ВЫСШЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

А.В.КУЗИН, С.В.ЛЕВОНИСОВА

БАЗЫ ДАННЫХ

Допущено
Учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлению подготовки дипломированных специалистов
654600 «Информатика и вычислительная техника»

Москва
ACADEMIA
2005

УДК 681.3(075.8)
ББК 32.81я73
К89

Рецензенты:
директор Красноярского оптико-электронного колледжа, д-р пед. наук *В. М. Демин*;
ведущий научный сотрудник ВЦ РАН, проф., д-р техн. наук *С. К. Дулин*

Кузин А. В.

К89 Базы данных: Учеб. пособие для студ. высш. учеб. заведений / А. В. Кузин, С. В. Левонисова. — М.: Издательский центр «Академия», 2005. — 320 с.

ISBN 5-7695-1796-4

Рассмотрены базовые вопросы теории проектирования баз данных, использование СУБД Access для создания баз данных, особенности разработки пользовательских приложений на основе СУБД Microsoft Access, а также архитектура системы баз данных.

Для студентов высших учебных заведений, обучающихся по направлению «Информатика и вычислительная техника».

УДК 681.3(075.8)
ББК 32.81я73

ISBN 5-7695-1796-4

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

За последние годы в нашей стране произошли значительные перемены, которые не могли не затронуть области информатики и вычислительной техники. Всего каких-нибудь десять лет назад работа с базами данных и электронными таблицами была уделом профессиональных программистов.

Системы управления базами данных (СУБД) не были предназначены для широкого пользователя.

Их основным потребителем был военно-промышленный комплекс.

С появлением огромного числа банков, акционерных обществ и частных компаний ситуация резко изменилась.

В настоящее время обработка и хранение информации являются важнейшими задачами.

Потеря информации или ее несвоевременное получение могут обернуться потерей денег. Именно этими обстоятельствами можно объяснить столь быстрый рост компьютерной техники и стремительное развитие электронных таблиц и систем управления базами данных в нашей стране и за рубежом.

Для оперативного, гибкого и эффективного управления предприятиями, фирмами и организациями различных форм собственности, телекоммуникационными средствами гражданского и военного назначения, информационно-вычислительными, экологическими, радиолокационными и радионавигационными системами широко внедряются системы автоматизированного управления, ядром которых являются базы данных (БД). При большом объеме информации и сложности производимых с ней операций проблема эффективности средств организации хранения, доступа и обработки данных приобретает особое значение.

Важность и значимость баз данных в современной жизни определяют серьезные требования, предъявляемые к

квалификации специалистов, создающих приложения на их основе.

В предлагаемом учебном пособии, предназначенном для студентов, обучающихся по направлению «Информатика и вычислительная техника», рассматриваются базовые вопросы теории проектирования баз данных и особенности разработки пользовательских приложений на основе СУБД Microsoft Access.

ГЛАВА 1
ОСНОВЫ ТЕОРИИ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

1.1. Определение и назначение баз данных. Системы управления базами данных

С самого начала развития вычислительной техники образовались два основных направления ее использования.

Первое направление — применение вычислительной техники для выполнения численных расчетов, которые слишком долго или вообще невозможно производить вручную.

Второе направление — использование средств вычислительной техники в автоматических или автоматизированных информационных системах. В самом широком смысле информационная система представляет собой программный комплекс, функции которого состоят в поддержке надежного хранения информации в памяти компьютера, выполнении специфических для данного приложения преобразований информации и/или вычислений, предоставлении пользователям удобного и легко осваиваемого интерфейса. Обычно объемы информации, с которыми приходится иметь дело таким системам, достаточно велики, а сама информация имеет довольно сложную структуру. Классическими примерами информационных систем являются банковские системы, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т.д.

Второе направление возникло несколько позже первого. Это связано с тем, что на заре вычислительной техники компьютеры обладали ограниченными возможностями. Надежное и долговременное хранение информации возможно только при наличии запоминающих устройств, сохраняющих информацию после выключения электрического питания. Оперативная память этим свойством обычно не обладает. Используемые в ранних ЭВМ два вида устройств внешней памяти — магнитные ленты и барабаны — были несовершенными. Магнитные ленты обладали достаточно большой емкостью, но по своей физической природе обеспечивали лишь последовательный доступ к данным. Магнитные барабаны, обеспечивая возможность произвольного доступа к данным, имели ограниченный размер. Появление новых носителей данных — в первую очередь, жестких дисков — дало толчок к работам по созданию информационных компьютерных систем.

Основу любой информационной системы составляет база данных, т.е. набор данных, организованных специальным образом.

В действующем в настоящее время Законе РФ «О правовой охране программ для электронных вычислительных машин и баз данных» от 23.09.92 № 3523-1 дается следующее определение: «База данных — это объективная форма представления и организации совокупности данных (например, статей, расчетов), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ».

Файл — это место фактического хранения информации. В файле различают структуру и собственно данные. Структура файла остается неизменной, а информация (данные) может изменяться при операциях обращения к нему.

В качестве основной структурообразующей единицы хранимых в файле данных принимается *хранимая запись*. Хранимые записи состоят из фиксированной совокупности *полей*, служащих для представления значений какого-либо типа (чисел, литерных строк, дат, булевских значений, денежных единиц и т.д.), и могут иметь формат фиксированной или переменной длины. Полям, как правило, присваиваются уникальные в данной базе имена, ассоциируемые с предметной областью. Если в качестве примера базы данных рассмотреть картотеку сотрудников некоторого абстрактного предприятия, то единицей хранимых данных может быть запись персональной информации по каждому сотруднику с полями: табельный номер (формат поля — целое число); фамилия, имя, отчество (формат поля — литерная строка определенной длины); дата рождения (формат поля — дата); заработная плата (формат поля — действительное число) и т.д.

Информационные системы ориентированы главным образом на хранение, выбор и модификацию постоянно существующей информации.

Структура информации зачастую очень сложна, и хотя структуры данных различны в разных информационных системах, между ними часто бывает много общего. На начальном этапе использования вычислительной техники для управления информацией проблемы структуризации данных решались индивидуально в каждой информационной системе.

Поскольку информационные системы содержат сложные структуры данных, дополнительные индивидуальные средства управления этими данными, являясь существенной частью информационных систем, практически повторялись от одной системы к другой. Стремление выделить общую часть информационных систем, ответственную за управление сложно структурированными данными, явилось первой побудительной причиной создания систем управления базами данных.

Компоненты наиболее полного варианта СУБД следующие:

• среда пользователя, дающая возможность непосредственного управления данными с клавиатуры;

• алгоритмический язык для программирования прикладных систем обработки данных, реализованный как интерпретатор (последний позволяет быстро создавать и отлаживать программы);

• компилятор для придания завершенной программе вида готового коммерческого продукта в форме независимого EXE-файла;

• программы-утилиты для быстрого программирования рутинных операций (генераторы отчетов, форм, таблиц, экранов, меню и других приложений).

Собственно СУБД — это инструментальная оболочка пользователя, а ввиду того, что такая среда ориентирована на немедленное удовлетворение запросов пользователя, — это всегда система-интерпретатор. Наличие в СУБД языка программирования позволяет создавать сложные системы обработки данных, ориентированные на конкретные задачи и конкретного пользователя.

1.2. Области применения баз данных

Автоматизированные информационные системы (АИС), основу которых составляют базы данных, появились в 60-х годах XX века в военной промышленности и бизнесе — там, где были накоплены значительные объемы полезных данных. Первоначально АИС были ориентированы лишь на работу с информацией фактического характера — числовыми или текстовыми характеристиками объектов. Затем по мере развития техники появилась возможность обработки текстовой информации на естественном языке.

Принципы хранения разных видов информации в АИС аналогичны, но алгоритмы ее обработки определяются характером информационных ресурсов. Соответственно различают два класса АИС: документальные и фактографические.

Документальные АИС служат для работы с документами на естественном языке. Наиболее распространенный тип документальных АИС — информационно-поисковые системы, предназначенные для накопления и подбора документов, удовлетворяющих заданным критериям. Эти системы могут выполнять просмотр и подборку монографий, публикаций в периодике, сообщений пресс-агентств, текстов законодательных актов и т.д.

Фактографические АИС оперируют фактическими сведениями, представленными в формализованном виде, и используются для решения задач обработки данных.

Обработка данных — специальный класс решаемых на ЭВМ задач, связанных с вводом, хранением, сортировкой, отбором и группировкой записей данных однородной структуры. К задачам этого класса относятся: учет товаров в магазинах и на складах;

начисление зарплаты; управление производством, финансами, телекоммуникациями и т. п.

Различают фактографические АИС оперативной обработки данных, подразумевающие быстрое обслуживание относительно простых запросов от большого числа пользователей, и фактографические АИС аналитической обработки, ориентированные на выполнение сложных запросов, требующих проведения статистической обработки исторических (накопленных за некоторый промежуток времени) данных, моделирования процессов предметной области и прогнозирования развития этих процессов.

Таким образом, АИС применяются в следующих областях:

- организация хранилищ данных;
- системы анализа данных;
- системы принятия решений;
- мобильные и персональные базы данных;
- географические базы данных;
- мультимедиа базы данных;
- распределенные информационные системы;
- базы данных для всемирной сети World Wide Web.

1.3. Информационная модель данных и ее состав

Каждая информационная система в зависимости от назначения имеет дело с той или иной частью конкретного мира, которую принято называть ее *предметной областью*. Анализ предметной области является необходимым начальным этапом разработки любой информационной системы. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных. Предметная область конкретной информационной системы рассматривается, прежде всего, как некоторая совокупность реальных *объектов*, которые представляют интерес для ее пользователей. Примерами объектов предметной области могут служить персональные ЭВМ, программные продукты и их пользователи. Каждый из этих объектов обладает определенным набором *свойств* (атрибутов). Так, например, компьютер характеризуется названием фирмы-производителя, идентификатором модели, типом микропроцессора, объемом оперативной и внешней памяти, типом графической карты и т. д.

Информационный объект — это описание некоторой сущности предметной области, т. е. *реального объекта*, *процесса*, *явления или события*. Информационный объект (сущность) образуется совокупностью логически взаимосвязанных атрибутов (свойств), представляющих собой качественные и количественные характеристики объекта (сущности).

Между объектами предметной области могут существовать связи, имеющие различный содержательный смысл. Эти связи могут быть обязательными или факультативными (необязательными).

Если вновь порожденный объект оказывается по необходимости связанным с каким-либо объектом предметной области, то между этими двумя объектами существует обязательная связь. В противном случае связь является факультативной.

Например, обязательная связь *Замещает* существует между двумя объектами СОТРУДНИК и ДОЛЖНОСТЬ в предметной области кадровой информационной системы, т. е. каждый принимаемый в организацию сотрудник зачисляется на какую-либо должность и не может быть сотрудника, не замещающего какой-либо должности. В то же время связь *Замещает* между типами объектов СОТРУДНИК и ДОЛЖНОСТЬ является факультативной, поскольку могут существовать вакантные должности.

Совокупность объектов предметной области и связей между ними характеризует *структуру* предметной области.

Множество объектов предметной области, значения атрибутов объектов и связи между ними могут изменяться во времени. Изменения могут сводиться к появлению новых или исключению из рассмотрения некоторых существующих объектов в предметной области, установлению новых или разрушению существующих связей между ними. Следовательно, с каждым моментом времени можно сопоставить некоторое *состояние* предметной области.

Информационно-логическая модель (*ИЛМ*) *— это совокупность информационных объектов* (*сущностей*) *предметной области и связей между ними.*

Процесс создания информационной модели начинается с определения концептуальных требований будущих пользователей БД.

Требования отдельных пользователей интегрируются в едином обобщенном представлении, которое называют концептуальной моделью данной предметной области (рис. 1.1). Такая модель ото-

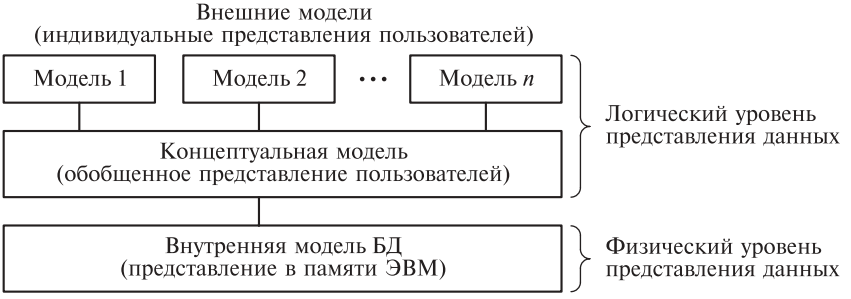

Рис. 1.1. Многоуровневое представление БД

бражает предметную область в виде взаимосвязанных объектов без указания способов их физического хранения.

Концептуальная модель представляет собой интегрированные концептуальные требования всех пользователей к базе данных данной предметной области. При этом усилия разработчика должны быть направлены в основном на структуризацию данных, принадлежащих будущим пользователям БД и выявление взаимосвязей между ними.

Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБД, что потребует ее изменения. Версия концептуальной модели, которая может быть реализована конкретной СУБД, называется *логической моделью.*

Логическая модель, отражающая логические связи между атрибутами объектов вне зависимости от их содержания и среды хранения, может быть реляционной, иерархической или сетевой.

Таким образом, *логическая модель отображает логические связи между информационными данными в данной концептуальной модели.*

Различным пользователям в информационной модели соответствуют различные подмножества ее логической модели, которые называются внешними моделями пользователей.

Таким образом, внешняя модель пользователя представляет собой отображение его концептуальных требований в логической модели и соответствует тем представлениям, которые этот пользователь получает о предметной области на основе логической модели. Следовательно, насколько хорошо спроектирована внешняя модель, настолько полно и точно информационная модель отображает предметную область и настолько полно и точно работает автоматизированная система управления этой предметной областью.

Логическая модель отображается в физическую память, которая может быть построена на электронных, магнитных, оптических, биологических или других принципах.

Внутренняя модель предметной области определяет размещение данных, методы доступа к ним и технику индексирования в данной логической модели и иначе называется физической моделью.

Информационные данные любого пользователя в БД должны быть независимы от всех других пользователей, т.е. не должны оказывать влияния на существующие внешние модели. Это положение отражает первый уровень независимости данных. С другой стороны, внешние модели пользователей никак не связаны с типом физической памяти, в которой будут храниться данные, и с физическими методами доступа к этим данным. Это положение отражает второй уровень независимости данных.

1.4. Три типа логических моделей баз данных

Ядром любой базы данных является модель данных. *Модель данных — это совокупность структур данных и операций их обработки.*

По способу установления связей между данными различают иерархическую, сетевую и реляционную модели.

Иерархическая модель позволяет строить базы данных с древовидной структурой, где каждый узел содержит свой тип данных (сущность). На верхнем уровне дерева в этой модели имеется один узел — корень, на следующем уровне располагаются узлы, связанные с этим корнем, затем узлы, связанные с узлами предыдущего уровня и т.д.

При этом каждый узел может иметь только одного предка (рис. 1.2).

Поиск данных в иерархической системе всегда начинается с корня. Затем производится спуск с одного уровня дерева на другой, пока не будет достигнут искомый уровень. Перемещения по системе от одной записи к другой осуществляются с помощью ссылок.

Основные достоинства иерархической модели — простота описания иерархических структур реального мира и быстрое выполнение запросов. Однако не всегда удобно каждый раз начинать поиск нужных данных с корня, а другого способа перемещения по базе в иерархических структурах нет.

Указанный недостаток снят в *сетевой* модели, где (по крайней мере, теоретически) возможны связи всех информационных объектов со всеми.

В примере, приведенном на рис. 1.3, каждый преподаватель может обучать многих (теоретически всех) студентов и каждый сту-

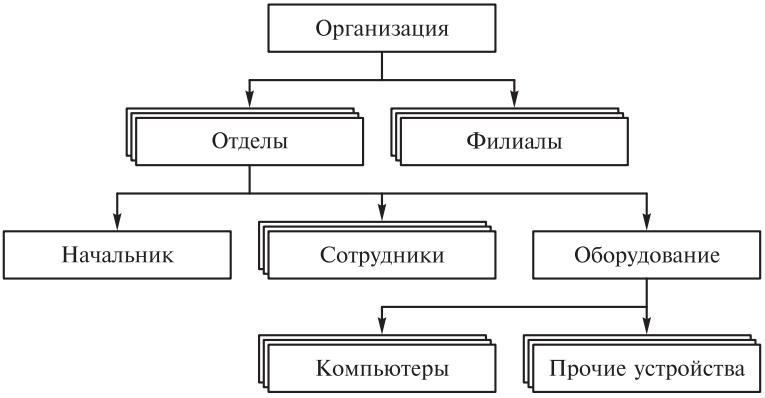

Рис. 1.2. Иерархическая древовидная структура модели БД

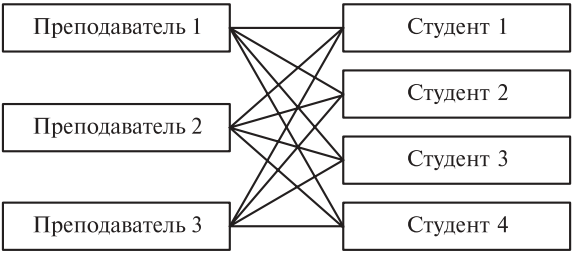

Рис. 1.3. Сетевая структура модели БД

дент может обучаться у многих (теоретически у всех) преподавателей. Поскольку на практике это, естественно, невозможно, приходится прибегать к некоторым ограничениям.

Использование иерархической и сетевой моделей ускоряет доступ к информации в базе данных. Однако, поскольку каждый элемент данных должен содержать ссылки на некоторые другие элементы, требуются значительные ресурсы как дисковой, так и основной памяти ЭВМ. Недостаточность основной памяти, конечно, снижает скорость обработки данных. Кроме того, для таких моделей характерна сложность реализации системы управления базами данных.

Реляционная модель (от англ. relation — отношение) была разработана в начале 70-х годов XX в. Коддом. Простота и гибкость этой модели привлекли к ней внимание разработчиков, и уже 80-х годах XX в. она получила широкое распространение. Таким образом реляционные СУБД оказались промышленным стандартом.

Реляционная модель опирается на систему понятий реляционной алгебры, важнейшими из которых являются таблица, строка, столбец, отношение и первичный ключ, а все операции в этом случае сводятся к манипуляциям с таблицами.

В реляционной модели информация представляется в виде прямоугольных таблиц, каждая из которых состоит из строк и столбцов и имеет имя, уникальное внутри базы данных.

Таблица отражает объект реального мира — *сущность*, а каждая ее строка (запись) отражает один конкретный экземпляр объекта — *экземпляр сущности*. Каждый столбец таблицы имеет уникальное для данной таблицы имя. Располагаются столбцы в соответствии с порядком следования их имен, принятом при создании таблицы.

В отличие от столбцов строки не имеют имен, порядок их следования в таблице не определен, а число — логически не ограничено. Так как строки в таблице не упорядочены, невозможно выбрать строку по ее позиции. Номер, имеющийся в файле у каждой

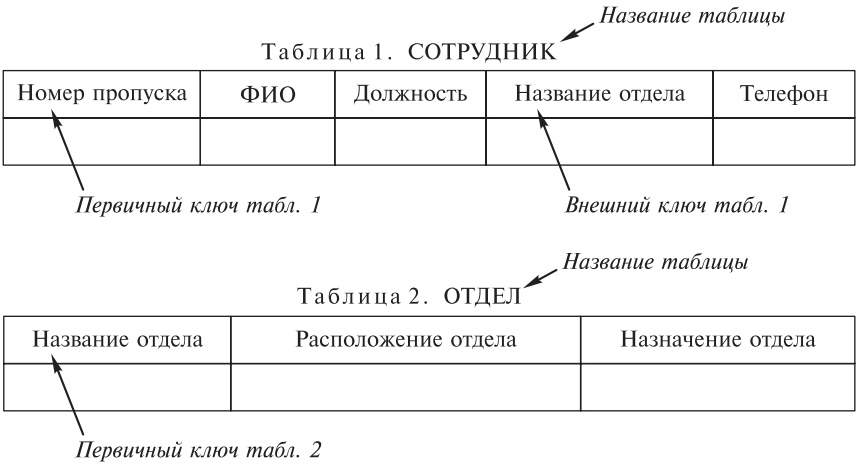

Рис. 1.4. Организация ссылки от одной таблицы к другой

строки, не характеризует ее, так как его значение изменяется при удалении строк из таблицы. Логически не существует первой и последней строк.

Реляционные системы исключили необходимость сложной навигации, поскольку данные представлены в них не в виде одного файла, а независимыми наборами, и для отбора данных используются операции реляционной алгебры — прикладной теории множеств.

В каждой таблице реляционной модели должен быть столбец (или совокупность столбцов), значение которого однозначно идентифицирует каждую ее строку. Этот столбец (или совокупность столбцов) и называется *первичным ключом* таблицы (рис. 1.4).

Если таблица удовлетворяет требованию уникальности первичного ключа, она называется *отношением*. В реляционной модели все таблицы должны быть преобразованы в отношения. Отношения реляционной модели связаны между собой. Связи поддерживаются внешними ключами. *Внешний ключ* — это столбец (совокупность столбцов), значение которого однозначно характеризует значения первичного ключа другого отношения (таблицы).

Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором та же совокупность столбцов является первичным ключом.

В приведенном на рис. 1.4 примере отношение СОТРУДНИК ссылается на отношение ОТДЕЛ через название отдела.

Схема реляционной таблицы (отношения) представляет собой совокупность имен полей, образующих ее запись:

НАЗВАНИЕ ТАБЛИЦЫ (Поле 1, Поле 2, ..., Поле *n*).

Например, для таблиц, показанных на рис. 1.4, имеем следующие схемы (курсивом выделены первичные ключи):

СОТРУДНИК (*Номер пропуска*, ФИО, Должность, Название отдела, Телефон);
ОТДЕЛ (*Название отдела*, Расположение отдела, Назначение отдела).

Объектно-ориентированная модель баз данных начала разрабатываться в связи с появлением объектно-ориентированных языков программирования в 90-е годы XX века. Такого рода базы хранят методы классов, а иногда и постоянные объекты классов, что позволяет осуществлять беспрепятственную интеграцию между данными и их обработкой в приложениях.

Доминирование реляционной модели в современных СУБД определяется:

наличием развитой теории (реляционной алгебры);

наличием аппарата сведения других моделей данных к реляционной модели;

наличием специальных средств ускоренного доступа к информации;

наличием стандартизированного высокоуровневого языка запросов к БД, позволяющего манипулировать ими без знания конкретной физической организации БД во внешней памяти.

1.5. Типы взаимосвязей в модели

На практике часто используются связи, устанавливающие различные виды соответствия между объектами «связанных» типов, — это один к одному (1:1), один ко многим (1:M), многие ко многим (M:M).

Связь *один к одному* означает, что каждому экземпляру первого объекта (*A*) соответствует только один экземпляр второго объекта (*B*) и, наоборот, каждому экземпляру второго объекта (*B*) соответствует только один экземпляр первого объекта (*A*).

Связь *один ко многим* означает, что каждому экземпляру одного объекта (*A*) может соответствовать несколько экземпляров другого объекта (*B*), а каждому экземпляру второго объекта (*B*) может соответствовать только один экземпляр первого объекта (*A*).

Связь *многие ко многим* означает, что каждому экземпляру одного объекта (*A*) могут соответствовать несколько экземпляров второго объекта (*B*) и, наоборот, каждому экземпляру второго

объекта (B) могут соответствовать тоже несколько экземпляров первого объекта (A).

Пример 1.1. Рассмотрим совокупность следующих информационных объектов:

СТУДЕНТ (*Номер студента*, ФИО, Дата рождения, Номер группы);
СТИПЕНДИЯ (*Номер студента*, Размер стипендии);
ГРУППА (*Номер группы*, Специальность);
ПРЕПОДАВАТЕЛЬ (*Код преподавателя*, ФИО, Должность).

Здесь информационные объекты СТУДЕНТ и СТИПЕНДИЯ связаны отношением один к одному, так как каждый студент может иметь только одну стипендию и каждая стипендия может быть назначена только одному студенту.

Информационные объекты ГРУППА и СТУДЕНТ связаны отношением один ко многим, так как одна группа может включать в себя много студентов, в то время как каждый студент может обучаться только в одной группе.

Информационные объекты СТУДЕНТ и ПРЕПОДАВАТЕЛЬ связаны отношением многие ко многим, так как один студент может обучаться у многих преподавателей и один преподаватель может обучать многих студентов.

1.6. Обеспечение непротиворечивости и целостности данных в базе

Для пользователей АИС важно, чтобы база данных отображала предметную область однозначно и непротиворечиво, т.е. чтобы она удовлетворяла условию целостности.

Выделяют два основных типа ограничений по условию целостности данных в базе.

1. Каждая строка таблицы должна отличаться от остальных ее строк значением хотя бы одного столбца.

Пример 1.2. Сотрудники одного отдела могут оказаться полными тезками, иметь одинаковые должность и телефон. Наличие в табл. 1.4 столбца *Номер пропуска* превращает ее в отношение. Таким образом, первое ограничение по условию целостности данных в базе обеспечивается наличием в таблице-отношении первичного ключа.

2. Внешний ключ не может быть указателем на несуществующую строку той таблицы, на которую он ссылается. Это ограничение называется ограничением целостности данных в базе по ссылкам.

Пример 1.3. В столбце *Название отдела* таблицы СОТРУДНИК (см. рис. 1.4) хранятся сведения о принадлежности сотрудников к отделу, т.е. этот столбец является внешним ключом для ссылки на таблицу ОТДЕЛ. Для обеспечения ограничения целостности данных по ссылкам каждое

название отдела из таблицы СОТРУДНИК должно принадлежать конкретному столбцу из таблицы ОТДЕЛ.

В реальных базах данных названия не делают ключевыми из-за их длины, замедляющей процесс поиска, и возможности изменения, создающей сложности с сопровождением системы.

1.7. Основы реляционной алгебры

Поскольку каждая таблица в реляционной БД является отношением, действия над ними базируются на операциях реляционной алгебры. Исключение составляют лишь операции создания и заполнения таблиц данными (присваивания), а также операции описания и переименования столбцов таблицы.

В теории реляционной алгебры отношение рассматривается как множество, строки таблицы называются *кортежами*, столбцы — *атрибутами*. Над отношениями выполняются традиционные операции теории множеств.

1. Ограничение отношения (выборка) — создание нового отношения отбором в него строк отношения-операнда (исходного отношения), которые удовлетворяют условию ограничения.

2. Проекция отношения — создание нового отношения отбором в него определенных столбцов отношения-операнда.

3. Объединение отношений — создание нового отношения, содержащего все кортежи отношений-операндов. При этом операнды должны иметь одинаковые атрибуты.

Пример 1.4. Объединить поступающие из цехов отчеты о выпуске новой продукции за прошедший месяц, содержащие следующие данные: номер цеха, код продукции, дату выпуска и количество выпущенной продукции, с данными общей таблицы ВЫПУСК ПРОДУКЦИИ, имеющей ту же структуру. Для этого к кортежам

ВЫПУСК ПРОДУКЦИИ (Номер цеха, Код продукции, Дата выпуска, Количество)

добавляют кортежи

НОВАЯ ПРОДУКЦИЯ (Номер цеха, Код продукции, Дата выпуска, Количество).

Поскольку атрибуты приведенных операндов совпадают, таблица НОВАЯ ПРОДУКЦИЯ объединяется с исходной.

4. Пересечение отношений — создание нового отношения, содержащего строки, общие для сравниваемых операндов. При этом операнды должны иметь одинаковые атрибуты.

Пример 1.5. Рассмотрим пересечение отношений с выполнением операций ограничения и проекции.

Таблица 1.1

ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ по математике

Группа	Номер зачетной книжки	ФИО студента	Дата	Дисциплина	Оценка
1	1	Иванов И. И.	10.01.05	Математика	Отлично
1	2	Петров П. П.	10.01.05	Математика	Хорошо
1	3	Сидоров С. С.	10.01.05	Математика	Удовлетворительно
1	4	Прохоров Н. И.	10.01.05	Математика	Отлично
1	5	Симонов В. В.	10.01.05	Математика	Хорошо

Таблица 1.2

ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ по физике

Группа	Номер зачетной книжки	ФИО студента	Дата	Дисциплина	Оценка
1	1	Иванов И. И.	17.01.05	Физика	Отлично
1	2	Петров П. П.	17.01.05	Физика	Хорошо
1	3	Сидоров С. С.	17.01.05	Физика	Удовлетворительно
1	4	Прохоров Н. И.	17.01.05	Физика	Отлично
1	5	Симонов В. В.	17.01.05	Физика	Отлично

Пусть имеется набор экзаменационных ведомостей — отношений с совпадающими атрибутами (табл. 1.1, 1.2):

ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ (Группа, Номер зачетной книжки, ФИО студента, Дата, Дисциплина, Оценка).

Требуется подготовить список студентов, получивших только отличные оценки, в виде таблицы со столбцами *Номер зачетной книжки* и ФИО *студента*.

Таблица 1.3

Результат операции ограничения для ведомости по математике

Группа	Номер зачетной книжки	ФИО студента	Дата	Дисциплина	Оценка
1	1	Иванов И. И.	10.01.05	Математика	Отлично
1	4	Прохоров Н. И.	10.01.05	Математика	Отлично

Таблица 1.4

Результат операции ограничения для ведомости по физике

Группа	Номер зачетной книжки	ФИО студента	Дата	Дисциплина	Оценка
1	1	Иванов И. И.	17.01.05	Физика	Отлично
1	4	Прохоров Н. И.	17.01.05	Физика	Отлично
1	5	Симонов В. В.	17.01.05	Физика	Отлично

Таблица 1.5

Результат операции проекции для ведомости по математике

Номер зачетной книжки	ФИО студента
1	Иванов И. И.
4	Прохоров Н. И.

Таблица 1.6

Результат операции проекции для ведомости по физике

Номер зачетной книжки	ФИО студента
1	Иванов И. И.
4	Прохоров Н. И.
5	Симонов В. В.

Для экзаменационных ведомостей нужной группы сначала выполним ограничение исходных отношений, отобрав из каждого из них в новое отношение кортежи, удовлетворяющие следующему условию:

Оценка = Отлично.

В результате получим списки отличников группы по дисциплинам (табл. 1.3, 1.4).

Теперь выполним проекцию полученных отношений, отобрав из каждого из них только атрибуты *Номер зачетной книжки* и *ФИО студента*. Получим новые списки отличников, в которых остались только номера зачетных книжек и фамилии студентов (табл. 1.5, 1.6).

Таким образом получим искомое отношение — СПИСОК ОТЛИЧНИКОВ, содержащее номера зачетных книжек и фамилии, общие для всех списков отличников (табл. 1.7).

Таблица 1.7

СПИСОК ОТЛИЧНИКОВ

Номер зачетной книжки	ФИО студента
1	Иванов И. И.
4	Прохоров Н. И.

5. Разность отношений — создание нового отношения, содержащего строки 1-го операнда, отсутствующие во 2-м операнде. При этом операнды должны иметь одинаковые атрибуты.

Пример 1.6. Требуется, используя ежемесячные отчеты цехов (см.

пример 1.4), подготовить сведения о выпуске новых видов продукции за последний квартал.

Для решения этой задачи выполняем ограничение отношения ВЫПУСК ПРОДУКЦИИ по следующему условию: дата выпуска меньше последней даты прошлого квартала.

Результат такого ограничения поместим в исходную таблицу.

Затем выполним следующее ограничение для исходной таблицы: дата выпуска меньше первой даты прошлого квартала. Полученный результат занесем в конечную таблицу.

Разность отношений исходной и конечной таблиц даст искомые сведения.

6. Произведение отношений — создание нового отношения, в котором имеются все атрибуты 1-го и 2-го операндов, а строки получены попарным сцеплением строк их отношений. Число кортежей — мощность нового отношения — равна произведению мощностей 1-го и 2-го отношений. При этом множества атрибутов отношений не должны пересекаться.

Произведение отношений используется при решении задач подбора пар из двух множеств, например поставщиков и потребителей. Для этого сначала составляют все возможные пары, а затем по конкретному критерию отбирают из них подходящие.

Пример 1.7. По двум заданным отношениям (табл. 1.8, 1.9) требуется найти произведение (табл. 1.10).

Таблица 1.8

ПОСТАВЩИК

Поставщик
Поставщик 1
Поставщик 2

Таблица 1.9

ПОТРЕБИТЕЛЬ

Потребитель
Потребитель 1
Потребитель 2

Таблица 1.10

Результат операции произведения

Поставщик	Потребитель
Поставщик 1	Потребитель1
Поставщик 1	Потребитель 2
Поставщик 2	Потребитель 1
Поставщик 2	Потребитель 2

7. Деление отношений — создание нового отношения, содержащего атрибуты 1-го операнда, отсутствующие во 2-м операнде, и кортежи 1-го операнда, которые совпали с кортежами 2-го операнда. Для выполнения этой операции 2-й операнд должен содержать лишь атрибуты, совпадающие с атрибутами 1-го.

Пример 1.8. Требуется отобрать студентов группы, получающих стипендию, используя список, содержащий следующие сведения: ФИО, дата рождения, шифр группы и признак наличия стипендии (да, нет).

Для решения задачи создадим вспомогательное отношение с атрибутами *Шифр группы* и *Признак наличия стипендии*. Затем заполним один кортеж этого отношения, поместив в него шифр заданной группы и отметку о получении стипендии (да).

В результате деления исходного списка на вспомогательное отношение получим искомый список с атрибутами *ФИО* и *Дата рождения*.

8. Соединение отношений — создание нового отношения, кортеж которого является результатом сцепления кортежей операндов (исходных отношений).

Различают соединения отношений двух видов: естественное и по условию.

При соединении отношений по условию производятся сцепление строк их операндов и проверка полученной строки на соответствие заданному условию. Если условие выполнено, то полученная строка включается в результирующее отношение.

При естественном соединении отношений производятся сцепление строк их операндов и включение полученной строки в результирующее отношение без проверки. Такое соединение используют, когда отношения-операнды обладают общими атрибутами.

Пример 1.9. Требуется соединить отношения СТУДЕНТ (табл. 1.11) и ОЦЕНКА (табл. 1.12), для которых общим атрибутом является *Номер зачетной книжки*.

Результат операции соединения представлен в табл. 1.13.

Таблица 1.11

СТУДЕНТ

ФИО	Дата рождения	Номер зачетной книжки
Иванов И. И.	22.12.80	1234
Петров П. П.	12.05.80	1235
Сидоров С. С.	30.09.80	1236

Таблица 1.12

ОЦЕНКА

Код дисциплины	Номер зачетной книжки	Оценка
1	1234	4
1	1235	3
2	1234	4
2	1235	3

Таблица 1.13

Результат операции соединения

ФИО	Дата рождения	Номер зачетной книжки	Код дисциплины	Номер зачетной книжки	Оценка
Иванов И. И.	22.12.80	1234	1	1234	4
Иванов И. И.	22.12.80	1234	2	1234	4
Петров П. П.	12.05.80	1235	1	1235	3
Петров П. П.	12.05.80	1235	2	1235	3
Сидоров С. С.	30.09.80	1236			

1.8. Нормализация баз данных

Одни и те же данные могут группироваться в таблицы различными способами. Группировка атрибутов в отношениях должна быть рациональной, т. е. минимизирующей дублирование данных и упрощающей процедуры их обработки и обновления. Устранение избыточности данных, являющееся одной из важнейших задач при проектировании баз данных, обеспечивается нормализацией.

Нормализация — это формальный аппарат ограничений на формирование таблиц (отношений), который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых данных и уменьшает трудозатраты на ведение (ввод, корректировку) базы данных.

Процесс нормализации заключается в разложении (декомпозиции) исходных отношений БД на более простые отношения. При этом на каждой ступени этого процесса схемы отношений приводятся в нормальные формы. Для каждой ступени нормализации имеются наборы ограничений, которым должны удовлет-

ворять отношения БД. Тем самым удаляется из таблиц базы избыточная неключевая информация.

Процесс нормализации основан на понятии функциональной зависимости атрибутов: атрибут *A* зависит от атрибута *B* ($B \rightarrow A$), если в любой момент времени каждому значению атрибута *B* соответствует не более одного значения атрибута *A*.

Зависимость, при которой каждый неключевой атрибут зависит от всего составного ключа и не зависит от его частей, называется *полной функциональной зависимостью*. Если атрибут *A* зависит от атрибута *B*, а атрибут *B* зависит от атрибута *C* ($C \rightarrow B \rightarrow A$), но обратная зависимость при этом отсутствует, то зависимость *C* от *A* называется *транзитивной*.

Общее понятие нормализации подразделяется на несколько нормальных форм.

Информационный объект (сущность) находится в ***первой нормальной форме*** (1НФ), когда все его атрибуты имеют единственное значение. Если в каком-либо атрибуте есть повторяющиеся значения, объект (сущность) не находится в 1НФ, и упущен, по крайней мере, еще один информационный объект (еще одна сущность).

Например, задано следующее отношение:

ПРЕДМЕТ (*Код предмета*, Название, Цикл, Объем часов, Преподаватели).

Это отношение не находится в 1НФ, так как атрибут *Преподаватели* подразумевает возможность наличия нескольких фамилий преподавателей в записи, относящейся к какому-то конкретному предмету, что соответствует участию нескольких преподавателей в ведении одной дисциплины.

Переведем атрибут с повторяющимися значениями в новую сущность, назначим ей первичный ключ (*Код преподавателя*) и свяжем с исходной сущностью ссылкой на ее первичный ключ (*Код предмета*). В результате получим две сущности, причем во вторую сущность добавятся характеризующие ее атрибуты:

ПРЕДМЕТ (*Код предмета*, Название, Цикл, Объем часов);
ПРЕПОДАВАТЕЛЬ (*Код преподавателя*, ФИО, Должность, Оклад, Адрес, Код предмета).

Полученные выражения соответствуют случаю, когда несколько преподавателей могут вести один предмет, но каждый преподаватель не может вести более одной дисциплины. А если учесть, что на самом деле один лектор может читать более одной дисциплины, так же как одну и ту же дисциплину могут читать несколько лекторов, необходимо отказаться от жесткой привязки препо-

давателя к предмету в сущности ПРЕПОДАВАТЕЛЬ, создав дополнительную сущность ИЗУЧЕНИЕ, которая будет показывать, как связаны между собой преподаватели и предметы:

ПРЕДМЕТ (*Код предмета*, Название, Цикл, Объем часов);
ПРЕПОДАВАТЕЛЬ (*Код преподавателя*, ФИО, Должность, Оклад, Адрес);
ИЗУЧЕНИЕ (*Код предмета*, *Код преподавателя*).

Информационный объект находится во ***второй нормальной форме*** (2НФ), если он уже находится в первой нормальной форме и каждый его неидентифицирующий (описательный) атрибут зависит от всего уникального идентификатора информационного объекта. Если некий атрибут не зависит полностью от уникального идентификатора информационного объекта, значит, он внесен в состав этого информационного объекта ошибочно и его необходимо удалить. Нормализация в этом случае производится путем нахождения существующего информационного объекта, к которому данный атрибут относится, или созданием нового информационного объекта, в который атрибут должен быть помещен.

Возвратившись к последнему примеру, заметим, что атрибут *Цикл* в сущности ПРЕДМЕТ, характеризующий принадлежность предмета к циклу гуманитарных, естественно-научных, общепрофессиональных или специальных дисциплин, не полностью зависит от уникального идентификатора *Код предмета*, так как разные предметы могут иметь одно и то же значение атрибута *Цикл*. Перенесем этот атрибут в новую сущность ЦИКЛ и получим четыре взаимосвязанных сущности:

ПРЕДМЕТ (*Код предмета*, Название, Объем часов, Код цикла);
ЦИКЛ (*Код цикла*, Название цикла);
ПРЕПОДАВАТЕЛЬ (*Код преподавателя*, ФИО, Должность, Оклад, Адрес);
ИЗУЧЕНИЕ (*Код предмета*, Код преподавателя).

Информационный объект находится в ***третьей нормальной форме*** (3НФ), если он уже находится во второй нормальной форме и ни один его описательный атрибут не зависит от каких-либо других описательных атрибутов. Атрибуты, зависящие от других неидентифицирующих атрибутов, нормализуются путем перемещения зависимого атрибута и атрибута, от которого он зависит, в новый информационный объект.

В данном случае неключевые атрибуты *Должность* и *Оклад* находятся в транзитивной зависимости. Опасность такой зависимости состоит в том, что несколько человек могут работать в одной и той же должности. При изменении должностного оклада в этом

случае нужно будет менять данные в каждой записи, содержащей эту должность, следовательно, требуется создать новую сущность ДОЛЖНОСТЬ с находящимися в транзитивной зависимости атрибутами *Название должности* и *Оклад* и сделать ссылку от сущности ПРЕПОДАВАТЕЛЬ на сущность ДОЛЖНОСТЬ:

ПРЕДМЕТ (*Код предмета*, Название, Объем часов, Код цикла);
ЦИКЛ (*Код цикла*, Название цикла);
ПРЕПОДАВАТЕЛЬ (*Код преподавателя*, ФИО, Код должности, Адрес);
ДОЛЖНОСТЬ (*Код должности*, Название должности, Оклад);
ИЗУЧЕНИЕ (*Код предмета*, Код преподавателя).

1.9. Средства ускоренного доступа к данным

Чтобы пользователь чувствовал себя комфортно, время ожидания ответа на запрос к БД не должно превышать нескольких секунд. В связи с этим требованием специально разрабатываются методы ускорения выборки, позволяющие обойтись без полного перебора строк при выполнении реляционных операций модификации отношений и отбора данных.

Наиболее эффективны методы индексирования и хеширования значений ключей отношения.

Индексирование — логическая сортировка строк таблицы — заключается в создании вспомогательных файлов, содержащих упорядоченные списки значений ключей отношения со ссылками на строку отношения, в которой они находятся. Индексные файлы занимают дополнительную память, но резко ускоряют поиск благодаря применению метода половинного деления. Для одного отношения может быть создано несколько индексов. Кроме того, можно создать индекс для нескольких отношений, если они содержат одинаковые атрибуты, что позволит ускорить выполнение операций соединения этих отношений.

Индексы позволяют находить строки, в которых значения ключевых полей совпадают с заданным значением или принадлежат заданному интервалу.

Хеширование (hashing) — использование хэш-функций, которые вычисляют вес строки таблицы по значению ее ключевых атрибутов. Результат вычисления хэш-функции — целое число в диапазоне физических номеров строк таблицы.

Идеальная хэш-функция должна давать разные значения веса для разных ключевых атрибутов. Но это не всегда возможно. На практике обычно используют простые хэш-функции, например $f(k) = k \bmod p$, где k — целое число, первичный ключ отношения; p — простое целое число; mod — операция, вычисляющая оста-

ток при целочисленном делении. Если ключевой атрибут — строка символов, то для вычисления $f(k)$ выбирается один из методов преобразования строки в число, например вычисление контрольной суммы.

Для организации доступа к данным при хешировании создается таблица с пустыми строками, которая заполняется следующим образом:

- по первичному ключу новой строки вычисляется значение хэш-функции $f(k)$ и результат трактуется как номер строки в созданной таблице;
- если строка уже занята, производится проверка следующих строк по специальному алгоритму до тех пор, пока не будет обнаружено свободное место.

Аналогично производится поиск нужной строки:

- если после вычисления $f(k)$ на месте в таблице, которое соответствует вычисленному значению, оказывается пустая строка, значит, искомой строки просто нет;
- если значение ключа совпало с искомым, поиск заканчивается;
- если же значение ключа не совпало с искомым, проверяются следующие строки таблицы до обнаружения строки с нужным ключом (в этом случае искомая строка найдена) или пустой строки (в этом случае искомая строка отсутствует).

Если таблица заполнена не более чем на 60 %, то для размещения новой или поиска существующей строки необходимо проверить в среднем не более двух ячеек. Хеширование используют для поиска строк по точному совпадению значения ключевого атрибута кортежа с нужным значением ключа.

1.10. Этапы проектирования баз данных

На этапе проектирования базы данных разработчик должен определить, из каких таблиц должна состоять база данных, какие данные нужно поместить в каждую таблицу и как связать таблицы.

Следовательно, в результате проектирования определяются логическая структура базы данных, т. е. состав реляционных таблиц, их структура и межтабличные связи.

Для создания базы данных необходимо располагать описанием выбранной предметной области, охватывающим реальные объекты и процессы, а также определить все необходимые источники информации для удовлетворения предполагаемых запросов пользователей и потребности в обработке данных. На основе такого описания определяются состав и структура данных предметной области, которые должны находиться в базе и обеспечивать выполнение необходимых запросов и задач пользователей. Структура дан-

ных предметной области может отображаться информационно-логической моделью, на основе которой легко создается реляционная база данных.

Этапы проектирования и создания базы данных:

- построение информационно-логической модели данных предметной области;
- определение логической структуры реляционной базы данных;
- конструирование таблиц базы данных;
- создание схемы данных;
- ввод данных в таблицы (создание записей);
- разработка необходимых форм, запросов, макросов, модулей, отчетов;
- разработка пользовательского интерфейса.

В процессе разработки модели данных необходимо выделить информационные объекты, соответствующие требованиям нормализации данных, и определить связи между ними. Полученная модель позволит создать реляционную базу данных без дублирования, в которой обеспечиваются однократный ввод данных при первоначальной загрузке и корректировках, а также целостность данных при внесении изменений.

При разработке модели данных используются два подхода.

1. Сначала определяются основные задачи, для решения которых строится база, выявляются потребности задач в данных и соответственно определяются состав и структура информационных объектов.

2. Сразу устанавливаются типовые объекты предметной области.

Наиболее рационально сочетание этих подходов, так как на начальном этапе, как правило, нет исчерпывающих сведений обо всех задачах. Использование такой технологии тем более оправдано, что гибкие средства создания реляционной базы данных допускают на любом этапе разработки внесение изменений и модифицирование ее структуры без ущерба для введенных ранее данных.

Процесс выделения информационных объектов предметной области, отвечающих требованиям нормализации, может производиться на основе интуитивного или формального подхода. Теоретические основы формального подхода разработаны известным американским ученым Дж. Мартином и изложены в его монографиях по организации баз данных.

При интуитивном подходе легко выявить информационные объекты, соответствующие реальным объектам, однако получаемая при этом информационно-логическая модель, как правило, требует дальнейших преобразований, в частности преобразования много-многозначных связей между объектами. При таком подходе в случае отсутствия достаточного опыта возможны существенные ошибки. Последующая проверка выполнения требований нор-

мализации обычно показывает необходимость уточнения информационных объектов.

Рассмотрим формальные правила выделения информационных объектов:

на основе описания предметной области выявить документы и их атрибуты, подлежащие хранению в базе данных;

определить функциональные зависимости между атрибутами;

выбрать все зависимые атрибуты и указать для каждого все его ключевые атрибуты, т.е. атрибуты, от которых он зависит;

сгруппировать атрибуты, одинаково зависимые от ключевых атрибутов. (Полученные группы зависимых атрибутов вместе с их ключевыми атрибутами образуют информационные объекты.)

При определении логической структуры реляционной базы данных на основе модели каждый информационный объект адекватно отображается реляционной таблицей, а связи между этими таблицами соответствуют связям между информационными объектами.

В процессе создания БД сначала конструируются таблицы, соответствующие информационным объектам построенной модели данных. Далее может создаваться схема данных, в которой фиксируются существующие логические связи между таблицами, соответствующие связям информационных объектов. В схеме данных могут быть заданы параметры поддержания целостности базы данных, если модель была разработана в соответствии с требованиями нормализации. Целостность данных означает, что в БД 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.

После формирования схемы данных осуществляется ввод непротиворечивых данных из документов предметной области.

На основе созданной базы формируются необходимые запросы, формы, макросы, модули, отчеты, производящие требуемую обработку данных и их представление.

С помощью встроенных средств и инструментов базы данных создается пользовательский интерфейс, позволяющий управлять процессами ввода, хранения, обработки, обновления и представления информации.

1.11. Проектирование базы данных на основе модели типа объект — отношение

Имеется целый ряд методик создания информационно-логических моделей. Наиболее популярна в настоящее время методика с использованием ERD (entity-relationship diagram). В русскоязычной литературе эти диаграммы называют объект — отношение либо