В. В. Кириллов,
Г. Ю. Громов

Введение в реляционные базы данных

- Базы данных и управление ими
- Реляционная модель данных
- SQL – стандартный язык для работы с реляционными базами данных
- Основы проектирования реляционных баз данных
- Создание приложений на языке SQL и его процедурных расширениях
- Хранимые процедуры
- Примеры создания баз данных
- Инструментарий для создания баз данных и приложений к ним

+CD

bhv

Владимир Кириллов

Геннадий Громов

Введение в реляционные базы данных

Санкт-Петербург

«БХВ-Петербург»

2009

УДК 681.3.06
ББК 32.973.26-018.2
К43

Кириллов, В. В.

К43 Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. — СПб.: БХВ-Петербург, 2009. — 464 с.: ил. + CD-ROM — (Учебная литература для вузов)

ISBN 978-5-94157-770-5

В книге рассматриваются основные понятия баз данных и систем управления ими (СУБД), моделей данных, положенных в основу баз данных и методов проектирования реляционных баз данных. Обсуждаются реляционные операции и основы теории нормализации отношений и приводятся примеры проектирования баз данных. Большое место уделено подробному описанию языка SQL — международного стандарта языка реляционных баз данных. Рассматриваются основные понятия, необходимые для изучения SQL и применения его на практике. Подробно рассмотрено манипулирование данными в интерактивном режиме, затронуты вопросы обеспечения безопасности хранимых данных, средств оптимизации запросов и создания прикладных программ. На прилагаемом к книге компакт-диске содержатся дистрибутивы СУБД OracleXE, SqlDeveloper, учебные базы данных и дополнительные материалы.

Для студентов и начинающих программистов

УДК 681.3.06
ББК 32.973.26-018.2

Рецензент — А. А. Бобцов, профессор, д. т. н., декан факультета компьютерных технологий и управления СПбГУ ИТМО

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Фото	*Кирилла Сергеева*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.09.08.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 37,41.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-94157-770-5

Оглавление

Введение

Основные идеи современной информационной технологии базируются на концепции баз данных. Согласно этой концепции, основой информационной технологии являются данные, которые должны быть организованы в базах данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем выдвинуло требование создания удобных общесистемных средств интеграции хранимых данных и управления ими. Это и привело к появлению в конце 1960-х годов первых промышленных систем управления базами данных (СУБД) — специализированных программных средств, предназначенных для организации и ведения баз данных. Сначала это были системы с инвертированными списками, иерархические и сетевые системы. В 1969 году была предложена реляционная модель данных *(см. главу 3)*, а в конце 1970-х и начале 1980-х годов стали появляться первые промышленные реляционные СУБД. Сейчас преобладающее большинство СУБД являются реляционными, несмотря на появление объектно-ориентированных СУБД. Это не в последнюю очередь связано с тем, что в конце 1990-х годов большинство ведущих производителей реляционных СУБД создали объектные надстройки к реляционной схеме, что привело к появлению объектно-реляционных СУБД, поддерживающих некоторые технологии, реализующие объектно-ориентированный подход.

Поэтому в данной книге рассматривается реляционная модель данных, реляционные СУБД и основной язык общения с этими СУБД — SQL.

Несмотря на совпадающую модель данных, положенную в основу таких СУБД, и использование ими языка, в основном поддерживающего стандарт SQL:2003 *(см. разд. 4.1)*, все они, в той или иной мере, отличаются друг от друга. Поэтому в иллюстрационных примерах часто будет использоваться реализация SQL:2003 Oracle Database 10g.

Книга адресована широкому кругу пользователей. Для освоения материала книги необходимы минимальные знания о компьютерах. Те же читатели, которые установят на своем персональном компьютере СУБД, базы данных и средства для работы с ними, размещенные на приложенном к книге компакт-диске, смогут достаточно детально изучить материал, связанный с реляционными базами данных и познакомиться с одной из лучших профессиональных СУБД.

Книга ориентирована на студентов высших учебных заведений, изучающих дисциплины "Базы данных", "Информационные системы", "Проектирование информационных систем", а также будет полезна специалистам в области информационных технологий. Это подтверждается отзывами о частично включенных в материал книги учебных пособиях [4, 5], получаемыми авторами уже более десяти лет, а также отзывами студентов направлений "Информатика и вычислительная техника", "Информационные системы" и "Прикладная математика и информатика", изучающих перечисленные ранее дисциплины в СПбГУ ИТМО.

Книга содержит 23 главы и 2 приложения. Главы сгруппированы в семь частей.

Первая часть знакомит с историей появления и основными понятиями баз данных, с моделями данных и реляционным подходом. В *главе 1* дается общее представление о возникновении концепции баз данных и их архитектуре. В *главе 2* рассматриваются основные понятия информационно-логического (инфологического) моделирования, язык моделирования, необходимость введения ключей и подходы к построению моделей. В *главе 3* разбираются основы реляционного подхода и манипулирования реляционными данными.

Вторая часть знакомит с основами языка SQL и его применением для получения информации из реляционных баз данных. В *главе 4* рассматриваются стандарты языка, его синтаксические конструкции, типы данных и разнообразные функции. *Глава 5* описывает предложение `SELECT`, выборку данных без использования и с использованием фразы для отбора данных, упорядочение и агрегирование данных. В *главе 6* продолжается изучение способов выборки данных, но уже из нескольких таблиц. Рассматриваются средства одновременной работы с множеством таблиц, соединения, вложенные подзапросы и объединение нескольких запросов.

В *третьей части* анализируются средства защиты данных, средства изменения содержимого базы данных, управления транзакциями и обеспечения параллельной работы. *Глава 7* описывает средства языка SQL, предназначенные для защиты данных. Здесь рассматривается создание и использование представлений. В *главе 8* обсуждаются особенности синтаксиса и применения предложений модификации данных. В *главе 9* дается определение тран-

закции и объясняется необходимость ее использования. Рассматриваются проблемы, возникающие в многопользовательском режиме работы, и их решение.

Четвертая часть описывает цели и процедуры проектирования, основы нормализации и пример проектирования конкретной базы данных. В *главе 10* обсуждаются цели проектирования, возможные ошибки в процессе проектирования и этапы проектирования. *Глава 11* посвящена функциональным и многозначным зависимостям, нормальным формам, процедурам нормализации. Здесь даются рекомендации по построению даталогической модели. В *главе 12* приводится подробный пример проектирования базы данных "Библиотека".

В *пятой части* рассматривается язык создания основных объектов базы данных, системный каталог и способы оптимизации запросов. В *главе 13* дается обзор языка определения данных, сведения о создании и изменении таблиц, а также последовательностей. *Глава 14* объясняет, зачем нужен системный каталог, описывает его структуру, приводит примеры использования. В *главе 15* обсуждаются вопросы, связанные с оптимизацией выполнения запросов и способов, позволяющих увеличить их производительность.

В *шестой части* рассматриваются различные варианты и средства создания приложений с использованием языка SQL и его процедурных расширений. *Глава 16* описывает статический и динамический SQL, а также интерфейс программирования приложений. В *главе 17* подробно анализируется одно из таких расширений — PL/SQL. Даются основы PL/SQL: достаточно подробно рассматриваются команды, обработка ошибок, курсоры и динамический SQL в PL/SQL. В *главе 18* приводятся синтаксис и примеры создания хранимых процедур, функций, триггеров и пакетов (модулей). Рассказывается о встроенных пакетах PL/SQL.

Седьмая часть достаточно подробно описывает инфологические модели двух связанных предметных областей из интегрированной информационной системы управления университетом. Вырезка "UCHEB" из базы данных этой информационной системы, относящаяся к рассматриваемым моделям, размещена на приложенном к книге компакт-диске. Она может использоваться в процессе изучения материала книги. В *главе 19* рассказывается о Государственных образовательных стандартах и основных образовательных программах. На основании этого материала создается инфологическая модель учебных планов, процедура построения которой рассматривается в следующей главе. *Глава 20* подробно анализирует подходы к построению инфологической модели, здесь приводится диаграмма сущность-связь "Учебные планы". В *главе 21* дается описание предметной области, связанной с успеваемостью

студентов, рассматривается процедура построения инфологической модели и приводится диаграмма сущность-связь "Итоговая успеваемость". *Глава 22* наполнена запросами, которые надо "перевести" на язык SQL и реализовать во время изучения книги. Здесь также даны возможные ответы к некоторым из этих запросов. В *главе 23* приводятся листинги и описания ряда приложений (функции, процедур и пакетов) базы данных "UCHEB".

В приложениях даны полезные справочные сведения о PL/SQL. *Приложение А* представляет собой краткое описание СУБД Oracle 10g Express Edition (XE). Здесь представлены инструкция по ее установке, краткое описание некоторых инструментов для работы с базами данных, расположенными на Oracle 10g XE, и инструкция по установке Oracle SQL Developer. В *приложении Б* приводится перечень инструментальных средств, инструкций и листингов, размещенных на компакт-диске к книге.

ЧАСТЬ I

ЧТО ТАКОЕ БАЗА ДАННЫХ И СУБД

Глава 1

Зачем нужны базы данных

1.1. Данные и ЭВМ

Восприятие реального мира можно соотнести с последовательностью разных, хотя иногда и взаимосвязанных, явлений. С давних времен люди пытались описать эти явления (даже тогда, когда не могли их понять). Такое описание называют *данными*.

Традиционно фиксация данных осуществляется с помощью конкретного средства общения (например, с помощью естественного языка или изображений) на конкретном носителе (например, камне или бумаге). Обычно данные (факты, явления, события, идеи или предметы) и их интерпретация (семантика) фиксируются совместно, так как естественный язык достаточно гибок для представления того и другого. Примером может служить утверждение "Стоимость авиабилета 115". Здесь "115" — данное, а "Стоимость авиабилета" — его семантика.

Нередко данные и интерпретация разделены. Например, "Расписание движения самолетов" может быть представлено в виде таблицы (рис. 1.1), в верхней части которой (отдельно от данных) приводится их интерпретация. Такое разделение затрудняет работу с данными (попробуйте быстро получить сведения из нижней части таблицы, если в ней будет намного больше строк).

Применение ЭВМ для ведения* и обработки данных обычно приводит к еще большему разделению данных и интерпретации. ЭВМ имеет дело главным образом с данными как таковыми. Большая часть интерпретирующей информации вообще не фиксируется в явной форме (ЭВМ не "знает", является ли "21.50" стоимостью авиабилета или временем вылета). Почему же это произошло?

---

* *Ведение* (сопровождение, поддержка) *данных* — термин, объединяющий действия по добавлению, удалению или изменению хранимых данных.

Интерпретация

Номер рейса	Дни недели	Пункт отправления	Время вылета	Пункт назначения	Время прибытия	Тип самолета	Стоимость билета

Данные

138	.2.4..7	Баку	21.12	Москва	0.52	ИЛ-86	115.00
57	..3..6.	Ереван	7.20	Киев	9.25	ТУ-154	92.00
1234	.2…6.	Казань	22.40	Баку	23.50	ТУ-134	73.50
242	1234567	Киев	14.10	Москва	16.15	Б-737	57.00
86	.23.5..	Минск	10.50	Сочи	13.06	ИЛ-86	78.50
137	1.3..6.	Москва	15.17	Баку	18.44	ИЛ-86	115.00
241	1234567	Москва	9.05	Киев	11.05	Б-737	57.00
577	1.3.5..	Рига	21.53	Таллинн	22.57	ЯК 42	21.50
78	..3..6.	Сочи	18.25	Баку	20.12	ТУ-134	44.00
578	.2.4.6.	Таллинн	6.30	Рига	7.37	ЯК 42	21.50

Рис. 1.1. К разделению данных и их интерпретации

Существует, по крайней мере, две исторические причины, по которым применение ЭВМ привело к отделению данных от интерпретации. Во-первых, ЭВМ не обладали достаточными возможностями для обработки текстов на естественном языке — основном языке интерпретации данных. Во-вторых, стоимость памяти ЭВМ была первоначально весьма велика. Память использовалась для хранения самих данных, а интерпретация традиционно возлагалась на пользователя. Пользователь закладывал интерпретацию данных в свою программу, которая "знала", например, что шестое вводимое значение связано со временем прибытия самолета, а четвертое — со временем его вылета. Это существенно повышало роль программы, так как вне интерпретации данные представляют собой не более чем совокупность битов на запоминающем устройстве.

Жесткая зависимость между данными и использующими их программами создает серьезные проблемы в ведении данных и делает их использование менее гибким.

Нередки случаи, когда пользователи одной и той же ЭВМ создают и используют в своих программах разные наборы данных, содержащие сходную информацию. Иногда это связано с тем, что пользователь не знает (либо не захотел узнать), что в соседней комнате или за соседним столом сидит сотрудник, который уже давно ввел в ЭВМ нужные данные. Чаще же потому,

что при совместном использовании одних и тех же данных возникает масса проблем.

Разработчики прикладных программ (написанных, например, на Бейсике, Паскале или Си) размещают нужные им данные в файлах, организуя их наиболее удобным для себя образом. При этом одни и те же данные могут иметь в разных приложениях совершенно разную организацию (разную последовательность размещения в записи, разные форматы одних и тех же полей и т. п.). Обобществить такие данные чрезвычайно трудно: например, любое изменение структуры записи файла, производимое одним из разработчиков, приводит к необходимости изменения другими разработчиками тех программ, которые используют записи этого файла.

Для иллюстрации обратимся к примеру, приведенному в книге У. Девиса "Операционные системы" (М., Мир, 1980):

> "Несколько лет назад почтовое ведомство (из лучших побуждений) пришло к решению, что все адреса должны обязательно включать почтовый индекс. Во многих вычислительных центрах это, казалось бы, незначительное изменение привело к ужасным последствиям. Добавление к адресу нового поля, содержащего шесть символов, означало необходимость внесения изменений в каждую программу, использующую данные этой задачи в соответствии с изменившейся суммарной длиной полей. Тот факт, что какой-то программе для выполнения ее функций не требуется знания почтового индекса, во внимание не принимался: если в некоторой программе содержалось обращение к новой, более длинной записи, то в такую программу вносились изменения, обеспечивающие дополнительное место в памяти.
>
> В условиях автоматизированного управления централизованной базой данных все такие изменения связаны с функциями управляющей программы базы данных. Программы, не использующие значения почтового индекса, не нуждаются в модификации — в них, как и прежде, в соответствии с запросами посылаются те же элементы данных. В таких случаях внесенное изменение неощутимо. Модифицировать необходимо только те программы, которые пользуются новым элементом данных".

1.2. Концепция баз данных

Активная деятельность по отысканию приемлемых способов обобществления непрерывно растущего объема информации привела к созданию в начале 60-х годов XX в. специальных программных комплексов, называемых *системами управления базами данных (СУБД)*.

Основная особенность СУБД — это наличие процедур для ввода и хранения не только самих данных, но и описаний их структуры. Файлы, снабженные описанием хранимых в них данных и находящиеся под управлением СУБД, стали называть банки данных, а затем *базы данных (БД)*.

Пусть, например, требуется сохранить расписание движения самолетов (см. рис. 1.1) и ряд других данных, связанных с организацией работы аэропорта (БД "Аэропорт"). Используя для этого одну из "русифицированных" реляционных СУБД, можно подготовить следующее описание расписания:

```
СОЗДАТЬ ТАБЛИЦУ Расписание
  (Номер_рейса         Целое
   Дни_недели          Текст (8)
   Пункт_отправления   Текст (24)
   Время_вылета        Время
   Пункт_назначения    Текст (24)
   Время_прибытия      Время
   Тип_самолета        Текст (8)
   Стоимость_билета    Валюта);
```

и ввести его вместе с данными в БД "Аэропорт".

Язык запросов СУБД позволяет обращаться за данными как из программ, так и с терминалов (рис. 1.2).

Сформировав запрос:

```
ВЫБРАТЬ  Номер_рейса, Дни_недели, Время_вылета
ИЗ ТАБЛИЦЫ  Расписание
ГДЕ  Пункт_отправления = 'Москва'
  И  Пункт_назначения  = 'Киев'
  И  Время_вылета > 17;
```

получим расписание "Москва—Киев" на вечернее время, а по запросу:

```
ВЫБРАТЬ  КОЛИЧЕСТВО(Номер_рейса)
ИЗ ТАБЛИЦЫ  Расписание
ГДЕ  Пункт_отправления = 'Москва'
  И  Пункт_назначения  = 'Минск';
```

получим количество рейсов "Москва—Минск".

Эти запросы не потеряют актуальности и при расширении таблицы:

```
ДОБАВИТЬ В ТАБЛИЦУ Расписание
   Длительность_полета  Целое;
```

как это было с программами обработки почтовых адресов при введении почтового индекса.

Рис. 1.2. Связь программ и данных при использовании СУБД

Однако за все надо расплачиваться: на обмен данными через СУБД требуется большее время, чем на обмен аналогичными данными прямо из файлов, специально созданных для того или иного приложения.

1.3. Архитектура СУБД

СУБД должна предоставлять доступ к данным любым пользователям, включая и тех, кто практически не имеет и (или) не хочет иметь представления о:

- физическом размещении в памяти компьютера данных и их описаний;
- механизмах поиска запрашиваемых данных;
- проблемах, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами);
- способах обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа;
- поддержании баз данных в актуальном состоянии
- и множестве других функций современных СУБД.

При выполнении основных из этих функций СУБД должна использовать различные описания данных. А что это за описания?

В 1975 году подкомитет SPARC (Standards Planning and Requirements Committee) американского национального института стандартов (American National Standards Institute, ANSI) выдвинул проект трехуровневой архитектуры СУБД (рис. 1.3):

1. *Внешний* (пользовательский).
2. Промежуточный (*концептуальный*).
3. *Внутренний* (физический).

В основе архитектуры ANSI/SPARC лежит концептуальный уровень. Этот уровень описывает данные и их взаимосвязи с наиболее общей точки зрения — концепции архитекторов (разработчиков) базы данных.

Внутренний уровень позволяет скрыть подробности физического хранения данных (носители, файлы ...) от концептуального уровня. Отделение внутреннего уровня от концептуального уровня обеспечивает, так называемую, *физическую независимость* данных.

На внешнем уровне описываются различные подмножества элементов концептуального уровня для представления данных различными пользователями и (или) их программами. Каждый пользователь получает в свое распоряжение часть представлений о данных, но полная концепция скрыта. Отделение внешнего уровня от концептуального уровня обеспечивает *логическую независимость* данных.

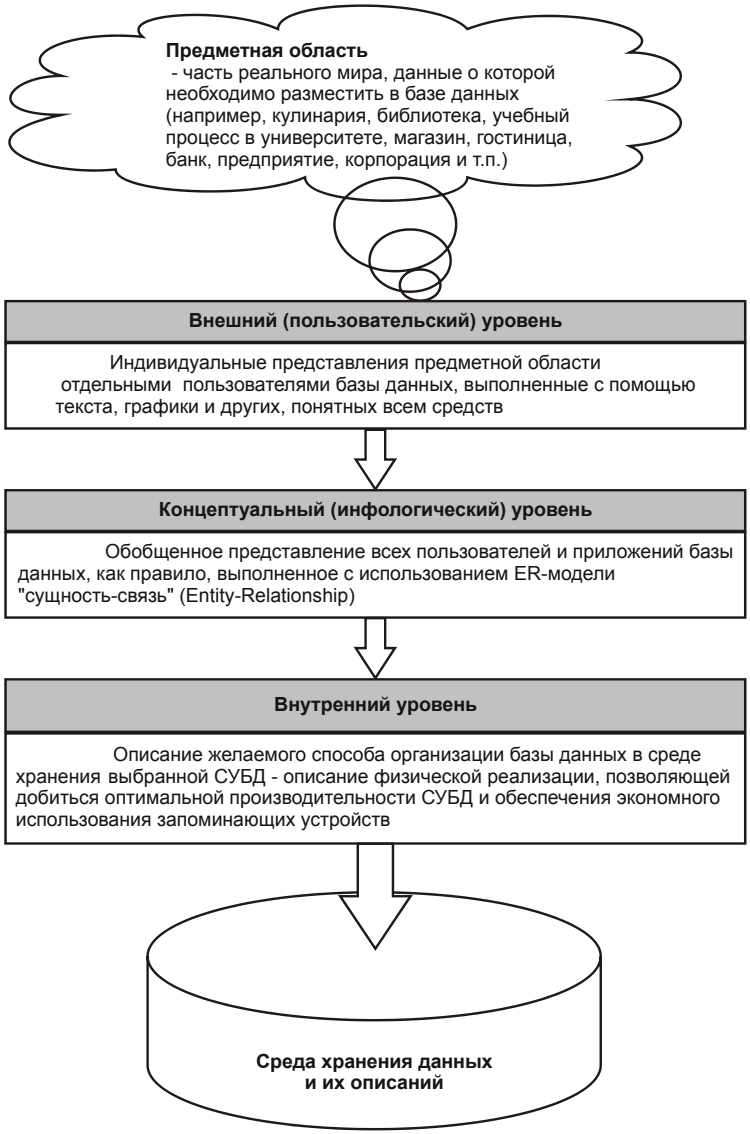

Рис. 1.3. Уровни моделей данных

В дальнейшем эта архитектура была включена в стандарты международной организации по стандартизации (International Organization for Standardization, ISO), членом которой с момента ее основания является Россия.

Естественно, что проект базы данных надо начинать с выбора *предметной области* (той части реального мира, данные о которой надо отразить в базе

данных) и выявления требований к ней отдельных пользователей (сотрудников организации, для которых создается база данных). Подробнее этот процесс будет рассмотрен далее, а здесь отметим, что проектирование обычно поручается человеку (группе лиц) — *администратору данных (АД)*. Объединяя частные представления о содержимом базы данных, полученные в результате опроса пользователей и (или) анализа их технических заданий, и свои представления о данных, которые могут потребоваться в будущих приложениях, АД сначала создает обобщенное неформальное описание создаваемой базы данных. Это описание, выполненное с использованием текста (на естественном языке), образцов входных и выходных документов, математических формул, таблиц, графики и других средств, понятных потенциальным пользователям и всем людям, работающим над проектированием базы данных, и есть концептуальная модель данных, которую часто называют логической или *информационно-логической (инфологической)* моделью.

Такая человеко-ориентированная модель полностью независима от физических параметров среды хранения данных и от той СУБД, которая будет использоваться для построения и ведения базы данных. В конце концов, этой средой может быть память человека, а не ЭВМ. Поэтому концептуальная модель не должна изменяться до тех пор, пока какие-то изменения в реальном мире не потребуют изменения в ней некоторых определений, чтобы эта модель продолжала отражать предметную область.

Если АД — это человек, отвечающий за стратегию и политику принятия решений, связанных с данными базы данных, то человек, обеспечивающий необходимую техническую поддержку для реализации принятых решений, называется *администратором базы данных* (АБД).

АБД выбирает конкретную СУБД и решает, как данные будут представлены в хранимой базе данных, т. е. осуществляет физическое проектирование базы данных — преобразование концептуальной модели в физическую модель.

Трехуровневая архитектура позволяет обеспечить *независимость хранимых данных* от использующих их программ. АБД может при необходимости переписать хранимые данные на другие носители информации и (или) реорганизовать их физическую структуру, изменив лишь физическую модель данных. АД может подключить к системе любое число новых пользователей (новых приложений), дополнив, если надо, концептуальную модель. Указанные изменения физической и концептуальной моделей не будут замечены существующими пользователями системы (окажутся "прозрачными" для них), так же как не будут замечены и новые пользователи. Следовательно, независимость данных обеспечивает возможность развития системы баз данных без разрушения существующих приложений.

Глава 2

Инфологическая модель данных "сущность-связь"

2.1. Основные понятия

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).

Сущность — любой различимый объект, факт, явление, событие, идея или предмет, информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет, женитьба, гроза, изобретение, боль и т. п. Необходимо различать такие понятия, как *тип сущности* и *экземпляр сущности*. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое. Экземпляр сущности относится к конкретной вещи в наборе. Например, типом сущности может быть `ГОРОД`, а экземпляром — `Москва`, `Киев` и т. д.

Атрибут — поименованная характеристика (свойство) сущности. Это любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Наименование атрибута должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей, например, `ЦВЕТ` может быть определен для многих сущностей: `СОБАКА`, `АВТОМОБИЛЬ`, `ДЫМ` и т. д. Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Примерами атрибутов для сущности `АВТОМОБИЛЬ` являются `ТИП`, `МАРКА`, `НОМЕРНОЙ ЗНАК`, `ЦВЕТ` и т. д. Здесь также существует различие

между типом и экземпляром. Тип атрибута `ЦВЕТ` имеет много экземпляров или значений (`Красный`, `Синий`, `Банановый`, `Белая ночь` и т. д.), однако каждому экземпляру сущности присваивается только одно значение атрибута.

Абсолютное различие между типами сущностей и атрибутами отсутствует. Атрибут является таковым только в связи с типом сущности. В другом контексте атрибут может выступать как самостоятельная сущность. Например, в расписании движения самолетов (см. рис. 1.1) город — это атрибут расписания, а в кодификаторе адресов город — тип сущности.

Ключ — минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Для сущности `Расписание` (см. *разд. 1.2*) ключом является атрибут `Номер_рейса` или набор: `Пункт_отправления`, `Время_вылета` и `Пункт_назначения` (при условии, что из пункта в пункт вылетает в каждый момент времени один самолет).

Связь — ассоциирование двух или более сущностей. Абсолютное различие между типами сущностей и связями отсутствует. Один и тот же факт может совершенно обоснованно рассматриваться или как сущность, или как связь. Например, в запросе — "с кем вступила в брак Алла Пугачева" брак — связь, а в запросе — "сколько браков было зарегистрировано в этом ЗАГСе в прошлом году" брак — сущность.

Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных — это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей.

2.2. Характеристика связей и язык моделирования

При построении инфологических моделей можно использовать язык *ER-диаграмм* — от англ. Entity-Relationship, т. е. сущность-связь. Этот язык был предложен в 1976 году сотрудником корпорации IBM Питером Ченом [7].

В нем предлагалось изображать сущности помеченными прямоугольниками, связи — помеченными ромбами, атрибуты — помеченными овалами. Над линиями, соединяющими прямоугольники, может проставляться степень связи (1 или буква M, заменяющая слово "много") и необходимое пояснение (рис. 2.1).

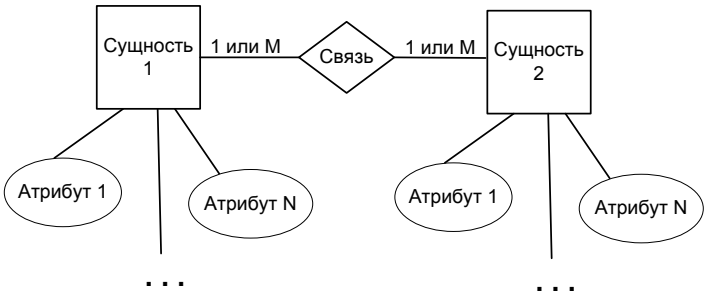

Рис. 2.1. Элементы ER-диаграмм языка, предложенного Ченом

Так, между двумя сущностями, например, А и Б возможны четыре вида связей.

Первый тип — связь *"один-к-одному" (1:1)*: в каждый момент времени каждому представителю (экземпляру) сущности А соответствует 1 или 0 представителей сущности Б. Например, студент может не "заработать" стипендию, получить обычную или одну из повышенных стипендий (рис. 2.2, *а*).

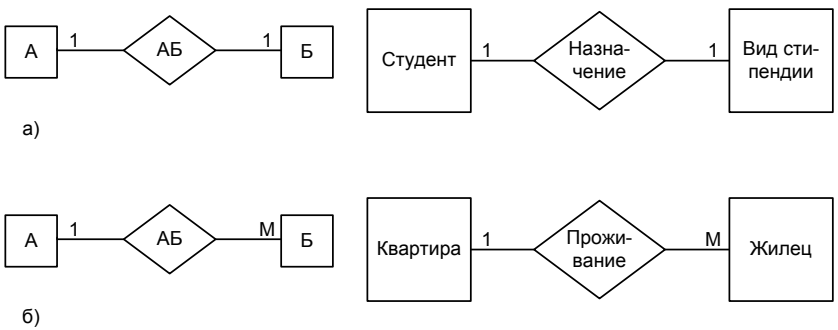

Рис. 2.2. Примеры простейших ER-моделей

Второй тип — связь *"один-ко-многим" (1:M)*: одному представителю сущности А соответствуют 0, 1 или несколько представителей сущности Б. Напри-

мер, квартира может пустовать, в ней может жить один или несколько жильцов (рис. 2.2, *б*).

Так как между двумя сущностями возможны связи в обоих направлениях, то существует еще два типа связи *"многие-к-одному" (M:1)* и *"многие-ко-многим" (M:M)*.

Пример 2.1. Если связь между сущностями МУЖЧИНЫ и ЖЕНЩИНЫ называется БРАК, то существует четыре возможных представления такой связи (рис. 2.3).

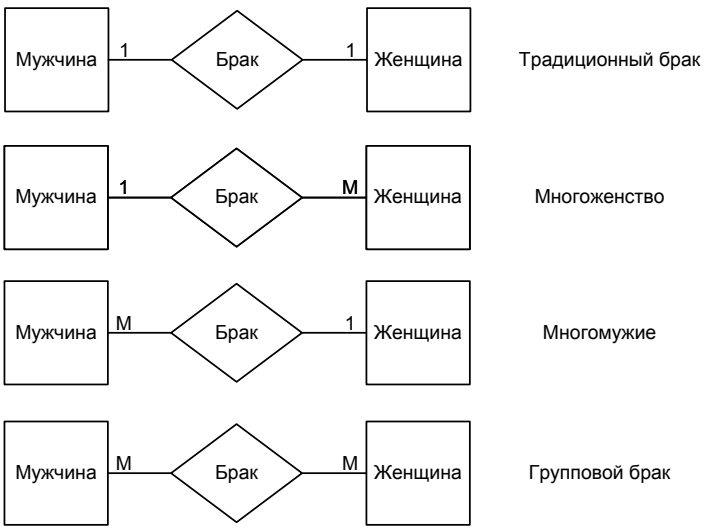

Рис. 2.3. Возможные связи между двумя сущностями

Характер связей между сущностями не ограничивается перечисленными. Существуют и более сложные связи:

- множество связей между одними и теми же сущностями (рис. 2.4, *а*): пациент, имея одного лечащего врача, может иметь также несколько врачей-консультантов; врач может быть лечащим врачом нескольких пациентов и может одновременно консультировать несколько других пациентов;
- тренарные связи (рис. 2.4, *б*): врач может назначить несколько пациентов на несколько анализов, анализ может быть назначен несколькими врачами нескольким пациентам и пациент может быть назначен на несколько анализов несколькими врачами;
- связи более высоких порядков, семантика (смысл) которых иногда очень сложна.

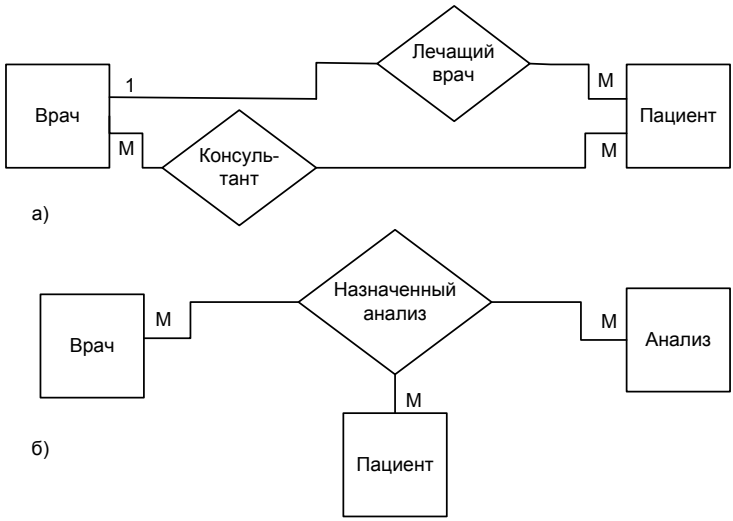

Рис. 2.4. Примеры сложных связей между сущностями

В приведенных примерах для повышения иллюстративности рассматриваемых связей не показаны атрибуты сущностей ER-диаграмм. Так, ввод атрибутов в описание брачных связей:

- фамилии, имена и отчества мужа и жены;
- даты и места рождения мужа и жены;
- даты и места регистрации брака;
- фамилий, присвоенных мужу и жене после регистрации;
- даты выдачи и номера свидетельства о браке

значительно усложнит ER-диаграмму и затруднит ее понимание.

Поэтому к настоящему времени появилось множество более удобных графических нотаций описания ER-диаграмм, поддержанных CASE-средствами (от Computer Aided Software/System Engineering) разработки информационных систем и редакторами деловой графики. В данной книге мы будем пользоваться нотациями системы построения диаграмм Microsoft Visio.

В этих нотациях сущность изображается в виде прямоугольника, в верхней части которого располагается имя сущности (рис. 2.5, *а*). В прямоугольнике перечислены атрибуты сущности. Атрибуты, расположенные сверху и отделенные от остальных горизонтальной линией, являются ключевыми.

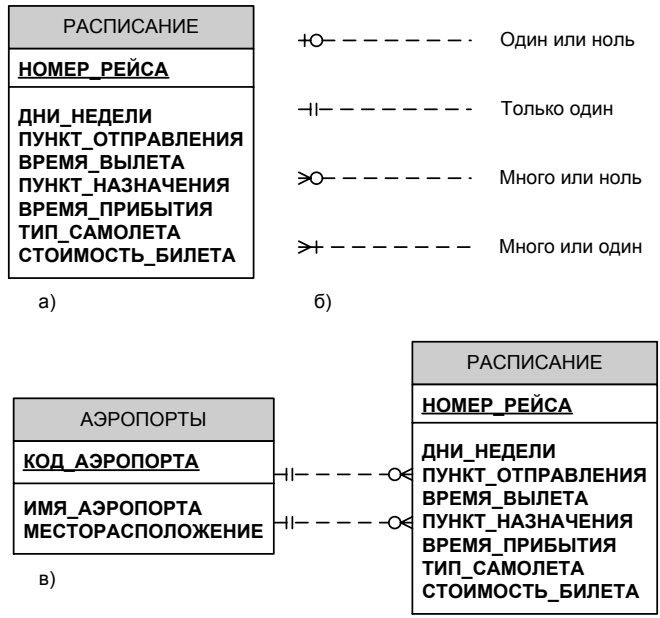

Рис. 2.5. Сущность (*а*), виды связей (*б*) и пример ER-диаграммы (*в*)

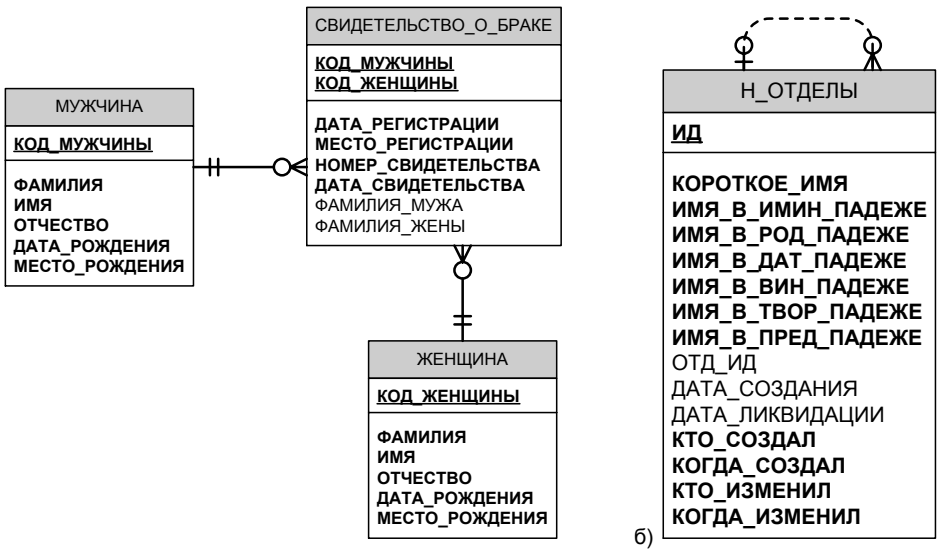

Рис. 2.6. Примеры связей в ER-диаграммах