящим в предикаты:

РОДИЛСЯ (Иванов А. А., 1982);

ЯВЛЯЕТСЯ СТУДЕНТОМ (Иванов А. А.).

Необходимо отметить, что в семантике естественных языков ситуация и взаимосвязь считаются почти синонимами. Ситуация содержит высказывание об объектах предметной области, которому можно приписать некоторую оценку истинности и представить в виде предиката после введения переменных. Таким образом, совокупность высказываний о предметной области можно трактовать как определение информационного пространства для базы данных.

На рис. 4.2 представлен один из подходов к классификации ситуаций в рамках предметной области.

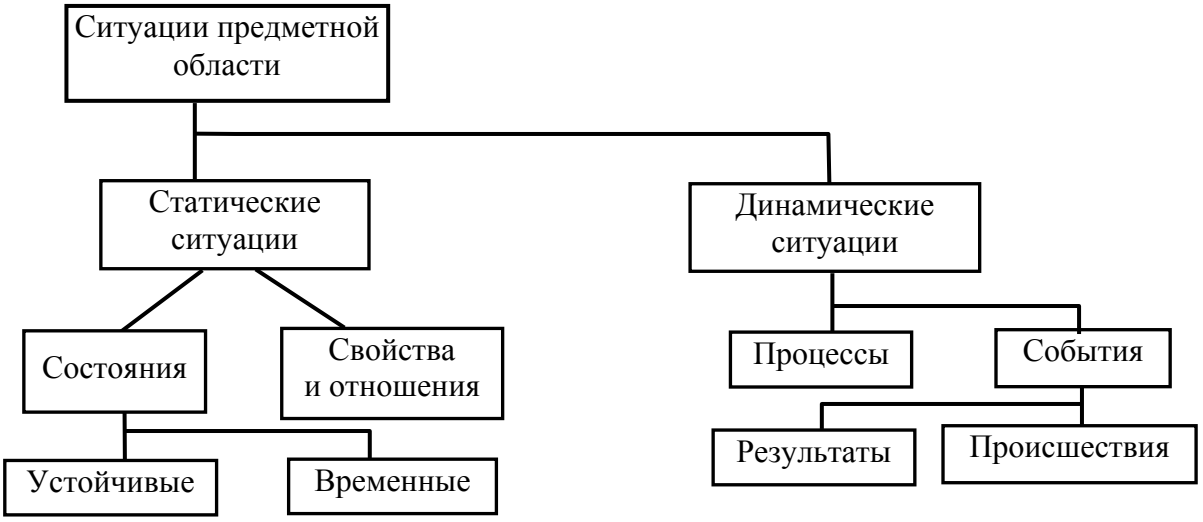

Рис. 4.2

Различают статические и динамические ситуации. Примерами статических ситуаций являются такие ситуации, как «иметь цвет», «иметь возраст». Примерами динамических ситуаций являются такие ситуации, как «включить утюг», «выпечь хлеб».

Необходимо обратить внимание на то, что ситуация также может представлять собой объект (см. рис. 4.2) и обладать свойствами. С другой стороны, приведенная классификация рассматривает свойства как специальный случай ситуаций. Подобная коллизия порождает неоднозначность при

моделировании предметной области базы данных. Приведенная классификация вводит в предметную область два важных аспекта — пространство и время, причем время как момент и как интервал. Предметная область существует в пространстве и во времени, т. е. ей присущи, как и реальному миру, временные и пространственные отношения и связи. Следует отличать реальное время внешнего мира и его отражение в базе данных и источниках информации.

В базе данных взаимосвязи, зависящие от времени, фиксируются только после их регистрации в базе данных. Таким образом, предметная область в каждый конкретный момент времени представляет собой выделенную совокупность определенных объектов и ситуаций, называемую состоянием предметной области (или снимком).

Понятие предметной области было введено в начале 80-х гг. прошлого века, когда учеными в области ИС была осознана необходимость использовать семантические модели для представления информации в компьютерных системах.

Предметная область — это целенаправленная первичная трансформация картины внешнего мира в некоторую умозрительную картину, определенная часть которой фиксируется в ИС в качестве алгоритмической модели фрагмента действительности.

Так же как требования к компьютерной системе формируются средствами естественного языка, так и информация в компьютерных системах представляется средствами особого языка с определенной семантикой. Такой подход впервые был представлен П. Ченом в 1976 г.

4.2. Понятие «модель данных»

Центральным понятием в области баз данных является понятие модели. Не существует однозначного определения этого термина, у разных авторов эта абстракция определяется с некоторыми различиями, но тем не менее можно выделить нечто общее в этих определениях.

Модель данных — это некоторая абстракция, которая, будучи приложима к конкретным данным, позволяет пользователям и разработчикам трактовать их уже как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними. Иными словами: модель данных — это совокупность структур данных и операций их обработки.

Так как СУБД имеет трехуровневую архитектуру, то понятие модели данных связано с каждым уровнем.

Физическая модель данных связана с организацией внешней памяти и структур хранения, используемых в данной операционной среде.

На концептуальном уровне модели данных наиболее важны для разработчиков БД, так как именно ими определяется тип СУБД.

Для внешнего уровня отдельных моделей данных нет, они лишь являются подсхемами концептуальных моделей данных.

4.3. Классификация моделей данных

В соответствии с рассмотренной ранее трехуровневой архитектурой мы сталкиваемся с понятием модели данных по отношению к каждому уровню (рис. 4.3)

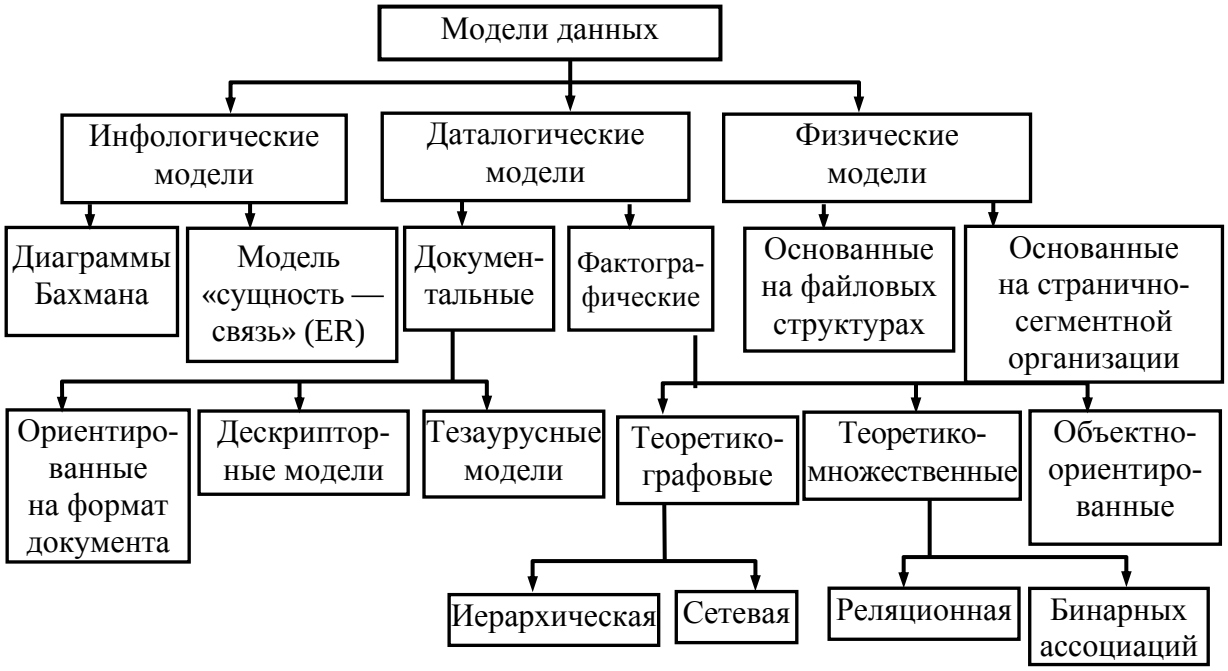

Рис. 4.3

В настоящий момент в качестве физических моделей используются различные методы размещения данных, основанные на файловых структурах: организация файлов прямого и последовательного доступа, индексных и инвертированных файлов, файлов, использующих различные методы хеширования, а также взаимосвязанных. Кроме того, современные СУБД широко используют страничную организацию данных. Физические модели данных, основанные на страничной организации, являются наиболее перспективными.

Наибольший интерес вызывают модели данных, используемые на концептуальном уровне. По отношению к ним внешние модели называются подсхемами и используют те же абстрактные категории, что и концептуальные модели данных.

Кроме трех рассмотренных уровней абстракции при проектировании БД существует еще один уровень, предшествующий им. Модель этого уровня должна выражать информацию о предметной области в виде, независимом от используемой СУБД. Такие модели называются инфологическими, или семантическими.

Инфологические модели данных отражают в естественной и удобной для разработчиков и других пользователей форме информационно-логический уровень абстрагирования, связанный с фиксацией и описанием объектов предметной области, их свойств и их взаимосвязей. Инфологические модели данных используются на ранних стадиях проектирования для

описания структур данных в процессе разработки приложения, а даталогические модели уже поддерживаются конкретной СУБД. В даталогическом аспекте рассматриваются вопросы представления данных в памяти информационной системы.

Документальные модели данных соответствуют представлению о слабоструктурированной информации, ориентированной в основном на свободные форматы документов и текстов на естественном языке, фактографические — хранят совокупность фактов интегрированных, иногда из различных документов.

Модели, *ориентированные на формат документов*, связаны прежде всего со стандартным общим языком разметки — SGML (Standart Generalised Markup Language), который был утвержден ISO в качестве стандарта еще в 80-х годах XX в. Этот язык предназначен для создания других языков разметки, он определяет допустимый набор тегов (ссылок), их атрибуты и внутреннюю структуру документа. Контроль над правильностью использования тегов осуществляется при помощи специального набора правил, называемых DTD-описаниями, которые используются программой клиента при разборе документа. Для каждого класса документов определяется свой набор правил, описывающих грамматику соответствующего языка разметки. С помощью SGML можно описывать структурированные данные, организовывать информацию, содержащуюся в документах, представлять ее в некотором стандартизованном формате. Но ввиду некоторой своей сложности SGML использовался в основном для описания синтаксиса других языков (наиболее известным из которых является HTML), и немногие приложения работали с SGML-документами напрямую.

Гораздо более простой и удобный язык HTML позволяет определять оформление элементов документа и имеет некий ограниченный набор инструкций — тегов, при помощи которых осуществляется процесс разметки. Инструкции HTML в первую очередь предназначены для управления процессом вывода содержимого документа на экране программы-клиента, тем самым определяя способ представления документа, но не его структуру. В качестве элемента гипертекстовой базы данных, описываемой HTML, используется текстовый файл, который может легко передаваться по сети с использованием протокола HTTP. Эта особенность, а также тот факт, что HTML является открытым стандартом и огромное количество пользователей имеет возможность применять возможности этого языка для оформления своих документов, безусловно повлияли на рост популярности HTML и сделали его сегодня главным механизмом представления информации в Интернете.

Однако HTML сегодня уже не удовлетворяет в полной мере требованиям, предъявляемым современными разработчиками к языкам подобного рода. И ему на смену был предложен новый язык гипертекстовой разметки, мощный, гибкий и удобный язык XML.

XML (Extensible Markup Language) — это язык разметки, описывающий целый класс объектов данных, называемых XML-документами. Он используется в качестве средства для описания грамматики других языков и кон-

троля над правильностью составления документов. То есть сам по себе XML не содержит никаких тегов, предназначенных для разметки, он просто определяет порядок их создания.

Тезаурусные модели основаны на принципе организации словарей, содержат определенные языковые конструкции и принципы их взаимодействия в заданной грамматике. Эти модели эффективно используются в системах-переводчиках, особенно многоязыковых переводчиках. Принцип хранения информации в этих системах и подчиняется тезаурусным моделям.

Дескрипторные модели — самые простые из документальных моделей, они широко использовались на ранних стадиях использования документальных баз данных. В этих моделях каждому документу соответствовал дескриптор — описатель. Этот дескриптор имел жесткую структуру и описывал документ в соответствии с теми характеристиками, которые требуются для работы с документами в разрабатываемой документальной БД. Например, для БД, содержащей описание патентов, дескриптор содержал название области, к которой относился патент, номер патента, дату выдачи патента и еще ряд ключевых параметров, которые заполнялись для каждого патента. Обработка информации в таких базах данных велась исключительно по дескрипторам, то есть по тем параметрам, которые характеризовали патент, а не по самому тексту патента.

Теоретико-графовые модели отражают совокупность объектов реального мира в виде графа взаимосвязанных информационных объектов. Математической основой таких моделей является теория графов.

Объектно-ориентированная модель перекликается с семантическими моделями данных. Принципы похожи на принципы объектно-ориентированных языков программирования. Структура таких моделей графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются типом.

Иерархическая модель данных является наиболее простой среди всех даталогических моделей. Исторически она появилась первой: именно эту модель поддерживает первая из зарегистрированных промышленных СУБД IMS фирмы IBM. Объекты иерархической модели данных связанны иерархическими отношениями и образуют ориентированный граф. Основные понятия иерархических структур: уровень, узел (совокупность свойств данных, описывающих объект), связь.

В *сетевой модели* данных при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

Реляционная модель данных обеспечивает ряд важных возможностей, которые делают управление БД и их использование относительно легким, устойчивым по отношению к ошибкам и предсказуемым. Она описывает данные с их естественной структурой, не добавляя каких-либо дополнительных структур, необходимых для машинного представления или для целей реализации; обеспечивает математическую основу для интерпретации выводимости, избыточности и непротиворечивости отношений; обеспечивает независимость данных от их физического представления, связей между данными и соображений реализации, связанных с эффективностью и подобными заботами. В реляционной модели данных данные представлены только в виде таблиц.

5. БАЗОВЫЕ ТИПЫ МОДЕЛЕЙ ДАННЫХ

5.1. Выбор модели

В модели данных описывается некоторый набор родовых понятий и признаков, которыми должны обладать все конкретные СУБД и управляемые ими базы данных, если они основываются на этой модели. Наличие модели данных позволяет сравнивать конкретные реализации, используя один общий язык. Модель данных определяет совокупность правил порождения структур данных, операций над ними, а также ограничений целостности, определяющих допустимые связи и значения данных, последовательность их изменения.

Можно по-разному характеризовать понятие модели данных. С одной стороны, модель данных — это способ структурирования данных, которые рассматриваются как некоторая абстракция в отрыве от предметной области. С другой стороны, модель данных — это инструмент представления концептуальной модели предметной области и динамики ее изменения в виде базы данных.

Модель является представлением реального мира объектов и событий, а также существующих между ними связей. Это некоторая абстракция, в которой акцент делается на самых важных и неотъемлемых аспектах деятельности организации, а все второстепенные свойства игнорируются. Таким образом, можно сказать, что модель данных представляет саму организацию. Модель должна отражать основные концепции, представленные в таком виде, который позволит проектировщикам и пользователям базы данных обмениваться конкретными и недвусмысленными мнениями о роли тех или иных данных в организации. Модель данных можно рассматривать как сочетание трех указанных ниже компонентов:

1. Структурная часть, т. е. набор правил, по которым может быть построена база данных.

2. Управляющая часть, определяющая типы допустимых операций с данными (сюда относятся операции обновления и извлечения данных, а также операции изменения структуры базы данных).

3. Набор (необязательный) ограничений поддержки целостности данных, гарантирующих корректность используемых данных.

Цель построения модели данных заключается в представлении данных в понятном виде. Если такое представление возможно, то модель данных можно легко применить при проектировании базы данных.

5.2. Категории моделей данных

Модели данных подразделяются на три категории:

1) объектные (object-based) модели данных;

2) модели данных на основе записей (record-based);

3) физические модели данных.

Первые две используются для описания данных на концептуальном и внешнем уровнях, а последняя — на внутреннем уровне.

5.2.1. Объектные модели данных

При создании объектных моделей данных используются следующие понятия:

Сущность — это отдельный элемент деятельности организации (сотрудник или клиент, место или вещь, понятие или событие), который должен быть представлен в базе данных.

Атрибут — это свойство, которое описывает некоторый аспект объекта и значение которого следует зафиксировать.

Связь — это ассоциативное отношение между сущностями.

Наиболее общими типами объектных моделей данных являются:

1. Модель типа «сущность — связь», или ER-модель (Entity-Relationship model). В настоящее время ER-модель стала одним из основных методов концептуального проектирования баз данных.

2. Семантическая модель.

3. Функциональная модель.

4. Объектно-ориентированная модель.

Объектно-ориентированная модель расширяет определение сущности с целью включения в него не только атрибутов, которые описывают состояние объекта, но и действий, которые с ним связаны, т. е. его поведение. В таком случае говорят, что объект инкапсулирует состояние и поведение.

5.2.2. Модели данных на основе записей

В модели на основе записей база данных состоит из нескольких записей фиксированного формата, которые могут иметь разные типы. Каждый тип записи определяет фиксированное количество полей, каждое из которых имеет фиксированную длину.

Существуют три основных типа логических моделей данных на основе записей:

иерархическая модель данных (Hierarchical Data model),

сетевая модель данных (Network Data model),

реляционная модель данных (Relational Data model).

Иерархическая модель данных (ИМД) представляет сбой совокупность элементов, связанных между собой по определенным правилам. Эта модель строится по принципу иерархии типов объектов, то есть один тип объекта является главным, а остальные, находящиеся на низших уровнях иерархии, — подчиненными. Между главным и подчиненными объектами устанавливается взаимосвязь «один ко многим». Иными словами, для данного главного типа объекта существует несколько подчиненных типов объектов. В то же время для каждого экземпляра главного объекта может быть несколько экземпляров подчиненных типов объектов.

Основные понятия иерархической структуры:

Узел (элемент) — совокупность атрибутов данных, описывающих некоторый объект (на схеме это вершины графа). Каждый узел, находящийся на более низком уровне, связан только с одним узлом, находящимся на более высоком уровне. Узел может иметь только одного родителя.

Иерархическое дерево имеет только одну вершину (корень), неподчиненную никакой другой вершине и находящуюся на самом верхнем (первом) уровне.

Таким образом, *иерархическая модель представляет собой древовидный граф с записями в виде узлов (сегментами) и множествами в виде ребер (связь).*

Примером простого иерархического представления может служить административная структура высшего учебного заведения: институт — отделение — факультет — студенческая группа (рис. 5.1).

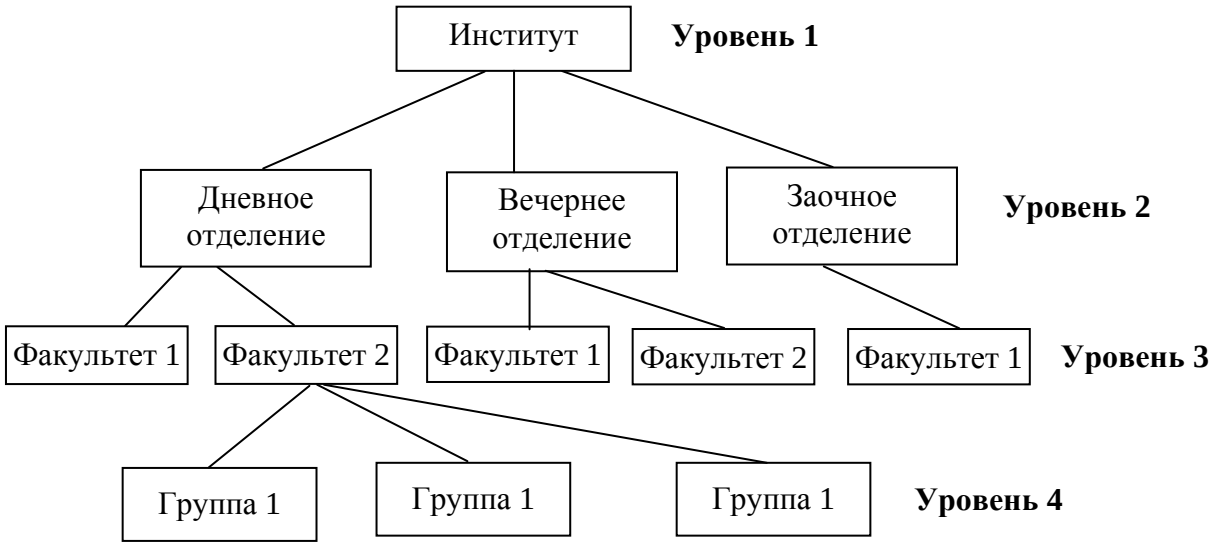

Рис. 5.1

На иерархической модели данных основано сравнительно ограниченное количество СУБД, в числе которых можно назвать зарубежные системы IMS, PC/Focus, Data Edge, а также отечественные системы Ока, ИНЭС и МИРИС.

Основанные на записях (логические) модели данных используются для определения общей структуры базы данных и высокоуровневого описания

ее реализации. Основной недостаток таких моделей заключается в том, что они не дают адекватных средств для явного указания ограничений, накладываемых на данные. В то же время в объектных моделях данных отсутствуют средства указания их логической структуры, но за счет предоставления пользователю возможности указать ограничения для данных они позволяют в большей мере представить семантическую суть хранимой информации.

При использовании сетевой или иерархической моделей от пользователя требуется знание физической организации базы данных, к которой он должен осуществлять доступ. Сетевые и иерархические системы используют навигационный подход (т. е. они указывают, как их следует извлечь).

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель удобна для работы с иерархически упорядоченной информацией. Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя. Иерархическая упорядоченность усложняет операции включения и удаления. Доступ к любой вершине возможен только через корневую, что увеличивает время доступа.

В сетевой модели данных (СМД) элементарные данные и отношения между ними представляются в виде ориентированной сети (вершины — данные, дуги — отношения). БД, описываемая сетевой моделью, состоит из нескольких областей. *Область — это поименованная часть базы данных, в которой могут содержаться экземпляры записей и наборов или части наборов.* Каждая область может обладать собственными уникальными физическими характеристиками. Области могут обрабатываться как по отдельности, так и вместе с другими областями.

Набор — это поименованная совокупность связанных записей. Набор может размещаться в одной или нескольких областях.

Сетевая БД состоит из набора записей и набора соответствующих связей. Сетевую модель можно представить как граф с записями в виде узлов графа и наборами в виде его ребер.

Для описания схемы сетевой БД используется две группы типов: «запись» и «связь». Тип «связь» определяется для двух типов «запись»: предка и потомка. Переменные типа «связь» являются экземплярами связей.

На форматирование связи особых ограничений не накладывается. Если в иерархических структурах запись-потомок могла иметь только одну запись-предка, то в сетевой модели данных запись-потомок может иметь произвольное число записей-предков (свободных родителей).

Физическое размещение данных в базах сетевого типа может быть организовано практически теми же методами, что и в иерархических базах. Пример простой сетевой структуры показан на рис. 5.2.

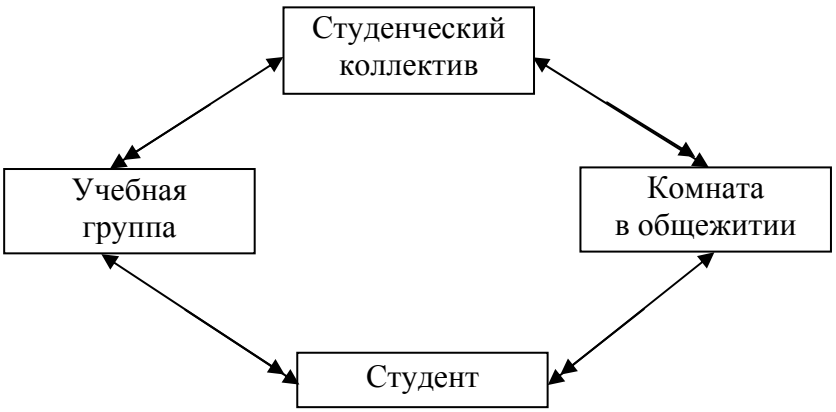

Рис. 5.2

К числу важнейших операций манипулирования данными баз сетевого типа можно отнести следующие:

поиск записи в БД;

переход от предка к первому потомку;

переход от потомка к предку;

создание новой записи;

удаление текущей записи;

обновление текущей записи;

включение записи в связь;

исключение записи из связи;

изменение связей и т. д.

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая представляет большие возможности в смысле допустимости образования произвольных связей.

Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Наиболее известными сетевыми СУБД являются IDMS, СЕТЬ, СЕТОР и КОМПАС. Системы на основе сетевой модели не получили широкого распространения на практике.

Реляционная модель данных (РМД) была предложена известным исследователем в области баз данных Е. Ф. Коддом в 1969 г., когда он был сотрудником фирмы IBM. Впервые основные концепции этой модели были опубликованы в 1970 г. Строгое изложение теории реляционных баз данных (реляционной модели данных) в современном понимании было предложено Кристофером Дейтом. Согласно Дейту, реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода:

структурной части;

манипуляционной части;

целостной части.

В структурной части модели фиксируется, что единственной структурой данных, используемой в реляционных БД, являются нормализованные *n*-арные отношения.

Вторая часть реляционной модели, манипуляционная часть, утверждает, что доступ к реляционным данным осуществляется при помощи реляционной алгебры или эквивалентного ему реляционного исчисления.

В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД:

1. Требование целостности сущностей, которое состоит в том, что любой кортеж любого отношения должен быть отличим от любого другого кортежа этого отношения, т. е. другими словами, любое отношение должно обладать первичным ключом.

2. Требование целостности по ссылкам (внешним ключам). Оно является более трудным. Сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений.

В течение долгого времени реляционный подход рассматривался как удобный формальный аппарат анализа баз данных, не имеющий практических перспектив, так как его реализация требовала слишком больших машинных ресурсов. Только с появлением персональных ЭВМ реляционные и близкие к ним системы стали распространяться, практически не оставив места другим моделям.

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

каждый элемент таблицы — один элемент данных, повторяющиеся группы отсутствуют;

все столбцы в таблице однородны, т. е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т. д.) и длину;

каждый столбец имеет уникальное имя;

одинаковые строки в таблице отсутствуют;

порядок следования строк и столбцов может быть произвольным.

Таблица такого рода называется отношением. База данных, построенная с помощью отношений, называется реляционной базой данных. Отношения представлены в виде таблиц.

Предложив реляционную модель данных, Э. Ф. Кодд создал и инструмент для удобной работы с отношениями — реляционную алгебру. Каждая операция этой алгебры использует одну или несколько таблиц (отношений) в качестве ее операндов и продуцирует в результате новую таблицу, т. е. позволяет разрезать или склеивать таблицы.

Основное отличие реляционной модели состоит в том, что иерархические и сетевые модели данных имеют связь по структуре, а реляционные — по значению.

Проектирование баз данных традиционно считалось очень трудной задачей. Реляционная технология значительно упрощает эту задачу. Разделением логического и физического уровней системы она упрощает процесс отображения уровня реального мира в структуру, которую система может прямо поддерживать. Поскольку реляционная структура сама по себе концептуально проста, она позволяет реализовывать небольшие и/или простые (и поэтому легкие для создания) базы данных, такие как персональные, сама возможность реализации которых никогда даже бы не рассматривалась в старых более сложных системах.

Реляционная модель данных особенно удобна для использования в базах данных распределенной архитектуры — она позволяет получать доступ к любым информационным элементам, хранящимся в узлах сети ЭВМ. Необходимо обратить особое внимание на высокоуровневый аспект реляционного подхода, который состоит во множественной обработке записей. Благодаря этому значительно возрастает потенциал реляционного подхода, который не может быть достигнут.

Основными достоинствами РМД является обеспечение высокой наглядности представления информации и повышенной эффективности ее обработки.

К недостаткам можно отнести трудоемкий процесс обеспечения целостности и непротиворечивости данных, которые хранятся в базе данных.

Объектно-ориентированная модель данных. Разработка систем объектно-ориентированных баз данных (так называемые технологии баз данных пятого поколения) началась в середине 80-х гг. XX в. в связи с необходимостью удовлетворения требований приложений, отличных от тех приложений обработки данных, которые характерны для систем реляционных баз данных (технология баз данных четвертого поколения). Попытки использования технологий реляционных баз данных в таких сложных приложениях, как автоматизированное проектирование (Computer Aided Design, CAD); автоматизированное производство (Computer Aided Manufacturing, CAM); технология программирования; системы, основанные на знаниях, и мультимедийные системы обнажили ограничения систем реляционных баз данных (РБД). В условиях, когда появилось новое поколение приложений баз данных, возникли потребности, которые лучше удовлетворялись применением объектно-ориентированных баз данных (ООБД).

Объектно-ориентированная база данных — база данных, в которой данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями. Результатом совмещения возможностей (особенностей) баз данных и возможностей объектно-ориентированных языков программирования являются Объектно-ориентированные системы управления базами данных (ООСУБД). ООСУБД позволяет работать с объектами баз данных так же, как с объектами в про-

граммировании на объектно-ориентированных языках программирования (ООЯП). ООСУБД расширяет языки программирования, вводя долговременные данные, управление параллелизмом, восстановление данных, ассоциированные запросы и другие возможности.

Некоторые объектно-ориентированные базы данных разработаны для плотного взаимодействия с такими объектно-ориентированными языками программирования как Python, Java, C#, Visual Basic, NET, C++, Objective-C и Smalltalk; другие имеют свои собственные языки программирования. ООСУБД использует точно такую же модель, что и объектно-ориентированные языки программирования.

Причиной появления систем объектно-ориентированных баз данных была потребность в более адекватном представлении и моделировании сущностей реального мира, поскольку ООБД обеспечивают гораздо более развитую модель данных, нежели традиционные — реляционные базы данных. Парадигма ООБД основывается на ряде базовых понятий: объект, идентифицируемость, класс, наследование, перегрузка, отложенное связывание.

В объектно-ориентированной модели данных любая сущность реального мира представляется всего одним понятием — объектом. С объектом ассоциируется состояние и поведение. Состояние объекта определяется значениями его свойств (атрибутов), которыми могут являться примитивные значения (строки или целые числа) и непримитивные объекты. Непримитивный объект, в свою очередь, состоит из набора свойств. Следовательно, объекты можно рекурсивно определять в терминах других объектов. Поведение объекта определяется с помощью методов, которые оперируют над состоянием объекта.

У каждого объекта имеется определяемый системой уникальный идентификатор. Объекты, обладающие одними и теми же свойствами и поведением, группируются в классы. Объект может быть экземпляром только одного класса или нескольких классов.

Классы организуются в иерархии классов. Подкласс наследует свойства и методы суперкласса; кроме того, подклассы могут обладать индивидуальными свойствами и методами. В некоторых системах (например, ORION) у класса можется более одного суперкласса (множественное наследование), тогда как в других системах число суперклассов ограничено одним (одиночное наследование).

В большинстве моделей допускается перегрузка унаследованных свойств и методов. Перегрузка состоит в замене домена свойств новым доменом или в замене одной реализации метода другой его реализацией.

Достоинством объектно-ориентированные баз данных является возможность представлять сложные объекты более непосредственным образом, нежели реляционные системы. Системы ООБД позволяют пользователям определять абстракции; облегчают проектирование некоторых связей; устраняют потребность в определяемых пользователями ключах; поддерживают новый набор предикатов сравнения; в некоторых случаях устраняют потребность

в соединениях и обеспечивают более высокую производительность, чем системы, основанные на реляционной модели; обеспечивают поддержку версий и длительных транзакций. Наконец, разработана объектная алгебра — хотя, возможно, пока и не столь детально, как реляционная алгебра.

Ожидалось, что объектно-ориентированные методы позволят технологии баз данных сделать своего рода квантовый переход. Однако, несмотря на указанные выше достижения, ООБД так и не смогли оказать значительного влияния на положение дел в этой области. И в модели, и технологии ООБД до сих пор сохраняются слабые места.

К недостаткам модели ООБД можно отнести отсутствие базовых средств, к которым пользователи систем баз данных привыкли. Можно отметить: минимальную оптимизацию и стандартную алгебру запросов; ограниченную поддержку ограничений целостности; ограниченную интеграцию с существующими объектно-ориентированными системами программирования; ограниченный выигрыш в производительности.

К объектно-ориентированным СУБД относятся POET, Jasmine, Versant, O2, ODB-Jupiter, Iris, Orion, Postgres.

Физические модели данных описывают то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Физических моделей данных не так много, как логических, а самыми популярными среди них являются обобщающая модель (Unifying model) и модель памяти кадров (Frame memory).

6. ТЕОРИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

6.1. Основные понятия реляционной модели

В математических дисциплинах понятию «таблица» соответствует понятие «отношение» (relation). *Отношение* представляет собой двумерную таблицу, содержащую некоторые данные. *Таблица* состоит из строк и столбцов и имеет имя, уникальное внутри базы данных. Таблица отражает тип объекта реального мира (сущность), а каждая ее строка — конкретный объект. *Сущность — объект любой природы, данные о котором хранятся в БД. Атрибуты — свойства, характеризующие сущность (столбцы). Степень отношения — количество столбцов.* Реляционной считается такая база данных, в которой все данные представлены для пользователя в виде прямоугольных таблиц значений данных, а все операции над базой данных сводятся к манипуляциям с таблицами. Так, таблица СОТРУДНИК содержит сведения о всех сотрудниках, работающих на предприятии, а ее строки являются наборами значений атрибутов конкретных сотрудников. Каждый столбец таблицы — это совокупность значений конкретного атрибута объекта. Схема отношения — список имен атрибутов, например, СОТРУДНИК (№, ФИО, Год рождения, Должность, Кафедра). *Домен* — совокупность значений атрибутов отношения (тип данных). *Кортеж* — строка таблицы.

Реляционная модель требует, чтобы *каждая строка таблицы была уникальной,* т. е. чтобы любые две строки различались значением хотя бы одного атрибута. *Кардинальность (мощность)* — количество строк в таблице.

Каждый столбец имеет имя, которое обычно записывается в верхней части таблицы (рис. 6.1). Оно должно быть уникальным в таблице, однако различные таблицы могут иметь столбцы с одинаковыми именами.

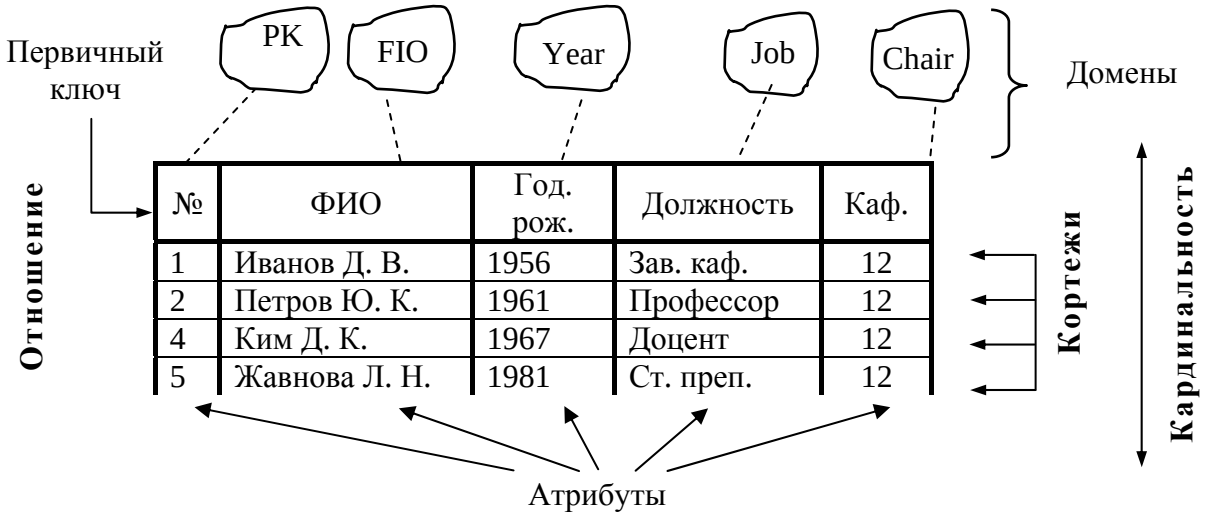

Рис. 6.1

Любая таблица должна иметь хотя бы один столбец; столбцы располагаются в определенном порядке, который создается при создании таблицы. В таблице может не быть ни одной строки, но обязательно должен быть минимум один столбец. В отличие от столбцов, строки не имеют имен; порядок их следования в таблице не определен, а количество логически не ограничено. Для того чтобы идентифицировать какую-либо строку таблицы необходимо создать определенный набор столбцов, который называется ключом. Он может состоять из одного столбца или охватывать все столбцы таблицы.

6.2. Типы ключей в базах данных. Понятие первичного ключа

Ключи можно разделить по признаку общности:

простой — сформирован из значений единственного поля, которые однозначно определяют каждую запись;

составной — сформирован из значений нескольких полей.

Также ключи разделяются:

1) по способу возникновения:

естественный — основан на уже существующем поле. Например, поле «фамилия»;

суррогатный — основан на добавленном искусственным путем отдельном поле для однозначной идентификации;

интеллектуальный[*] — основан на естественном ключе путем добавления дополнительного поля. Например:

Ключ	Фамилия	Имя	Отчество
ТИА	Тихонов	Игорь	Алексеевич

[*] Примеряется крайне редко и лишь для при необходимости добавления сокращенных кодов (например, ТИА-0701).

2) по ограничению целостности данных:

первичный ключ — служит как ограничение целостности в рамках одной таблицы для однозначной идентификации, конкретно поле первичного ключа не может повторяться или быть пустым;

внешний ключ (вторичный) — служит как ограничение целостности связей нескольких таблиц, конкретно подчиненная таблица не может ссылаться на несуществующие записи главной таблицы (что позволяет строить целостные модели данных).

При изображении таблиц первичные ключи таблиц принято выделять. Например, соответствующие поля часто подчеркивают. А Microsoft Access выделяет ключевые поля полужирным шрифтом.

Первичные ключи. Поскольку строки в реляционной таблице не упорядочены, нельзя выбрать строку по ее номеру в таблице. В таблице нет первой, последней или тринадцатой строки. Каким же образом можно указать в таблице конкретную строку, например строку для доцента 1945 г. рождения?

В правильно построенной реляционной базе данных в каждой таблице есть один или несколько столбцов, значения в которых во всех строках разные. *Ключ — это столбец (может быть несколько столбцов), добавляемый к таблице и позволяющий установить связь с записями в другой таблице.* Правила хорошего тона при разработке структур баз данных, а также чисто практические соображения должны побудить разработчика всегда определять первичный ключ для таблицы базы данных. Первичный ключ (сокращенно РК — Primary Key) — столбец, значения которого во всех строках различны. Первичные ключи могут быть логическими (естественными) и суррогатными (искусственными). Ключи, которые можно использовать в качестве первичных, называются *потенциальными* или *альтернативными* ключами.

РК-ключ однозначно определяет соответствующую строку таблицы. Если ключ состоит из одного поля, его часто называют простым, если из нескольких — составным. Первичный ключ не может быть полностью или частично пустым (иметь значение Null).

Выбор ключа на практике является довольно сложной задачей, поскольку состояние отношения обычно не является константой: кортежи постоянно добавляются, удаляются или меняют значения атрибутов. В результате набор, который является ключом в данный момент, в общем случае вовсе не обязательно будет таковым через некоторое время. Например: владелец маленькой фирмы заказал систему для ведения кадров и зарплаты. В ходе анализа выясняется, что у него в данный момент работают три человека: Иванов, Петров и Сидоров. Мы убеждаемся, что фамилия однозначно определяет работника, и принимаем решение: сделать атрибут «Фамилия» ключом в таблице, где хранится перечень работников. Система работает некоторое время без проблем. Благодаря наведенному порядку фирма процветает все больше, и в конечном итоге персонал вырастает до сотни человек. Внезапно

при приеме очередного работника система дает сбой. Выясняется, что его фамилия тоже Иванов. Таким образом, нарушается уникальность ключа, что делает невозможным дальнейшую работу системы. Пытаясь спасти положение и расширяя ключ на столбцы «Имя» и «Отчество» проблема не решается окончательно, а всего лишь уменьшается вероятность очередного сбоя по причине нарушения уникальности ключа. Итак, оказывается, что одной математики здесь недостаточно. Нужно применить знание предметной области для гарантии уникальности ключа. В данном случае опыт подсказывает, что однофамильцы — далеко не редкое явление в коллективе, а совпадение имени-отчества — тоже вполне предсказуемая ситуация. Зачастую проблема решается введением, так называемого суррогатного ключа — к имеющимся атрибутам, которые отражают свойство реального объекта, добавляется еще один (или более), искусственного происхождения, сам способ получения которого гарантирует уникальность. К суррогатным ключам относятся табельные номера сотрудников, регистрационные номера автомобилей, серийные номера изделий и т. д. Тот же номерок, получаемый в гардеробе, также можно отнести к суррогатным ключам, которым гардеробщик однозначно помечает персону, чтобы отличить ее от сотен подобных при выдаче вещей. Нередко суррогатный ключ вводят и из соображений эффективности. Если, например, в таблице имеется длинный составной ключ, то разработчики часто вводят дополнительный короткий числовой суррогатный ключ и именно его делают первичным. Нередко так поступают даже при наличии простого ключа, имеющего неудобный (неэффективный для поиска) тип данных, например, строковый. Подобные операции уже не имеют отношения к теории, но сплошь и рядом встречаются на практике.

Итак, суррогатный ключ — это дополнительное служебное поле, добавленное к уже имеющимся информационным полям таблицы, единственное предназначение которого — служить первичным ключом. Значение этого поля не образуется на основе каких-либо других данных из БД, а генерируется искусственно.

Составной ключ. На сайтах прогнозов погоды нередко представляют информацию следующим образом: для каждой даты указывают прогнозируемую температуру ночью, утром, днем и вечером. Для хранения указанной информации можно использовать вид, приведенный в табл. 6.1.

Таблица 6.1

Дата	Время суток	Температура

В этой таблице ни поле «Дата», ни «Время суток», ни «Температура» не являются ключами — в каждом из этих полей значения могут повторяться. Зато комбинация полей «Дата» + «Время суток» является уникальной и однозначно определяет строку таблицы. Это и есть составной ключ.

Нередко встречается ситуация, в которой выбор ключа не является однозначным. Возьмем еще один пример. Допустим, в дополнение к фамилии, имени, отчеству, ИНН, дате рождения требуется хранить серию и номер общегражданского паспорта, серию и номер заграничного паспорта. Таблица будет иметь вид, приведенный в табл. 6.2.

Таблица 6.2

ИНН	Серия общегражданского паспорта	Номер общегражданского паспорта	Серия заграничного паспорта	Номер заграничного паспорта	Фамилия	Имя	Отчество	Дата рождения

В этой таблице можно выбрать целых три ключа. Один из них — простой (ИНН), два другие — составные («Серия» + «Номер общегражданского паспорта» и «Серия» + «Номер заграничного паспорта»). В такой ситуации разработчик выбирает наиболее удобный с точки зрения организации БД ключ (в общем случае — ключ, на поиск значения которого требуется наименьшее время). Выбранный ключ в этом случае часто называют главным, или первичным ключом, а другие комбинации столбцов, из которых можно сделать ключ, — возможными, или альтернативными ключами. Подводя итоги можно сказать, что хотя бы один возможный ключ или потенциальный ключ в таблице имеется всегда, т. к. строки не могут повторяться и, следовательно, комбинация всех столбцов гарантированно является возможным ключом. Также можно утверждать, что ключ — это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность должна, но не обязана обладать хотя бы одним возможным ключом.

6.3. Многотабличные БД, связи между таблицами, внешние ключи

На практике однотабличные базы данных встречаются достаточно редко, поскольку с точки зрения моделирования базой данных предметной области наличие одной таблицы означает наличие одной сущности. В свою очередь, наличие нескольких сущностей обычно означает наличие связей между ними.

Рассмотрим пример, позволяющий продемонстрировать связи в многотабличных БД.

Пусть существует школа, в которой есть ученики, сгруппированные по классам, и учителя, преподающие некоторые предметы. Можно сразу выделить четыре сущности в виде таблиц: ученики, учителя, классы и предметы.

Далее необходимо решить вопрос об атрибутах сущностей — какую именно информацию они будут хранить. Пусть для каждого ученика необходимо хранить фамилию и имя, для класса — номер параллели и букву, идентифицирующую класс внутри параллели, для учителя — фамилию, имя и отчество, для предмета — только его название.

Теперь следует решить вопрос с первичными ключами. Таблицы учеников и учителей в принципе не имеют ключа, поэтому мы введем в них суррогатный числовой ключ — номер. Таблицы классов и предметов, вообще говоря, имеют ключи. В таблице классов ключ составной, его образуют атрибуты «Номер параллели» + «Буква», а в таблице предметов простой ключ состоит из единственного поля — названия предмета. В результате получится набор таблиц, соответствующих описываемым сущностям (рис. 6.2).

Таблица «Ученик»

Номер ученика	Фамилия	Имя

Таблица «Учитель»

Номер учителя	Фамилия	Имя	Отчество

Таблица «Класс»

Номер класса	Номер параллели	Буква

Таблица «Предмет»

Номер предмета	Название предмета

Рис. 6.2

Понятно, что сущности существуют не сами по себе — они связаны некоторыми отношениями, которые уже обозначены. Чтобы их связать, необходимо ввести дополнительные поля и даже дополнительные таблицы. Разберемся с отношениями между сущностями по порядку (рис. 6.3)

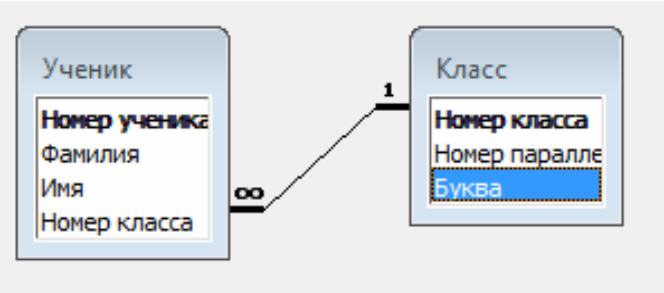

Рис. 6.3

Чтобы отнести ученика к некоторому классу, можно ввести в таблицу «Ученик» дополнительное поле «Номер класса». Понятно, что его тип должен полностью совпадать с типом поля «Номер класса» в таблице «Класс». Теперь можно связать таблицы «Ученик» и «Класс» по совпадающим значениям полей «Номер класса» (эти поля названы одинаково, так как на практике так часто поступают, чтобы легко ориентироваться в связываемых полях). Одной записи в таблице «Класс» может соответствовать много записей в таблице «Ученик», поскольку трудно представить класс из одного ученика. О таких таблицах говорят, что они связаны отношением «один ко многим». А поле «Номер класса» в таблице «Ученик» называют внешним ключом. Таким образом, назначение внешних ключей — связывание таблиц. Внешний ключ всегда ссылается на первичный ключ связанной таблицы (т. е. внешний ключ находится на стороне «многих»). *Внешний ключ — это столбец, значения которого соответствуют значениям первичного ключа другой связанной таблицы.* Связанный с внешним, первичный ключ называют родительским, хотя этот термин используется редко.

Проанализировав предметную область, заметим, что тип связи между сущностями «Учитель» и «Предмет» иной, нежели рассмотренный выше. Ведь не только несколько учителей могут вести один предмет, но и один учитель может вести несколько предметов. Таким образом, между этими сущностями имеется связь «многие ко многим», необходимо ввести дополнительные поля. Связи «многие ко многим» всегда разрешаются посредством введения дополнительной таблицы. Составим таблицу «Учитель — Предмет», имеющую структуру, приведенную на рис. 6.4.

Таблица «Учитель — Предмет»

Номер учителя	Номер предмета

Рис. 6.4

Эта таблица имеет составной ключ, образованный из двух ее полей. И таблица «Учитель», и таблица «Предмет» связаны с данной таблицей отношением «один-ко-многим». Соответственно, в таблице «Учитель — Предмет» имеются два внешних ключа (оба — части составного первичного ключа, что не воспрещается), служащие для связи с соответствующими таблицами.

На практике, помимо рассмотренных отношений «один ко многим» и «многие ко многим», встречается и отношение «один к одному». С точки зрения теории такое отношение интереса не представляет, так как две таблицы, связанные отношением «один к одному», всегда можно просто объединить в одну. Тем не менее, в реальных базах данных отношение «один-к-одному» применяется для оптимизации обработки данных. Например, в базе данных хранится очень много разнообразной информации о людях — данные их документов, телефоны, адреса и пр. Скорее всего, большая часть

этих данных будет использоваться очень редко. А часто требуются лишь фамилия, имя, отчество и телефон. Тогда имеет смысл организовать две таблицы и связать их отношением «один-к-одному». В одной небольшой таблице хранить часто используемую информацию, в другой — остальную. Естественно, что таблицы, связанные отношением «один-к-одному», имеют один и тот же первичный ключ.

Итак, отношения, которые могут существовать между записями двух таблиц:

«один-к-одному» — каждой записи из одной таблицы соответствует одна запись в другой таблице. Связь «один-к-одному» чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две;

«один-ко-многим» — каждой записи из одной таблицы соответствует несколько записей другой таблице. Левая сущность (со стороны «один») называется *родительской*, правая (со стороны «много») — *дочерней*;

«многие-к-одному» — множеству записей из одной таблице соответствует одна запись в другой таблице;

«многие-ко-многим» — множеству записей из одной таблицы соответствует несколько записей в другой таблице. Тип связи «много-ко-многим» является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа «один-ко-многим» путем создания промежуточной сущности.

Тип отношения в создаваемой связи зависит от способа определения связываемых полей:

Отношение «один-ко-многим» создается в том случае, когда только одно из полей является полем первичного ключа или уникального индекса.

Отношение «один-к-одному» создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы.

Отношение «многие-ко-многим» фактически является двумя отношениями «один-ко-многим» с третьей таблицей, первичный ключ которой состоит из полей внешнего ключа двух других таблиц.

6.4. Правила целостности

Логические ограничения, накладываемые на данные, называются *ограничениями целостности.*

Под целостностью базы данных понимается то, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область (правильная) информация.

СУБД должна контролировать соответствие данных заданным ограничениям при переводе БД из одного состояния в другое. Использование ограничений связано также с адекватностью отражения предметной области. Реляционная модель определяет два общих правила целостности базы данных: целостность объектов и ссылочная целостность.

1. *Правило целостности* объектов очень простое. Оно требует, чтобы первичные ключи таблиц не содержали неопределенных (пустых) значений.

2. *Правило ссылочной целостности* требует, чтобы внешние ключи не содержали несогласованных с родительскими ключами значений, т. е. наличие системы правил, используемых для поддержания связей между записями в связанных таблицах. Возвращаясь к рассмотренному выше примеру, мы должны потребовать, например, чтобы ученики относились лишь к классу, номер которого указан в таблице «Классы».

Большинство СУБД умеют следить за целостностью данных (разумеется, это требует соответствующих усилий и от разработчика на этапе описания структур данных). В частности, для поддержания ссылочной целостности используются механизмы каскадирования операций. Каскадирование подразумевает, что при удалении записи из родительской таблицы, связанной с другой таблицей отношением «один-ко-многим», из таблицы «многих» автоматически (самой СУБД, без участия пользователя) удаляются все связанные записи. И это естественно, ведь такие записи «повисают в воздухе», они более ни с чем не связаны.

7. РЕЛЯЦИОННЫЕ ОПЕРАЦИИ НАД ОТНОШЕНИЯМИ

7.1. Общие сведения

РМД стала первой работоспособной моделью данных, поскольку имела эффективный инструментарий — операции реляционной алгебры. Основной единицей обработки является отношение, а не его кортежи. К отношениям можно применить систему операций, позволяющих получить одни отношения из других. Исключение составляют операции создания и заполнения таблиц, а также операции описания и переименования столбцов. Результатом запроса к реляционной БД может быть новое отношение, вычисленное на основе имеющихся отношений.

Результатом любой операции реляционной алгебры всегда является таблица, которую можно потом использовать в других операциях. Все запросы на обработку данных выполняются посредством реляционных операций.

Реляционная алгебра включает две группы операций.

1. *Традиционные операции* над множествами (модифицированные с учетом того, что их операндами являются отношения) — объединение, пересечение, разность (вычитание), декартово произведение.

2. *Специальные реляционные операции* — выборка, проекция, соединение.

7.2. Объединение

При выполнении операции *объединения* (UNION) (∪) двух отношений с одинаковыми заголовками производится отношение, включающее все кортежи, которые входят хотя бы в одно из отношений-операндов, которое имеет тот же состав атрибутов и совокупность кортежей исходных отношений. В результирующее отношение по определению не включаются дубликаты кортежей. Ниже приведены исходные отношения: R1 (табл. 7.1), R2 (табл. 7.2) и результат объединения R (табл. 7.3).

Таблица 7.1

ФИО	Год рождения	Должность	Кафедра
Иванов И.И.	1948	Зав. кафедрой	12
Сидоров С.С.	1953	Доцент	12
Козлов К.К.	1980	Ассистент	32

Таблица 7.2

ФИО	Год рождения	Должность	Кафедра
Цветкова Н.Н.	1965	Доцент	13
Петрова П.П.	1953	Ст. препод.	12
Козлов К.К.	1980	Ассистент	12

Таблица 7.3

ФИО	Год рождения	Должность	Кафедра
Иванов И.И.	1948	Зав. кафедрой	12
Сидоров С.С.	1953	Доцент	12
Козлов К.К.	1980	Ассистент	32
Цветкова Н.Н.	1965	Доцент	13
Петрова П.П.	1953	Ст. препод.	12

7.3. Пересечение

Операция пересечения (INTERSECT) ($\bigcap$) двух отношений с одинаковыми заголовками производит отношение, включающее все кортежи, которые входят в оба отношения-операнда. Результирующее отношение $RP = R1 \bigcap R2$ содержит кортежи, которые есть в каждом из исходных. Результат имеет тот же состав атрибутов, что и исходные отношения. Пересечение отношений R1 и R2 дает отношение RP (табл. 7.4).

Таблица 7.4

ФИО	Год рождения	Должность	Кафедра
Козлов К.К.	1980	Ассистент	32

Отношение, являющееся разностью (MINUS) двух отношений с одинаковыми заголовками, включает все кортежи, входящие в отношение «первый операнд», но ни один из них не входит в отношение, которое является вторым операндом. В результате строится новое отношение $RV = R1 - R2$ с идентичным набором атрибутов, содержащее кортежи первого отношения R1, которые не входят в отношение R2. Вычитание отношения R2 из R1 дает отношение RV (табл. 7.5).

Таблица 7.5

ФИО	Год рождения	Должность	Кафедра
Иванов И.И.	1948	Зав. кафедрой	12
Сидоров С.С.	1953	Доцент	12

7.4. Произведение

При выполнении декартова *произведения* (PRODUCT) (×) двух отношений, пересечение заголовков которых пусто, производится отношение, кортежи которого производятся путем объединения кортежей первого и второго операндов. В результате образуется новое отношение, $RD = R1 \times R2$, которое

включает все атрибуты исходных отношений. Результирующее отношение состоит из всевозможных сочетаний кортежей исходных отношений. Число кортежей (мощность) отношения произведения равно произведению мощностей исходных отношений.

Декартово произведение отношений R1 (табл. 7.6) и R2 (табл. 7.7) дает новое отношение RD (табл. 7.8), которое содержит все атрибуты исходных отношений. В него целесообразно добавить атрибут «Оценка для записи результатов экзамена».

Таблица 7.6

Номер студента	ФИО студента
11	Жак Д.И
12	Жуков П.П.
13	Коган С.С.

Таблица 7.7

Код дисциплины	Наименование
Д1	Математика
Д2	Информатика

Таблица 7.8

Номер студента	ФИО студента	Код дисциплины	Наименование	Оценка
11	Жак Д.И	Д1	Математика	5
12	Жуков П.П.	Д1	Математика	3
13	Коган С.С.	Д1	Математика	5
11	Жак Д.И	Д2	Информатика	5
12	Жуков П.П.	Д2	Информатика	4
13	Коган С.С.	Д2	Информатика	4

7.5. Выборка

Требуется извлечь кортежи из отношения, которые удовлетворяют заданным условиям. *Выборка* (SELECT) (σ) выполняется над одним отношением R. Для отношения по заданному условию (предикату) осуществляется выборка подмножества кортежей. Результирующее отношение имеет ту же структуру, что и исходное, но число его кортежей будет меньше (или равно) числа кортежей исходного отношения. Например, выбрать студентов, сдавших математику на отлично («Код дисциплины» = Д1) AND («Оценка» = 5) (табл. 7.9).

Таблица 7.9

Номер студента	ФИО студента	Код дисциплины	Наименование	Оценка
11	Жак Д.И	Д1	Математика	5
13	Коган С.С.	Д1	Математика	5

7.6. Проекция

Требуется извлечь заданные атрибуты (колонки) из отношения. *Проекция* (PROJECT) (π) выполняется над одним отношением R. Операция формирует новое отношение RPR с заданным подмножеством атрибутов исходного отношения R. Оно может содержать меньше кортежей, так как после отбрасывания в исходном отношении R части атрибутов (и возможного исключения первичного ключа) могут образоваться кортежи-дубли, которые из результирующего отношения исключаются по определению.

Ниже приведен пример исходного отношения R (табл. 7.10) и результат проекции этого отношения на два его атрибута — «Должность» и «Номер отдела» (табл. 7.11).

Таблица 7.10

ФИО	Номер отдела	Должность
Иванов И.И.	01	Инженер
Петров П.П.	02	Инженер
Нестеров Н.Н.	01	Инженер
Никитин К.К.	02	Лаборант

Таблица 7.11

Номер отдела	Должность
01	Инженер
02	Инженер
03	Лаборант

7.7. Соединение

Требуется соединить две таблицы по их общим атрибутам. *Соединение* (JOIN) (∏) выполняется для заданного условия соединения над двумя логически связанными отношениями. Исходные отношения R1 и R2 имеют разные структуры, в которых есть одинаковые атрибуты — внешние ключи. Операция соединения формирует новое отношение, структура которого является совокупностью всех атрибутов исходных отношений. Результирующие кортежи формируются соединением каждого кортежа из R1 с теми кортежами R2, для которых выполняется условие соединения. В зависимости от этого условия соединение называется: *естественным* — равенство значений общих атрибутов отношений R1 и R2; *эквисоединением* — равенство значений атрибутов, входящих в условие соединения; *тета-соединением* — другой знак сравнения.

Операция соединения имеет большое значение для РБД, так как в процессе нормализации отношений исходное отношение разбивается на несколько более мелких отношений, которые при выполнении запросов пользователя требуется, как правило, вновь соединять для восстановления исходного отношения.

8. ЖИЗНЕННЫЙ ЦИКЛ БАЗЫ ДАННЫХ

8.1. Этапы проектирования

Под *жизненным циклом* базы данных понимаются этапы развития БД, начиная от анализа предметной области и заканчивая эксплуатацией БД.

Этапы жизненного цикла базы данных изображаются аналогично развитию любой программной системы, однако в них есть определенная специфика, касающаяся только баз данных:

проектирование базы данных;

проектирование приложений;

реализация БД;

разработка специальных средств администрирования;

эксплуатация базы данных.

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

1) системный анализ и словесное описание информационных объектов предметной области;

2) проектирование инфологической модели предметной области — частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например в терминах ER-модели;

3) выбор СУБД;

4) даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели данных (допустим, логическое описание на языке SQL);

5) физическое проектирование БД, то есть выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения (это делает администратор БД, располагает сегменты, функции).

Если учитывать, что между вторым и третьим этапами необходимо принять решение, с использованием какой-то стандартной СУБД, то условно процесс проектирования БД можно представить последовательностью выполнения пяти соответствующих этапов.

Целью разработки любой базы данных является хранение и использование информации о какой-либо предметной области. Для реализации этой цели имеются следующие инструменты:

1. Реляционная модель данных — удобный способ представления данных предметной области.

2. Язык SQL — универсальный способ манипулирования такими данными.

Однако очевидно, что для одной и той же предметной области реляционные отношения можно спроектировать множеством различных способов.

С точки зрения проектирования БД в рамках системного анализа необходимо осуществить первый этап, т. е. провести подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Желательно, чтобы данное описание позволяло корректно определить все взаимосвязи между объектами предметной области. Предметная область состоит из реальных и абстрактных объектов, которые называют сущностями. Можно считать также, что она состоит из классов сущностей. Классификация основывается на сходстве и учитывает характеристики, общие для нескольких сущностей. Выбор характеристик для группировки сущностей в классы произволен и осуществляется прагматически, в зависимости от целей анализа. В предметной области также рассматриваются некоторые общие свойства, которыми обладают сущности, которые их классифицируют, связывают и т. д. в данной предметной области. Они могут обозначаться как классификации, правила, законы или ограничения, касающиеся состояния и поведения сущностей в предметной области. Все то, что считается частью предметной области, зависит от времени, то есть выбранные объекты и события могут со временем изменяться. Это в равной степени относится и к классификациям, правилам, законам, но, как правило, частота их изменения будет меньше.

8.2. Системный анализ предметной области

Предметная область — это часть реального мира, данные о которой мы хотим отразить в базе данных. Например, в качестве предметной области можно выбрать бухгалтерию какого-либо предприятия, отдел кадров, банк, магазин и т. д. Предметная область бесконечна и содержит как достаточно важные понятия и данные, так и малозначащие или вообще незначительные данные. Так, если в качестве предметной области выбрать учет товаров на складе, то понятия «накладная» и «счет-фактура» являются достаточно важными понятиями, а то, что сотрудница, принимающая накладные, имеет двоих детей, — это для учета товаров неважно. Однако с точки зрения отдела кадров данные о наличии детей являются существенно важными. Таким образом, важность данных зависит от выбора предметной области.

Модель предметной области. Модель предметной области — это наши знания о предметной области. Знания могут существовать как в виде неформальных знаний в мозгу эксперта, так и выражаться формально при помощи

каких-либо средств. В качестве таких средств могут выступать текстовые описания предметной области, наборы должностных инструкций, правила ведения дел в компании и т. п. Опыт показывает, что текстовый способ представления модели предметной области крайне неэффективен. Гораздо более информативными и полезными при разработке баз данных являются описания предметной области, выполненные при помощи специализированных графических нотаций. Имеется большое количество методик описания предметной области. Из наиболее известных можно назвать методику структурного анализа SADT (Structured Analisys and Design Technique) и основанную на нем IDEF0, которая используется для моделирования бизнес-процессов в организационных системах и имеет развитые процедуры поддержки коллективной работы, диаграммы потоков данных Гейна-Сарсона, методику объектно-ориентированного анализа UML, и др. Модель предметной области описывает скорее процессы, происходящие в предметной области и данные, используемые этими процессами. От того, насколько правильно смоделирована предметная область, зависит успех дальнейшей разработки приложений.

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

1. Функциональный подход — применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД. В этом случае мы можем четко выделить минимальный необходимый набор объектов предметной области, которые должны быть описаны.

2. Предметный подход — когда информационные потребности будущих пользователей БД жестко не фиксируются. Они могут быть многоаспектными и весьма динамичными. То есть нельзя точно выделить минимальный набор объектов предметной области, которые необходимо описывать. В описание предметной области в этом случае включаются такие объекты и взаимосвязи, которые наиболее характерны и наиболее существенны для нее. БД, конструируемая при этом, называется предметной, она может быть использована при решении множества разнообразных, заранее не определенных задач. Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, требуемой для решения конкретных задач, которая должна храниться в БД; формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения; описанием выходных документов, которые должны генерироваться в системе, и входных документов, которые служат основанием для заполнения данными БД.

9. КОНЦЕПТУАЛЬНОЕ (ИНФОЛОГИЧЕСКОЕ) ПРОЕКТИРОВАНИЕ

9.1. Понятие инфологической модели

Приступая к следующему этапу проектирования, необходимо помнить, что решаемая задача составляет только часть предметной области. Если мы хотим построить гибкую легко наращиваемую систему, то в первую очередь следует выделить наименьший неделимый в пределах нашей задачи объект описания (минимальный объект описания). Невозможно управлять городским пассажирским потоком, не отслеживая передвижение каждого отдельного человека. Сложно составить школьный журнал, если вся информация об ученике будет собрана в единственный объект.

Однако как жизнь невозможна без законов, так и моделирование не существует без правил. На данном этапе проектирования предметная область представляется моделью, выполненной без ориентации на используемые в дальнейшем программные и технические средства — инфологической моделью.

Проектирование БД начинается с предварительной структуризации предметной области: объекты реального мира подвергаются классификации, фиксируется совокупность подлежащих отображению в БД типов объектов. Для каждого типа объектов фиксируется совокупность свойств, посредством которых будут описываться конкретные объекты этого типа в БД, виды отношений (взаимосвязей) между этими объектами. Затем решаются вопросы о том, какая информация об этих объектах должна быть представлена в БД, как ее представить с помощью данных.

Инфологическая (концептуальная) модель применяется на втором этапе проектирования БД, то есть после словесного описания предметной области. В связи с этим *под инфологической моделью (ИЛМ) понимают описание предметной области, выполненное с использованием специальных языковых средств, не зависящих от используемых в дальнейшем программных средств.*

Цель инфологического моделирования — обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным

языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка). *Основными конструктивными элементами* инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Инфологическая модель должна строиться вне зависимости от того, будете ли вы в дальнейшем использовать какую-либо СУБД или пользоваться другими программными средствами для реализации своей информационной системы.

К концептуальной модели (КМ) предъявляются следующие требования:

адекватное отображение предметной области (язык для представления модели должен обладать достаточными выразительными возможностями для отображения явлений, имеющих место в предметной области, а сама модель должна содержать всю необходимую и достаточную информацию для дальнейшего проектирования системы);

непротиворечивость (модель отражает взгляды и потребности всех пользователей системы, а также обычно является результатом работы многих специалистов, поэтому целостное описание предметной области должно быть проверено на непротиворечивость);

однозначная трактовка модели всеми ее пользователями (обеспечивается формализованностью языка и четким его пониманием всеми участниками процесса создания информационной системы);

легкость восприятия разными категориями пользователей (обеспечивается выбором соответствующего языка моделирования);

конечность модели (несмотря на то, что реальный мир, отображаемый в КМ, является по своей природе бесконечным, инфологическая модель является конечной, что обеспечивается четким ограничением предметной области);

легкость модификации (в концептуальную модель по разным причинам часто приходится вводить новые объекты или модифицировать существующие; ИЛМ должна в связи с этим обладать свойством легкой расширяемости, обеспечивающим ввод новых данных без изменения раннее определенных. То же самое можно сказать и об удалении и корректировке данных);

возможность композиции и декомпозиции модели (в связи с большой размерностью реальных инфологических моделей должна обеспечиваться возможность ее композиции и декомпозиции). Желательно, чтобы язык спецификации концептуальной модели был одинаково применим как при ручном, так и при автоматизированном проектировании информационных систем.

Последнее предъявляет к языку дополнительные требования — он должен:

быть вычисляемым, то есть восприниматься и обрабатываться ЭВМ;

использовать дружелюбные пользователю интерфейсы, в частности графические;

быть независимым от оборудования и других ресурсов, которые подвержены частым изменениям;

использовать средства тестирования КМ, а также иметь аппарат для указания того, что спецификация завершена и по ней может быть выполнена ге-

нерация структур баз данных. При автоматизированном проектировании все изменения, внесенные в КМ, должны быть автоматически отражены в связанных с модифицируемым элементом компонентах банка данных.

Желательно, чтобы КМ строили специалисты, работающие в предметной области, для которой создается модель, а не проектировщики систем машинной обработки данных. Если в силу определенных причин это невозможно обеспечить, то необходимо, чтобы первые могли хотя бы проверить сделанное другим специалистом описание, чтобы убедиться, что специфика предметной области воспринята и отображена правильно. Концептуальная модель является средством коммуникации разнообразных коллективов как конечных пользователей, так и разработчиков. Информация из КМ корреспондирует со словарной системой и другими компонентами банка данных.

9.2. Компоненты инфологической модели

Инфологическая модель предметной области включает в себя ряд компонентов (рис. 9.1). Центральной компонентой инфологической модели является описание объектов предметной области и связей между ними (ER-модель).

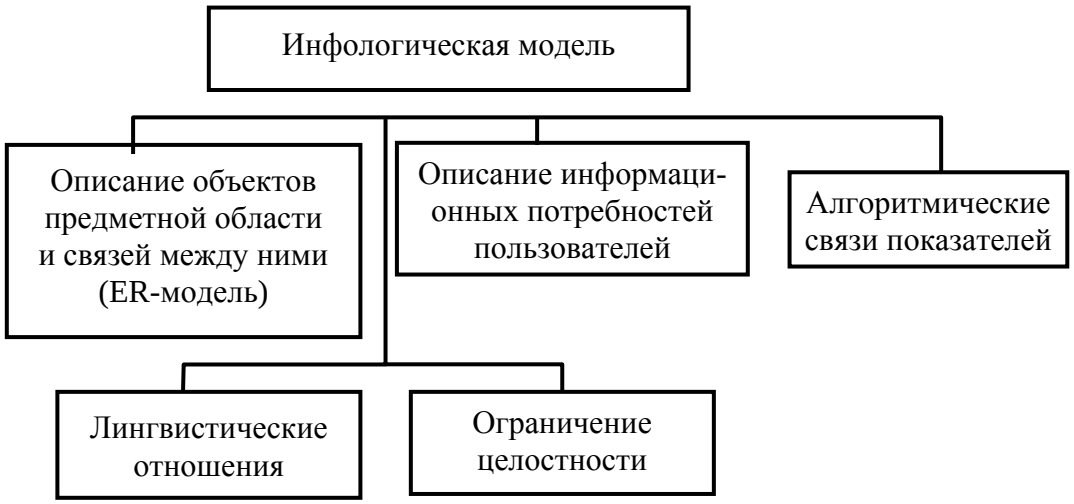

Рис. 9.1

Описание предметной области (ПО) всегда представлено в какой-то знаковой системе. Следовательно, кроме отношений, присущих предметной области, возникают еще и отношения, обусловленные особенностями отображения ПО в языковой среде. Поэтому при построении ИЛМ должны учитываться такие лингвистические категории, как синонимия, омонимия, изоморфизм и др.

Кроме того, в инфологической модели должны быть отражены и алгоритмические зависимости между показателями. Обычно для этих целей используются графы взаимосвязи показателей, отражающие, какие показатели служат исходными для вычисления других. Расчетные формулы и алгоритмы вычислений также в каком-то виде должны быть представлены в ИЛМ.

Следующим компонентом инфологической модели является описание информационных потребностей пользователей. Для этих целей используются специальные языковые средства. Они должны отражать тип запроса, объемно-частотные характеристики, режим использования данных и т. п.

Нельзя сказать, что в настоящее время существует какой-либо стандарт или хотя бы общепринятый способ построения инфологической модели. Для описания ИЛМ используются как языки аналитического (описательного) типа, так и графические средства. Последние в настоящее время приобретают все большую популярность. Существует ряд средств автоматизации проектирования, для которых исходные данные представляются в виде совокупности графических схем. Графическое представление является наиболее наглядным и простым для восприятия и анализа, поэтому мы воспользуемся в дальнейшем именно графическим способом отображения модели.

9.3. Модель «сущность — связь»

В качестве инструмента для построения семантических моделей данных на этапе инфологического проектирования является неформальная модель «сущность — связь» (ER-модель — Entity-Relationship). Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. Основными понятиями модели «сущность — связь» являются: сущность, связь и атрибут.

Любой фрагмент предметной области может быть представлен как множество сущностей, между которыми существует некоторое множество связей.

Сущность — это реальный или представляемый объект, информация о котором должна сохраняться в проектируемой системе. Сущность имеет имя, уникальное в пределах системы. Сущность соответствует некоторому классу однотипных объектов, то есть в системе существует множество экземпляров данной сущности. Примеры сущностей: люди, продукты, студенты и т. д. Примеры экземпляров сущностей: конкретный человек, конкретный продукт, конкретный студент и т. д.

Сущности не обязательно должны быть непересекающимися. Например, экземпляр сущности СТУДЕНТ, также принадлежит сущности ЛЮДИ.

Объект, которому соответствует понятие сущности, имеет свой набор *атрибутов* — характеристик, определяющих свойства данного объекта. Атрибут должен иметь имя, уникальное в пределах данной сущности.

Пример: Рассмотрим множество пищевых продуктов, имеющихся в магазине. Каждый продукт можно представить следующими характеристиками: код продукта, продукт, срок хранения, условия хранения. В дальнейшем для определения сущности и ее атрибутов будем использовать обозначение вида «Продукты» («КодПродукта», «Продукт», «ЕдиницаИзмерения», «СрокХранения», «УсловияХранения»).

Множество допустимых значений (область определения) атрибута называется доменом.

Например, атрибут «СрокХранения» хранит информацию о количестве дней, в течение которых продукт годен к употреблению. То есть этот атрибут принадлежит домену «КоличествоДней», который задается интервалом целых чисел больших нуля, поскольку количество дней отрицательным быть не может.

Набор атрибутов сущности должен быть таким, чтобы можно было однозначно найти требуемый экземпляр сущности. Например, сущность ПРОДУКТЫ однозначно определяется атрибутом «КодПродукта», поскольку все коды продуктов различны.

Отсюда определяется ключ сущности — это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.

Сущность ПРОДАЖА с атрибутами «ДатаПродажи», «КодПродукта», «Количество», «Цена» содержит информацию о продаже продуктов за конкретный день. Для этой сущности ключом будут атрибуты «ДатаПродажи» и «КодПродукта», поскольку за день могут быть проданы несколько продуктов, а конкретный продукт может быть продан в разные дни. Исключение любого атрибута из ключа не позволит однозначно найти требуемый экземпляр сущности, т. е. будет нарушено условие минимальности. Обозначим эту сущность как ПРОДАЖА («ДатаПродажи», «КодПродукта», «Количество», «Цена»). Ключевой атрибут выделен подчеркиванием.

Между сущностями могут быть установлены связи.

Связь — это ассоциация, установленная между несколькими сущностями и показывающая, как взаимодействуют сущности между собой. Например, в магазине происходит продажа продуктов, т. е. между сущностями ПРОДУКТЫ и ПРОДАЖА существует связь «происходит» (обычно, но необязательно, связь обозначается глаголом или двойным названием сущностей, между которыми установлена эта связь). Так как продукты в магазин поставляют поставщики, то между сущностями ПРОДУТЫ и ПОСТАВЩИКИ существует связь «поставка» или «продукты — поставщики». Также могут существовать и связи между экземплярами одной и той же сущности (рекурсивная связь), например связь «родитель — потомок» между экземплярами сущности ЧЕЛОВЕК.

Связь также может иметь атрибуты. Например, для связи «продукты — поставщики» можно задать атрибуты «ДатаПоставки», «Цена» и т. д.

Связь, существующая между двумя сущностями, называется бинарной связью.

Связь, существующая между n сущностями, называется n-арной связью.

Рекурсивная связь — это связь между экземплярами одной сущности.

Общеизвестно, что любую *n*-арную связь всегда можно заменить множеством бинарных, однако первые лучше отображают семантику предметной области.

То число экземпляров сущностей, которое может быть ассоциировано через связь с экземплярами другой сущности, называют *степенью связи*. Рассмотрение степеней особенно полезно для бинарных связей.

Очень важным свойством модели «сущность — связь» является то, что она может быть представлена в виде графической схемы (диаграммы). Это значительно облегчает анализ предметной области. Существует несколько вариантов обозначения элементов диаграммы «сущность — связь» (нотаций). Для обозначения сущностей, связей и атрибутов будем использовать нотацию Чена, а для обозначения степеней и кардинальностей связей — нотацию Мартина (табл. 9.1).

Таблица 9.1

Обозначение	Пояснение
Имя сущности	Независимая сущность
Имя сущности	Зависимая сущность
Имя атрибута	Атрибут
Имя атрибута	Ключевой атрибут
Имя связи	Связь
	Связь степени 1, необязательный класс принадлежности
	Связь степени 1, обязательный класс принадлежности
	Связь степени *N*, необязательный класс принадлежности
	Связь степени *N*, обязательный класс принадлежности
	Связь от зависимой к независимой сущности

Рассмотрим теперь существующие степени бинарных связей:

«один-к-одному» (обозначается 1 : 1). Это означает, что в такой связи в каждый момент времени каждому экземпляру сущности *A* соответствует 1 или 0 экземпляров сущности *B*. Прямоугольники обозначают сущности, а ромб — связь. Так как степень связи для каждой сущности равна 1, то они соединяются одной линией (рис. 9.2).

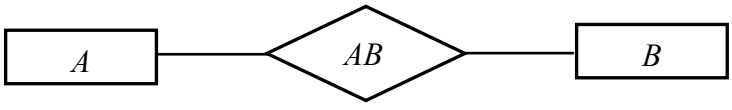

Рис. 9.2

«один-ко-многим» (1 : *N*). Одному экземпляру сущности *A* соответствуют 0, 1 или *N* экземпляров сущности *B*. Графически степень связи *N* отображается древообразной линией (рис. 9.3).

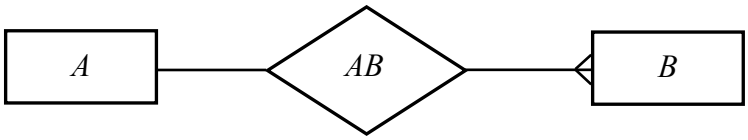

Рис. 9.3

«многие-к-одному» (N : 1). Эта связь аналогична отображению 1 : N. Одному экземпляру сущности B соответствуют 0, 1 или N экземпляров сущности A (рис. 9.4).

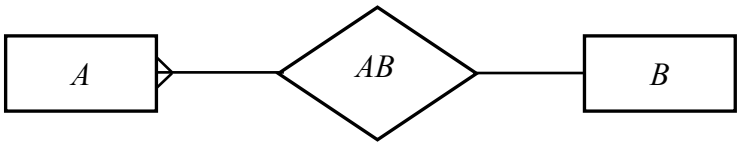

Рис. 9.4

«многие-ко-многим» (M : N). В этом случае одному экземпляру сущности A соответствуют 0, 1 или N экземпляров сущности B, и наоборот, одному экземпляру сущности B соответствуют 0, 1 или N экземпляров сущности A (рис. 9.5).

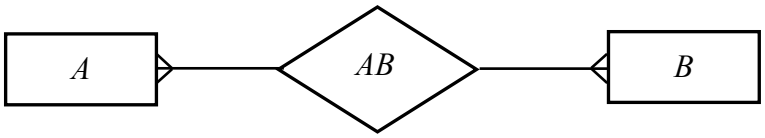

Рис. 9.5

Другой важной характеристикой связи, помимо ее степени, является *класс принадлежности* входящих в нее сущностей. Существует *обязательный* и *необязательный* классы принадлежности. Если все экземпляры сущности участвуют в связи, то класс принадлежности этой сущности — обязательный. Если хотя бы один экземпляр сущности не участвует в связи, то класс принадлежности необязательный.

Если существование сущности x зависит от существования сущности y, то x называется *зависимой* сущностью (иногда сущность x называют слабой, а сущностьy — сильной).

Степень связи для сильной сущности всегда будет 1 и обязательный класс принадлежности. Класс принадлежности и степень связи для зависимой сущности могут быть любыми. Например, в магазине происходит продажа продуктов. Продукт не может быть продан, если его нет в магазине. Поэтому сущность ПРОДАЖИ является зависимой от сущности ПРОДУКТЫ (рис. 9.6)

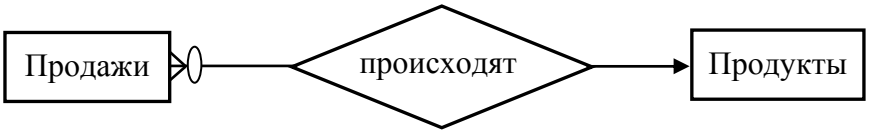

Рис. 9.6

Продукт МОЖЕТ быть продан в разные дни (а может быть вообще не продан), конкретная продажа связана только с одним продуктом. Таким образом, степень связи $N:1$, сущность ПРОДАЖИ имеет необязательный, а сущность ПРОДУКТЫ — обязательный классы принадлежности (в самом деле, продажа без продукта теряет смысл).

Между одними и теми же сущностями могут существовать несколько связей, например, с одной стороны продукты в магазин поставляются заказчиками, с другой стороны, чтобы продукты были поставлены в магазин, необходимо заказать поставщикам необходимые продукты. Таким образом, между сущностями ПРОДУКТЫ и ПОСТАВЩИКИ существуют связи «Поставляют» и «Заказаны» (рис. 9.7). Каждый продукт ДОЛЖЕН быть заказан одному или нескольким поставщикам, каждый поставщик МОЖЕТ получить заказ на один или несколько продуктов или вообще не получить заказ.

Модальность «Может» означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть и не связан ни с одним экземпляром. Модальность «Должен» означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности. Связь может иметь разную модальность с разных концов.

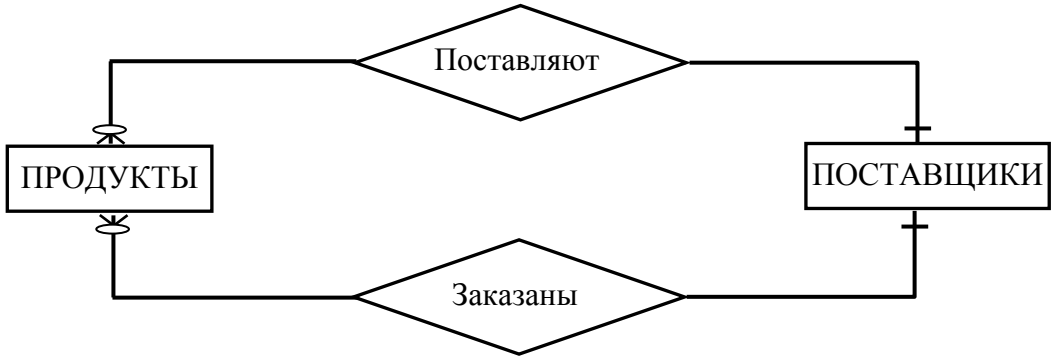

Рис. 9.7

9.4. Построение ER-диаграммы

В процессе построения диаграммы можно выделить несколько этапов:

1) определение списка сущностей выбранной предметной области;

2) определение списка атрибутов сущностей;

3) описание связей между сущностями (степени, классы принадлежности связей, а также атрибуты связей, если они необходимы);

4) организация данных в виде диаграммы «сущность — связь».

В качестве примера построим диаграмму, отображающую связь данных для информационной системы учета продажи продуктов в магазине. БД должна хранить информацию о продуктах, поставляемых в магазин, их ежедневной продаже, заказах на поставку продуктов, а также о поставщиках продуктов. оставим список сущностей, необходимых для реализации поставленной задачи:

1. ПРОДУКТЫ. Для этой сущности необходимы следующие атрибуты:

код продукта — уникальный идентификатор, ключевой атрибут;

продукт — название продукта;

единица измерения — литры, килограммы, штуки и т. п.;

срок хранения в днях — для определения даты окончания срока годности продукта;

условия хранения — температура, влажность и т. п.

2. ПОСТАВЩИКИ должны иметь атрибуты:

код поставщика — уникальный идентификатор, ключевой атрибут;

поставщик — название организации или ФИО физического лица;

код города — выделим отдельно город, где находится поставщик, для удобства дальнейшей работы (например, для поиска);

адрес — поскольку город выделен в отдельный атрибут, то в адресе остается улица и дом (а также квартира — для физического лица);

ФИО директора;

телефон;

факс.

3. ПРОДАЖИ должны иметь атрибуты:

дата продажи;

код продукта — какой именно продукт был продан;

количество — сколько продано этого продукта в тех единицах измерения, которые указаны для этого продукта, в сущности ПРОДУКТ;

цена продажи — цена при продаже за единицу продукта.

4. ГОРОДА — поскольку мы выделили отдельно город из адреса поставщика, то возникает необходимость в этой сущности:

код города — уникальный идентификатор, ключевой атрибут;

город.

Сократив для удобства названия атрибутов, получим список сущностей:

ПРОДУКТЫ (КодПрод, Продукт, ЕдИзм, СрокХран(дней), УсловияХран);

ПОСТАВЩИКИ (КодПост, Поставщик, КодГорода, Адрес, ФИОдиректора, Телефон, Факс);

ПРОДАЖИ (ДатаПродажи, КодПрод, Количество, ЦенаПродажи).

Нужно обратить внимание, что в этой сущности ключ составной, поскольку каждый день продается множество продуктов, и конкретный продукт может быть продан в разные дни;

ГОРОДА (КодГорода, Город).

Рассмотрим связи, существующие между описанными выше сущностями:

1. Продукты в магазин поставляются поставщиками, т. е. существует связь M : *N* «Поставляют» между сущностями ПРОДУКТЫ и ПОСТАВЩИКИ. Эта связь имеет следующие атрибуты:

дата поставки;

код поставщика — какой поставщик поставил этот продукт;

код продукта — какой именно продукт был поставлен;

количество продукта — сколько поставлено этого продукта в тех единицах измерения, которые указаны для этого продукта, в сущности ПРОДУКТ;

цена поставки — цена при поставке за единицу продукта;

дата изготовления — дата изготовления продукта.

Ключом будет составной атрибут: ДатаПоставки, КодПост, КодПрод.

2. Продукты должны быть заказаны поставщикам, т. е. существует связь M : N «Заказаны» между сущностями ПРОДУКТЫ и ПОСТАВЩИКИ. Эта связь имеет следующие атрибуты:

дата заказа;

код поставщика — какому поставщику заказан этот продукт;

код продукта — какой именно продукт был заказан;

количество заказа — сколько поставлено этого продукта в тех единицах измерения, которые указаны для этого продукта в сущности ПРОДУКТ.

Ключом будет составной атрибут: ДатаЗаказа, КодПост, КодПрод.

3. В магазине происходит продажа продуктов, т. е. существует связь N : 1 «Происходит» между сущностями ПРОДАЖИ и ПРОДУКТЫ.

4. Поставщики находятся в определенном городе, т. е. существует связь N : 1 «Находятся» между сущностями ПОСТАВЩИКИ и ГОРОДА.

После объединения всех фрагментов в общую модель и добавления атрибутов, получится ER-диаграмма (рис. 9.8).

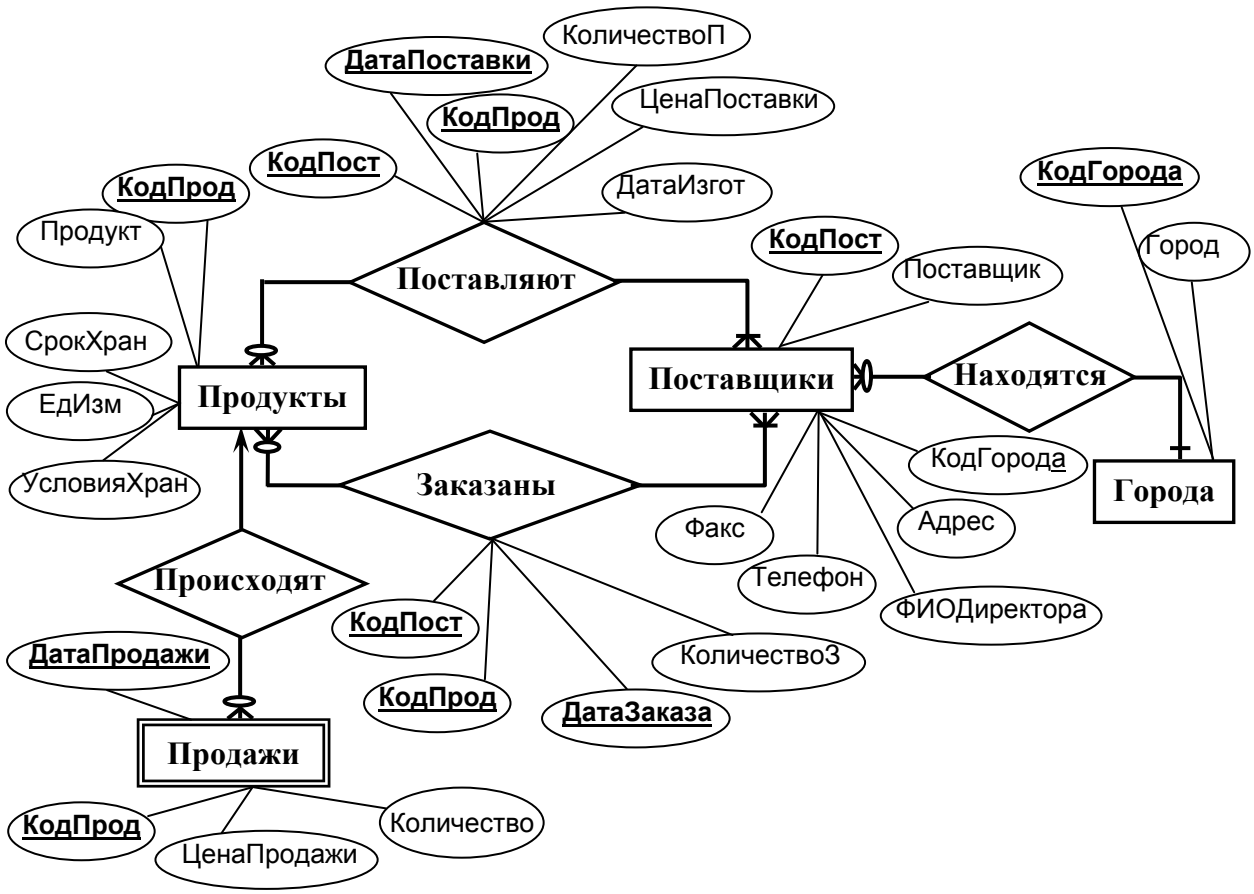

Рис. 9.8

10. ЛОГИЧЕСКОЕ И ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

10.1. Переход от модели «сущность — связь» к реляционной модели

ER-модель используется на ранних стадиях проектирования БД. Являясь концептуальной моделью, она не учитывает особенности конкретной СУБД (допустимые типы и наименования полей и таблиц, ограничения целостности и т. п.). Наиболее часто для решения перечисленных задач используется переход к логической модели базы данных от концептуальной модели, представленной в виде ER-диаграммы. Существует алгоритм однозначного преобразования ER-модели в реляционную модель данных (т. е. осуществляется переход от инфологического моделирования к логическому проектированию схемы реляционной БД). Этот алгоритм положен в основу работы многих CASE-систем — инструментальных средств автоматизированного проектирования баз данных:

1. Каждой сущности ER-модели ставится в соответствие отношение реляционной модели. При этом на имена отношений накладываются ограничения, присущие конкретной СУБД.

2. Каждый атрибут сущности становится атрибутом соответствующего отношения. На имена атрибутов отношения также накладываются ограничения выбранной СУБД. Для каждого атрибута задается конкретный допустимый в СУБД тип данных и обязательность или необязательность данного атрибута (т. е. допустимость или недопустимость Null-значений).

3. Первичный ключ сущности становится первичным ключом соответствующего отношения. Атрибуты, входящие в первичный ключ отношения, автоматически получают свойство отсутствия неопределенных значений (Not Null).

4. В каждое отношение, соответствующее сущности со стороны «многие» (связь $1 : N$), добавляется набор атрибутов сущности со стороны «один», являющихся первичным ключом сущности со стороны «один». В отношении, соответствующим сущности со стороны «многие», этот набор атрибутов становится внешним ключом (рис. 10.1).

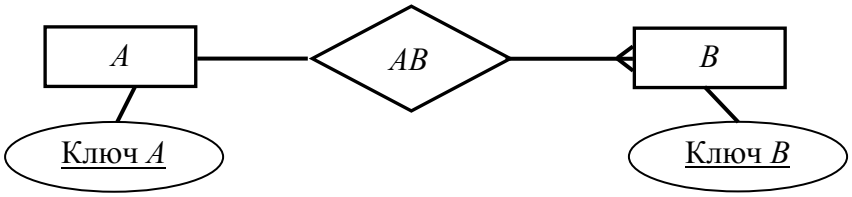

Рис. 10.1

5. Для моделирования необязательного класса принадлежности у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений. При обязательном классе принадлежности атрибуты получают свойство отсутствия неопределенных значений.

6. Разрешение связей типа M : *N*. В соответствие связи становится отношение, имеющее атрибуты, которые в сущностях являются первичными ключами, а в новом отношении будут внешними ключами. Первичным ключом нового отношения будет совокупность внешних ключей (рис. 10.2).

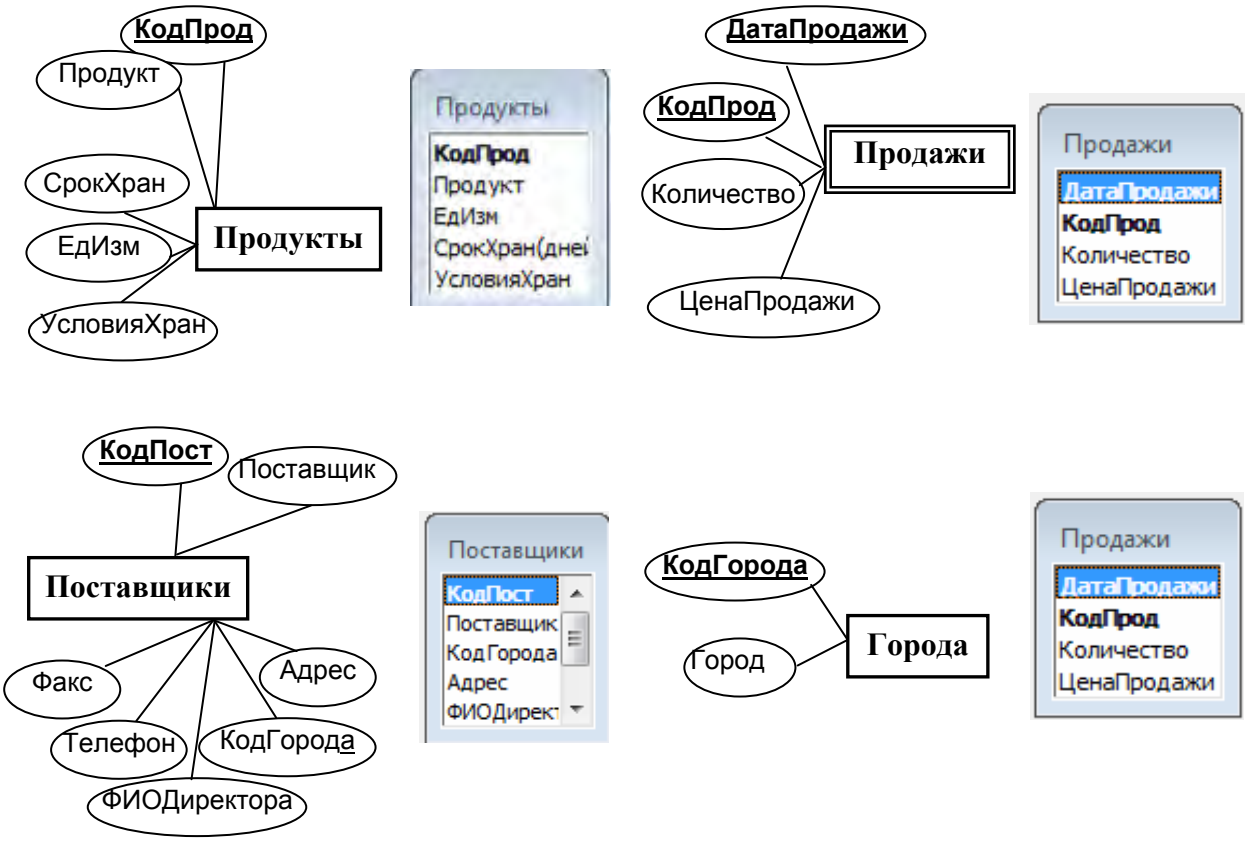

Рис. 10.2

В качестве примера преобразования ER-модели в реляционную модель данных рассмотрим окончательный вариант ER-диаграммы (см. рис. 9.7).

В указанной модели мы имеем дело со следующими сущностями: ПРОДУКТЫ, ПОСТАВЩИКИ, ГОРОДА, ПРОДАЖИ. Следовательно, в реляционной модели будут участвовать четыре отношения с такими же именами.

Далее в процессе задания конкретных типов данных для каждого атрибута отношений (домены могли быть определены уже на этапе инфологического моделирования) получаем отношения, приведенные на рис. 10.2. Табл. 10.1 содержит типы данных для каждого атрибута.

Первичные ключи отношений, описанных в табл. 10.1, помечены знаком V.

Степень связи между сущностями ПОСТАВЩИКИ и ГОРОДА — $N:1$, поэтому первичный ключ «КодГорода» (сущности ГОРОДА) должен войти в сущность Поставщики в качестве внешнего ключа (это сделано еще на этапе создания модели «сущность — связь»). Степень связи между сущностями ПРОДАЖИ и ПРОДУКТЫ — $N:1$, поэтому первичный ключ «КодПрод» (сущности ПРОДУКТЫ) должен войти в сущность ПРОДАЖИ в качестве внешнего ключ (это сделано еще на этапе создания модели «сущность — связь»).

Таблица 10.1

Атрибут	Тип данных (СУБД Access)	Допустимость Null-значений	Первичный ключ	Внешний ключ
Отношение Продукты				
КодПрод	Целое	Нет	V	
Продукт	Текстовый (30)	Нет		
ЕдИзм (5)	Текстовый	Да		
СрокХран(дней)	Целое	Да		
УсловияХран	Текстовый	Да		
Отношение Поставщики				
КодПост	Целое	Нет	V	
Поставщик	Текстовый (50)	Нет		
КодГорода	Целое	Да	V	
Адрес	Текстовый	Да		
ФИОдиректора	Текстовый (50)	Да		
Телефон	Текстовый (15)	Да		
Факс	Текстовый (15)	Да		
Отношение Продажи				
ДатаПродажи	Дата/время	Нет	V	
КодПрод	Целое	Нет		V
Количество	Одинарное с плавающей точкой	Да		
ЦенаПродажи	Денежный	Да		
Отношение Города				
КодГорода	Целое	Нет	V	
Город	Текстовый (30)	Нет		
Отношение Поставки				
ДатаПоставки	Дата/время	Нет	V	
КодПост	Целое	Нет		
КодПрод	Целое	Нет		

Окончание табл. 10.1

Атрибут	Тип данных (СУБД Access)	Допустимость Null-значений	Первичный ключ	Внешний ключ
КоличествоП	Одинарное с плавающей точкой	Да		
ДатаИзгот	Дата/время	Да		
Отношение Заказы				
ДатаЗаказа	Дата/время	Нет	V	
КодПост	Целое	Нет		
КодПрод	Целое	Нет		
КоличествоЗ	Одинарное с плавающей точкой	Да		

Для внешнего ключа «КодГорода» (отношение ПОСТАВЩИКИ) устанавливаем свойство «Допустимость Null-значений» «Да», так как в модели «сущность — связь» сущность ПОСТАВЩИКИ имела необязательный класс принадлежности; для внешнего ключа «КодПост» (отношение ПОСТАВЩИКИ) устанавливаем свойство «Допустимость Null-значений» «Нет», поскольку этот внешний ключ входит в состав первичного ключа (рис. 10.3).

В данном примере две связи имеют степень M : *N*. Это связи «Поставляют» и «Заказаны». Следовательно, дополнительно появляются еще два отношения «Поставки» и «Заказы» соответственно.

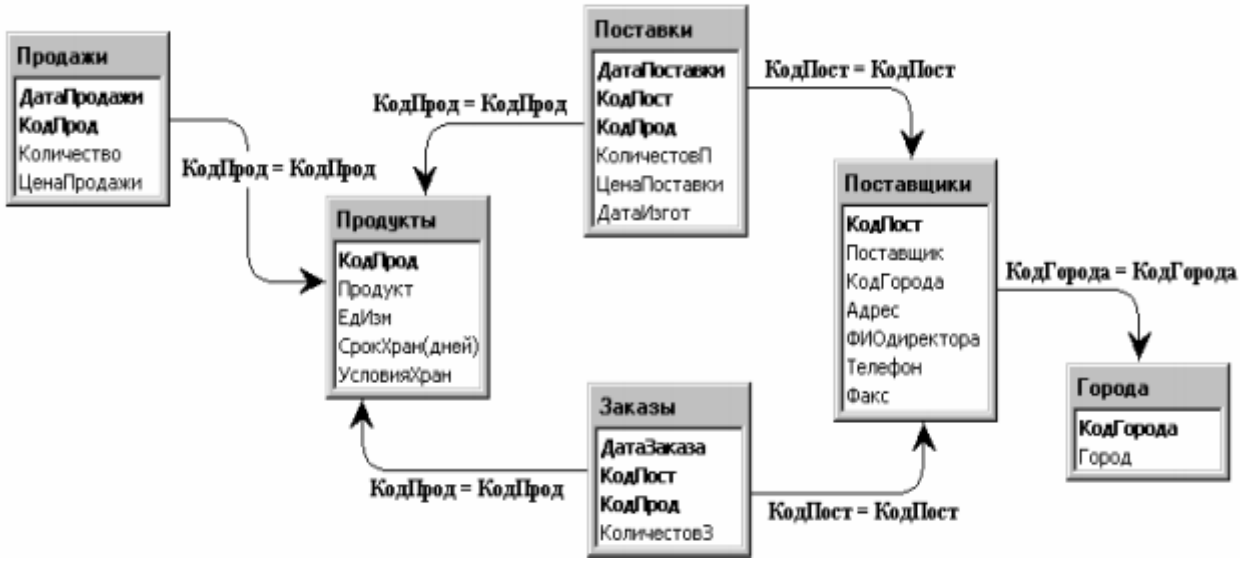

Рис. 10.3

10.2. Логическое проектирование базы данных

Следующим этапом жизненного цикла БД, является этап даталогического или логического проектирования БД. При проектировании логической структуры реляционной базы данных определяется оптимальный состав таблиц для хранения исходной информации, разрабатывается схема БД. Для каждой таблицы указывается ее название, перечень полей и первичный ключ. Идентифицируются связи между таблицами. В рамках логического проектирования БД могут формулироваться ограничения целостности, приниматься решения о создании индексов и т. д.

Схема БД — совокупность схем отношений, адекватно моделирующих абстрактные объекты предметной области и семантические связи между этими объектами. Основой анализа корректности схемы являются анализ функциональных зависимостей между атрибутами отношений БД. Некоторые функциональные зависимости являются нежелательными из-за побочных эффектов и аномалий, возникающих при модификации БД.

На этапе инфологического моделирования была построена модель «сущность — связь», и с помощью алгоритма перехода к реляционной модели получена схема БД (см. рис. 10.3), т. е. был начат этап логического проектирования. Для продолжения процесса проектирования необходимо проверить полученную схему БД на отсутствие избыточных функциональных зависимостей и при необходимости нормализовать схему БД.

Процесс нормализации может быть проведен уже с концептуальной моделью «сущность — связь», тогда после перехода к реляционной модели получим нормализованную схему БД.

Построение схемы БД может быть выполнено двумя путями:

1) декомпозиция (разбиение), когда исходное множество отношений, входящих в схему БД, заменяется другим множеством отношений (их число при этом возрастает), являющимися проекциями исходных отношений;

2) синтез, то есть компоновка из заданных исходных элементарных зависимостей между объектами предметной области схемы БД.

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

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

10.3. Физическое проектирование базы данных

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решение о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования уже определена логическая структура базы данных (которая описывает отношения и ограничения в рассмат-

риваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных (реляционной, сетевой или иерархической). Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Физическое проектирование базы данных — процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

В процессе создания базы данных на компьютере сначала осуществляется конструирование ее таблиц средствами Access. Далее создается схема данных, в которой устанавливаются логические связи таблиц. В схеме данных базы могут быть заданы параметры поддержания связной целостности данных, если модель данных была разработана в соответствии с требованиями нормализации.

Связная целостность данных означает, что в базе данных установлены и корректно поддерживаются взаимосвязи между записями разных таблиц при загрузке, добавлении и удалении записей в связанных таблицах, а также при изменении значений ключевых полей. При обеспечении связной целостности в подчиненной таблице не может существовать запись, для которой отсутствует связанная запись в главной таблице.

11. ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ НА ОСНОВЕ ПРИНЦИПОВ НОРМАЛИЗАЦИИ

11.1. Нормализация отношений

После формирования в Access схемы данных можно приступать к вводу данных в базу — загрузке с документов предметной области, являющихся источниками данных. В практических приложениях пользователя обычно не используется ввод непосредственно в таблицы, а применяются специально создаваемые экранные формы, выполняющие роль интерфейса пользователя.

Каждый программист обычно по-своему проектирует базу данных для программы, над которой работает. У одних это получается лучше, у других — хуже. Качество спроектированной БД в немалой степени зависит от опыта и интуиции программиста, однако существуют правила, помогающие улучшить проектируемую БД, которые носят рекомендательный характер и называются нормализацией базы данных.

Проектирование базы данных, основанное на построении нормализованной модели данных предметной области, позволяет легко получить логическую структуру реляционной базы данных Access, в которой автоматически поддерживается целостность и непротиворечивость данных.

Нормализация представляет собой действия по последовательному преобразованию исходной (ненормализованной) таблицы в нормализованные отношения в первой нормальной форме 1НФ, 2НФ, 3НФ, нормальной форме Бойса-Кодда (НФБК), 4НФ, 5НФ, иначе говоря, процесс нормализации данных заключается в устранении избыточности данных в таблицах.

Нормализация — это итерационный обратный процесс декомпозиции начального отношения на несколько простейших отношений меньшей размерности. Под *обратимостью процесса* понимают то, что операция объединения отношений, полученных в результате декомпозиции, должна дать начальное отношение, то есть при выполнении декомпозиции должно выполняться условие объединения без потерь информации. Полученный в результате нормализации состав атрибутов отношений БД должен отвечать следующим требованиям:

между атрибутами не должно быть нежелательных функциональных зависимостей;

группирование атрибутов должно обеспечивать минимальное дублирование данных, их обработку и возобновление без осложнений и аномалий.

Вместе с тем полученные в результате декомпозиции отношения не должны утратить функциональных зависимостей начального отношения, так как это может привести к искажению семантики данного отношения.

Аппарат нормализации также разработал Кодд. Каждая нормальная форма ограничивает тип допустимых зависимостей между атрибутами. Кодд выделил три нормальных формы (сокращенные названия 1НФ, 2НФ и 3НФ). Наиболее совершенная из них — 3НФ. Сейчас уже известны и определенны 4НФ и 5НФ.

Нормализацию отношений выполняют в несколько шагов. Первая итерация (первый шаг) — сведение отношений к первой нормальной форме (1НФ), затем ко второй и к третьей.

В ходе нормализации формат отношений становится все более ограниченным (строгим) и менее восприимчивым к аномалиям обновления. При работе с реляционной моделью данных важно понимать, что для создания отношений приемлемого качества обязательно только выполнение требований первой нормальной формы (1НФ). Все остальные формы могут использоваться по желанию проектировщиков. Но для того чтобы избежать аномалий обновления, нормализацию рекомендуется выполнять как минимум до третьей нормальной формы (3НФ).

Основные свойства нормальных форм:

каждая следующая нормальная форма улучшает свойства предыдущей нормальной формы;

при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Цели, которые преследуются при построении наиболее эффективной структуры данных:

обеспечить быстрый доступ к данным;

исключить ненужное повторение данных, которое может являться причиной ошибок при вводе, а также привести к нерациональному использованию дискового пространства;

обеспечить целостность данных, т. о. чтобы при изменении одних объектов автоматически происходило соответствующее изменение связанных с ними объектов.

Первая нормальная форма (1НФ) требует, чтобы каждое поле таблицы БД было неделимым (атомарным) и не содержало повторяющихся групп. Все записи в одном столбце (атрибуте) должны иметь один и тот же тип. Каждый столбец должен иметь уникальное имя, но порядок следования столбцов в таблице несуществен. Неделимость означает, что в таблице не должно быть полей, которые можно разбить на более мелкие поля. Например, если в одном поле объединить фамилию студента и группу, в которой он учится, требование неделимости соблюдаться не будет. Первая нормальная форма требует, чтобы данные были разбиты по двум полям.

Под понятием повторяющиеся группы подразумевают поля, содержащие одинаковые по смыслу значения (табл. 11.1).

Таблица 11.1

№	Студент 1	Студент 2	Студент 3
1	Иванов И.И.	Петров П.П.	Сидоров С.С.

Такую таблицу можно сделать, однако она нарушает правило первой нормальной формы. Поля «Студент 1», «Студент 2» и «Студент 3» содержат одинаковые по смыслу объекты, их требуется поместить в одно поле «Студент». В табл. 11.1 поля описывают студентов в формате «Фамилия И.О.» Однако если оператор будет вводить эти описания в формате «Фамилия Имя Отчество», то нарушается также правило неделимости. В этом случае каждое такое поле следует разбить на три отдельных поля, так как поиск может вестись не только по фамилии, но и по имени или по отчеству.

Отношение находится в первой нормальной форме, если на пересечении каждого столбца и каждой строки находятся только элементарные (неделимые) значения атрибутов.

Вторая нормальная форма (2НФ) требует, чтобы таблица удовлетворяла всем требованиям первой нормальной формы и любое не ключевое поле однозначно идентифицировалось полным набором ключевых полей. Рассмотрим пример: некоторые студенты посещают спортивные платные секции, и ВУЗ взял на себя оплату этих секций (табл. 11.2).

Таблица 11.2

№ студента	Секция	Плата
100	Плавание	100
110	Скейтборд	150
112	Теннис	175
254	Плавание	100
260	Теннис	175

Ключом этой таблицы служат поля «№ студента» — «Секция». Однако данная таблица также содержит отношение «Секция» — «Плата». Если удалить запись студента № 110, то потеряются данные о стоимости секции по скейтборду. А после этого нельзя будет ввести информацию об этой секции, пока в нее не запишется хотя бы один студент. Говорят, что такое отношение подвержено как аномалии удаления, так и аномалии вставки.

В соответствие с требованиями второй нормальной формы, каждое не ключевое поле должно однозначно зависеть от ключа. Поле «Плата» в приведенном примере содержит сведения о стоимости данной секции и никоим образом не зависит от ключа — номера студента. Таким образом, чтобы удовлетворить требованию второй нормальной формы, данную таблицу следует разбить на две таблицы, каждая из которых зависит от своего ключа (табл. 11.3, 11.4).

Таблица 11.3

№ студента	Секция
100	Плавание
110	Скейтборд
112	Теннис
254	Плавание
260	Теннис

Ключ: № студента

Таблица 11.4

Секция	Плата
Плавание	100
Скейтборд	150
Теннис	175

Ключ: Секция

Получены две таблицы, в каждой из которых не ключевые данные однозначно зависят от своего ключа. *Отношение является второй нормальной формой, если оно находится в первой нормальной форме, и каждый неключевой атрибут (т. е. не являющийся составной частью первичного ключа) функционально полно зависит от первичного ключа.* 2НФ применяется к таблицам, которые имеют составной ключ или частично зависимое поле, зависящее только от части ключа.

Для приведения к 2НФ необходимо:

вынести все частично зависимые поля в отдельную таблицу;

определить ключевые поля;

установить отношения между таблицами.

Третья нормальная форма (3НФ) требует, чтобы в таблице не имелось транзитивных зависимостей между не ключевыми полями, то есть, чтобы значение любого поля, не входящего в первичный ключ, не зависело от другого поля, также не входящего в первичный ключ. Допустим, в студенческой базе данных есть таблица с расходами на спортивные секции (табл. 11.5).

Таблица 11.5

Секция	Плата	Количество студентов	Общая стоимость
Плавание	100	2	200
Скейтборд	150	1	150
Теннис	175	2	350

Как нетрудно заметить, ключевым полем здесь является поле «Секция». Поля «Плата» и «Количество студентов» зависят от ключевого поля и не зависят друг от друга. Однако поле «Общая стоимость» зависит от полей «Плата» и «Количество студентов», которые не являются ключевыми, следовательно, нарушается правило третьей нормальной формы.

Поле «Общая стоимость» в данном примере можно спокойно убрать из таблицы, ведь если потребуется вывести такие данные, нетрудно будет перемножить значения полей «Плата» и «Количество студентов» и создать для вывода вычисляемое поле. *Отношение является третьей нормальной формой, если оно находится во второй нормальной форме и каждый его неключевой атрибут непосредственно (не транзитивно) зависит от первичного ключа, т. е все поля, не входящие в полный первичный ключ, должны зависеть от него и не зависеть друг от друга.*

Таким образом, нормализация данных подразумевает, что вначале проектируется сама база данных: планируется, какие таблицы и поля в ней будут, какого типа и размера. Затем каждая таблица приводится к первой нормальной форме. После этого полученные таблицы приводятся ко второй, а затем и к третьей нормальной форме, после чего можно утверждать, что база данных нормализована.

Однако такой подход имеет и недостатки: если требуется разработать программный комплекс для крупного предприятия, база данных будет довольно большой. При нормализации данных можно получить сотни взаимосвязанных между собой таблиц. С увеличением числа нормализованных таблиц уменьшается восприятие программистом базы данных в целом, то есть можно потерять общее представление обо всей базе данных, запутаться в связях. Кроме того, поиск в чересчур нормализованных данных может быть замедлен. Отсюда вывод: при работе с данными большого объема необходимо искать компромисс между требованиями нормализации и собственным общим восприятием базы данных.

Главная цель нормализации базы данных — устранение избыточности и дублирования информации. В идеале при нормализации надо добиться, чтобы любое значение хранилось в базе в одном экземпляре, причем значение это не должно быть получено расчетным путем из других данных, хранящихся в базе.

На практике приводить отношения к 1НФ и 2НФ приходится достаточно редко. Чаще хватает просто просмотреть отношения, чтобы убедиться, что они не нарушают эти нормальные формы. С 3НФ ситуация несколько сложнее, так как транзитивная зависимость не всегда так явно бросается в глаза. Поэтому нарушение 3НФ — наиболее частая проблема, с которой приходится иметь дело.

Вообще, нормализация — это фактически исправление огрехов, допущенных при проектировании БД. Лучше просто не допускать этих огрехов с самого начала, чем потом оптимизировать.

11.2. Избыточное дублирование данных и аномалии в базах данных

В реляционной БД отношения должны быть нормализованны. Структура реляционных отношений в нормализованной базе данных должна быть оптимальной, то есть наиболее устойчивой к внесению изменений в данные и связи между ними. Следовательно, в ненормализованном отношении могут появиться аномалии, связанные с выполнением операций поддержки его в актуальном состоянии. Выявить фактическое проявление той или иной аномалии можно, учитывая конкретную семантику данных.

Аномалиями называют такую ситуацию в таблицах БД, которая приводит к противоречиям в БД, либо существенно усложняет обработку данных.

Существуют разные виды аномалий:

1. Аномалия обновления — появление в базе данных несогласованности данных при выполнении операций вставки, удаления, модификации записей.

2. Аномалии модификации — появление записей с противоречащими значениями в некоторых столбцах при изменении значений соответствующих полей одной записи.

3. Аномалии удаления — удаление лишней информации при удалении записи, т. е. при удалении фактов, относящихся к одной сущности, мы непроизвольно удаляем факты, относящиеся к другой сущности.

4. Аномалии вставки — добавление лишней информации или возникновение противоречащих значений в некоторых столбцах при вставке новой записи.

Указанные аномалии связаны с избыточностью данных в БД. Следует различать избыточное и не избыточное дублирование данных.

Неизбыточное дублирование возникает из необходимости хранить идентичные данные, так как важен сам факт их идентичности, и удаление хотя бы одного представителя идентичных данных приведет к невосполнимой потере информации.

Избыточное дублирование (избыточность) обычно связано с необходимостью задания значения всех атрибутов отношения, при этом дублируемые данные не являются необходимыми, и в случае потери (удаления) могут быть восстановлены по данным одного или нескольких отношений БД.

На уровне логического моделирования определяются реляционные отношения и атрибуты этих отношений. На этом уровне нельзя определить какие-либо физические структуры хранения (индексы, хеширование и т. п.). Единственное, чем можно управлять — это распределением атрибутов по различным отношениям. Можно описать мало отношений с большим количеством атрибутов или много отношений с малым количеством атрибутов. Таким образом, необходимо попытаться ответить на вопрос — влияет ли количество отношений и количество атрибутов в отношениях на скорость выполнения операций обновления данных. Основными операциями, изменяющими состояние базы данных, являются операции вставки, обновления и удаления записей. В базах данных, требующих постоянных изменений (складской учет, системы продаж билетов и т. п.), производительность определяется скоростью выполнения большого количества небольших операций вставки, обновления и удаления.

Например, вставка записи производится в одну из свободных страниц памяти, выделенной для данной таблицы. СУБД постоянно хранит информацию о наличии и расположении свободных страниц. Если для таблицы не созданы индексы, то операция вставки выполняется фактически с одинаковой скоростью независимо от размера таблицы и от количества атрибутов в таблице. Если в таблице имеются индексы, то при выполнении операции вставки записи индексы должны быть перестроены. Таким образом, скорость выполнения операции вставки уменьшается при увеличении количества индексов у таблицы и мало зависит от числа строк в таблице.

Прежде чем обновить или удалить запись, ее необходимо найти. Если таблица не индексирована, то единственным способом поиска является последовательное сканирование таблицы до нахождения нужной записи. В этом случае скорость операций обновления и удаления существенно увеличивается с увеличением количества записей в таблице и не зависит от количества атрибутов. Но на самом деле неиндексированные таблицы практически никогда не используются. Для каждой таблицы обычно объявляется один или несколько индексов, соответствующих потенциальным ключам. При помощи этих индексов поиск записи производится очень быстро и практически не зависит от количества строк и атрибутов в таблице (но некоторая зависимость имеется). Если для таблицы объявлено несколько индексов, то при выполнении операций обновления и удаления эти индексы должны быть перестроены, на что тратится дополнительное время. Таким образом, скорость выполнения операций обновления и удаления также уменьшается при увеличении количества индексов у таблицы и мало зависит от числа строк в таблице.

Можно предположить, что чем больше атрибутов имеет таблица, тем больше для нее будет объявлено индексов. Эта зависимость, конечно, не прямая, но при одинаковых подходах к физическому моделированию обычно так и происходит. То есть, чем больше атрибутов имеют отношения, разработанные в ходе логического моделирования, тем медленнее будут выполняться операции обновления данных за счет затраты времени на перестройку большего количества индексов.

11.3. Ограничения целостности в реляционной модели данных

Одним из основополагающих понятий в технологии баз данных является понятие *целостности*. В общем случае это понятие связано с тем, что база данных отражает в информационном виде некоторый объект реального мира или совокупность взаимосвязанных объектов реального мира. В реляционной модели объекты реального мира представлены в виде совокупности взаимосвязанных отношений. *Целостность данных — это механизм поддержания соответствия базы данных предметной области или защиты данных от неверных изменений или разрушений.*

Поддержка целостности в реляционной модели данных в ее классическом понимании включает в себя три аспекта:

1) структурную целостность;
2) языковую целостность;
3) ссылочную целостность.

Эти три вида целостности определяют допустимую форму представления и обработки информации в реляционных БД. Для определения некоторых ограничений, связанных с содержанием БД, используется другой вид целостности — *семантическая целостность*.

Структурная целостность подразумевает, что реляционная СУБД может работать только с реляционными отношениями. А реляционное отношение, в свою очередь, должно удовлетворять ограничениям, накладываемым на него в классической теории реляционных БД (отсутствие одинаковых кортежей и, следовательно, наличие первичного ключа, отсутствие упорядоченности атрибутов и кортежей).

Требование структурной целостности осуществляется с помощью двух ограничений:

1. При добавлении кортежей в отношение проверяется уникальность их первичных ключей.

2. Не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение.

В дополнение к структурной целостности необходимо рассмотреть проблему неопределенных Null-значений. *Неопределенное значение* интерпретируется в реляционной модели как значение, неизвестное на данный момент времени. При сравнении неопределенных значений не действуют стандартные правила сравнения: одно Null-значение никогда не считается равным другому Null-значению. Для выявления равенства значения некоторого атрибута неопределенному применяют стандартные предикаты:

<Имя атрибута> Is Null;

<Имя атрибута> Is Not Null.

Языковая целостность состоит в том, что реляционная СУБД должна обеспечивать языки описания и манипулирования данными не ниже стандарта SQL. Не должны быть доступны иные низкоуровневые средства манипулирования данными, не соответствующие стандарту. Именно поэтому доступ к информации, хранимой в базе данных, и любые изменения этой информации могут быть выполнены только с использованием операторов языка SQL.

При установлении связи между отношениями возникает необходимость поддержания целостности по ссылкам, которая означает обеспечение одного из заданных принципов взаимосвязи между экземплярами кортежей взаимосвязанных отношений:

1) кортежи подчиненного отношения уничтожаются при удалении кортежа основного отношения, связанного с ними;

2) кортежи основного отношения модифицируются при удалении кортежа основного отношения, связанного с ними, при этом на месте ключа родительского отношения ставится неопределенное Null-значение.

Ссылочная целостность обеспечивает поддержку непротиворечивого состояния БД в процессе модификации данных при выполнении операций добавления или удаления.

Кроме указанных ограничений целостности, которые в общем виде не определяют семантику БД, вводится понятие семантической поддержки целостности.

Структурная, языковая и ссылочная целостность определяют правила работы СУБД с реляционными структурами данных. Требования поддержки этих трех видов целостности говорят о том, что каждая СУБД должна уметь это делать, а разработчики должны это учитывать при построении баз данных с использованием реляционной модели. Требования поддержки целостности достаточно абстрактны, они определяют допустимую форму представления и обработки информации в реляционных базах данных. Но с другой стороны, эти аспекты никак не касаются содержания базы данных. Для определения некоторых ограничений, связанных с содержанием базы данных, требуются другие методы, которые и сведены в поддержку семантической целостности.

12. РЕАЛИЗАЦИЯ РЕЛЯЦИОННОЙ МОДЕЛИ В СРЕДЕ MS ACCESS

12.1. Создание таблиц

Рассмотрим процесс реализации реляционной модели учета продажи продуктов в магазине, в СУБД MS Access. Создаваемую базу данных назовем БД «Магазин».

В СУБД MS Access терминология несколько отличается от терминологии реляционных моделей (табл. 12.1).

Таблица 12.1

Термины реляционной модели	Термины СУБД MS Access
Отношение	Таблица
Атрибут	Поле
Кортеж	Запись
Связь	Связь

Имена таблиц (и других объектов Access) и полей должны подчиняться следующим правилам:

должны содержать не более 64 символов;

могут включать любую комбинацию букв, цифр, пробелов и специальных символов за исключением точки (.), восклицательного знака (!), надстрочного символа (`) и квадратных скобок ([ ]);

не должны начинаться с символа пробела;

не должны включать управляющие символы (с кодами ASCII от 0 до 31);

не должны включать прямые кавычки (") в именах таблиц, представлений и хранимых процедур в проекте MS Access.

Хотя пробелы внутри имен полей, элементов управления и объектов являются допустимыми, в большинстве примеров в документации MS Access имена полей записываются без пробелов. Пробелы в именах могут при некоторых обстоятельствах вызывать конфликты в программах Visual Basic.

Одним из способов создания таблиц в MS Access является создание таблиц в *режиме конструктора*. Последовательность действий при создании таблицы:

1. Запустить режим конструктора таблиц (рис. 12.1).

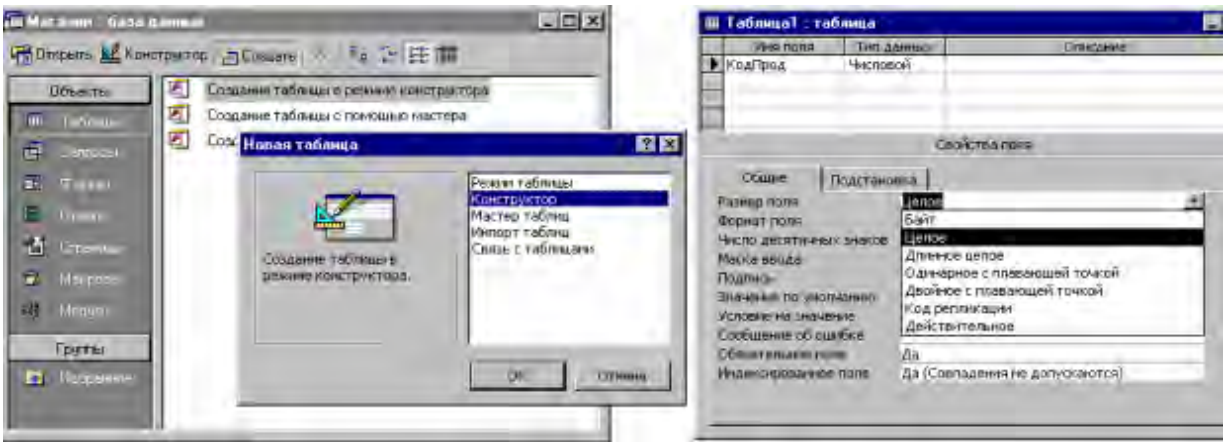

Рис. 12.1

2. В столбце «Имя поля» ввести имя очередного поля таблицы.
3. В столбце «Тип данных» ввести или выбрать из списка тип данных поля:

«Текстовый» (Text) — для хранения текста, комбинации букв и цифр (тип данных по умолчанию), максимальный размер 255 символов;

«MEMO» — для хранения длинного текста или чисел, например, примечания или описания, максимальный размер 65 535 символов;

«Числовой» (Number) — для хранения числовых данных, используемых для математических вычислений, за исключением финансовых расчетов;

«Дата/время» (Date/Time) — хранение дат и/или времени от 100 года до 9999 года (8 байт);

«Денежный» (Currency) — хранение значения валют. Денежный тип используется для предотвращения округлений во время вычислений. Предполагает до 15 символов в целой части числа и 4 — в дробной (8 байт);

«Счетчик» (AutoNumber) — для автоматической нумерации. Поле счетчика может генерировать три типа чисел: последовательно возрастающие на единицу, случайные числа, а также коды репликации. Созданный для записи номер уже не может быть удален или изменен (удалить можно только всю запись);

«Логический» (Yes/No) — может принимать одно из двух значений: Да или Нет, Истина или Ложь, Вкл или Выкл (размер 1 бит);

«Поле объекта OLE» (OLE Object) — объекты (например, документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звуки и другие двоичные данные), созданные в других программах, использующих протокол OLE. Объекты могут быть связанными или внедренными в таблицу Microsoft Access;

«Гиперссылка» (Hyperlink) — поле для хранения гиперссылок. Гиперссылка может иметь вид пути UNC, либо URL-адреса. Адрес гиперссылки может состоять максимум из четырех частей, каждая из которых может содержать до 2048 символов;

«Мастер подстановок» (Lookup Wizard) — создает поле, позволяющее выбрать значение из другой таблицы или из списка значений, используя поле со списком. При выборе данного параметра в списке типов данных запускается мастер для автоматического определения этого поля.

4. В столбце «Описание» ввести пояснение назначения поля (не обязательно).

5. В нижней части конструктора расположены две вкладки: «Общие» — для задания различных свойств поля, «Подстановка» — для организации подстановки значений в поле из списка или из другой таблицы. На вкладке «Общие» необходимо уточнить свойства поля:

«Размер поля» — ввести или выбрать из списка размер поля;

«Формат поля» — ввести или выбрать из списка, для отображения данных в постоянном формате. Например, если свойство «Формат поля» для полей типа «Дата/время» установлено на «Краткий формат даты», то все вводимые данные будут отображаться в следующем формате: 25.02.04. Если же пользователь базы данных введет число в виде 25-фев-04 (или в другом допустимом виде), то при сохранении записи формат даты будет преобразован в «Краткий формат даты»;

«Число десятичных знаков» — ввести или выбрать из списка (для числовых типов данных);

«Маска ввода» — нажмите кнопку «Построителя …», чтобы запустить «Мастер масок ввода», и следуйте инструкциям диалоговых окон мастера. «Маска ввода» используется для форматирования данных и управления вводимыми значениями. В основном маски ввода используются в текстовых полях и полях даты/времени, а также в числовых и денежных;

«Значение по умолчанию» — можно ввести значение (при необходимости), которое будет присваиваться полю каждый раз при вводе новой записи;

«Условие на значение» — можно задать условие, которому должно удовлетворять вводимое значение поля, например, срок хранения продуктов задается положительным числом, тогда «Условие на значение» будет иметь вид: «> 0»;

«Сообщение об ошибке»* — сообщение, выдаваемое при нарушении условия на значение поля для условия из предыдущего примера, сообщение об ошибке будет: «Срок хранения не может быть отрицательным!»;

Обязательное поле «Да/Нет»:

«Нет» — допустимость неопределенных значений (Null-значений);

«Да» — недопустимость неопределенных значений;

«Индексированное поле» — индекс служит для ускорения поиска и сортировки полей, но замедляет обновление. Могут быть установлены значения:

«Нет» — нет индекса;

«Да (Допускаются совпадения)» — индексированное поле, разрешены дублирующие значения поля в разных записях;

«Да (Совпадения не допускаются)» — индексированное поле, значения поля уникальны для всей таблицы.

6. Повторить пункты 2—5 для создания всех полей таблицы.

---

* Свойства поля «Условие на значение» и «Сообщение об ошибке» можно использовать для задания семантической целостности БД

7. Установить первичный ключ таблицы. Для этого выделить нужное поле (или несколько полей для составного первичного ключа) и нажать соответствующую кнопку на панели инструментов.

8. Сохранить таблицу и выйти из режима конструктора.

В соответствии с разработанной реляционной моделью (см. рис. 10.2), необходимо создать все таблицы (отношения), входящие в эту модель. При построении реляционной модели подробно рассматривалась, из каких атрибутов состоит каждое отношение, какие типы данных должны иметь атрибуты, какие атрибуты являются первичным ключом отношения, а также допустимость Null-значений для атрибутов (таблицы «Поставщики», «Города»). Используя данные реляционной модели, создадим в режиме конструктора таблицы «Продукты» (рис. 12.2), «Поставщики», «Поставки» (рис. 12.3), «Заказы», «Продажи», «Города».

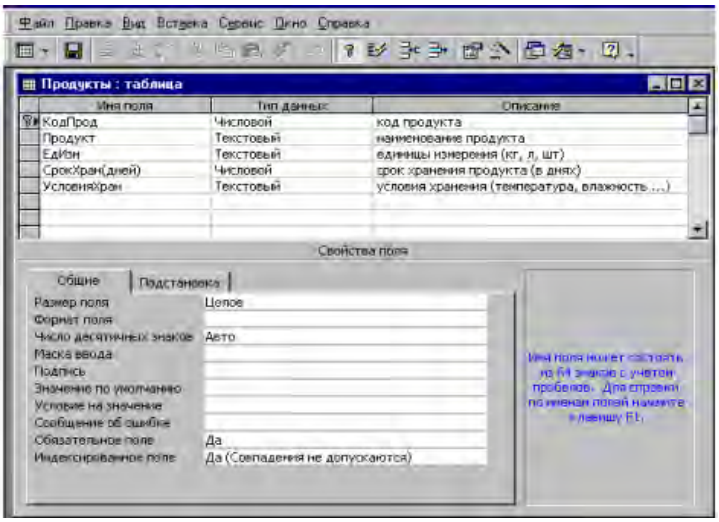

Рис. 12.2

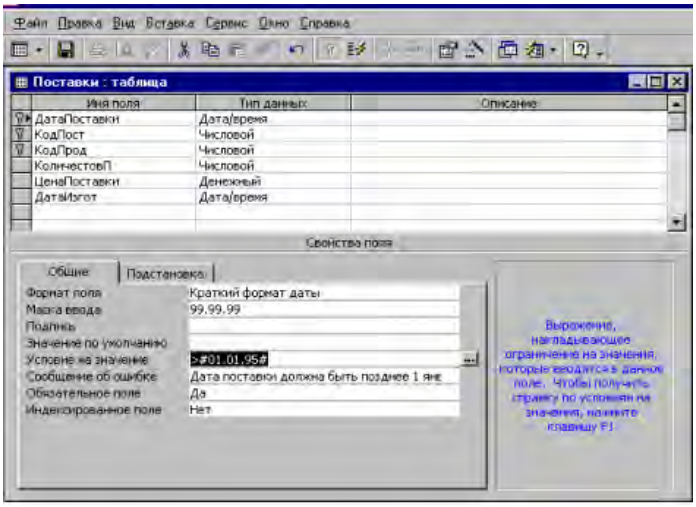

Рис. 12.3

После того как все таблицы БД созданы, необходимо установить связи между таблицами и задать ограничения ссылочной целостности в окне схемы данных.

12.2. Построение схемы данных. Задание ограничений целостности

Для построения схемы данных необходимо:

1. Открыть окно схемы данных, нажав кнопку на панели инструментов  (или выбрать пункты меню «Сервис», «Схема данных»).

2. Открыть окно для добавления таблиц на схему данных  (рис. 12.4), нажав кнопку на панели инструментов (или выбрать пункты меню «Связи», «Добавить таблицу»).

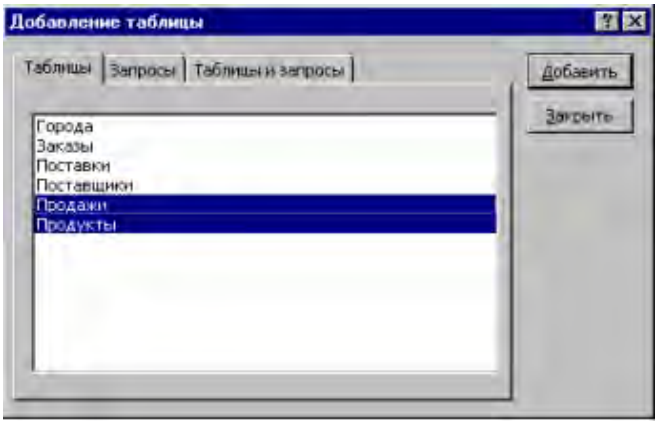

Рис. 12.4

3. Добавить таблицы на схему данных, выделив нужные таблицы в списке таблиц и нажав кнопку «Добавить», закрыть окно «Добавление таблицы».

4. Установить связи и правила ссылочной целостности между таблицами.

Связь устанавливается между полями таблиц (между первичным ключом основной таблицы и внешним ключом подчиненной таблицы). Для этого перетащите мышкой поле первичного ключа из основной таблицы на поле внешнего ключа подчиненной таблицы (или наоборот внешний ключ на первичный), в результате будет открыто окно изменения связей (рис. 12.5).

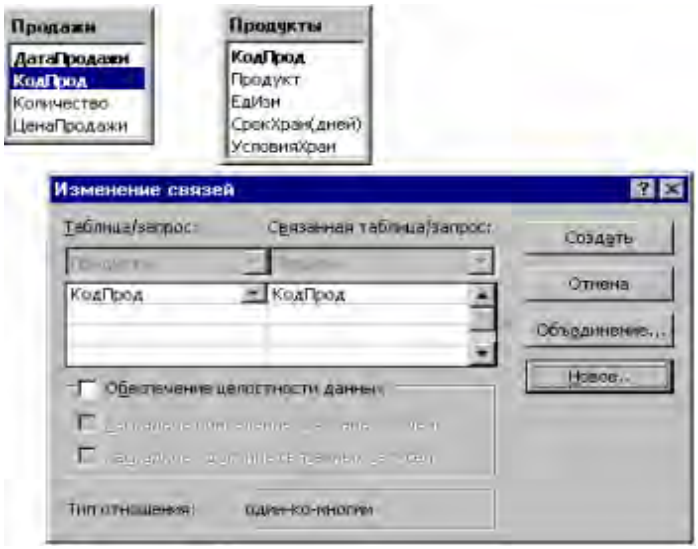

Рис. 12.5

Задайте правила ссылочной целостности путем установки соответствующих флажков:

обеспечение целостности данных;

каскадное обновление связанных полей;

каскадное удаление связанных полей.

Для обеспечения режима «Ограничения» обновления (и/или удаления) необходимо установить флаг «Обеспечение целостности данных».

Для обеспечения режима «Каскадирование» обновления (и/или удаления) необходимо установить флаги:

«Обеспечение целостности данных»;

«Каскадное обновление (и/или удаление) связанных полей».

Для подтверждения установленных режимов целостности и создания связи между таблицами нажмите кнопку «ОК» (рис. 12.6)

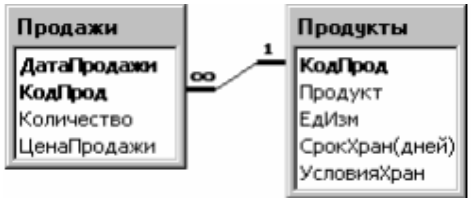

Рис. 12.6

5. Повторить пункты 2—4 для создания связей между остальными таблицами.

Теперь необходимо рассмотреть, какие правила ссылочной целостности необходимо установить в БД «Магазин» (табл. 12.2), реляционная модель которой приведена на рис. 22 и строим окончательную схему данных БД «Магазин» (рис. 12.7).

Таблица 12.2

Связь между таблицами	Обновление	Удаление
Продукты – Продажи	Каскадировать	Каскадировать
Продукты – Поставки	Каскадировать	Ограничить
Продукты – Заказы	Каскадировать	Каскадировать
Поставщики – Поставки	Каскадировать	Ограничить
Поставщики – Заказы	Каскадировать	Каскадировать
Города – Поставщики	Каскадировать	Ограничить

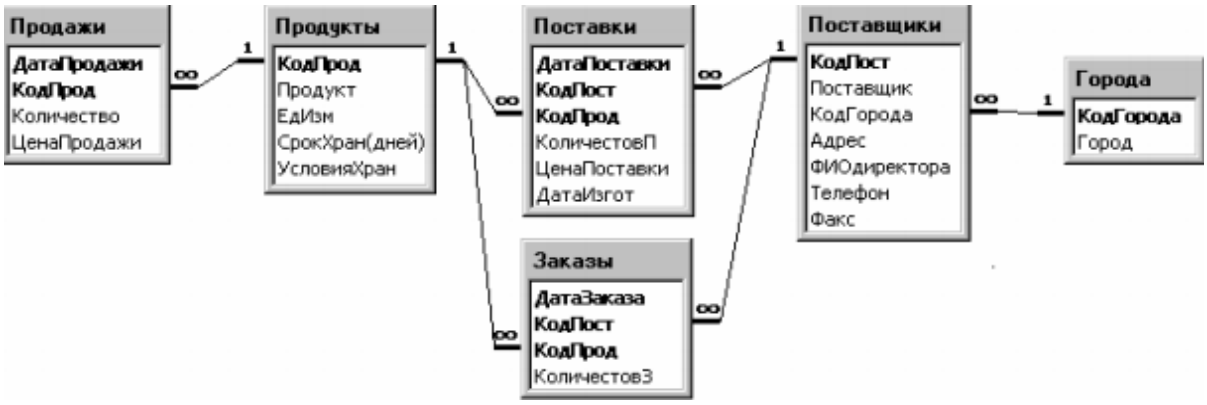

Рис. 12.7

13. ЯЗЫКИ ОПИСАНИЯ ЗАПРОСОВ

13.1. Табличный язык запросов QBE

Хранимые в базе данные можно обрабатывать вручную, последовательно просматривая и редактируя данные в таблицах, с помощью имеющихся в СУБД соответствующих средств. Для выполнения эффективности применяют запросы, позволяющие производить множественную обработку данных, то есть одновременно вводить, редактировать и удалять множество записей, а также выбирать данные из таблиц.

Запрос представляет собой специальным образом описанное требование, определяющие состав производимых над базой данных операций по выборке, удалению или модификации хранимых данных. Для подготовки запросов с помощью различных СУБД чаще всего используются два основных языка описания запросов: язык QBE (Query By Example) — язык запросов по образцу; язык SQL (Structured Query Language) — структурированный язык запросов. Главное отличие между ними — способ формирования запросов: язык QBE предлагает ручное или визуальное формирование запроса, в то время как использование SQL означает программирование запроса.

Теоретической основой языка QBE является реляционное исчисление с переменными — доменами. Язык QBE позволяет задавать сложные запросы к БД путем заполнения предлагаемой СУБД запросной формы. Такой способ задания запросов обеспечивает высокую наглядность и не требует указания алгоритма выполнения операции — достаточно описать образец ожидаемого результата. В каждой из современной реляционной СУБД имеется свой вариант языка QBE.

В языке QBE используется визуальный подход для организации доступа к информации в базе данных, основанный на применении шаблонов запросов. Применение QBE осуществляется путем задания образцов значений в шаблоне запроса, предусматривающем такой тип доступа к базе данных, который требуется в данный момент, например получение ответа на некоторый вопрос.

Язык QBE был разработан компанией IBM в 1970-х годах XX в. и предназначался для пользователей, заинтересованных в выборе информации из баз данных. Этот язык получил у пользователей столь широкое признание, что в настоящее время в той или иной мере он реализован практически во

всех популярных СУБД, включая и Microsoft Access. Средства поддержки языка QBE в СУБД Microsoft Access весьма просты в эксплуатации, но в то же время представляют пользователям достаточно широкий спектр возможностей работы с данными. Средства языка QBE могут использоваться для ввода запросов к информации, сохраняемой в одной или нескольких таблицах, а так же для определения набора полей, которые должны присутствовать в результирующей таблице. Отбор записей может проводится по конкретному или общему критерию и предусматривать выполнение необходимых вычислений на основе информации, сохраняемой в таблицах. Кроме того, средства языка QBE можно использовать для выполнения различных операций над таблицами, например, для вставки и удаления записей, модификации значений полей или создания новых полей и таблиц.

На языке QBE можно задавать запросы однотабличные и многотабличные. С помощью запросов на языке QBE выполняются следующие основные операции:

выборка данных;

вычисление над данными;

вставка новых записей;

удаление записей;

модификация (изменение) данных.

Результатом выполнения запроса является новая таблица, называемая ответной, или обновленная исходная таблица.

Выборка, вставка, удаление и модификация могут производиться безусловно или в соответствии с условиями, задаваемыми с помощью логических выражений. Вычисления над данными задаются с помощью арифметических выражений, в ответ появляются новые поля, называемые вычисляемыми.

Запросная форма имеет вид таблицы, имя и названия полей которой совпадают с именем и названиями полей соответствующей исходной таблицы. Чтобы узнать имена доступных таблиц БД, в языке QBE предусмотрен запрос на выборку имен таблиц. Названия полей исходной таблицы могут вводиться в шаблон вручную или автоматически. Во втором случае используется запрос на выборку заголовков столбцов.

Чаще всего используется тип запросов, который принято называть запросами на выборку. Они позволяют просматривать, анализировать и вносить изменения в данные, сохраняемые в одной или нескольких таблицах. При выполнении запроса на выборку СУБД Microsoft Access помещает выбранные данные в динамический набор данных, представляющий собой обновляемый набор записей, зависящий от таблицы или запроса и рассматриваемый как отдельный объект. Исключением являются лишь запросы, использующие специфические возможности языка SQL, которые отсутствуют в языке QBE. В современных СУБД (например, в Access и Visual FoxPro) многие действия по подготовке запросов с помощью языка QBE выполняются визуально с помощью мыши. В частности, визуальное связывание таблиц при подготовке запроса выполняется не элементами примеров, а просто протаскиванием мышью поля одной таблицы к полю другой.

13.2. Язык SQL: основные понятия и компоненты

СУБД Microsoft Access при создании запроса с использованием средств QBE неявно формирует эквивалентный оператор языка SQL, предназначенный для выполнения указанных действий.

SQL часто называют языком эсперанто для систем управления базами данных. Действительно, в мире нет другого языка для работы с базами данных, который бы настолько широко использовался в программах. Первый стандарт SQL появился в 1986 г. и к настоящему времени завоевал всеобщее признание. Его можно использовать даже при работе с нереляционными СУБД. В отличие от других программных средств (языки Си и Кобол), которые являются прерогативой программистов-профессионалов, SQL применяется специалистами из самых разных областей. Язык SQL стал фактически стандартным языком доступа к базам данных. Нужно заметить, что в настоящее время ни одна система не реализует стандарт SQL в полном объеме. Кроме того, во всех диалектах языка имеются нестандартные возможности. Таким образом, можно сказать, что каждый диалект — это надмножество некоторого подмножества стандарта SQL, что затрудняет переносимость приложений, разработанных для одних СУБД в другие СУБД.

SQL символизирует собой Структурированный Язык Запросов. Запросы — вероятно наиболее часто используемый аспект SQL.

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории, например, вместо «отношений» используются «таблицы», вместо «кортежей» — «строки», вместо «атрибутов» — «колонки» или «столбцы».

Стандарт языка SQL хотя и основан на реляционной теории, но во многих местах отходит он нее. Например, отношение в реляционной модели данных не допускает наличия одинаковых кортежей, а таблицы в терминологии SQL могут иметь одинаковые строки. Имеются и другие отличия.

SQL является примером языка преобразования данных. Он предназначен для работы с таблицами для преобразования входных данных к требуемому выходному виду. Язык SQL, который определен стандартом ISO, имеет два основных компонента:

1. Язык DDL (Data Definition Language), предназначенный для определения структур базы данных и управления доступом к данным.

2. Язык DML (Data Manipulation Language), предназначенный для выборки и обновления данных.

Язык SQL относительно прост в изучении. Это непроцедурный язык, поэтому в нем необходимо указывать, какая информация должна быть получена, а не как ее можно получить. Иначе говоря, язык SQL не требует указания методов доступа к данным. Как и большинство современных языков, SQL поддерживает свободный формат записи операторов, т. е. при вводе отдельные элементы операторов не связаны с фиксированными позициями на экране. Структура команд задается набором ключевых слов, представляющих собой обычные слова английского языка, такие как CREATE TABLE (Создать таблицу).

В архитектуре «Клиент — Сервер» язык SQL занимает очень важное место. Именно он используется как язык общения клиентского программного обеспечения с серверной СУБД, расположенной на удаленном компьютере. Так, клиент посылает серверу запрос на языке SQL, а сервер интерпретирует его, выполняет запрос и отсылает клиенту результат.

13.3. Основные операторы SQL

Оператор SQL состоит из зарезервированных слов, а также из слов, определяемых пользователем. Зарезервированные слова являются постоянной частью языка SQL и имеют определенное значение. Их следует записывать именно так, как указано в стандарте, и нельзя разбивать на части для переноса из одной строки в другую. Слова, определяемые пользователем, задаются самим пользователем (в соответствии с определенными синтаксическими правилами) и представляют собой имена различных объектов базы данных — таблиц, столбцов, представлений, индексов и т. д. Слова в операторе размещаются в соответствии с установленными синтаксическими правилами. Хотя в стандарте это не указано, многие диалекты языка SQL требуют задания в конце оператора некоторого символа, обозначающего окончание его текста; как правило, с этой целью используется точка с запятой (;).

Большинство компонентов операторов SQL не чувствительны к регистру. Это означает, что могут использоваться любые буквы — как строчные, так и прописные. Одним важным исключением из этого правила являются символьные литералы — данные, которые должны вводиться точно так же, как были введены соответствующие им значения, хранящиеся в базе данных. Например, если в базе данных хранится значение фамилии «SMITH1», а в условии поиска указан символьный литерал «Smith1», то эта запись не будет найдена.

Поскольку язык SQL имеет свободный формат, отдельные операторы SQL и их последовательности будут иметь более удобный для чтения вид при использовании отступов и выравнивания. Рекомендуется придерживаться следующих правил:

1. Каждая конструкция в операторе должна начинаться с новой строки.
2. Начало каждой конструкции должно быть обозначено таким же отступом, что и начало других конструкций оператора.
3. Если конструкция состоит из нескольких частей, каждая из них должна начинаться с новой строки с некоторым отступом относительно начала конструкции, что будет указывать на их подчиненность.
4. Для определения формата операторов SQL следует применять следующую расширенную форму системы обозначений BNF (Backus Naur Form — форма Бэкуса — Наура).
5. Прописные буквы используются для записи зарезервированных слов и должны указываться в операторах точно так же, как это будет показано.
6. Строчные буквы используются для записи слов, определяемых пользователем.

7. Вертикальная черта указывает на необходимость выбора одного из нескольких приведенных значений, например a | b | c.

8. Фигурные скобки определяют обязательный элемент, например {a}.

9. Квадратные скобки определяют необязательный элемент, например [a].

10. Многоточие (...) используется для указания необязательной возможности повторения конструкции от нуля до нескольких раз, например {a b}, [c...]. Эта запись означает, что после a или b может следовать от нуля до нескольких повторений c, разделенных запятыми.

На практике для определения структуры базы данных (в основном ее таблиц) используются операторы DDL, а для заполнения этих таблиц данными и выборки из них информации с помощью запросов — операторы DML.

Основные категории команд языка SQL:

DDL — язык определения данных;

DML — язык манипулирования данными;

DQL — язык запросов;

DCL — язык управления данными;

команды администрирования данных;

команды управления транзакциями.

Чаще всего возникает задача построения запросов на извлечение данных. Для этих целей используется SQL-оператор SELECT. Синтаксис оператора SELECT имеет следующий вид:

SELECT[ALL|DISTINCT](<Список полей>|*);

FROM <Список таблиц>;

[WHERE <Предикат-условие выборки или соединения>];

[GROUP BY <Список полей результата>];

[HAVING <Предикат-условие для группы>];

[ORDER BY <Список полей, по которым упорядочить вывод>].

Предикат — языковое выражение, обозначающее какое-либо свойство или отношение.

Например, чтобы вывести список преподавателей необходимо создать запрос на выборку (рис. 13.1, 13.2)

SELECT Фамилия, Имя, Отчество

FROM Преподаватели

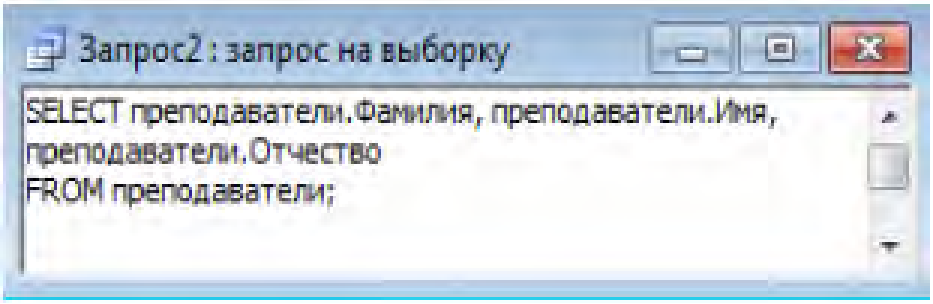

Рис. 13.1

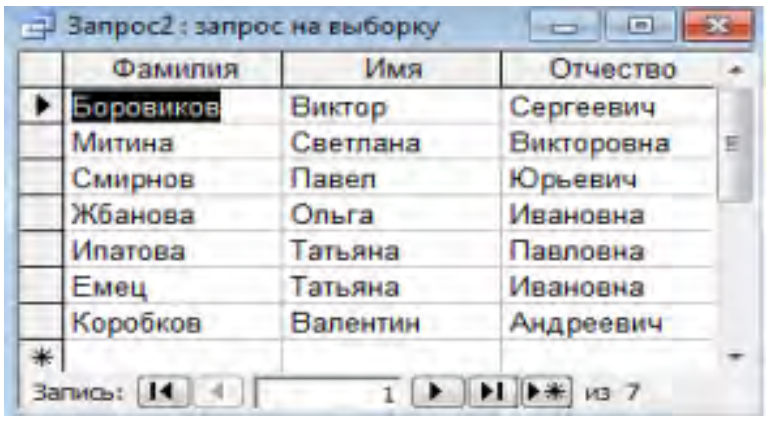

Рис. 13.2

Чтобы узнать дату рождения студента Иванова, необходимо задать запрос (рис. 13.3):

```
SELECT студенты.Фамилия, студенты.[дата рождения]
FROM студенты
WHERE (((студенты.Фамилия)="Иванов"));
```

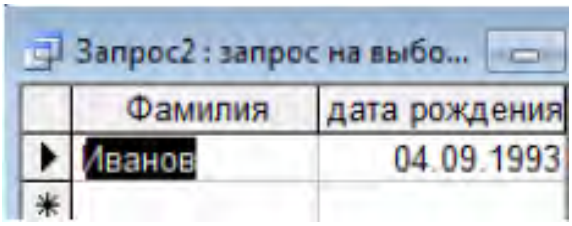

Рис. 13.3

В параметрических запросах всегда используют значение параметра, которое запрашивается во время ввода (рис. 13.4).

```
SELECT *
FROM студенты
WHERE (((студенты[Номер группы])=[укажите номер группы]));
```

Рис. 13.4

14. ФАЙЛОВЫЕ СТРУКТУРЫ, ИСПОЛЬЗУЕМЫЕ ДЛЯ ХРАНЕНИЯ ИНФОРМАЦИИ В БАЗАХ ДАННЫХ

14.1. Классификация файлов и файловых структур

В каждой СУБД по-разному организованы хранение и доступ к данным, однако существуют некоторые файловые структуры, которые имеют общепринятые способы организации и широко применяются практически во всех СУБД.

Классификация в системах баз данных файлов и файловых структур, которые используются для хранения информации во внешней памяти, приведена на рис. 14.1.

Рис. 14.1

С точки зрения пользователя, файлом называется поименованная линейная последовательность записей, расположенных на внешних носителях. Так как *файл — это линейная последовательность записей*, то всегда в файле можно определить текущую запись, предшествующую ей и следующую за

ней. Всегда существует понятие первой и последней записи файла. В соответствии с методами управления доступом различают устройства внешней памяти с произвольной адресацией (магнитные и оптические диски) и устройства с последовательной адресацией (магнитофоны, стримеры).

На устройствах с произвольной адресацией теоретически возможна установка головок чтения-записи в произвольное место мгновенно. Практически существует время позиционирования головки, которое весьма мало по сравнению со временем считывания-записи. В устройствах с последовательным доступом для получения доступа к некоторому элементу требуется «перемотать (пройти)» все предшествующие ему элементы информации. На устройствах с последовательным доступом вся память рассматривается как линейная последовательность информационных элементов.

14.2. Файлы прямого и последовательного доступа

Файлы с постоянной длиной записи, расположенные на устройствах прямого доступа (УПД), являются *файлами прямого доступа*.

В этих файлах физический адрес расположения нужной записи может быть вычислен по номеру записи (NZ). Каждая файловая система СУФ — система управления файлами поддерживает некоторую иерархическую файловую структуру, включающую чаще всего неограниченное количество уровней иерархии в представлении внешней памяти.

Для каждого файла в системе хранится следующая информация:

имя файла;

тип файла (например, расширение или другие характеристики);

размер записи;

количество занятых физических блоков;

базовый начальный адрес;

ссылка на сегмент расширения;

способ доступа (код защиты).

Для файлов с постоянной длиной записи адрес размещения записи с номером K может быть вычислен по формуле

$$BA + (K - 1)LZ + 1, \tag{14.1}$$

где BA — базовый адрес; LZ — длина записи.

Таким образом, если всегда можно определить адрес, на который необходимо позиционировать механизм считывания записи, то устройства прямого доступа делают это практически мгновенно, поэтому для таких файлов чтение произвольной записи практически не зависит от ее номера. Файлы прямого доступа обеспечивают наиболее быстрый доступ к произвольным записям, их использование считается наиболее перспективным в системах баз данных.

На устройствах последовательного доступа могут быть организованы файлы только последовательного доступа. Файлы с переменной длиной записи всегда являются файлами последовательного доступа. Они могут быть организованы двумя способами:

1. Конец записи отличается специальным маркером.

Запись 1	X	Запись 2	X	Запись 3	X

2. В начале каждой записи записывается ее длина.

LZ1	Запись!	LZ2	Запись2	LZ3	Запись 3

Здесь *LZN* — длина *N*-й записи.

14.3. Доступ по ключу. Хеширование

Файлы с прямым доступом обеспечивают наиболее быстрый (по номеру записи в базах) способ доступа. Не всегда можно хранить информацию в виде файлов прямого доступа, чаще всего в базах данных необходим поиск по первичному или возможному ключам, иногда необходима выборка по внешним ключам, но во всех этих случаях известно значение ключа, но не известен номер записи, который соответствует этому ключу.

При организации файлов прямого доступа в некоторых очень редких случаях возможно построение функции, которая по значению ключа однозначно вычисляет адрес (номер записи файла):

$$NZ = F(K), \tag{14.2}$$

где *NZ* — номер записи; *K* — значение ключа; *F*( ) — функция.

Функция *F*() при этом должна быть линейной, чтобы обеспечивать однозначное соответствие. Однако часто бывает, что значения ключей разбросаны по нескольким диапазонам. В этом случае не удается построить взаимнооднозначную функцию, либо эта функция будет иметь множество незадействованных значений, которые соответствуют недопустимым значениям ключа. В подобных случаях применяют различные методы хеширования (рандомизации) и создают специальные хеш-функции. *Хеширование есть разбиение множества ключей (однозначно характеризующих элементы хранения и представленных, как правило, в виде текстовых строк или чисел) на непересекающиеся подмножества (наборы элементов), обладающие определенными свойствами.*

Суть методов хеширования состоит в том, что берется значения ключа или некоторые его характеристики) и используется для начала поиска, то есть вычисляется некоторая хэш-функция *h*(*k*), а полученное значение используется в качестве адреса начала поиска. То есть не требуется полного взаимнооднозначного соответствия, но, с другой стороны, для повышения скорости ограничивается время этого поиска (количество дополнительных шагов) для окончательного получения адреса. Таким образом, допускается, что нескольким разным ключам может соответствовать одно значение хэш-функции (то есть один адрес). Подобные ситуации называются *коллизиями*. Значения ключей, которые имеют одно и то же значение хэш-функции, называются *синонимами*.

Поэтому при использовании хеширования как метода доступа необходимо принять два независимых решения:

1) выбрать хэш-функцию;

2) выбрать метод разрешения коллизий.

14.4. Стратегии разрешения коллизий

Существует множество различных стратегий разрешения коллизий, но для примера достаточно рассмотреть две достаточно распространенные.

1. Стратегия разрешения коллизий с областью переполнения. Эта стратегия условно может быть названа стратегией с областью переполнения. При выборе этой стратегии область хранения разбивается на две части:

основную область;

область переполнения.

Для каждой новой записи вычисляется значение хеш-функции, которое определяет адрес ее расположения, после чего запись заносится в основную область в соответствии с полученным значением хэш-функции. Если вновь заносимая запись имеет значение функции хеширования такое же, которое использовала другая запись, уже имеющаяся в БД, то новая запись заносится в область переполнения на первое свободное место, а в записи-синониме, которая находится в основной области, делается ссылка на адрес вновь размещенной записи в области переполнения. Если же уже существует ссылка в записи-синониме, которая расположена в основной области, то тогда новая запись получает дополнительную информацию в виде ссылки и уже в таком виде заносится в область переполнения.

При этом цепочка синонимов не разрывается, но мы не просматриваем ее до конца, чтобы расположить новую запись в конце цепочки синонимов, а всегда располагаем новую запись на второе место в цепочке синонимов, что существенно сокращает время размещения новой записи. Теперь оно составляет не более двух обращений к диску, учитывая, что номер первой свободной записи в области переполнения хранится в виде системной переменной.

При поиске произвольной записи также сначала вычисляется значение ее хэш-функции и считывается первая запись в цепочке синонимов, которая расположена в основной области. Если искомая запись не соответствует первой в цепочке синонимов, то далее поиск происходит перемещением по этой цепочке, пока не будет обнаружена требуемая запись. Скорость поиска зависит от длины цепочки синонимов, поэтому качество хэш-функции определяется ее максимальной длиной. Хорошим результатом может считаться наличие не более 10 синонимов в цепочке.

При удалении произвольной записи сначала определяется ее место расположения. Если удаляемой является первая запись в цепочке синонимов, то после удаления на ее место в основной области заносится вторая (следующая) запись в цепочке синонимов, при этом все указатели (ссылки на синонимы) сохраняются.

Если же удаляемая запись находится в середине цепочки синонимов, то необходимо провести корректировку указателей: в записи, предшествующей удаляемой, в цепочке ставится указатель из удаляемой записи. Если это последняя запись в цепочке, то все равно механизм изменения указателей такой же, то есть в предшествующую запись заносится признак отсутствия следующей записи в цепочке, который ранее хранился в последней записи.

2. Организация стратегии свободного замещения. При этой стратегии файловое пространство не разделяется на области, но для каждой записи добавляется два указателя: на предыдущую запись в цепочке синонимов и на следующую запись в цепочке синонимов. Отсутствие соответствующей ссылки обозначается специальным символом (например, нулем). Для каждой новой записи вычисляется значение хэш-функции, и если данный адрес свободен, то запись попадает на заданное место и становится первой в цепочке синонимов. Если адрес, соответствующий полученному значению хэш-функции, занят, то по наличию ссылок определяется, является ли запись, расположенная по указанному адресу, первой в цепочке синонимов. Если да, то новая запись располагается на первом свободном месте и для нее устанавливаются соответствующие ссылки: она становится второй в цепочке синонимов, на нее ссылается первая запись, а она ссылается на следующую, если таковая есть. Если запись, которая занимает требуемое место, не является первой записью в цепочке синонимов, значит, она занимает данное место «незаконно» и при появлении «законного владельца» должна быть «выселена», т. е. перемещена на новое место. Механизм перемещения аналогичен занесению новой записи, которая уже имеет синоним, занесенный в файл. Для этой записи ищется первое свободное место, и корректируются соответствующие ссылки: в записи, которая является предыдущей в цепочке синонимов для перемещаемой записи, заносится указатель на новое место перемещаемой записи, указатели же в самой перемещаемой записи остаются прежние.

После перемещения «незаконной» записи вновь вносимая запись занимает свое законное место и становится первой записью в новой цепочке синонимов. Механизмы удаления записей во многом аналогичны механизмам удаления в стратегии с областью переполнения. Однако еще раз кратко опишем их. Если удаляемая запись является первой записью в цепочке синонимов, то после удаления на ее место перемещается следующая (вторая) запись из цепочки синонимов и проводится соответствующая корректировка указателя третьей записи в цепочке синонимов, если таковая существует.

Если же удаляется запись, которая находится в середине цепочки синонимов, то производится только корректировка указателей: в предшествующей записи указатель на удаляемую запись заменяется указателем на следующую за удаляемой запись, а в записи, следующей за удаляемой, указатель на предыдущую запись заменяется на указатель на запись, предшествующую удаляемой.

15. ПЕРСПЕКТИВЫ РАЗВИТИЯ БД И СУБД

Современные БД являются основой многочисленных информационных систем. Информация, накопленная в них, является чрезвычайно ценным материалом, и в настоящий момент широко распространяются способы обработки баз данных с точки зрения извлечения из них дополнительных знаний, методов, связанных с обобщением и различными дополнительными приемами обработки данных. БД (Хранилища данных, Data Warehouse) в данной концепции выступают как хранилища информации.

Для работы с Хранилищами данных наиболее значимым становится так называемый интеллектуальный анализ данных (ИАД, Data Mining) — процесс выявления значимых корреляций, образцов и тенденций в больших объемах данных. Учитывая высокие темпы роста объемов накопленной в современных хранилищах данных информации, невозможно недооценить роль ИАД. По мнению специалистов Gartner Group, уже в 1998 г. ИАД вошел в десятку важнейших информационных технологий. В последние годы началось активное внедрение технологии ИАД. Ее активно используют как крупные корпорации, так и более мелкие фирмы, которые серьезно относятся к вопросам анализа и прогнозирования своей деятельности. Естественно, на рынке программных продуктов стали появляться соответствующие инструментальные средства.

Особенно широко методы ИАД применяются в бизнес-приложениях аналитиками и руководителями компаний. Для этих категорий пользователей разрабатываются инструментальные средства высокого уровня, позволяющие решать достаточно сложные практические задачи без специальной математической подготовки. Актуальность использования ИАД в бизнесе связана с жесткой конкуренцией, возникшей вследствие перехода от «рынка продавца» к «рынку покупателя». В этих условиях особенно важно качество и обоснованность принимаемых решений, что требует строгого количественного анализа имеющихся данных. При работе с большими объемами накапливаемой информации необходимо постоянно оперативно отслеживать динамику рынка, что практически невозможно без автоматизации аналитической деятельности.

В бизнес-приложениях наибольший интерес представляет интеграция методов интеллектуального анализа данных с технологией оперативной аналитической обработки данных (On-Line Analytical Processing, OLAP). OLAP использует многомерное представление агрегированных данных для быстрого доступа к важной информации и дальнейшего ее анализа. Системы OLAP обеспечивают аналитикам и руководителям быстрый последовательный интерактивный доступ к внутренней структуре данных и возможность преобразования исходных данных с тем, чтобы они позволяли отразить структуру системы нужным для пользователя способом. Кроме того, OLAP-системы позволяют просматривать данные и выявлять имеющиеся в них закономерности либо визуально, либо простейшими методами (такими, как линейная регрессия), а включение в их арсенал нейросетевых методов обеспечивает существенное расширение аналитических возможностей.

В основе концепции оперативной аналитической обработки OLAP лежит многомерное представление данных. Термин OLAP ввел Кодд (E. F. Codd) в 1993 г. Он рассмотрел недостатки реляционной модели, в которой невозможно «объединять, просматривать и анализировать данные с точки зрения множественности измерений, то есть самым понятным для корпоративных аналитиков способом», и определил общие требования к системам OLAP, расширяющим функциональность реляционных СУБД и включающим многомерный анализ как одну из своих характеристик.

По Кодду, многомерное концептуальное представление (multi-dimensional conceptual view) является наиболее естественным взглядом управляющего персонала на объект управления. Оно представляет собой множественную перспективу, состоящую из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям данных определяется как многомерный анализ. Каждое измерение включает направления консолидации данных, состоящие из серии последовательных уровней обобщения, где каждый вышестоящий уровень соответствует большей степени агрегации данных по соответствующему измерению. Так, измерение «Исполнитель» может определяться направлением консолидации, состоящим из уровней обобщения «предприятие — подразделение — отдел — служащий». Измерение «Время» может даже включать два направления консолидации — «год — квартал — месяц — день» и «неделя — день», поскольку счет времени по месяцам и по неделям несовместим. В этом случае становится возможным произвольный выбор желаемого уровня детализации информации по каждому из измерений. Операция спуска (Drilling Down) соответствует движению от высших ступеней консолидации к низшим; напротив, операция подъема (Rolling Up) означает движение от низших уровней к высшим.

Следующим новым направлением в развитии систем управления базами данных является направление, связанное с отказом от нормализации отношений. Во многом нормализация отношений нарушает естественные иерархи-

ческие связи между объектами, которые достаточно распространены в нашем мире. Возможность сохранять их на концептуальном (но не на физическом) уровне позволяет пользователям более естественно отражать семантику предметной области. В настоящий момент уже существует теоретическое обоснование работы с ненормализованными отношениями и практические реализации подобных систем.

Дальнейшим расширением в структурных преобразованиях являются объектно-ориентированные базы данных. В объектно-ориентированной парадигме предметная область моделируется как множество классов взаимодействующих объектов. Каждый объект характеризуется набором свойств, которые являются как бы его пассивными характеристиками и набором методов работы с этим объектом. Атрибуты объекта могут принимать определенное множество допустимых значений, набор конкретных значений атрибутов объекта определяет его состояние. Используя методы работы с объектом, можно изменять значение его атрибутов и тем самым как бы изменять состояние самого объекта. Множество объектов с одним и тем же набором атрибутов и методов образует класс объектов. Объект должен принадлежать только одному классу (если не учитывать возможности наследования). Допускается наличие примитивных предопределенных классов, объекты-экземпляры которых не имеют атрибутов: целые, строки и т. д. Класс, объекты которого могут служить значениями атрибута объектов другого класса, называется *доменом* этого атрибута.

Одной из наиболее перспективных черт объектно-ориентированной парадигмы является принцип наследования. Допускается порождение нового класса на основе уже существующего, и этот процесс называется наследованием. В этом случае новый класс, называемый подклассом существующего класса (суперкласса), наследует все атрибуты и методы суперкласса. В подклассе могут быть определены дополнительные атрибуты и методы. Различаются случаи простого и множественного наследования. В первом случае подкласс может определяться только на основе одного суперкласса, во втором случае суперклассов может быть несколько. Если в языке или системе поддерживается единичное наследование, набор классов образует древовидную иерархию. При поддержании множественного наследования классы связаны в ориентированный граф с корнем, называемый решеткой классов. Объект подкласса считается принадлежащим любому суперклассу этого класса.

Можно считать, что наиболее важным качеством объектно-ориентированной базы данных (ООБД), которое позволяет реализовать объектно-ориентированный подход, является учет поведенческого аспекта объектов.

В прикладных информационных системах, основывавшихся на БД с традиционной организацией (вплоть до тех, которые базировались на семантических моделях данных), существовал принципиальный разрыв между структурной и поведенческой частями. Структурная часть системы поддерживалась всем аппаратом БД, ее можно было моделировать, верифицировать и т. д., а поведенческая часть создавалась изолированно. В частности, отсут-

ствовали формальный аппарат и системная поддержка совместного моделирования и гарантий согласованности структурной (статической) и поведенческой (динамической) частей. В среде ООБД проектирование, разработка и сопровождение прикладной системы становятся процессом, в котором интегрируются структурный и поведенческий аспекты. Конечно, для этого нужны специальные языки, позволяющие определять объекты и создавать на их основе прикладную систему.

Специфика применения объектно-ориентированного подхода для организации и управления БД потребовала уточненного толкования классических концепций и некоторого их расширения. Прежде всего, возникло направление, которое предполагает возможность хранения объектов внутри реляционной БД, тогда дополнительно необходимо предусмотреть хранение и использование специфических методов работы с этими объектами, а это в свою очередь требует расширения стандарта языка SQL. Частично это уже сделано в новом стандарте SQL3, однако там далеко не все вопросы получили однозначное разрешение.

Однако часть разработчиков придерживается мнения о необходимости полного отказа от реляционной парадигмы и перехода на объектно-ориентированную парадигму. Для перехода к объектно-ориентированным БД стандарт объектного проектирования был дополнен стандартизованными средствами доступа к базам данных (стандарт ODMG93).

Следующим направлением развития баз данных является появление так называемых темпоральных баз данных, то есть баз данных, чувствительных ко времени. Фактически БД моделирует состояние объектов предметной области в некоторый текущий момент времени. Однако в ряде прикладных областей необходимо исследовать именно изменение состояний объектов во времени. Если использовать чисто реляционную модель, то требуется строить и хранить дополнительно множество отношений, имеющих одинаковые схемы, отличающиеся временем существования или снятия данных. Гораздо перспективнее и удобнее для этого использовать специальные механизмы снятия срезов по времени для определенных объектов БД. Основной тезис темпоральных систем состоит в том, что для любого объекта данных, созданного в момент времени $t1$ и уничтоженного в момент времени $t2$, в БД сохраняются (и доступны пользователям) все его состояния во временном интервале ($t1$, $t2$). При обозначении интервала квадратные скобки означают, что граница интервала включена в него, а круглые скобки означают, что точка на временной оси, соответствующая границе интервала, не включается в интервал. И действительно, если объект уничтожен в момент времени $t2$, то в этой точке временной оси он уже не существует, поэтому мы оставляем правую границу временного интервала открытой.

Еще одним из перспективных направлений развития баз данных является направление, связанное с объединением технологии экспертных систем и баз данных и развитие так называемых дедуктивных баз данных. Эти базы основаны на выявлении новых знаний из баз данных не путем запросов или

аналитической обработки, а путем использования правил вывода и построения цепочек применения этих правил для вывода ответов на запросы. Для этих баз данных существуют языки запросов, отличные от классического SQL. В экспертных системах также знания экспертов хранятся в форме правил, чаще всего используются так называемые продукционные правила типа «если описание ситуации, то описание действия». Хранение подобных правил и организация вывода на основании имеющихся фактов под силу современным СУБД.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

1. Выбрать вариант задания в соответствии с номером в журнале группы. Разработать реляционную базу данных из нескольких взаимосвязанных таблиц, с описанием предметной области и требованиями пользователей к информации.

2. Уточнив и дополнив заданную предметную область, выявить необходимый набор сущностей, определить требуемый набор атрибутов для каждой сущности, определить связи между объектами.

3. Создать структуры таблиц с ключевыми полями. Хотя бы одна таблица должна содержать составной ключ. Индексированное поле выбирается в качестве ключа лишь в том случае, когда никакое другое не подходит в качестве ключа. Все таблицы должны быть приведены к третьей нормальной форме — 3НФ. Заполнить таблицы данными.

4. Количество данных в таблицах должно обеспечивать выдачу не менее 3—5 записей по каждому запросу задания. База данных должна содержать 4—5 таблиц с обязательными полями и 1—3 добавленными самостоятельно (если это возможно). Установить связи между таблицами.

5. Создать формы для ввода информации в удобном для пользователя формате.

6. Создать запросы на выборку в соответствии с заданием. Создать параметрический запрос. Создать запросы на обновление и удаление. Создать перекрестный запрос. Создать запрос для создания отчета.

7. Создать простой отчет и отчет на основе ранее созданного запроса.

8. Создать кнопочную форму для работы со всеми созданными ранее объектами базы данных (таблицы, формы, запросы, отчеты). Предусмотреть в форме выход из базы данных.

Вариант 1. Страховая компания

Описание предметной области. Вы работаете в страховой компании. Вашей задачей является отслеживание финансовой деятельности компании. Компания имеет различные филиалы по всей стране. Каждый филиал характеризуется названием, адресом и телефоном. Деятельность компании организована следующим образом: к вам обращаются различные лица с целью

заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков, договор заключается по определенному виду страхования (например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора вы фиксируете дату заключения, страховую сумму, вид страхования, тарифную ставку и филиал, в котором заключался договор.

Обязательные таблицы:

Договоры (Номер договора, Дата заключения, Страховая сумма, Тарифная ставка, и др.).

Вид страхования (Вид страхования, Наименование и др).

Филиал (Наименование филиала, Адрес, Телефон).

Развитие постановки задачи. Нужно учесть, что договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон), нужно еще хранить филиал, в котором работают агенты. Кроме того, исходя из базы данных, нужно иметь возможность рассчитывать заработную плату агентам. Заработная плата составляет некоторый процент от страхового платежа (страховой платеж — это страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор. Внести в структуру таблиц изменения, учитывающие эти факты. Добавить запросы. Добавить новые отношения.

Вариант 2. Туристическая фирма

Описание предметной области. Вы работаете в туристической компании. Ваша компания работает с клиентами, продавая им путевки. Вашей задачей является отслеживание финансовой стороны деятельности фирмы. Работа с клиентами в компании организована следующим образом: у каждого клиента, пришедшего к вам, собираются некоторые стандартные данные — фамилия, имя, отчество, адрес, телефон. После этого сотрудники выясняют у клиента, куда он хотел бы поехать отдыхать. При этом ему демонстрируются различные варианты, включающие страну проживания, особенности местного климата, имеющиеся отели разного класса. Наряду с этим обсуждается возможная длительность пребывания и стоимость путевки. В случае если удалось договориться и найти для клиента приемлемый вариант, вы регистрируете факт продажи путевки (или путевок, если клиент покупает сразу несколько), фиксируя дату отправления. Иногда вы решаете предоставить клиенту некоторую скидку.

Обязательные таблицы:

Маршруты (Название маршрута, Страна, Климат, Длительность, Отель, Стоимость и т. д.).

Путевки (Дата отправления, Продолжительность, Скидка и т. д.).

Клиенты (Фамилия, Имя, Отчество, Адрес, Телефон и т. д.).

Развитие постановки задачи. Фирма работает с несколькими отелями в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля. Скидки, которые предоставляет фирма, фиксированы. Например, при покупке более одной путевки, предоставляется скидка 5 %. Скидки могут суммироваться. Внести в структуру таблиц изменения, учитывающие эти факты. Добавить запросы. Добавить новые отношения.

Вариант 3. Прокат автомобилей

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

Обязательные таблицы:

Автомобили (Марка, Стоимость, Стоимость проката, Тип и т. д.).

Клиенты (Фамилия, Имя, Отчество, Адрес, Телефон и т. д.).

Выданные автомобили (Дата выдачи, Дата возврата и т. д.).

Развитие постановки задачи. Стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от времени его проката, а также от года выпуска. Нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов. Внести в структуру таблиц изменения, учитывающие эти факты. Добавить запросы. Добавить новые отношения.

Вариант 4. Платная поликлиника

Описание предметной области. Вы являетесь руководителем службы планирования платной поликлиники. Вашей задачей является отслеживание финансовых показателей работы поликлиники. В поликлинике работают врачи различных специальностей, имеющие разную квалификацию. Каждый день в поликлинику обращаются больные. Все больные проходят обязательную регистрацию, при которой в базу данных заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения). Каждый больной может обращаться в поликлинику несколько раз, нуждаясь в различной медицинской помощи. Все обращения больных фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается дата обращения.

Обязательные таблицы:

Врачи (Фамилия, Имя, Отчество, Специальность, Категория и т. д.).

Пациенты (Фамилия, Имя, Отчество, Год рождения и т. д.).

Обращения (Дата обращения, Диагноз, Стоимость).

Развитие постановки задачи. При обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Общая стоимость лечения зависит от стоимости тех консультаций и процедур, которые назначены пациенту. Кроме того, для определенных категорий граждан предусмотрены скидки. Внести в структуру таблиц изменения, учитывающие эти факты. Добавить запросы. Добавить новые отношения.

Вариант 5. Гостиница

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

Обязательные таблицы:

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

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

Поселение (Дата поселения, Дата освобождения, Примечание).

Развитие постановки задачи. Необходимо не только хранить информацию по факту сдачи номера клиенту, но и осуществлять бронирование номеров. Для постоянных клиентов, а также для определенных категорий клиентов предусмотрена система скидок. Скидки могут суммироваться. Внести в структуру таблиц изменения, учитывающие этот факт, и изменить существующие запросы. Добавить новые запросы.

Вариант 6. Ломбард

Описание предметной области. Вы работаете в ломбарде. Вашей задачей является отслеживание финансовой стороны его работы. Деятельность компании организована следующим образом: к вам обращаются различные лица с целью получения денежных средств под залог определенных товаров, у каждого из приходящих к вам клиентов вы запрашиваете фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости прине-

сенного в качестве залога товара вы определяете сумму, которую готовы выдать на руки клиенту, свои комиссионные, а также определяете срок возврата денег. Если клиент согласен, то ваши договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается у вас. В случае если в указанный срок не происходит возврата денег, товар переходит в вашу собственность.

Обязательные таблицы:

Клиенты (Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Название, Примечание).

Сдача в ломбард (Описание товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

Развитие постановки задачи. После перехода прав собственности на товар ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке. (Например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы.) Помимо текущей, нужно хранить все возможные значения цены для данного товара. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 7. Реализация готовой продукции

Описание предметной области. Вы работаете в компании, занимающейся оптово-розничной продажей различных товаров. Вашей задачей является отслеживание финансовой стороны ее работы. Деятельность компании организована следующим образом: компания торгует товарами разного вида. Каждый из этих товаров характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В компанию обращаются покупатели. Для каждого из них вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с покупателем количество купленного им товара и дату покупки.

Обязательные таблицы:

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

Покупатели (Телефон, Контактное лицо, Адрес).

Сделки (Дата сделки, Количество, Признак оптовой продажи).

Развитие постановки задачи. Теперь ситуация изменилась. Выяснилось, что обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 8. Бюро по трудоустройству

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

Обязательные таблицы:

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

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

Сделки (Код работодателя, Должность, Комиссионные).

Развитие постановки задачи. Оказалось, что база данных не совсем точно описывает работу бюро. В базе фиксируется только сделка, а информация по открытым вакансиям не хранится. Кроме того, для автоматического поиска вариантов необходимо вести справочник «Виды деятельности». Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 9. Нотариальная контора

Описание предметной области. Вы работаете в нотариальной конторе. Вашей задачей является отслеживание финансовой стороны работы компании. Деятельность нотариальной конторы организована следующим образом: фирма готова предоставить клиенту определенный комплекс услуг. Для наведения порядка вы формализовали эти услуги, составив их список с описанием каждой услуги. При обращении к вам клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ. В документе указываются услуга, сумма сделки, комиссионные (доход конторы), описание сделки.

Обязательные таблицы:

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

Сделки (Сумма, Комиссионные, Описание).

Услуги (Название, Описание).

Развитие постановки задачи. Теперь ситуация изменилась. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Также компания предоставляет в рамках одной сделки

различные виды скидок, которые могут суммироваться. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 10. Курсы повышения квалификации

Описание предметной области. Вы работаете в учебном заведении и занимаетесь организацией курсов повышения квалификации. В вашем распоряжении имеются сведения о сформированных в зависимости от специальности и отделения группах студентов. В каждую из них включено определенное количество студентов. Проведение занятий обеспечивает штат преподавателей, для каждого из которых в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки вы получаете информацию о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Также хранятся сведения о типе проводимых занятий (лекции, практика), предмете и оплате за час.

Обязательные таблицы:

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

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

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

Развитие постановки задачи. В результате работы с базой данных выяснилось, что размер почасовой оплаты зависит от предмета и типа занятия. Каждый преподаватель может вести не все предметы, а только некоторые. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 11. Фирма по продаже запчастей

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

Обязательные таблицы:

Поставщики (Название, Адрес, Телефон).

Детали (Название, Артикул, Цена, Примечание).

Поставки (Количество, Дата).

Развитие постановки задачи. Теперь ситуация изменилась. Выяснилось, что цена детали может меняться от поставки к поставке. Поставщики заранее ставят вас в известность о дате изменения цены и о ее новом значении. Нужно хранить не только текущее значение цены, но и всю историю изменения цен. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 12. Определение факультативов для студентов

Описание предметной области. Вы работаете в высшем учебном заведении и занимаетесь организацией факультативов. В вашем распоряжении имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Преподаватели кафедры должны обеспечить проведение факультативных занятий по некоторым предметам. По каждому факультативу установлены определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами появляется информация о том, на какие факультативы записался каждый из них. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра вы заносите информацию об оценках, полученных студентами на экзаменах.

Обязательные таблицы:

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

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

Учебный план (Оценка, Количество часов).

Развитие постановки задачи. Теперь ситуация изменилась. Выяснилось, что некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 13. Распределение учебной нагрузки

Описание предметной области. Вы работаете в высшем учебном заведении и занимаетесь распределением нагрузки между преподавателями кафедры. В вашем распоряжении имеются сведения о преподавателях кафедры, включающие анкетные данные, информацию об их ученой степени, занимаемой административной должности и стаже работы. Преподаватели должны обеспечить проведение занятий по некоторым предметам. По каждому из них установлено определенное количество часов. В результате распределения нагрузки у вас должна получиться информация следующего рода: «Такой-то преподаватель проводит занятия по такому-то предмету с такой-то группой».

Обязательные таблицы:

Преподаватели (Фамилия, Имя, Отчество, Ученая степень, Должность, Стаж).

Предметы (Название, Курс, на котором читается).

Нагрузка (Номер группы, Количество часов).

Развитие постановки задачи. Теперь ситуация изменилась. Выяснилось, что все проводимые занятия делятся на лекционные и практические. По каждому виду занятий устанавливается свое количество часов. Кроме того, данные о нагрузке нужно хранить несколько лет. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 14. Грузовые перевозки

Описание предметной области. Вы работаете в компании, занимающейся перевозками грузов. Вашей задачей является отслеживание стоимости перевозок с учетом заработной платы водителей. Компания осуществляет перевозки по различным маршрутам. Для каждого маршрута определено некоторое название, вычислено примерное расстояние и установлена некоторая оплата для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов хранится полная информация о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия.

Обязательные таблицы:

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

Водители (Фамилия, Имя, Отчество, Стаж).

Проделанная работа (Дата отправки, Дата возвращения, Премия).

Развитие постановки задачи. Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему оплаты. Так, оплата водителям теперь должна зависеть не только от маршрута, но и от стажа. Кроме того, перевозку могут осуществлять два водителя. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 15. Учет телефонных переговоров

Описание предметной области. Вы работаете в коммерческой службе телефонной компании, предоставляющей абонентам телефонные линии для междугородних переговоров. Вашей задачей является отслеживание стоимости междугородних телефонных переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

Обязательные таблицы:

Абоненты (Номер телефона, ИНН, Адрес).

Города (Название, Тариф дневной, Тариф ночной).

Переговоры (Дата, Количество минут, Время суток).

Развитие постановки задачи. Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему скидок. Так, стоимость минуты теперь уменьшается в зависимости от длительности разговора, размер скидки для каждого города разный. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 16. Библиотека

Описание предметной области. Вы являетесь руководителем библиотеки. Библиотека решила зарабатывать деньги, выдавая напрокат некоторые книги, имеющиеся в небольшом количестве. Вашей задачей является отслеживание финансовых показателей работы. У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги определяется для каждой из них залоговая стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). В библиотеку обращаются читатели, которые регистрируются в картотеке, содержащей стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку несколько раз. Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата.

Обязательные таблицы:

Книги (Название, Автор, Залоговая стоимость, Стоимость проката, Жанр).

Читатели (Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные книги (Дата выдачи, Дата возврата).

Развитие постановки задачи. Теперь ситуация изменилась. Несложный анализ показал, что стоимость проката книги должна зависеть не только от самой книги, но и от срока ее проката. Кроме того, необходимо добавить систему штрафов за вред, нанесенный книге, и систему скидок для некоторых категорий читателей. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 17. Ведение заказов

Описание предметной области. Вы работаете в компании, занимающейся оптовой продажей различных товаров. Вашей задачей является отслеживание финансовой стороны ее работы. Деятельность компании организована следующим образом: компания торгует товарами определенного вида. Каждый из этих товаров характеризуется ценой, справочной информацией и при-

знаком наличия или отсутствия доставки. В компанию обращаются заказчики, для каждого из которых вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с заказчиком количество купленного им товара и дату покупки.

Обязательные таблицы:

Товары (Цена, Доставка, Описание).

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

Заказы (Количество, Дата).

Развитие постановки задачи. Теперь ситуация изменилась. Выяснилось, что доставка разных товаров может производиться способами, различными по цене и скорости. Нужно хранить информацию о том, какими способами может осуществляться доставка каждого товара, и какой вид доставки (а соответственно, какую стоимость доставки) выбрал клиент при заключении сделки. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 18. Выдача банком кредитов

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

Обязательные таблицы:

Виды кредитов (Название, Условия получения, Ставка, Срок).

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

Кредиты (Сумма, Дата выдачи).

Развитие постановки задачи. Теперь ситуация изменилась. После проведения различных исследований выяснилось, что используемая система не позволяет отслеживать динамику возврата кредитов. Для устранения этого недостатка вы приняли решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 19. Учет телекомпанией стоимости прошедшей в эфире рекламы

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

Обязательные таблицы:

Передачи (Название, Рейтинг, Стоимость минуты).

Реклама (Дата, Длительность в минутах).

Заказчики (Название, Банковские реквизиты, Телефон, Контактное лицо).

Развитие постановки задачи. В результате эксплуатации базы данных выяснилось, что необходимо также хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 20. Занятость актеров театра

Описание предметной области. Вы являетесь коммерческим директором театра, в ваши обязанности входит вся организационно-финансовая работа, связанная с привлечением актеров и заключением контрактов. Вы организовали дело следующим образом: каждый год театр осуществляет постановку различных спектаклей. Каждый спектакль имеет определенный бюджет. Для участия в конкретных постановках в определенных ролях привлекаются актеры. С каждым из актеров вы заключаете персональный контракт на определенную сумму. Каждый из актеров имеет некоторый стаж работы, некоторые из них удостоены различных наград и званий.

Обязательные таблицы:

Актеры (Фамилия, Имя, Отчество, Звание, Стаж).

Спектакли (Название, Год постановки, Бюджет).

Занятость актеров в спектакле (Роль, Стоимость годового контракта).

Развитие постановки задачи. В результате эксплуатации базы данных выяснилось, что в рамках одного спектакля на одну и ту же роль привлекается несколько актеров. Контракт определяет базовую зарплату актера, а по итогам реально отыгранных спектаклей актеру назначается премия. Кроме того, в базе данных нужно хранить информацию за несколько лет. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 21. Техническое обслуживание станков

Описание предметной области. Ваше предприятие занимается ремонтом станков и другого промышленного оборудования. Вашей задачей является отслеживание финансовой стороны деятельности предприятия. Клиентами компании являются промышленные предприятия, оснащенные различным сложным оборудованием. В случае поломок оборудования они обращаются к вам. Ремонтные работы организованы следующим образом: все станки проклассифицированы по странам-производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта вы фиксируете вид станка и дату начала ремонта.

Обязательные таблицы:

Виды станков (Вид станка, Страна, Год выпуска, Марка).

Виды ремонта (Название, Продолжительность, Стоимость, Примечания).

Ремонт (Дата начала, Примечания).

Развитие постановки задачи. Теперь ситуация изменилась. Несложный анализ показал, что нужно не просто подразделять станки по видам, а иметь информацию о том, сколько раз ремонтировался тот или иной конкретный станок. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 22. Учет телефонных переговоров

Описание предметной области. Вы работаете в коммерческой службе телефонной компании, которая предоставляет абонентам телефонные линии для междугородних переговоров. Вашей задачей является отслеживание стоимости междугородних телефонных переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

Обязательные таблицы:

Абоненты (Номер телефона, ИНН, Адрес).

Города (Название, Тариф дневной, Тариф ночной).

Переговоры (Дата, Количество минут, Время суток).

Развитие постановки задачи. Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему скидок. Так, стоимость минуты теперь уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 23. Учет внутриофисных расходов

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

Обязательные таблицы:

Отделы (Название, Количество сотрудников).

Виды расходов (Название, Описание, Предельная норма).

Расходы (Сумма, Дата).

Развитие постановки задачи. Теперь ситуация изменилась. Оказалось, что нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 24. Инвестирование свободных средств

Описание предметной области. Вы являетесь руководителем аналитического центра инвестиционной компании, занимающейся вложением денежных средств в ценные бумаги. Ваши клиенты — предприятия, которые доверяют управлять их свободными денежными средствами на определенный период. Вам необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и компании, и клиенту. При работе с клиентом для вас весьма существенной является информация о предприятии: название, вид собственности, адрес и телефон.

Обязательные таблицы:

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

Инвестиции (Котировка, Дата покупки, Дата продажи).

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

Развитие постановки задачи. При эксплуатации базы данных стало понятно, что необходимо хранить историю котировок каждой ценной бумаги. Также помимо вложений в ценные бумаги, существует возможность вкладывать деньги в банковские депозиты. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 25. Ювелирная мастерская

Описание предметной области. Вы работаете в ювелирной мастерской, осуществляющей изготовление ювелирных изделий для частных лиц на заказ. Работа производится с определенными материалами (платина, золото, серебро, различные драгоценные камни и т. д.). При обращении потенциального клиента вы определяете, какое именно изделие ему необходимо. Все изготавливаемые изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), выполнены из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы).

Обязательные таблицы:

Изделия (Название, Тип, Код материала, Вес, Цена).

Материалы (Название, Цена за грамм).

Продажи (Дата продажи, Фамилия покупателя, Имя покупателя, Отчество покупателя).

Развитие постановки задачи. В процессе опытной эксплуатации базы данных выяснилось, что ювелирное изделие может состоять из нескольких материалов. Постоянным клиентам мастерская предоставляет скидки. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 26. Интернет-магазин

Описание предметной области. Вы являетесь сотрудником коммерческого отдела компании, продающей различные товары через Интернет. Вашей задачей является отслеживание финансовой составляющей ее работы. Работа компании организована следующим образом: на Интернет-сайте представлены (выставлены на продажу) некоторые товары. Каждый из них имеет некоторое название, цену и единицу измерения (штуки, килограммы, литры). Для проведения исследований и оптимизации работы магазина вы пытаетесь собирать данные с клиентов. При этом для вас определяющее значение имеют стандартные анкетные данные, а также телефон и адрес электронной почты для связи. В случае приобретения товаров на сумму свыше 5000 р. клиент переходит в категорию постоянных и получает скидку на каждую покупку в размере 2 %. По каждому факту продажи вы автоматически фиксируете клиента, товары, количество, дату продажи, дату доставки.

Обязательные таблицы:

Товары (Название, Цена, Единица измерения).

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

Продажи (Дата продажи, Дата доставки, Количество).

Развитие постановки задачи. В результате эксплуатации базы данных выяснилось, что иногда возникают проблемы, связанные с нехваткой информации о наличии нужных товаров на складе в нужном количестве. Обычно

клиенты в рамках одного заказа покупают не один вид товара, а несколько. Исходя из суммарной стоимости заказа, компания предоставляет дополнительные скидки. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 27. Распределение дополнительных обязанностей

Описание предметной области. Вы работаете в коммерческой компании и занимаетесь распределением дополнительных разовых работ. Вашей задачей является отслеживание хода их выполнения. Компания имеет определенный штат сотрудников, каждый из которых получает определенный оклад. Время от времени возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных должностных обязанностей сотрудников. Для наведения порядка в этой сфере деятельности вы проклассифицировали все виды дополнительных работ, определив сумму оплаты по факту их выполнения. При возникновении дополнительной работы определенного вида вы назначаете ответственного, фиксируя дату начала. По факту окончания фиксируется дата и выплачивается дополнительная сумма к зарплате с учетом классификации.

Обязательные таблицы:

Сотрудники (Фамилия, Имя, Отчество, Оклад).

Виды работ (Название вида, Описание, Оплата за день).

Работы (Дата начала, Дата окончания).

Развитие постановки задачи. Теперь ситуация изменилась. Выяснилось, что некоторые из дополнительных работ являются достаточно трудоемкими и срочными, что требует привлечения к их выполнению нескольких сотрудников. Также оказалось, что длительность работ в каждом конкретном случае различна. Соответственно, нужно заранее планировать длительность работы и количество сотрудников, занятых ее выполнением. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 28. Парикмахерская

Описание предметной области. Вы работаете в парикмахерской, обслуживающей клиентов в соответствии с их пожеланиями и некоторым каталогом различных видов стрижки. Так, для каждой стрижки определены название, принадлежность полу (мужская, женская), стоимость работы. Для наведения порядка, по мере возможности, составляется база данных клиентов, запоминаются их анкетные данные (фамилия, имя, отчество). Начиная с пятой стрижки клиент переходит в категорию постоянных и получает скидку в 3 % при каждой последующей стрижке. После окончания очередной работы документом фиксируются стрижка, клиент и дата производства работ.

Обязательные таблицы:

Стрижки (Название, Пол, Стоимость).

Клиенты (Фамилия, Имя, Отчество, Пол, Признак постоянного клиента).

Работа (Код клиента, Дата).

Развитие постановки задачи. Теперь ситуация изменилась. У парикмахерской появился филиал, и вы хотели бы видеть раздельную статистику по филиалам. Кроме того, стоимость стрижки может меняться с течением времени. Нужно хранить не только последнюю цену, но и все данные по изменению цены. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 29. Химчистка

Описание предметной области. Вы работаете в химчистке, осуществляющей прием у населения вещей для выведения пятен. Для наведения порядка вы составляете базу данных клиентов, запоминая их анкетные данные (фамилия, имя, отчество). Начиная с третьего обращения клиент переходит в категорию постоянных и получает скидку в 3 % при чистке каждой последующей вещи. Все оказываемые услуги подразделяются на виды, имеющие название, тип и стоимость, зависящую от сложности работ. Работа с клиентом первоначально состоит в определении объема работ, вида услуги и ее стоимости. Если клиент согласен, он оставляет вещь (при этом фиксируются услуга, клиент и дата приема) и забирает ее после обработки (при этом фиксируется дата возврата).

Обязательные таблицы:

Виды услуг (Название, Тип, Стоимость).

Клиенты (Фамилия, Имя, Отчество, Признак постоянного клиента).

Услуги (Дата приема, Дата возврата).

Развитие постановки задачи. Теперь ситуация изменилась. У химчистки появился филиал, и вы хотели бы видеть раздельную статистику по филиалам. Кроме того, вы решили делать надбавки за срочность и сложность работ. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Вариант 30. Сдача в аренду торговых площадей

Описание предметной области. Вы работаете в крупном торговом центре, сдающем в аренду коммерсантам свои торговые площади. Вашей задачей является наведение порядка в финансовой сфере работы торгового центра. Работа торгового центра построена следующим образом: в результате планирования определяется некоторое количество торговых точек в пределах здания, которые могут сдаваться в аренду. Важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. Со всех потенциальных клиентов вы собираете стандартные данные (название, адрес,

телефон, реквизиты, контактное лицо). Каждому потенциальному клиенту показываются имеющиеся свободные площади. При достижении соглашения оформляется договор, фиксируется в базе данных торговая точка, клиент, период (срок) аренды.

Обязательные таблицы:

Торговые точки (Название торговой точки, Этаж, Площадь, Наличие кондиционера, Стоимость аренды в день).

Клиенты (Название, Реквизиты, Адрес, Телефон, Контактное лицо).

Аренда (Дата начала, Дата окончания).

Развитие постановки задачи. В результате эксплуатации базы данных выяснилось, что некоторые клиенты арендуют сразу несколько торговых точек. Помимо этого, необходимо собирать информацию о ежемесячных платежах, поступающих от арендаторов. Внести в структуру таблиц поправки, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

СЛОВАРЬ СПЕЦИАЛИЗИРОВАННЫХ ТЕРМИНОВ

Администратор базы данных — лицо или группа лиц, отвечающих за выработку требований к базе данных, ее проектирование, создание, эффективное использование и сопровождение. В группу администрирования базы данных входят: системные и прикладные программисты, операторы и специалисты по техническому обслуживанию и др.

Актуализация базы данных — обновление (добавление, изменение или удаление) данных.

Аномалии актуализации — принятое в англоязычной литературе по базам данных название проблем обновления данных, вызванных избыточным дублированием данных. Устраняются путем нормализации реляционных таблиц.

Аномалия добавления — невозможность ввода одних данных из-за отсутствия других.

Аномалия модификации — изменение значения одного данного ведет к просмотру всей реляционной таблицы.

Аномалия удаления — потеря информации, напрямую не связанной с данными, которые удалены.

Атрибут — существенный признак.

Атрибут ключевой — атрибут, входящий в состав какого-либо ключа.

Атрибут неключевой — атрибут, который не входит в состав ключей.

Атрибут отношения — пара (*A, D*), где *A* — имя атрибута, а *D* — множество допустимых значений этого атрибута. Множество *D* называют доменом атрибута.

Атрибут сущности (связи) — конструктивный элемент ER-модели, предназначенный для задания свойства сущности (связи).

База данных — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области

База данных реляционная — конечная совокупность реляционных таблиц, объединенных смысловым содержанием, а также процедурами контроля целостности и обработки данных в интересах некоторых конечных пользователей.

Банк данных — разновидность информационной системы, в которой информационные средства представлены базой данных или системой баз данных, а программные средства содержат систему управления базами данных.

Банк знаний — автоматизированная система, содержащая различные виды знаний о предметной области.

Данные — информация, представленная в форме, удобной для хранения, передачи и переработки.

Декартово произведение — операция реляционной алгебры, применяемая к двум таблицам. В результате формируется таблица, строки которой получаются сцеплением каждой строки одной таблицы с каждой строкой другой таблицы.

Домен атрибута — множество допустимых значений атрибута.

Жизненный цикл базы данных — история развития базы данных. Делится на три этапа: проектирование, реализация, эксплуатация.

Запрос — специальным образом описанное обращение пользователя к базе данных, содержащее задание на получение или изменения информации, хранящейся в базе.

Защита данных — средства предотвращения несанкционированного доступа к базе данных.

Знания — это зафиксированная и проверенная практикой обработанная информация, которая использовалась и может многократно использоваться для принятия решений.

Избыточное дублирование данных — повторение данных в базе, вызванное наличием зависимостей между ними. При избыточном дублировании повторяются не просто данные, а информация о предметной области. Избыточное дублирование относится к нежелательным свойствам базы данных, в особенности для OLTP-систем. Оно ведет к увеличению объема памяти, необходимого для физического хранения базы данных, и времени обработки. Но самое неприятное последствие избыточного дублирования — появление аномалий актуализации. Устраняется путем нормализации таблиц.

Индекс — средство ускорения операции поиска строк в реляционной таблице, а также выполнения других операций, использующих поиск: извлечение, модификация и т. д.

Индексный файл — файл, в котором хранится информация индекса.

Инфологическая модель предметной области — семантическая структурная модель, отражающая классы однородных объектов, свойства объектов и связи между классами объектов предметной области.

Информационная система — система, включающая в себя информационные, программные, технические, языковые и организационные средства. Эти средства предназначены для сбора, накопления, передачи, поиска и обработки данных.

Класс принадлежности необязательный — возможны экземпляры сущности, свободные от связи.

Класс принадлежности обязательный — каждый экземпляр сущности участвует в связи.

Класс принадлежности сущности к связи — характеристика бинарной связи, определяющая обязательность или необязательность участия в связи экземпляров сущности.

Ключ реляционной таблицы — это столбец (может быть несколько столбцов), добавляемый к таблице и позволяющий установить связь с записями в другой таблице.

Ключ составной — ключ, состоящий из нескольких атрибутов.

Ключ сущности — минимальный набор атрибутов, значения которых уникальны для каждого экземпляра.

Ключ таблицы внешний — набор атрибутов одной (основной) таблицы, являющийся ключом другой (подчиненной) таблицы. Применяется для связывания реляционных таблиц.

Ключ естественный — основан на уже существующем поле.

Ключ суррогатный — основан на добавленном искусственным путем отдельном поле для однозначной идентификации.

Ключ интеллектуальный — основан на естественном ключе путем добавления дополнительного поля.

Конечные пользователи — группа лиц, в интересах которой создается база данных. В зависимости от особенностей создаваемой информационной системы круг ее пользователей может существенно различаться.

Кортеж отношения — элемент отношения, соответствующий строке реляционной таблицы.

Метаданные (данные о данных) — специальная компонента СУБД, предназначенная для централизованного хранения информации об организации базы данных (структурах и типах данных, кодах защиты и разграничениях доступа к данным со стороны пользователей, внешних носителях и физических адресах размещения данных в памяти системы и т. п.). Метаданные иногда называют словарем данных.

Модель — описание или аналогия, используемая для представления чего-либо, что не может быть рассмотрено непосредственно; упрощенные абстракции событий или условий реального мира.

Модель данных — абстракция, предназначенная для определения правил структурирования, процессов динамического изменения и допустимых состояний взаимосвязанных данных.

Модель данных иерархическая — теоретико-графовая модель данных. Основная структура данных — списковая структура типа корневого ориентированного графа (набора записей, связанных между собой указателями, при этом выделенная запись — корень дерева — является владельцем набора).

Модели данных классические — реляционная, сетевая, иерархическая модели данных.

Модели данных компоненты — основные структуры данных, множество операций над возможными структурами данных и ограничения целостности структур данных.

Модель данных многомерная — обобщение классической реляционной модели данных. Основная структура данных — многомерная таблица. Относится к теоретико-множественным моделям данных.

Модель данных объектно-ориентированная — теоретико-графовая модель данных. Данные структурируются подобно иерархической модели. Однако вершинами дерева выступают не записи, а объекты. Между объектами и функциями их обработки устанавливаются взаимосвязи.

Модель данных постреляционная — расширение классической реляционной модели данных. Снимает ограничение нормализованности таблиц. Допускает неопределенные и составные значения полей таблиц. Относится к теоретико-множественным моделям данных.

Модель данных реляционная — модель данных, основной структурой данных которой является реляционная таблица. Относится к теоретико-множественным моделям данных. Модель данных сетевая — теоретико-графовая модель данных. Основная структура данных — списковая структура типа ориентированного графа произвольного вида с выделенной вершиной.

Модель «сущность — связь» (ER-модель) — система условных обозначений для описания инфологической модели предметной области.

Независимость данных логическая — возможность изменения одного приложения без корректировки других приложений, работающих с той же базой данных.

Независимость данных от приложений — свойство базы данных, обеспечивающее устойчивость ее к развитию. Предполагает логическую и физическую независимость.

Независимость данных физическая — возможность переноса данных с одних носителей на другие без изменения приложений.

Неизбыточность базы данных — свойство, означающее, что в базе данных нет ничего лишнего, т. е. удаление из нее какого-либо элемента данных ведет к потере информации о предметной области.

Непротиворечивость базы данных — свойство базы данных, которое подразумевает, что все хранящиеся данные удовлетворяют определенным условиям — ограничениям целостности.

Непроцедурный язык — язык, обеспечивающий средства определения того, что требуется, а не того, как это получить.

Нормализация — процесс приведения базы данных к виду, в котором она будет соответствовать правилам нормальных форм

Нормальная форма — некоторый образец, которому должна соответствовать реляционная таблица во избежание аномалий актуализации.

Объединение — операция реляционной алгебры, создающая теоретико-множественное объединение двух таблиц, идентичных по структуре.

Ограничения целостности — условия, которым должны удовлетворять хранимые в базе данные.

Отношение — представление реляционной таблицы на языке математики: подмножество декартова произведения доменов атрибутов.

Первичный ключ реляционной таблицы — служит как ограничение целостности в рамках одной таблицы для однозначной идентификации, конкретно поле первичного ключа не может повторятся или быть пустым.

Пересечение — операция реляционной алгебры, создающая теоретико-множественное пересечение двух таблиц, идентичных по структуре.

Поддержка целостности базы данных — контроль целостности базы данных и восстановление базы в случае утраты ее целостного состояния.

Предметная область — часть реального мира, являющаяся объектом информационного представления в базе данных.

Приложение — программа или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи.

Приложения внешние — приложения, разработанные вне СУБД.

Приложения СУБД — приложения, разработанные с помощью языковых средств СУБД.

Проект базы данных логический — определение логической структуры (схемы) базы данных в терминах модели данных.

Проект базы данных физический — определение запоминающих устройств, методов доступа и индексов, которые будут использоваться в базе данных.

Проектирование базы данных — первый этап жизненного цикла базы данных, во время которого создаются семантические и синтаксические модели разной степени формализации: от словесного описания предметной области до определения структуры базы данных на формальном языке описания данных СУБД.

Проектирование базы данных даталогическое — выбор модели данных и СУБД, разработка логического и физического проектов (синтаксических моделей) базы данных.

Проектирование базы данных инфологическое — изучение, анализ и моделирование предметной области — создание семантических моделей базы данных.

Проекция — операция реляционной алгебры, создающая таблицу путем удаления некоторых столбцов из существующей таблицы.

Процедура — предписанная последовательность действий для осуществления какого-либо дела.

Процедурный язык — язык, обеспечивающий пошаговый способ описания решения задачи.

Разность — операция реляционной алгебры, создающая теоретико-множественную разность двух таблиц, идентичных по структуре.

Разработчик базы данных — категория пользователей, которые взаимодействуют с базой данной во время ее проектирования, создания и реорганизации. Группу разработчиков составляют: системные аналитики (анализируют и моделируют предметную область), проектировщики структур баз данных, прикладные программисты (создают приложения), администратор базы данных, администраторы приложений (координируют работу прикладных программистов при разработке приложений).

Реляционная алгебра — система запросов для реляционных баз данных. Составляет теоретическую основу реальных процедурных языков манипули-

рования данными. Операнды и результаты всех операций над ними — отношения. Отношение как ответ на запрос формируется по шагам на основе реляционного выражения.

Реляционное выражение — выражение реляционной алгебры.

Реляционное исчисление — система запросов для реляционных баз данных. Составляет теоретическую основу реальных непроцедурных языков манипулирования данными (например, SQL и QBE). Позволяет выражать запросы с помощью предикатов формальной логики. Запрос на языке реляционного исчисления содержит лишь сведения о том, что необходимо получить из базы данных, но не уточняется, как это сделать.

Реляционные операции — операции реляционной алгебры — операции над таблицами (отношениями).

Реляционная таблица — двумерная таблица, в которой представлены данные об однородных и различимых объектах или связях предметной области. Основная единица хранения данных в реляционной базе данных.

Связь — конструктивный элемент ER-модели, предназначенный для указания отношения между сущностями.

Связь бинарная — связь, отражающая отношение между двумя типами сущностей.

Связывание реляционных таблиц — средство реляционных СУБД, позволяющее установить явные связи (ссылки) между таблицами для их согласованного обновления.

Селекция — операция реляционной алгебры, выбирающая строки из таблицы на основании заданного условия.

Семантика — смысл, содержание, толкование. Семантика данных — смысл (информационное содержание) данных.

Сеть — совокупность компьютеров, объединенных средствами передачи данных.

Синтаксис — правила записи.

Система запросов — это формальная система выразительных средств, предназначенных для записи запросов к базе данных. Всякая система запросов — некоторый теоретический язык запросов.

Система обработки данных — информационная система, основной функцией которой является обработка данных. Среди них выделяют OLTP-системы и OLAP-системы.

Система управления базами данных — комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования баз данных.

Ссылочная целостность — условие, накладываемое на содержание взаимосвязанных реляционных таблиц: для каждого значения внешнего ключа основной таблицы всегда должна найтись строка подчиненной таблицы с таким же значением первичного ключа.

Структурная целостность — условие, подразумевающее, что реляционная СУБД может работать только с реляционными отношениями.

Соединение (естественное соединение) — операция реляционной алгебры, применяемая к двум таблицам. Строки результирующей таблицы получаются сцеплением только тех строк исходных таблиц, для которых имеет место равенство значений одноименных атрибутов.

Степень связи — характеристика бинарной связи, определяющая однократность или многократность участия в связи экземпляров сущностей.

Степень связи «многие-ко-многим» — один экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и, наоборот, один экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

Степень связи «один-к-одному» — один экземпляр одной сущности может быть связан только с одним экземпляром другой сущности.

Степень связи «один-ко-многим» — один экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, но один экземпляр второй сущности связан не более чем с одним экземпляром первой сущности.

Сущность — конструктивный элемент ER-модели, предназначенный для задания класса однородных и различимых между собой объектов предметной области. Классу соответствует тип сущности, конкретному объекту из класса — экземпляр сущности.

Схема отношения — множество имен атрибутов отношения.

Схема реляционной базы данных (схеме данных) — список, содержащий имена реляционных таблиц базы данных. Для каждой таблицы указываются имена атрибутов и множество ключей.

Трехуровневая схема представления метаданных — стандарт представления метаданных в современных СУБД. Обеспечивает независимость базы данных от приложений. Схема предложена американским национальным институтом стандартов ANSI.

Файл — именованный набор связанных записей.

Целостность базы данных — свойство, означающее, что база данных содержит полную, достоверную и непротиворечивую информацию о предметной области.

Целостность базы данных логическая — отсутствие логических ошибок в базе данных, т. е. все данные удовлетворяют заданным ограничениям целостности.

Целостность базы данных физическая — свойство базы данных, предполагающее, что данные не утрачены.

Целостность реляционной таблицы — свойство, означающее, что таблица имеет ключ и является нормализованной (все поля таблицы определены и атомарны, в таблице нет избыточного дублирования данных).

Язык манипулирования данными — совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными: ввод, модификацию и выборку.

ANSI (American National Standards Institute) — американский национальный институт стандартизации.

CASE-средство (Computer Aided Software Engineering) — инструментарий, обеспечивающий автоматизацию проектирования информационных систем.

ER-диаграмма (Entity-Relationship diagram) — объектно-связное представление предметной области в терминах ER-модели.

ER-модель (Entity-Relationship model) — модель «сущность — связь». Это система условных обозначений для описания инфологической модели предметной области.

OLTP (Online Transaction Processing) — системы оперативной обработки данных. Основными функции для них являются обновление и обработка данных.

OLAP (Online Analytical Processing) — системы оперативного анализа данных. Их основные функции: накопление, анализ и обработка данных. Базируются на многомерной модели данных.

QBE (Query-by-Example) — язык запросов по образцу. Применяется для построения запросов в режиме конструктора.

SQL (Structured Query Language) — структурированный язык запросов. Является стандартом языков манипулирования данными в реляционных СУБД.

Библиографический список

1. *Карпова, И. П.* Базы данных: учебное пособие / И. П. Карпова. — СПб. : Питер, 2013. — 240 с.

2. *Кузовкин, А. В.* Управление данными / А. В. Кузовкин, А. А. Цыганов, Б. А. Щукин. — М. : Академия, 2010. — 256 с.

3. *Ревунков, Г. И.* Базы и банки данных и знаний / Г. И. Ревунков, Э. Н. Самохвалов, В. В. Чистов. — М. : Высшая школа, 2002. — 367 с.

4. *Дейт, К. Дж.* Введение в системы баз данных / К. Дж. Дейт. — М. : Вильямс, 2005. — 1316 с.

5. *Михеева, Е. В.* Информационные технологии в профессиональной деятельности / Е. В. Михеева. — М. : Академия, 2008. — 379 с.

Учебное электронное издание

Катеринина Светлана Юрьевна
Усков Юрий Иванович

УПРАВЛЕНИЕ ДАННЫМИ

Учебное пособие

Начальник РИО *М. Л. Песчаная*
Редактор *И. Б. Чижикова*
Компьютерная правка и верстка *М. А. Денисова*

Минимальные систем. требования:
PC 486 DX-33; Microsoft Windows XP; Internet Explorer 6.0; Adobe Reader 6.0.

Подписано в свет 26.05.2015.
Гарнитура «Таймс». Уч.-изд. л. 7,3. Объем данных 4,8 Мбайт.

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Волгоградский государственный архитектурно-строительный университет»
Редакционно-издательский отдел
400074, Волгоград, ул. Академическая, 1
http://www.vgasu.ru, info@vgasu.ru