Содержание

Глава 1
Введение в базы данных

Базы данных всегда были важнейшей темой при изучении информационных систем. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных.

Многие студенты считают этот предмет приятным и интересным, даже несмотря на его сложность. Проектирование и разработка базы данных требуют и искусства, и умения. Понимание пользовательских требований и перевод их в эффективный проект базы данных можно назвать творческим процессом. Преобразование этих проектов в физические базы данных с помощью функционально полных и высокопроизводительных приложений — инженерный процесс. Оба процесса полны сложностей и приятных интеллектуальных головоломок.

Поскольку сейчас существует большая необходимость в развитии технологии баз данных, навыки, которые вы разовьете, и знания, которые вы получите в процессе изучения этого курса, будут востребованы. Цель книги — предоставить твердое обоснование фундаментальных положений технологии баз данных, чтобы вы могли начать успешную карьеру в этой области, если вам этого захочется.

В этой главе мы обсудим, что, как и почему в базах данных. Мы поймем, почему используются базы данных, расскажем, какие существуют компоненты системы базы данных и как разрабатывать такие системы. Глава завершится экскурсом в историю баз данных.

Почему используются базы данных?

Цель базы данных — помочь людям и организациям вести учет определенных вещей. На первый взгляд, эта цель кажется скромной, и вы, возможно, удивитесь, зачем нам нужна такая сложная технология и целый курс, посвященный этому

предмету. Большинство из нас может вспомнить ситуации, в которых нам требуется отслеживать некоторые вещи. Я, например, составляю список дел, которые нужно сделать на этой неделе, список покупок в магазине, список расходов для налоговой декларации и так далее. Почему не делать то же самое для информационных систем?

На самых ранних стадиях развития информационных технологий использовались списки — набитые на перфокарте и написанные на магнитной ленте. Со временем, однако, стало ясно, что только немногие проблемы можно решить с помощью таких списков. В следующем разделе мы обсудим такие проблемы, а затем опишем, как построить базы данных для их решения.

Проблемы списков

Рассмотрим список в табл. 1.1, который используется фирмой Lakeview Equipment Rentals. Эта фирма занимается прокатом оборудования, такого как краны и экскаваторы, строительным подрядчикам. В табл. 1.1 показан лишь небольшой кусок деятельности этой фирмы, но даже этот маленький пример иллюстрирует целый ряд важных проблем со списками. Таблица имеет 10 столбцов: Job (проект), Contractor (подрядчик), Phone (телефон), Equipment type (тип оборудования), Equipment number (номер оборудования), Daily rate (плата за день), Start date (дата начала), End date (дата окончания), Days (количество дней), Charge (стоимость).

В первую очередь обратим внимание на следующее. Что случится, если у подрядчика изменится номер телефона? Предположим, например, что у фирмы KH Services телефон станет 213-444-9988. Чтобы содержать список в порядке, нам нужно сделать изменения в четырех строках. Если мы не изменим все эти строки, мы получим противоречащие друг другу данные и не сможем установить, какой же из этих номеров верен. Далее предположим, что список отражает прокатную деятельность фирмы за весь квартал или год. В таком списке могут быть сотни или тысячи строк. Чтобы записать измененный телефонный номер, нам понадобится произвести поиск по списку и сделать исправления в каждой строке, которая содержит фирму KH Services. Этот процесс очень долог и чреват ошибками.

Еще одна проблема возникает, когда мы удаляем из такого списка данные. Скажем, фирма RB Partnership решила не брать в прокат экскаватор (backhoe), так что мы должны удалить последнюю строку. В процессе такого удаления не только теряются данные о прокате, но также теряются номер телефона фирмы RB Partnership и тот факт, что эта фирма работает над проектом под названием Village Square и что она согласилась взять в прокат экскаватор за 750 долларов в день. А ведь мы, скорее всего, просто хотели удалить данные о дате проката.

Такого же типа и следующая ситуация. Что делать, когда у нас есть только некоторые (но не все) данные? Допустим, мы знаем, что университет Swaging имеет телефон 206-555-8989 и что он работает над реконструкцией моста на Центральной улице. Мы хотим записать эти данные, но куда их поместить в списке? Ведь эта компания еще ничего не брала у нас в прокат.

Таблица 1.1. Список Lakeview Equipment

Job	Contractor	Phone	Equipment type	Equipment number	Daily rate	Start date	End date	Days	Charge
Sea View Bldg	KH Sercices	213-222-1181	Backhoe	10400	$750	17.06.2002	19.06.2002	3	$2250
Highland Center	Comstock, Inc	232-492-3383	Backhoe	10400	$750	24.06.2002	24.06.2002	1	$750
Sea View Bldg	KH Sercices	213-222-1181	Middle crane	335	$350	17.06.2002	3.07.2002	17	$5950
Long Plaza	KH Sercices	213-222-1181	Backhoe	10020	$650	1.07.2002	3.07.2002	3	$1950
Sea View Bldg	KH Sercices	213-222-1181	Scaffolding		$135	15.06.2003			
Highland Center	Comstock, Inc	232-492-3383	Middle crane	335	$400	1.07.2002	8.07.2002	8	$3200
Village Square	RB Partnership	508-555-3233	Backhoe	10020	$750	8.07.2002	11.07.2002	4	$3000

Другая проблема состоит в несовместимости данных. Мы можем сделать простую опечатку в имени подрядчика и случайно написать KJ Services вместо KH Services. Пользователи этого списка не поймут, то ли у нас появился новый клиент, то ли это ошибка. Такие ошибки не могут возникнуть, если мы выберем подрядчика из списка имеющихся.

Есть и более тонкие несоответствия. Проверим, например, четвертую и пятую строки. Обе они — о прокате экскаватора с номером 10020. В четвертой строке ежедневная плата 650 долларов, а в последней — 750 долларов. Является ли это несовпадение следствием опечатки, или арендная плата меняется в зависимости от дня? Или, может быть, для разных клиентов существуют разные цены? В этом случае KH Services платит за экскаватор 650 долларов в день, а RB Partnership — 750. Если это правда, то мы должны убедиться, что каждый раз, когда мы вводим сочетание фирмы KH Services с экскаватором под номером 10020, мы должны вводить плату 650 долларов в день.

Другая проблема касается отсутствующих данных. Например, в записи о прокате строительных лесов нет конечной даты. Ошибка ли это, или это означает что-нибудь другое? Это может значить, что прокат еще не закончен или что компания KH Services хочет держать леса неопределенно долго.

Проблемы совместно используемых данных

Другие проблемы использования списка всплывают, когда мы рассматриваем данные, которые совместно используются многими людьми в компании Lakeview Equipment Rentals. Компания хочет дать каждому доступ к именам подрядчиков и к телефонным номерам из общего источника данных. Таким образом, если кто-нибудь изменит телефонный номер, то это изменение нужно сделать в общем источнике данных компании. В противном случае сотрудникам придется изменить эти данные на каждом компьютере компании.

Однако если данные о подрядчике используются совместно, возникают другие проблемы. Бухгалтерия хочет вести учет счетов подрядчиков и платежей. Сотрудники отдела проката хотят отслеживать координаты подрядчиков, встречи и заказы. Отдел по работе с клиентами хочет знать, какие проблемы возникали с определенными подрядчиками и как они решались. Все эти отделы не обязательно хотят использовать данные совместно с другими отделами. Бухгалтерия, например, не хочет, чтобы кто-то другой имел доступ к данным о счетах и платежах. Отдел по работе с клиентами не хочет, чтобы кто-то в компании знал, какие проблемы есть у клиентов. Таким образом, отделы готовы совместно использовать только некоторые данные, но не все.

Существуют и другие проблемы использования списков, но идея уже понятна: нужен более сильный способ хранения данных.

Базы данных как группы связанных таблиц

Одна серьезная проблема со списками, подобными изображенному в табл. 1.1, состоит в том, что они содержат данные на различные темы. Помните вашу учительницу родного языка? Она наверняка говорила вам, что абзац должен быть посвящен только одной теме. Если у вас есть абзац, содержащий две темы или более, следует его разбить на несколько, каждый из которых говорит только об одной теме. Это суть процесса под названием *нормализация*, который мы глубоко исследуем в главе 4.

Проверим список в табл. 1.1. Сколько тем он содержит? Одна — строительные проекты, другая — подрядчики, третья — оборудование и четвертая — прокат. На рис. 1.1 показаны результаты разбиения списка на отдельные компоненты. Каждый из этих компонентов представляет собой таблицу данных, поэтому мы будем называть их *таблицами*. Таких таблиц получилось четыре — по количеству тем: CONTRACTOR (подрядчики), JOB (проекты), EQUIPMENT (оборудование) и RENTAL (прокат).

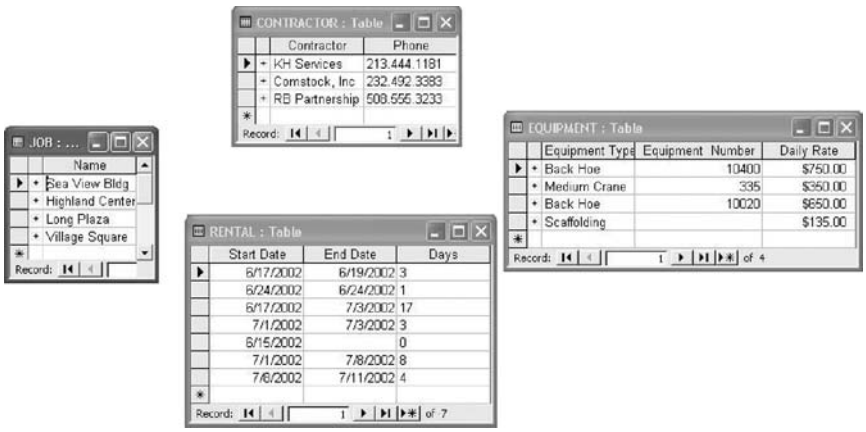


Рис. 1.1. Данные Lakeview Equipment в таблице

Когда список разбивается на четыре таблицы, многие проблемы, которые мы обсуждали, исчезают. Если компания KH Services меняет свой телефон на 213-444-9988, мы делаем это изменение только однажды — в таблице CONTRACTOR. Если мы хотим удалить запись о прокате, мы всего лишь удаляем строку из таблицы RENTAL, не теряя при этом никаких данных о проектах, подрядчиках или оборудовании. Если мы хотим добавить данные о новом клиенте, то просто добавляем эти данные в таблицу CONTRACTOR.

Что касается противоречивости данных, то мы можем разработать наше приложение баз данных так, что когда кому-нибудь понадобится сделать новую запись о прокате, он просто выберет запись в таблице CONTRACTOR. Таким образом, ошибку сделать нельзя. Это также дает нам возможность контролировать появление новых клиентов. Можно сделать так, что только определенные люди, например, бухгалтеры, смогут добавлять новые строки в таблицу CONTRACTOR.

А как насчет противоречивых данных о стоимости проката в табл. 1.1? Мне кажется, тут и начинается веселье. На рис. 1.1 стоимость проката в день (Daily rate) помещена в таблицу EQUIPMENT. Это значит, что оборудование должно даваться в прокат за одну и ту же плату всем клиентам. Так спроектирована эта таблица. Является ли это правильным? Это зависит от бизнес-правил компании Lakeview. Если, однако, плата зависит от каждого конкретного случая проката, то столбец Daily rate нужно поместить в таблицу RENTAL. Если каждый подрядчик платит за прокат по собственным расценкам, нужно построить новую таблицу.

Главное здесь не в решении этой частной проблемы. Главное — что проектирование таблицы должно отражать порядки организации, которая ее использует. В этой книге мы потратим довольно много времени и усилий на подробный разбор каждой такой ситуации.

Разбиение на таблицы также позволяет нам добавлять больше данных по каждой теме, чем появляется в списке в табл. 1.1. На рис. 1.2, например, показаны дополнительные данные для подрядчиков. При необходимости мы можем разработать приложения нашей базы данных так, что только определенный персонал сможет знакомится с этими дополнительными данными.

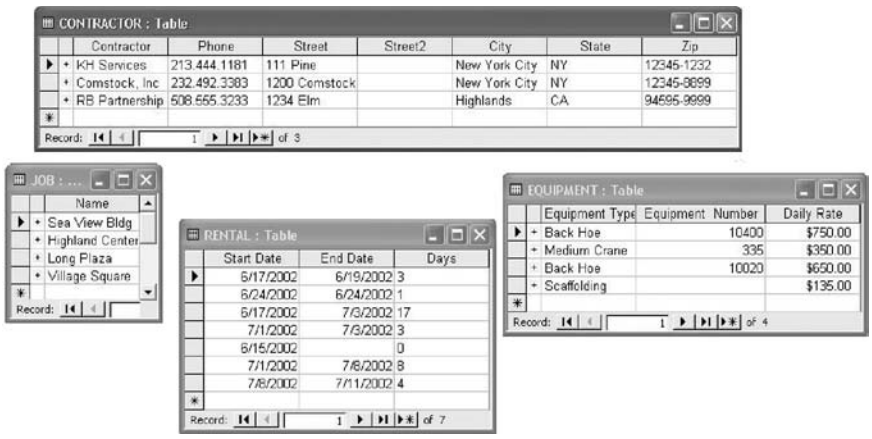


Рис. 1.2. Таблица CONTRACTOR и дополнительные данные

Связи

Сейчас вас может испугать следующий вопрос. Данные на рис. 1.1 и 1.2 хороши, но где же связи между ними? Как можно узнать, какой подрядчик взял напрокат какое оборудование, для какого проекта и на какой срок? По данным на этих рисунках невозможно сказать, что к чему относится.

Мы пришли к одному из фундаментальных вопросов теории баз данных. Этот вопрос мы будем обсуждать в главах 5 и 8 в теме о проектировании баз данных. Сейчас рассмотрим рис. 1.3, на котором изображен один из способов построить связи между таблицами.

Каждой строке в каждой таблице дается уникальный идентификатор ID. Этот идентификатор для пользователей фирмы Lakeview не имеет никакого смысла;

его цель — просто пронумеровать все строки таблицы. Значения идентификатора мы будем использовать для представления связи со строками таблицы RENTAL.

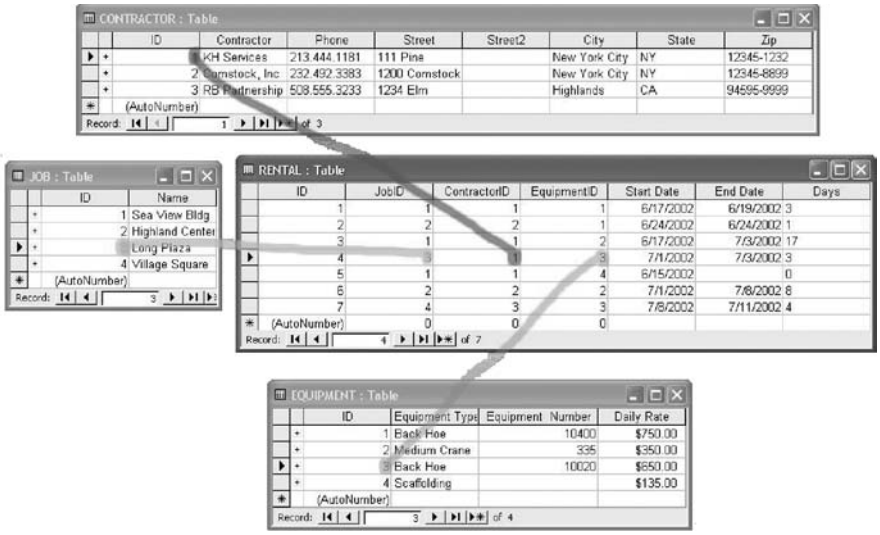


Рис.1.3. Таблицы Lakeview и связи между ними

Рассмотрим четвертую строку таблицы RENTAL на рис. 1.3. Значение 3 в столбце, обозначенном JobID (идентификатор проекта) означает, что этот прокат относится к проекту, который имеет в таблице JOB идентификатор, равный 3. Это строка, содержащая проект Long Plaza, который соотносится с данными в табл. 1.1. Подобным же образом располагаются в таблице прокат для подрядчика, имеющий идентификатор 1 (KH Services), и для оборудования с ID, равным 3 (экскаватор номер 10020).

Список в табл. 1.1. содержит столбец, обозначенный Charge (стоимость), но такого столбца нет на рис. 1.3. Причина в том, что мы можем вычислить этот столбец, если он нам понадобится. Так и будет в базе данных. Стоимость будет вычисляться каждый раз, а не храниться. Хорошо это спроектировано или нет, опять же зависит от порядков в Lakeview. Следствие такого проектирования состоит в том, что если значение платы за день в таблице RENTAL меняется, то все долги при следующем вычислении будут основаны уже на новой цене. Таким образом, мы можем получить значение стоимости, отличное от того, что пользователь платил в свое время. Подобные ситуации мы еще будем обсуждать в последующих главах.

Подведем итог. Чтобы справиться с проблемами, которые возникают со списками, подобными показанному в табл. 1.1, мы разбили список на четыре таблицы, каждая из которых посвящена определенной теме. После этого мы связали таблицы с помощью уникального идентификатора. Как вы потом узнаете, этот уникальный идентификатор называется *ключом*. Ключ, который представляет связь, вроде JobID в таблице RENTAL, называется *внешним ключом*. Эти термины мы будем использовать много раз в наших последующих обсуждениях.

Объединение таблиц

«Но где же наш список?» — спросите вы. На рис. 1.3. все изображено красиво и правильно, но пользователи в фирме Lakeview, по крайней мере некоторые из них, захотят увидеть список типа того, с которого мы начинали. Где он?

В главах 6–8 будет изучаться язык, который является настоящим подъязыком данных. Называется он SQL (Structured Query Language, язык структурированных запросов) и используется для манипуляций с таблицами типа изображенных на рис. 1.3. SQL — это промышленный стандарт, который поддерживается всеми основными системами управления базами данных (СУБД). Приведенный ниже оператор SQL используется в Access для объединения таблиц и отображения их в том виде, как в табл. 1.1.

```
SELECT JOB.Name, CONTRACTOR.Contractor, CONTRACTOR.Phone,
 EQUIPMENT.[Equipment Type], EQUIPMENT.[Equipment Number],
 EQUIPMENT.[Daily Rate], RENTAL.[Start Date], RENTAL.[End Date],
 RENTAL.Days,[Daily Rate]*[Days] AS Charge
FROM JOB, EQUIPMENT, CONTRACTOR, RENTAL
WHERE CONTRACTOR.ID = RENTAL.ContractorID
ANDEQUIPMENT.ID = RENTAL.ContractorID
ANDJOB.ID = RENTAL.JobID;
```

Результат выполнения этого оператора показан на рис. 1.4. Там содержится список, с которого мы начинали, но с исправленными ошибками.

Rental Data Query : Select Query

Job	Contractor	Phone	Equipment Type	Equipment Number	Daily Rate	Start Date	End Date	Days	Charge
Sea View Bldg	KH Services	213.444.1181	Back Hoe	10400	$750.00	6/17/2002	6/19/2002	3	$2,250.00
Highland Center	Comstock, Inc	232.492.3363	Back Hoe	10400	$750.00	6/24/2002	6/24/2002	1	$750.00
Sea View Bldg	KH Services	213.444.1181	Medium Crane	335	$350.00	6/17/2002	7/3/2002	17	$5,950.00
Long Plaza	KH Services	213.444.1181	Back Hoe	10020	$650.00	7/1/2002	7/3/2002	3	$1,950.00
Sea View Bldg	KH Services	213.444.1181	Scaffolding		$135.00	6/15/2002		0	$0.00
Highland Center	Comstock, Inc	232.492.3363	Medium Crane	335	$350.00	7/1/2002	7/8/2002	8	$2,800.00
Village Square	RB Partnership	508.556.3233	Back Hoe	10020	$650.00	7/8/2002	7/11/2002	4	$2,600.00

Record: 1 of 7

Рис. 1.4. Исходный список, построенный на основе таблиц

Приведенный выше SQL-оператор можно использовать, чтобы определить общую структуру. Но понять все в подробностях он не поможет. Когда вы закончите чтение этой книги, вы легко сможете писать такие и более сложные операторы.

Что такое система обработки базы данных?

На рис. 1.5 показаны четыре основных элемента базы данных. Слева показано, как пользователи выполняют свои задачи с помощью базы данных. Они вводят новые данные, модифицируют существующие и удаляют ненужные. Кроме того, они разными способами читают данные: посредством форм, запросов и путем генерации отчетов. Следующий компонент, приложение баз данных, представляет собой множество из одной или более компьютерных программ, которые служат

посредниками между пользователем и СУБД (программой, обрабатывающей базу данных). Приложение создает формы, запросы и отчеты, посылает данные пользователю и получает их от него, а также преобразует действия пользователя в запросы для управления данными с помощью СУБД.

Цель СУБД состоит в получении запросов от приложений и в преобразовании этих запросов в файлы ввода или вывода базы данных. В большинстве случаев СУБД посылает SQL-операторы и переводит их в инструкции операционной системы компьютера для чтения и записи данных в файлы базы данных. Теперь рассмотрим функции программы приложения и СУБД более подробно.

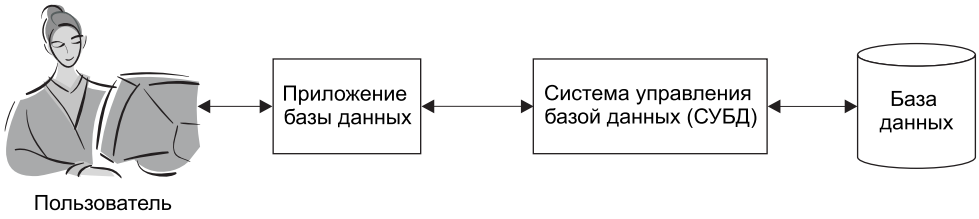


Рис. 1.5. Компоненты системы базы данных

Функции прикладных программ

На рис. 1.6 перечислены функции прикладных программ баз данных и СУБД. В первую очередь приложение создает и обрабатывает формы. Веб-приложение, например, генерирует HTML и другие конструкции веб-форм, которые заставляют форму отображаться на компьютере пользователя. Когда пользователь заполняет форму и посылает данные обратно, приложение определяет, какие таблицы данных нуждаются в модификации, и посылает запросы к СУБД, чтобы вызвать необходимую модификацию. Если во время этого процесса возникают ошибки, приложение получает сообщение об ошибке и генерирует подходящее сообщение для пользователя или осуществляет какое-нибудь другое действие.

Вторая функция прикладной программы, показанная на рис. 1.6, — создание и передача запросов. Сначала приложение генерирует запрос к СУБД. Такие запросы почти всегда пишутся на языке SQL. После обработки запроса его результаты форматируются и передаются пользователю.

Третья функция похожа на вторую. Приложение запрашивает данные у СУБД (опять с помощью SQL) и форматирует результаты запроса в виде отчета.

Кроме создания форм, запросов и отчетов, приложение также производит другие действия по изменению базы данных в соответствии с логикой, специфичной для этого приложения. Например, в некотором приложении пользователь запросил 10 единиц определенного товара. Теперь предположим, что когда прикладная программа произвела запрос базы данных (через СУБД), она нашла только 8 единиц в наличии. Что следует сделать? Это зависит от логики данного конкретного приложения. Возможно, следует не брать ничего со склада и уведомить пользователя о возникшей ситуации, или же взять 8 единиц, а 2 оформить как отложенный заказ. Могут быть и другие варианты. В любом случае, реализация соответствующей логики — задача прикладной программы.

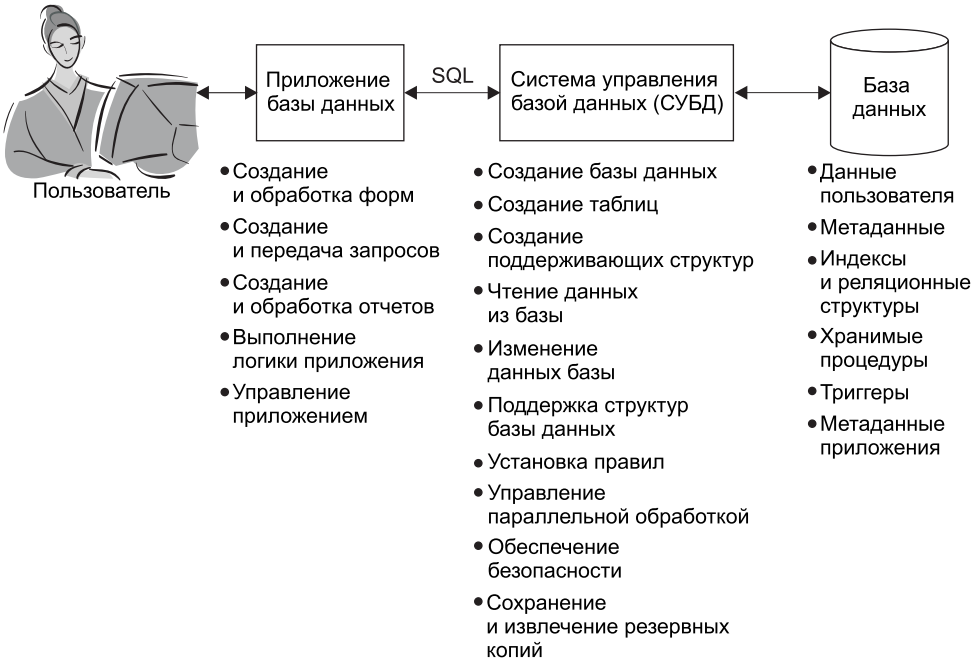


Рис. 1.6. Функции и содержимое компонентов обработки базы данных

Последняя функция прикладной программы, указанная на рис. 1.6, — управление приложением. Существует два способа осуществлять это управление. Во-первых, приложение должно быть написано так, чтобы пользователю были видны только определенные опции. Приложение может сгенерировать меню с вариантами для пользователя. При этом оно должно убедиться, что пользователю доступны только нужные варианты. Во-вторых, приложение должно управлять данными с помощью СУБД. Оно может, например, указать СУБД сделать определенный набор изменений в данных. СУБД может быть приказано произвести все эти изменения или не делать ни одного из них.

Функции СУБД

В противоположность прикладным программам, которые часто пишутся самими компаниями, их использующими, почти все СУБД являются коммерческими продуктами. К коммерческим СУБД относятся Oracle от корпорации Oracle, DB2 от корпорации IBM, а также Access и SQL Server от корпорации Microsoft. Существуют десятки СУБД, но эти четыре захватили львиную долю рынка.

Функции СУБД перечислены на рис. 1.6. СУБД используется для создания самой базы данных, таблиц и других поддерживаемых структур — например, индексов.

Следующие две функции СУБД — чтение и изменение данных в базе данных. Для этого СУБД получает запросы SQL (или какие-нибудь другие запросы)

и преобразует их в действия по отношению к базе данных. Другая функция СУБД — поддержка всех структур базы данных. Например, время от времени может понадобиться изменить формат таблицы или другую поддерживаемую структуру. Для таких изменений программисты используют СУБД.

При работе с большинством СУБД можно устанавливать правила, касающиеся значений данных. Например, рассмотрим рис. 1.3: что случится, если пользователь ошибочно введет значение 4 в столбец ContractorID (идентификатор подрядчика) в таблице RENTAL? Такого подрядчика нет, и это значение вызовет множество ошибок. Чтобы предотвратить такую ситуацию, можно объявить СУБД, что любое значение ContractorID в таблице RENTAL должно уже быть значением идентификатора в таблице CONTRACTOR. Если такого значения нет, вставка и запрос о модификации разрешаться не будут. Такие правила, которые называются *ограничениями ссылочной целостности*, устанавливаются СУБД.

Последние три функции СУБД, указанные на рис. 1.6, относятся к управлению базой данных. СУБД контролирует работу, следя, чтобы изменения одного пользователя не пересекались с изменениями другого. Эта важная (и сложная) функция будет обсуждаться в главе 9. Кроме того, СУБД содержит систему безопасности, которая используется для проверки того, что только авторизованные пользователи выполняют определенные действия с базой данных. Пользователям могут не позволить знакомиться некоторыми данными, а также ограничить их действия только определенными изменениями в определенных данных.

Наконец, база данных, как централизованное хранилище данных, является ценным имуществом организации. Рассмотрим, например, ценность базы данных книг в компании типа Amazon.com. Ввиду крайней важности этой базы данных нужно принимать меры по предотвращению потерь данных из-за случайных ошибок, проблем аппаратного обеспечения или природных катастроф. СУБД обеспечивает возможность резервного копирования данных из базы данных и восстановления их в случае необходимости.

Определение и компоненты базы данных

На рис. 1.6 компонент системы базы данных, находящийся справа, — сама база данных. Перед тем как продолжать, мы должны определить термин *база данных* и описать ее компоненты.

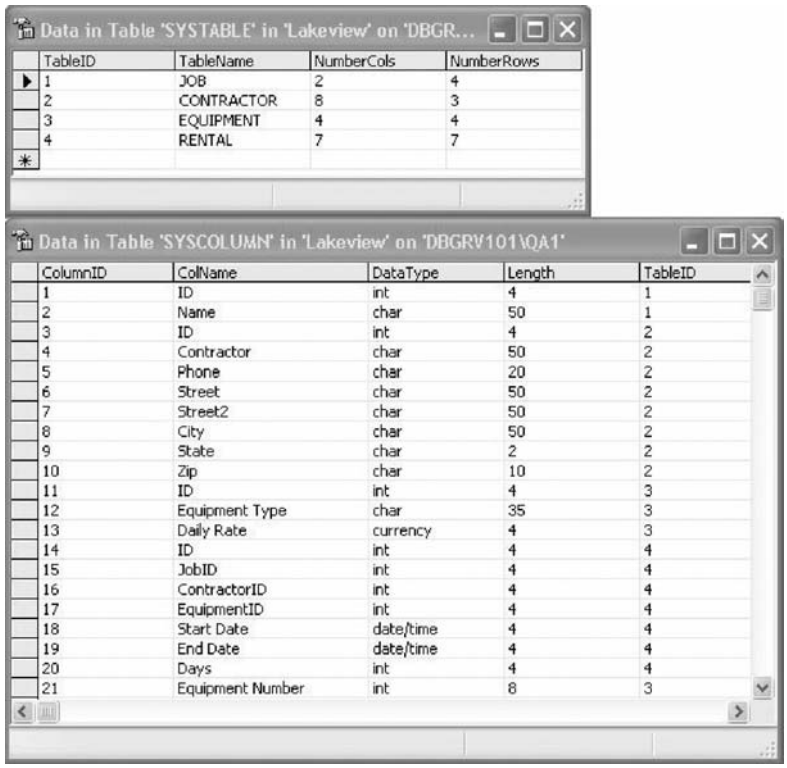
В общем случае, можно сказать, что *база данных — это самодокументированное собрание интегрированных записей*. Для всех реляционных баз данных (а это сегодня почти все базы данных, и только этот тип рассматривается в данной книге) это определение можно модифицировать так: *база данных — это самодокументированное собрание связанных таблиц.*

Ключевые слова в этом определении — *самодокументированность* и *связанные таблицы*. Вы уже примерно представляете, что мы подразумеваем под связанными таблицами. Примерами являются таблицы JOB, CONTRACTOR, EQUIPMENT и RENTAL. Ваше понимание этого углубится в главах 5 и 8.

Под *самодокументированностью* мы подразумеваем, что описание структуры базы данных содержится в самой базе данных. Благодаря этому мы всегда можем узнать содержимое базы данных, просто посмотрев на нее. Нам не нужно смотреть куда-то еще. Эта ситуация похожа на ситуацию с библиотекой в вашем студгородке. Можно сказать, что находится в библиотеке, просмотрев карточки каталога.

Данные о структуре базы данных называются метаданными. Примерами метаданных служат имена таблиц, имена столбцов и таблицы, которым они принадлежат, а также свойства таблиц и столбцов, и так далее.

На рис. 1.7 показаны метаданные для базы данных Lakeview, которую мы рассматривали ранее. Таблица под названием SYSTABLES содержит данные о каждой из таблиц базы данных, а вторая таблица (SYSCOLUMNS) содержит данные о каждом из столбцов и о связях столбцов с таблицами. Этот рисунок — только пример метаданных. Таблицы, подобные этим, существуют внутри баз данных, обрабатываемых продуктами типа Oracle, SQL Server или DB2, но они более сложные. Заметим, однако, что сами метаданные содержатся в таблицах. Это значит, что осведомленный персонал может использовать SQL для запросов к метаданным, так же, как и к пользовательским данным.



Data in Table 'SYSTABLE' in 'Lakeview' on 'DBGR...

TableID	TableName	NumberCols	NumberRows
1	JOB	2	4
2	CONTRACTOR	8	3
3	EQUIPMENT	4	4
4	RENTAL	7	7

Data in Table 'SYSCOLUMN' in 'Lakeview' on 'DBGRV101\QA1'

ColumnID	ColName	DataType	Length	TableID
1	ID	int	4	1
2	Name	char	50	1
3	ID	int	4	2
4	Contractor	char	50	2
5	Phone	char	20	2
6	Street	char	50	2
7	Street2	char	50	2
8	City	char	50	2
9	State	char	2	2
10	Zip	char	10	2
11	ID	int	4	3
12	Equipment Type	char	35	3
13	Daily Rate	currency	4	3
14	ID	int	4	4
15	JobID	int	4	4
16	ContractorID	int	4	4
17	EquipmentID	int	4	4
18	Start Date	date/time	4	4
19	End Date	date/time	4	4
20	Days	int	4	4
21	Equipment Number	int	8	3

Рис. 1.7. Пример метаданных

Все поставщики СУБД предоставляют набор средств отображения структуры баз данных. Например, на рис. 1.8 показано использование команды Describe в базе данных Oracle. Как видно, эта команда выдает список имен и свойств столбцов в таблице.

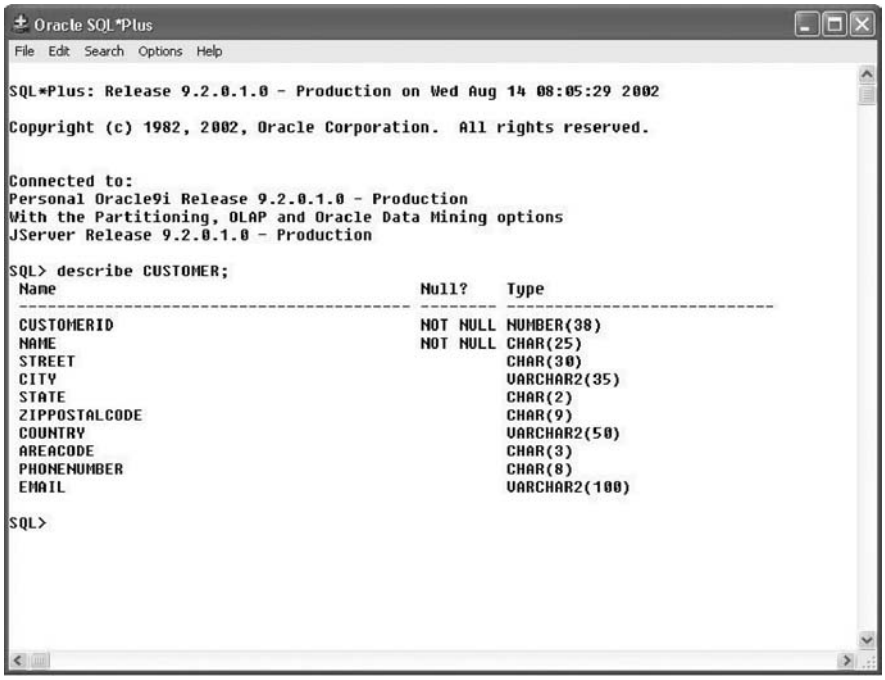


Рис. 1.8. Использование команды Dercribe из Oracle

Согласно рис. 1.5, база данных содержит пользовательские данные и метаданные, как было только что описано. Кроме того, база данных имеет индексы и другие структуры, которые существуют для облегчения представления баз данных. Впоследствии мы поговорим об этих структурах подробнее. Сейчас скажем только, что индекс похож на предметный указатель в конце книги и показывает, где искать определенные записи в таблице. Например, можно построить индекс EquipmentID (идентификатор оборудования) в таблице RENTAL. Этот индекс можно использовать для быстрого нахождения всех строк в таблице RENTAL, имеющих определенное значение идентификатора.

Хранимая процедура — это программа, которая хранится в базе данных. Некоторые хранимые процедуры являются утилитами для базы данных. Например, Lakeview может иметь хранимую процедуру, которая удаляет все данные о прокате более чем годовалой давности и все платежи, которые были сделаны за прокат. Такой хранимой процедуре может потребоваться проверка множества таблиц в базе данных и реализация сложной логики. Другие хранимые процедуры реализуют логику приложения. Например, можно написать хранимую процедуру для генерации списка задолженностей.

Триггер — это процедура, которая выполняется при возникновении определенных ситуаций в данных. База данных Lakeview может, скажем, иметь триггер `CustomerCheck`, который проверяет, имеет ли клиент хорошую репутацию, перед сохранением каких-либо новых данных о прокате для него. При попытке добавить строку в таблицу `RENTAL` (прокат) СУБД сначала загрузит и обработает триггер, а потом уже сделает изменение. В этом случае триггер не позволит вставку новой строки, если репутация клиента плохая. Подобно хранимым процедурам, триггеры содержатся в базе данных. Хранимые процедуры пишутся или на языке, уникальном для конкретной СУБД, или на языках общего назначения типа Java. О хранимых процедурах и триггерах вы узнаете подробнее из глав 10 и 11.

Наконец, некоторые базы данных содержат метаданные приложений — простые данные, которые описывают элементы приложения, такие как формы и отчеты. Например, Microsoft Access содержит метаданные приложения как часть своих баз данных.

Три примера систем баз данных

Технология баз данных может использоваться в широком спектре приложений. На одном его конце исследователь может использовать технологию баз данных для отслеживания результатов экспериментов, проводимых в лаборатории. В такой базе данных может быть всего несколько таблиц, и каждая из них может иметь максимум несколько сотен строк. Исследователь может быть единственным пользователем такого приложения.

На другом конце спектра — гигантские базы данных, которые поддерживают международные организации. Такие базы данных содержат сотни таблиц по миллиону строк данных каждая и используются одновременно тысячами пользователей. Эти базы данных работают круглосуточно и без выходных. Сделать даже обычную копию такой базы данных — трудная задача.

В этом разделе показаны три разных приложения баз данных: одно — для единичного пользователя, второе — для малого бизнеса и третье — для большого правительственного агентства.

Малярная фирма Мэри Ричардс

Мэри Ричардс — профессиональный маляр, владеет и управляет небольшой компанией, состоящей из нее самой, еще одного профессионального маляра и, когда это необходимо, наемных работников. Мэри занимается этим бизнесом уже 10 лет и приобрела за это время репутацию высококвалифицированного маляра, работающего за умеренную плату. Большую часть заказов она получает от постоянных клиентов, нанимающих ее для покраски домов, а также от их знакомых. Кроме того, некоторое количество заказов Мэри получает от строительных подрядчиков и профессиональных дизайнеров интерьера.

Клиенты помнят Мэри намного лучше, чем она их. Порою она бывает смущена, когда клиент звонит ей и говорит что-нибудь такое: «Здравствуйте, Мэри, это

Джон Мэплз. Вы красили мой дом три года назад». При этом звонящий подразумевает, что Мэри вспомнит его и работу, которую она для него сделала; но если учесть, что Мэри красит более 50 домов в год, для нее это будет затруднительно. Ситуация становится еще хуже, когда клиент заявляет: «Моей соседке понравилось, как вы покрасили наш дом, и она хочет, чтобы вы и у нее сделали что-нибудь подобное».

Чтобы несколько разгрузить свою память и лучше организовать учет деятельности фирмы, Мэри наняла консультанта для разработки базы данных, которую она могла бы хранить на своем персональном компьютере. В базе данных должны храниться записи о клиентах, заказах и поставщиках клиентов, представленные в форме таблиц (рис. 1.9).

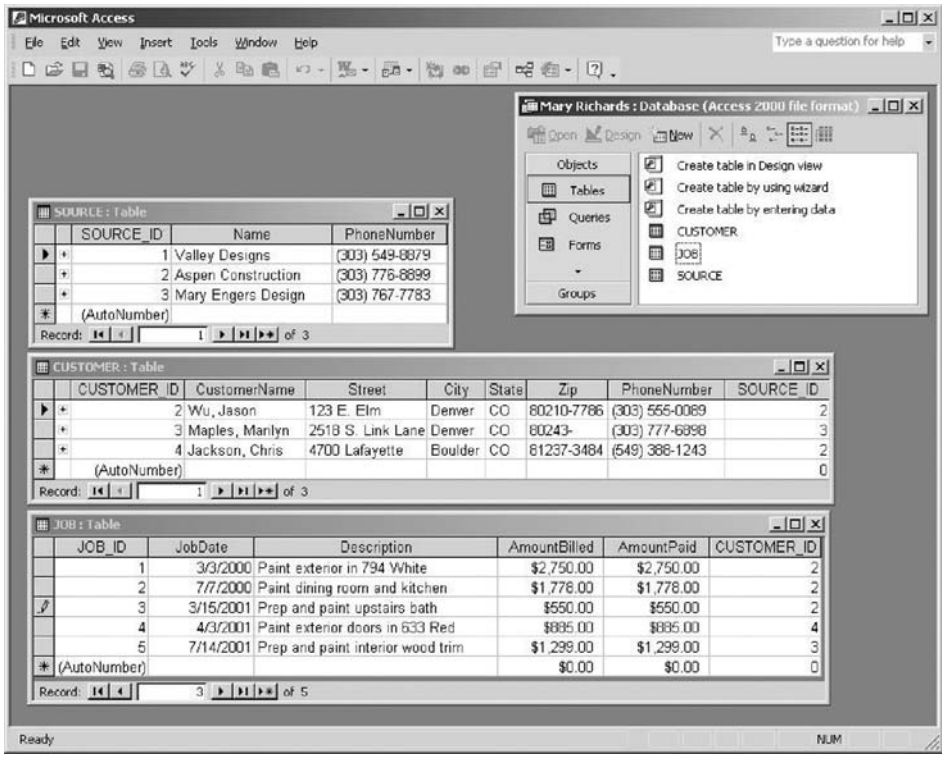


Рис. 1.9. Таблицы данных для малярной фирмы Мэри Ричардс

Как мы уже говорили, СУБД хранит данные в таблицах и извлекает их оттуда. К сожалению, эти данные, будучи представлены в форме таблиц, не слишком полезны для Мэри. Ей, скорее, хотелось бы знать, как клиенты и заказы связаны друг с другом — например, какие работы были выполнены ею для конкретного клиента или какие клиенты были направлены к ней конкретным человеком.

Чтобы предоставить Мэри такую возможность, консультант создал *прикладную программу* (application program), которая обрабатывает *формы* (forms) для

ввода данных и формирует *отчеты* (reports). Рассмотрим пример, представленный на рис. 1.10. В изображенную здесь форму Мэри вводит личные данные клиента — имя, телефон и адрес. Далее она вводит сведения о работах, выполненных для клиента, и указывает, кто направил этого клиента к ней. Эти данные могут быть затем отражены в отчете, пример которого показан на рис. 1.11. Другие функции этой базы данных включают оценку стоимости заказа, учет поставщиков клиентов и создание наклеек с адресом для рекламных буклетов, которые Мэри время от времени рассылает.

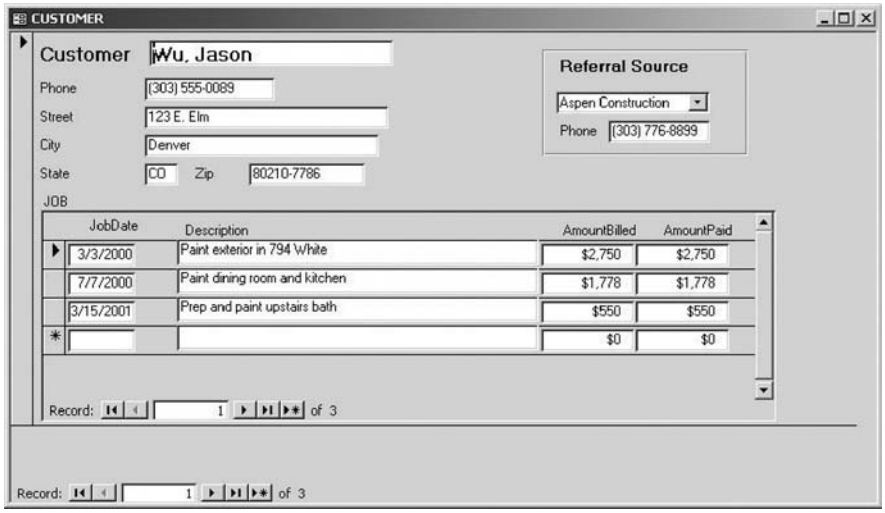


Рис. 1.10. Пример формы ввода данных для малярной фирмы Мэри Ричардс

Прикладная программа и СУБД обрабатывают заполненную форму и переписывают данные из нее в таблицы, подобные изображенным на рис. 1.9. Аналогичным образом прикладная программа и СУБД извлекают данные из этих таблиц для создания отчетов, таких, как на рис. 1.11.

Еще раз вернувшись к рис. 1.9, обратите внимание, что строки таблицы содержат перекрестные ссылки и поэтому оказываются связанными друг с другом. Для каждой работы (`JOB`) указан номер клиента (`CUSTOMER_ID`), заказавшего эту работу, и для каждого клиента (`CUSTOMER`) указан номер поставщика клиента (`SOURCE_ID`), то есть человека, направившего этого клиента к Мэри. Эти ссылки используются для объединения данных в формы и отчеты (рис. 1.10 и 1.11).

Как вы можете догадаться, Мэри вряд ли имеет представление о том, как проектировать таблицы, создавать их с помощью СУБД и разрабатывать прикладную программу для создания форм и отчетов. Но ваших знаний о технологии баз данных к моменту окончания этого курса должно хватить для того, чтобы суметь разработать такую базу данных и прикладную программу для работы с ней. Вы должны будете также уметь проектировать таблицы и манипулировать ими для создания довольно сложных форм и отчетов.

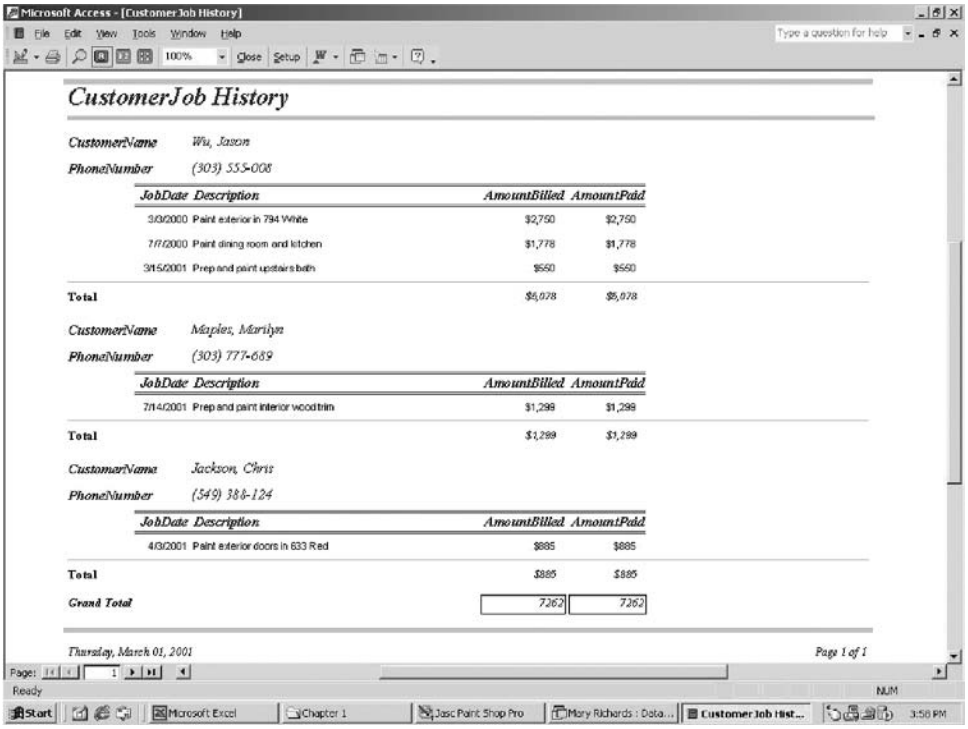


Рис. 1.11. Пример отчета для малярной фирмы Мэри Ричардс

Бюро проката музыкальных инструментов Treble Clef Music

База данных Мэри Ричардс называется *однопользовательской* (single-user), поскольку в каждый конкретный момент времени к ней обращается только один пользователь. В некоторых случаях такое ограничение неприемлемо: иногда требуется, чтобы одновременно к базе данных могли обращаться несколько человек с различных компьютеров. Такие *многопользовательские* (multi-user) базы данных являются более сложными, поскольку СУБД и прикладные программы должны заботиться о том, чтобы действия одного пользователя не противоречили действиям другого.

Бюро проката Treble Clef Music использует базу данных для учета сдаваемых в аренду музыкальных инструментов. Для этого требуется многопользовательская база данных, поскольку в периоды наплыва клиентов выдачей музыкальных инструментов могут одновременно заниматься несколько служащих. Кроме того, менеджер также должен иметь доступ к базе данных, чтобы определить момент, когда необходимо будет заказать большее количество определенных инструментов. При этом менеджер не должен мешать процессу выдачи инструментов.

Бюро Treble Clef Music имеет локальную сеть, соединяющую несколько персональных компьютеров с сервером, на котором находится база данных (рис. 1.12).

У каждого из служащих есть доступ к прикладной программе, позволяющей работать с тремя видами форм (рис. 1.13). Форма CUSTOMER содержит информацию о клиенте, форма RENTAL AGREEMENT представляет договор аренды и используется для учета выдачи и возврата инструментов, а форма INSTRUMENT содержит сведения об инструменте и историю его аренды.

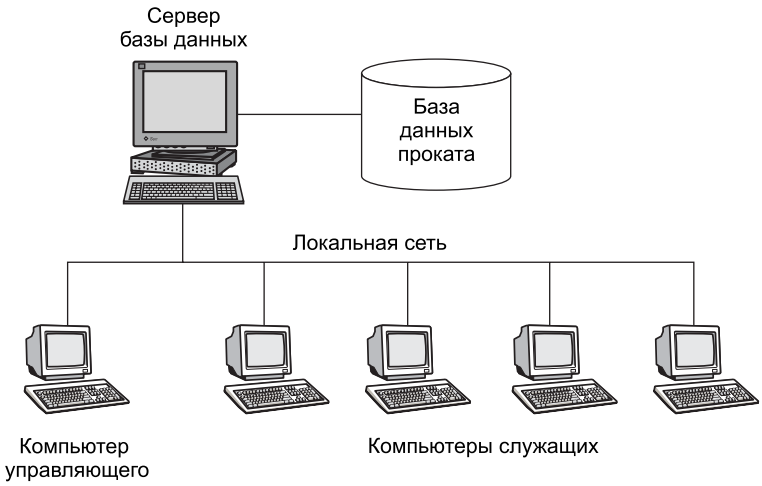


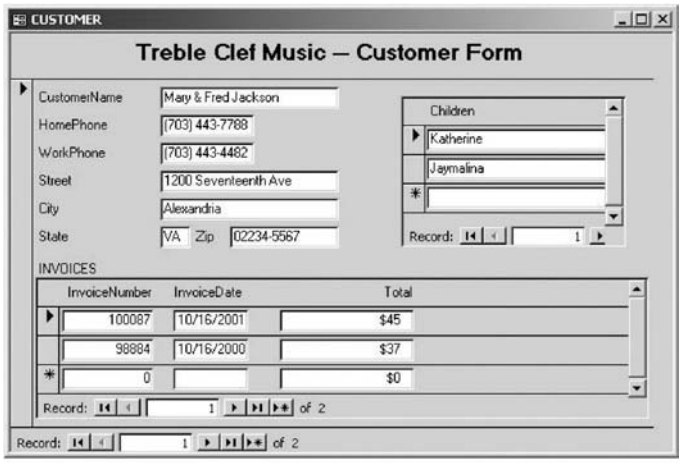
Рис. 1.12. Локальная сеть бюро проката Treble Clef Music

Чтобы уяснить проблемы, которые необходимо преодолеть в многопользовательской базе данных, представьте себе, что произойдет, если два клиента одновременно попытаются взять напрокат один и тот же кларнет си-бемоль. СУБД и прикладная программа должны каким-то образом обнаружить эту ситуацию и сообщить служащим, что им следует выбрать другой инструмент.

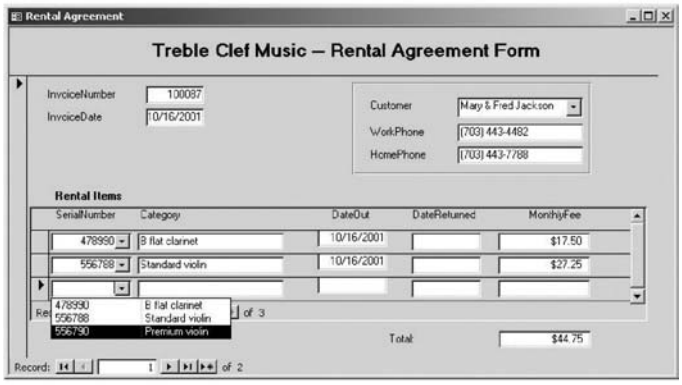
Бюро регистрации автомобилей и выдачи прав

Рассмотрим теперь еще более обширное приложение технологии баз данных — государственное бюро регистрации автомобилей и выдачи водительских прав. В него входят 52 центра, где принимаются экзамены по вождению и осуществляется выдача и продление прав, а также 37 офисов, занимающихся регистрацией транспортных средств.

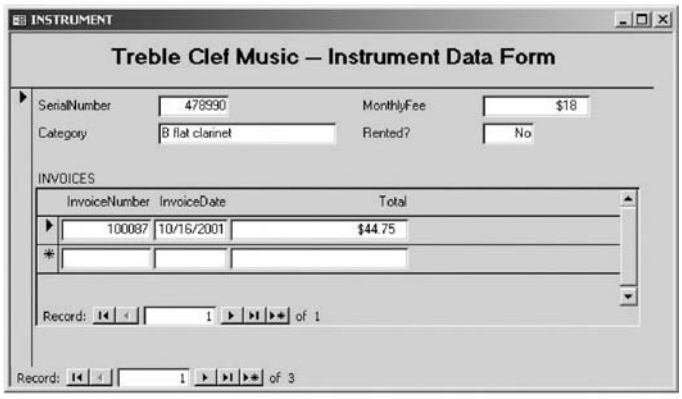
Персонал этих офисов использует в своей работе базу данных. Прежде чем конкретному лицу будут выданы или продлены права, в базе данных просматриваются сведения об этом лице на предмет наличия нарушений правил дорожного движения, ДТП или арестов. На основе этих данных принимается решение, могут ли права быть продлены, и если да, то будут ли в них какие-либо ограничения. Аналогичным образом, персонал в отделе регистрации автомобилей обращается к базе данных, чтобы определить, был ли автомобиль зарегистрирован ранее, а если был, то на чье имя, и нет ли каких-либо из ряда вон выходящих причин, по которым в регистрации следует отказать.



а



б



в

Рис. 1.13. Формы, используемые бюро проката Treble Clef Music: *а* — форма Customer, *б* — форма Rental Agreement, *в* — форма Instrument Data

У этой базы данных сотни пользователей, включая не только персонал, занимающийся регистрацией и выдачей прав, но и служащих финансового управления, а также сотрудников правоохранительных органов. Не удивительно, что база является большой и сложной и имеет более 40 таблиц, причем некоторые из них содержат сотни тысяч строк данных. Компоненты этой системы показаны на рис. 1.14.

Заметьте, что приложения написаны на разных языках и, вероятно, в разные периоды времени. Oracle обрабатывает две разные базы данных по поручению этих приложений. Поскольку эти приложения очень важны, система должна быть доступна круглосуточно и без выходных. Такой уровень доступности делает простые задачи, например резервное копирование, очень трудными.

Большие организационные базы данных, подобные только что рассмотренной нами, были первыми приложениями технологии баз данных. Подобные системы находятся в эксплуатации уже в течение 20 или 30 лет и за этот период неоднократно модифицировались в соответствии с менявшимися требованиями времени. Существуют организационные базы данных, предназначенные для ведения счетов в банках и других финансовых институтах, учета готовой продукции и комплектующих на складах больших предприятий, обработки медицинской документации в госпиталях и страховых компаниях, а также для правительственных нужд.

Сегодня многие организации модифицируют прикладные программы своих баз данных, чтобы дать клиентам возможность обращаться к этим данным и даже вносить в них изменения через Интернет. Если вы работаете в большой организации, то вас вполне могут подключить к подобному проекту.

Сравнение четырех типов баз данных

Приведенные примеры демонстрируют возможные варианты использования технологии баз данных. Есть сотни тысяч баз данных, похожих на ту, что имеется в малярной фирме Мэри Ричардс, — однопользовательские базы данных с относительно небольшим количеством данных, скажем, менее 10 Мбайт. Формы и отчеты для этих баз данных имеют обычно довольно простой вид.

У других баз данных, подобных той, что используется в бюро проката Treble Clef Music, несколько пользователей, но общее их количество обычно не превышает 20–30 человек. Они содержат умеренное количество данных — например 50 или 100 Мбайт. Формы и отчеты должны быть достаточно сложными, чтобы поддерживать несколько различных деловых функций.

Самые большие размеры у баз данных, подобных той, которую мы рассматривали для случая бюро регистрации автомобилей, — у них сотни пользователей и триллионы байт данных. Для работы с этими базами данных используется множество различных приложений, у каждого из которых свои собственные формы и отчеты. Характеристики этих типов данных сведены в табл. 1.2.

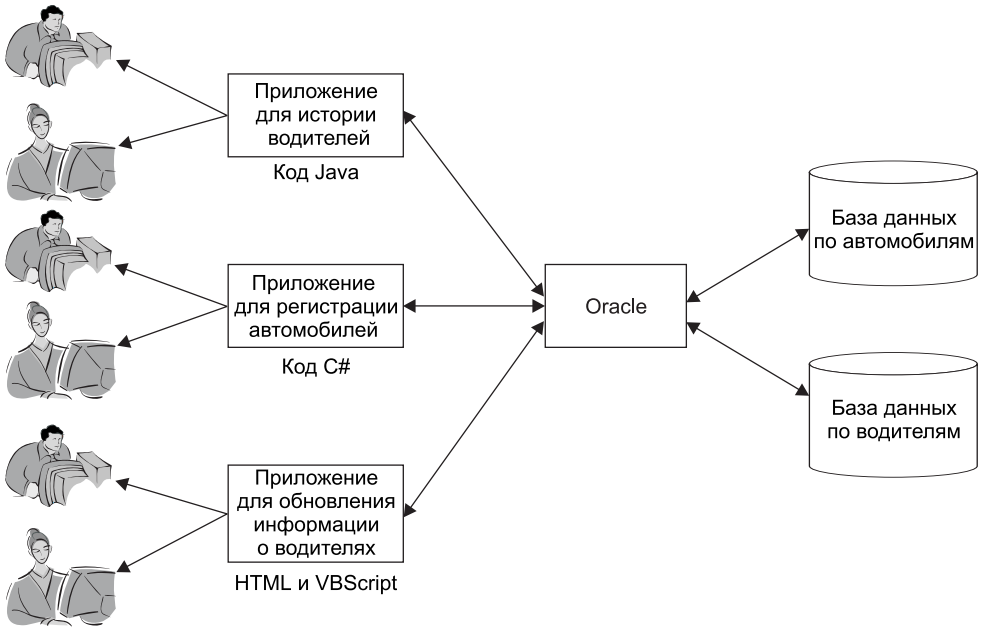


Рис. 1.14. Организационная система баз данных

Прочтя книгу, вы сможете проектировать и реализовывать базы данных наподобие тех, что используются в малярной фирме Мэри Ричардс и в бюро проката Treble Clef Music. Возможно, вы будете еще не в состоянии создавать такие большие и сложные базы данных, как та, что используется в бюро регистрации транспортных средств, но, тем не менее, сможете быть полезным членом команды по разработке и созданию такой базы данных. Вы также сможете создавать небольшие или средних размеров базы данных, использующие Интернет-технологии.

Таблица 1.2. Характеристики разных типов баз данных

Тип	Пример	Число пользователей	Размер базы данных
Персональная	Малярная фирма Мэри Ричардс	1	<10 Мбайт
Для рабочей группы	Бюро проката музыкальных инструментов Treble Clef Music	<25	<100 Мбайт
Организационная	Бюро регистрации автомобилей и выдачи прав	сотни или тысячи	>1 Тбайт, возможно, несколько баз данных

Как построить систему баз данных

Процесс построения базы данных по большей части совпадает с процессом построения любой другой информационной системы. Как показано в табл. 1.3, существуют три основных фазы: формулирование требований, проектирование и реализация. Большая часть этой книги посвящена второй фазе, хотя в некото-

рых главах мы будем рассматривать также создание запросов и кода приложения. Однако сначала мы познакомимся с разработкой базы данных.

Таблица 1.3. Обзор фаз построения базы данных

Фаза построения базы данных	Базы данных	Приложения
Фаза формулирования требований	Построение модели данных Задание элементов данных Определение ограничений и правил	Определение требований приложения
Фаза проектирования	Таблицы Отношения Индексы Ограничения Хранимые процедуры и триггеры	Формы Отчеты Запросы Код приложения
Фаза реализации	Создание таблиц Создание отношений Создание ограничений Написание хранимых процедур и триггеров Заполнение базы данных Тестирование	Создание форм Создание отчетов Создание запросов Написание кода приложения Тестирование

Фаза формулирования требований

Во время этой фазы разрабатывается модель данных: типы элементов данных, их длина и другие свойства. Кроме того, на данные накладываются ограничения и правила.

Модель данных — это логическое представление структуры базы данных. Моделирование данных очень важно, поскольку и база данных, и все ее структуры зависят от модели данных. Если модель данных некорректна, результат будет разочаровывающим. Учитывая важность модели данных, я посвятил ей следующие две главы.

На рис. 1.15 показан пример диаграммы модели данных для фирмы Lakeview Rentals. На рисунке представлен пример диаграммы сущность—связь — средства выражения моделей данных, которое стало промышленным стандартом. На этом рисунке `JOB`, `CONTRACTOR`, `RENTAL` и `EQUIPMENT` являются сущностями. Линии представляют связи между этими сущностями. Остальными подробностями пока можно пренебречь. Нужно только знать, что такие диаграммы существуют для документирования моделей данных.

Как вы увидите позже, во время фазы формулирования требований необходимо определить свойства элементов данных, такие как тип данных, максимальная длина, требуется ли значение и т. д. Кроме того, следует определить значения элементов данных и правила обработки данных. Приведем пример ограничения данных: значение `Equipment type` (тип оборудования) должно быть из следующего

множества: ['Backhoe', 'Small crane', 'Medium crane', 'Large crane', 'Scaffolding'] (экскаватор, малый кран, средний кран, большой кран, строительные леса). Ограничения и правила мы будем обсуждать в следующих двух главах.

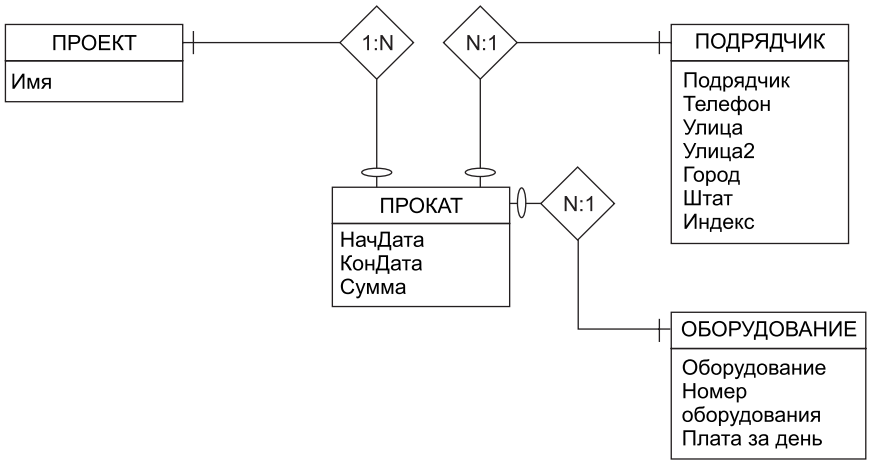


Рис. 1.15. Диаграмма сущность—связь для базы данных Lakeview Rentals

Фаза проектирования

Во время фазы проектирования модель данных преобразуется в таблицы и отношения. На рис. 1.16 показан пример диаграммы структуры данных — диаграммы, используемой для отображения таблиц и их отношений. На этой диаграмме линия между таблицами EQUIPMENT и RENTAL представляет связь. Вилка по направлению к таблице RENTAL означает, что данная строка в таблице EQUIPMENT может относиться ко многим строкам таблицы RENTAL. Эту и подобные диаграммы мы будем использовать при обсуждении проектирования баз данных в главах 5 и 8. В этих главах вы узнаете значение и других элементов на этом рисунке.

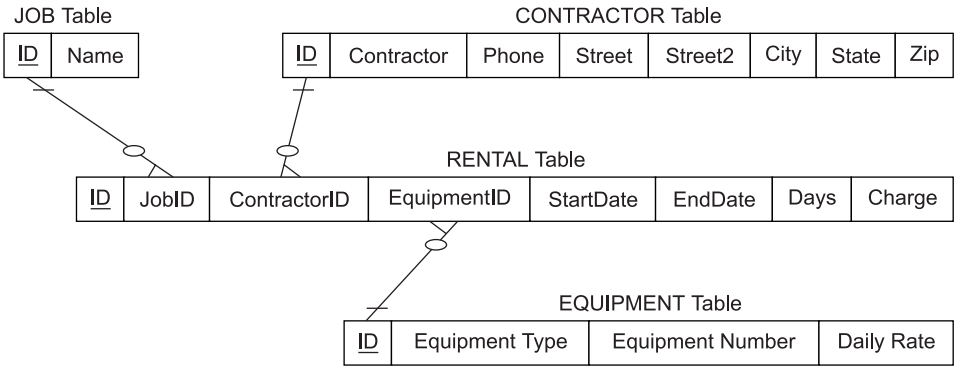


Рис. 1.16. Пример диаграммы структуры данных для Lakeview Rentals

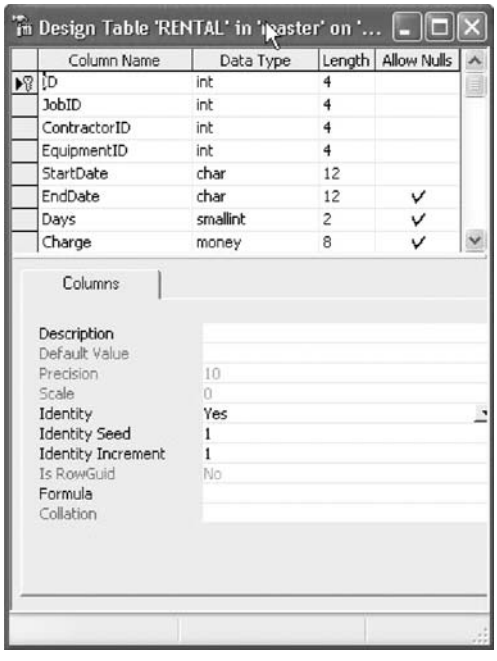
Необходимость в индексах определяется во время фазы проектирования базы данных, и иногда также задаются характеристики индексов (я говорю « иногда», поскольку не все СУБД поддерживают эту спецификацию). Механизмы ограничений, хранимые процедуры и триггеры также проектируются. Об этом вы узнаете из глав 10–12.

```
CREATE TABLE EQUIPMENT(
    ID                  Int             Primary Key,
    EquipmentType       Char (35)       Not Null,
    EquipmentNumber     Char (25)       Not Null,
    DailyRate           Number (6,2));

CREATE TABLE RENTAL(
    ID                  Int             Primary Key,
    Jobl D              Int             Not Null,
    ContractorID        Int             Not Null,
    EquipmentID         Int             Not Null,
    StartDate           Char (12)       Not Null,
    EndDate             Char (12),
    Days                SmallInt,
    Charge              Number (8,2));

ALTER TABLE RENTAL ADD CONSTRAINT EquipmentFK
    FOREIGN KEY (EquipmentID) REFERENCES RENTAL;
```

а



б

Рис. 1.17. Создание таблиц: *а* — с помощью текстовых команд, *б* — с помощью графических средств

Фаза реализации

Как показано в табл. 1.3, в фазе реализации создаются таблицы и связи. Для создания таблиц используются два способа: с помощью SQL и через средство графического проектирования. На рис. 1.17, *а* показаны SQL-операторы для создания таблиц `EQUIPMENT` и `RENTAL` и для определения связей между ними. Такие SQL-операторы можно вводить в базу данных для определения таблиц и связей. На рис. 1.17, *б* показан другой прием — использование средства графического проектирования для определения таблицы `CONTRACTOR` в SQL Server. Большинство СУБД позволяют использовать оба способа. Аналогичным образом, в большинстве СУБД можно задавать ограничения на данные как с помощью SQL, так и с помощью графических средств. О том, как писать SQL-операторы для этих целей, вы узнаете в главах 7 и 8.

Во время реализации также пишутся и тестируются хранимые процедуры и триггеры. Как это делается в Oracle, вы узнаете в главе 10, а в SQL Server — в главе 11. Наконец, база данных заполняется данными, и система тестируется.

Некоторые информационные системы на этом уже готовы. Обычно же когда база данных и ее приложение завершены, необходимо еще модифицировать их в соответствии с новыми требованиями, которые разрабатываются во время фазы реализации. Такие модификации также проходят эти же три фазы (формулирования требований, проектирования и реализации), но это происходит в контексте существующих баз данных и приложений. Изменение существующих баз данных может быть очень трудным, и этот процесс, называемый *перепроектированием базы* данных, мы будем обсуждать в главе 8.

Разработка приложения

Разработка приложения осуществляется параллельно с разработкой базы данных. Поскольку эта книга посвящена базам данных, мы не будем слишком подробно останавливаться на разработке приложения. В главе 7 будет обсуждаться использование SQL для приложений, а в главах 10 и 11 будет описываться написание хранимых процедур и триггеров. Однако по большей части третий столбец табл. 1.3 мы оставим для ваших занятий по системному программированию.

Перед тем как перейти к моделированию данных, мы завершим эту главу кратким экскурсом в историю баз данных.

Краткая история баз данных

В табл. 1.4 сведена история развития технологии баз данных. До середины 1960-х годов почти все компьютерные хранилища данных были на магнитных лентах. Поскольку лента может обрабатываться только последовательно, данные должны были храниться в виде списков (или последовательных файлов, как они назывались). Однако, как вы узнали в начале этой главы, хранение даже простейших данных в таком формате чревато большими проблемами.

Таблица 1.4. Краткая история баз данных

Период	Технология	Примечания
До 1968	Обработка файлов	Предшествовала обработке баз данных. Данные хранились в виде списков. Характер обработки определялся всеобщим использованием в качестве носителя магнитной ленты
1968–1980	Иерархические и сетевые модели	Эра обработки нереляционных баз данных. Выдающейся иерархической моделью данных была DL/I фирмы IBM. Первая СУБД называлась IMS. Выдающейся сетевой моделью данных была модель DBTG фирмы CODASYL. Самой популярной сетевой СУБД была IDMS
1980 – наст. время	Реляционная модель данных	Реляционная модель данных впервые была опубликована в 1970 году. Реализовываться в коммерческих приложениях начала в 1980 году. IBM выпустила DB2, среди других продуктов выделяется Oracle. Реляционный язык SQL стал промышленным стандартом
1982	Первые СУБД для микрокомпьютеров	Фирма Ashton-Tate разработала dBase, Microrim — R:Base, а Borland — Paradox
1985	Развитие интереса к объектно-ориентированным СУБД (ООСУБД)	С развитием объектно-ориентированного программирования были предложены ООСУБД. Коммерческий успех их невелик, в первую очередь потому, что преимущества не оправдывают перевод миллиардов байтов данных организаций в новый формат. Продолжают развиваться и сейчас
1991	Компания Microsoft выпустила Access	Персональная СУБД, созданная как элемент Windows. Постепенно вытеснила с рынка все другие персональные СУБД
1995	Первые приложения баз данных для Интернета	Базы данных стали ключевым компонентом Интернет-приложений. Популярность Интернета существенно повысила необходимость в базах данных и требования к ним
1997	Применение XML к обработке баз данных	Использование XML решило проблемы, которые долго стояли перед базами данных. Ведущие производители стали интегрировать XML в свои СУБД

Ранние модели баз данных

С коммерческим успехом хранилищ на дисках в середине 1960-х стало возможным получение непоследовательного, или прямого, доступа к записям. Базы данных стали разрабатываться по-другому. Изначально стали успешными две конкурирующие архитектуры, или модели. Корпорация IBM разработала и внедрила DL/I (Data Language One, язык данных один), который моделировал данные в базах данных в форме иерархий, или деревьев (см. рис. 1.18, *а*). Эта модель, которая была разработана совместно с промышленными предприятиями, легко могла использоваться для поддержки данных, таких как сметы материалов и списки деталей, но для общих целей мало подходила. Представление неиерархических сетевых данных (рис. 1.18, *б*) было громоздким.

После этого CODASYL, группа, которая разрабатывала стандарты для языка COBOL, в 1970 году создала модель под названием DBTG (Data Base Task Group, группа задач баз данных). Модель DBTG была готова к представлению как иерархических, так и сетевых данных. Один раз эта модель предлагалась в качестве национального стандарта, но не была принята в первую очередь из-за своей сложности. Однако это была основа для ряда коммерчески успешных СУБД в семидесятых и восьмидесятых годах прошлого века. Наиболее успешным был продукт корпорации Cullinane под названием IDMS.

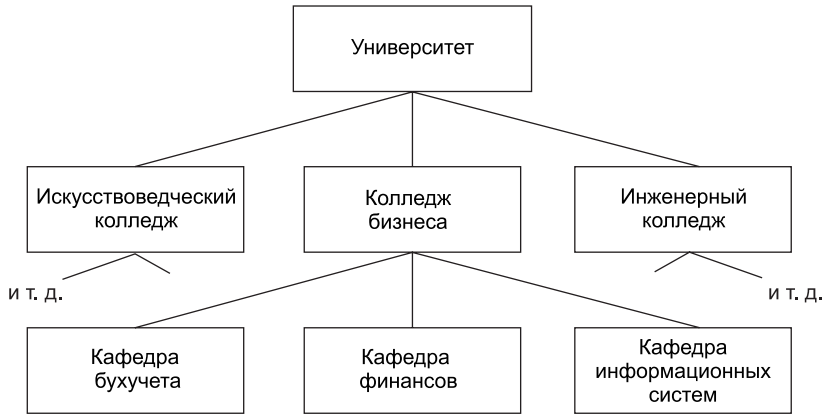


Рис. 1.18. *а* — пример иерархии: университетская организация, *б* — пример сети: классы и студенты

Реляционная модель

Реляционная модель, которая представляет данные в формате, показанном на рис. 1.3, впервые была предложена Е. Ф. Коддом в 1970 году. Кодд работал в IBM, и после десяти лет исследований, разработки и лоббирования на уровне корпорации он с коллегами убедил IBM разработать несколько СУБД, основанных на

реляционной модели. Наиболее известным из этих продуктов является DB2 — СУБД, которая активно используется и поныне.

Между тем другие корпорации (такие как Oracle, Ingres, Sybase и Informix) тоже разработали СУБД, основанные на реляционной модели. SQL Server был разработан в Sybase и в конце восьмидесятых годов продан Microsoft. Сегодня DB2, Oracle и SQL Server являются наиболее выдающимися коммерческими СУБД.

СУБД для персональных компьютеров

С появлением большого числа микрокомпьютеров стало возможно иметь персональные базы данных. В результате был разработан ряд СУБД для персональных компьютеров. Наиболее успешной из них была dBase — продукт корпорации Ashton-Tate. Еще среди ранних персональных СУБД можно назвать R:base корпорации Microrim и Paradox от Borland.

Поскольку компьютеры обладали довольно большой вычислительной мощностью, персональные СУБД предоставляли больше графических интерфейсов пользователя. Кроме того, со временем под влиянием этих продуктов изменились и интерфейсы больших организационных СУБД. На рис. 1.17 показан этот момент. Технология того, что показано на рис. 1.17, *а*, разрабатывалась для символьно-ориентированных интерфейсов, которые были распространены для СУБД, предшествовавших персональным компьютерам. На рис. 1.17, *б* показан пример графического интерфейса, который появился с возникновением персональных СУБД.

Объектно-ориентированные СУБД (ООСУБД)

Объектно-ориентированное программирование начало развиваться в середине восьмидесятых годов прошлого века и привело к созданию объектно-ориентированных СУБД. Целью этих продуктов была способность хранить объекты из объектно-ориентированного программирования (например, из языков C++ или Java) в базе данных, не преобразуя их в реляционный формат.

В настоящее время ООСУБД не имеют коммерческого успеха. Их использование требует от организаций преобразовывать свои базы данных из реляционного в ООСУБД-формат. Кроме того, большинство крупных организаций имеют более старые приложения, не основанные на объектно-ориентированном программировании. Программы в таких приложениях потребовалось бы как-то сопровождать новыми ООСУБД. Таким образом, высокая стоимость перевода существующих баз данных и информационных систем из реляционных СУБД в ООСУБД задерживает их широкое распространение.

Были разработаны объектно-ориентированные СУБД, такие как Oracle 8i и 9i, что позволило создавать как реляционное, так и объектное представление данных одной и той же базы данных. Эти СУБД начинают получать коммерческий успех, но не такой, как чисто реляционные СУБД. Объектно-ориентированные СУБД мы рассмотрим в главе 16.

Недавняя история

В 1991 году Microsoft выпустила Access, который на несколько лет вытеснил с рынка все остальные СУБД. Частично это произошло благодаря тому, что Access был интегрирован в Microsoft Office, и Microsoft смогла использовать свое влияние на рынке и монополию в связи с Windows для смещения других продуктов. Правда, Microsoft нужно отдать справедливость: Access — суперпродукт. Он доминирует на рынке, потому что это легкая в использовании и сильная СУБД.

Как все знают, использование Интернета распространилось в середине девяностых годов. Но немногие знают, что именно это сильно повысило значение и важность технологии баз данных. Как только ранние статические веб-страницы уступили дорогу динамическим, как только стали иметь успех компании типа Amazon.com и как только большие организации начали использовать Интернет для публикации своих данных, все большее и большее количество сайтов стало зависеть от баз данных. Эта тенденция продолжается и по сей день.

Наконец, в последние годы появился и стал широко использоваться язык XML, который представляет собой технологию для поддержки веб-сайтов, но был расширен для проведения важных решений, связанных с базами данных. Язык XML мы будем обсуждать в главе 13. Пока же нам осталось осознать, что интеграция технологии баз данных с XML является ведущим пунктом области баз данных сегодня и будет важна еще много лет в будущем.

Резюме

Хотя обработка баз данных всегда была важной темой, популярность Интернета сделала ее еще и одной из самых нужных специальностей. Навыки, которые вы разовьете, и знания, которые вы приобретете, будут чрезвычайно востребованы.

Цель базы данных — помочь людям и организациям вести учет различных вещей. Хотя для этой цели можно использовать списки, они вызывают множество проблем. Их сложно изменять без возникновения несоответствий, удаления из списков могут иметь непредвиденные последствия, а неполные данные трудно записывать. Кроме того, вводя данные, легко вызвать их противоречивость. Наконец, различные части организации хотят поддерживать некоторые данные совместно, а некоторые — исключительным образом. Это трудно организовать при использовании списков.

Базы данных состоят из групп реляционных таблиц. В большинстве случаев каждая таблица содержит данные по определенной теме. Поддержка данных таким образом решает все проблемы, перечисленные для списков. Связи в таблицах представляются разными способами. В этой главе связи представлялись путем присвоения каждой строке уникального идентификатора и использования этого идентификатора для связи строки одной таблицы со строкой другой таблицы. Для представления связей использовались и внешние ключи. Таблицы можно создавать с помощью языка SQL, который является промышленным стандартом для обработки таблиц.

Система базы данных состоит из четырех основных элементов: пользователи, приложения базы данных, СУБД и сама база данных. Пользователи применяют базу данных для решения своих задач. Приложения производят формы, запросы и отчеты, выполняют логику приложения и управляют обработкой базы. СУБД создает, обрабатывает и администрирует базу данных. База данных — это самодокументированное собрание интегрированных записей. Она содержит пользовательские данные, метаданные, индексы, хранимые процедуры, триггеры и метаданные приложения. Хранимая процедура — это программа, которая обрабатывает участок базы данных и хранится в базе данных. Триггер — это процедура, которая вызывается при наступлении определенного события. На рис. 1.6 показаны функции компонентов базы данных.

Технология баз данных может использоваться в широком спектре приложений. Некоторые базы данных используются одним человеком, другие — группой людей, а третьи — большими организациями. В табл. 1.2 показаны некоторые характеристики этих разных типов баз данных.

Подобно всем информационным системам, системы баз данных разрабатываются в течение трех фаз: формулирования требований, проектирования и реализации. Во время фазы формулирования требований разрабатывается модель данных, или логическое представление структуры базы данных. Модели данных важны, потому что от них зависит проектирование базы данных и приложения. Диаграмма сущность—связь — средство, используемое для представления модели данных.

Модель данных преобразуется в таблицы и связи на фазе проектирования. Также проектируются индексы, ограничения, хранимые процедуры и триггеры. Диаграммы структур данных иногда используются для таблиц документов и их связей. Во время фазы реализации создаются таблицы, связи и ограничения, пишутся хранимые процедуры и триггеры, база данных заполняется данными и тестируется. Сегодня таблицы и связанные с ними конструкции создаются с помощью SQL или графических средств, являющихся частью СУБД.

Разработка приложения идет параллельно с разработкой базы данных. Большая часть этой книги посвящена разработке базы данных, но проектирование и построение запросов и создание кода приложения также будут рассматриваться в последующих главах. Существенные события в истории обработки баз данных показаны в табл. 5.

Вопросы группы I

1. Почему обработка баз данных сегодня важнее, чем раньше?
2. Сформулируйте цель базы данных.
3. Перечислите проблемы, которые могут возникать при использовании списков для учета чего-либо.
4. Основываясь на вашем ответе на вопрос 3, скажите, когда, по вашему мнению, можно использовать списки для учета чего-либо.

При ответе на вопросы 5–9 используйте следующий список:

НаучРуководитель	ТелНаучРуководителя	Студент	Специализация	ЗавКафедрой
Джонс	221-23-45	Паркс	бухучет	Грин
Квейл	223-44-44	Штайн	инф.технологии	Мастерз
Розенблюм	281-39-44	Джонсон	бухучет	Грин
Розенблюм	281-39-44	Гонзалес	бухучет	Грин
Джонс	221-23-45	Рики	бухучет	Грин

5. Опишите проблемы, которые могут возникнуть, когда научный руководитель меняет свой номер телефона.
6. Предположим, студент Штайн бросил учебу, и вторая строка удаляется. Опишите побочные эффекты от этого удаления.
7. Как можно использовать этот список для записи того факта, что Смит — заведующий кафедрой компьютерных наук?
8. Предположим, что в последней строке значение последнего столбца изменилось с Грин на Абернати. Какие проблемы возникнут при интерпретации этого списка?
9. Переделайте этот список в набор из трех взаимосвязанных таблиц. Для представления связей используйте прием, показанный на рис. 1.3.
10. Объясните, почему ни одна из проблем, описанных в вопросах 5–8, не может возникнуть в таблицах, которые вы построили при ответе на вопрос 9.
11. Используя в качестве примера оператор SQL, показанный в разделе «Объединение таблиц», составьте подобный оператор для объединения таблиц, которые вы построили при ответе на вопрос 9. Если в каком-нибудь из имен столбцов есть пробелы, поставьте имена столбцов в квадратные скобки [ ]. Поскольку в вашем результате нет вычислений, вам не понадобится выражение типа [Daily Rate]*[Days] As Charge.
12. Назовите четыре компонента системы базы данных.
13. Перечислите функции приложения базы данных.
14. Для чего предназначена СУБД? Перечислите ее функции.
15. Определите термин *база данных*.
16. Перечислите содержимое базы данных.
17. Что такое метаданные? Приведите пример.
18. В чем разница между хранимой процедурой и триггером?
19. Объясните различия между следующими базами данных: личной, для рабочей группы и организационной.
20. Назовите задачи фазы формулирования требований при разработке системы базы данных.
21. Что такое модель данных? Почему такие модели важны?
22. Назовите задачи фазы проектирования при разработке системы базы данных.

23. Назовите задачи фазы реализации при разработке системы базы данных.
24. Назовите две модели данных, которые предшествовали реляционной модели.
25. Опишите недостатки двух моделей, о которых идет речь в предыдущем вопросе.
26. Кто первый предложил реляционную модель?
27. Как персональные СУБД повлияли на организационные СУБД?
28. Назовите три ранних персональных СУБД.
29. Какова цель ООСУБД?
30. Почему ООСУБД не имеют коммерческого успеха?
31. Что такое объектно-реляционная СУБД?
32. Какая технология сейчас на переднем крае в области баз данных?

Вопросы группы II

33. Предположим, что Новоорлеанское товарищество деревянных лодок (фиктивная организация) публикует ежемесячный бюллетень, на который оно тратит 35 долларов в год. Пусть для рассылки этих бюллетеней товарищество хранит следующие данные в виде списка:

    Имя, Улица, НомерДома, Город, Штат, Индекс, НачДата, КонДата, СуммСтоимость.

    Какие проблемы могут возникнуть при хранении данных в таком списке?
34. Предположим, что подписчик этого бюллетеня на три года изменил свой адрес. Нужно ли изменять адресные данные для всей подписки, или только для текущей? Почему?
35. Разбейте список из вопроса 33 на две таблицы — ПОДПИСЧИК и ПОДПИСКА. Постройте связи между таблицами, используя в качестве примера рис. 1.3. Как такое расположение может исправить проблемы, которые вы нашли, отвечая на вопрос 33? Как оно влияет на ваш ответ на вопрос 34?
36. Предположим, что товарищество решило опубликовать два издания: ежеквартальный путеводитель (зима, весна, лето и осень) и ежемесячное новостное издание. Предположим, что стоимость путеводителя — 15 долларов в год, а новостного издания — 35 долларов в год. Пусть для этой подписки общество хранит следующие данные:

    Имя, Улица, НомерДома, Город, Штат, Индекс, Издание, НачДата, КонДата, СуммСтоимость.

    Какие проблемы могут возникнуть при хранении этих данных в виде списка?
37. Разбейте список из вопроса 36 на две таблицы: ПОДПИСЧИК и ПОДПИСКА. Постройте связи между таблицами, используя в качестве примера рис. 1.3. Как такое расположение может исправить проблемы, которые вы нашли, отвечая на вопрос 36? Объясните, почему такое двухтабличное расположение лучше, чем одна таблица.

38. Предположим, что стоимость путеводителя — 15 долларов в год, а новостного издания — 35 долларов в год, но стоимость обоих изданий сразу — 40 долларов в год. Повлияет ли это обстоятельство на проектирование таблиц для вопроса 37? Почему? Как товарищество должно действовать в этой ситуации?
39. Практически любой член товарищества, имеющий компьютерные навыки, может создавать и обрабатывать список. Однако разработка базы данных и приложения базы данных уже требует специализированных знаний и умений. Если бы товарищество попросило вас помочь определить, окупятся ли затраты на базу данных, как бы вы рассуждали? Какие вопросы вы бы задали? Какой анализ вы бы провели?

Вопросы к проекту FiredUp

FiredUp, Inc. — небольшая компания, владельцами которой являются Керт и Джули Роубердс. Эта компания, находящаяся в Брисбейне, Австралия, производит и продает легкие походные горелки под названием FiredNow. Керт, работавший раньше аэрокосмическим инженером, изобрел и запатентовал сопло, которое не дает огню в горелке погаснуть даже при очень сильном ветре — до 90 миль в час. Джули, промышленный дизайнер по образованию, разработала элегантную складную конструкцию, которая имеет небольшой вес и габариты, проста в установке и весьма устойчива. Семья Роубердс производит горелки в своем гараже и продает клиентам напрямую, принимая заказы через Интернет, по факсу и по почте.

Владельцы FiredUp хотели бы отслеживать проданные горелки на тот случай, если им понадобится войти в контакт с покупателями относительно дефектов в изделиях и по другим вопросам, связанным с качеством изделий. Соответственно, они ввели форму регистрации продукта для каждой из горелок. Форма содержит следующие данные: ИмяПокупателя, Улица. НомерДома, Город, Штат, Индекс, Страна, НомерТелефона, ДатаПокупки, СерийныйНомер.

- Предположим, фирма FiredUp хранит свои данные в списке. Опишите пять потенциальных проблем, которые у нее могут возникнуть при этом.
- Предположим, что вместо списка они решили создать персональную базу данных со следующими таблицами: КЛИЕНТ (Имя, Улица, Дом, Квартира, Город, Штат, ПочтовыйИндекс, Страна, ЭлектронныйАдрес, Телефон) и ПОКУПКА (ДатаПокупки, СерийныйНомер).

1. Создайте таблицу с данными, структура которой будет соответствовать таблице КЛИЕНТ. В таблице должно быть по меньшей мере четыре строки. (Для ответа на вопросы 1–7 достаточно ввести данные в текстовом редакторе.)
2. Какой из столбцов таблицы КЛИЕНТ можно использовать для однозначной идентификации строки в таблице? Такой столбец иногда называется *первичным ключом*, как вы позже узнаете из этой книги.
3. Создайте таблицу с данными, структура которой будет соответствовать таблице ПОКУПКА. В таблице должно быть по меньшей мере четыре строки.

4. Какой из столбцов таблицы ПОКУПКА можно использовать в качестве первичного ключа для этой таблицы?
5. Имея определенные выше таблицы, невозможно связать конкретного покупателя с его горелкой. Один из способов сделать это — добавить столбец СерийныйНомер из таблицы ПОКУПКА в таблицу КЛИЕНТ. После этого таблица КЛИЕНТ будет выглядеть следующим образом: КЛИЕНТ (Имя, Улица, Дом, Квартира, Город, Штат, ПочтовыйИндекс, Страна, ЭлектронныйАдрес, Телефон). Скопируйте данные из таблицы КЛИЕНТ и добавьте к ним столбец СерийныйНомер. Назовите получившуюся таблицу КЛИЕНТ1.
6. Связь двух таблиц можно представить и по-другому: поместить столбец ЭлектронныйАдрес из таблицы КЛИЕНТ в таблицу ПОКУПКА. После этого таблица ПОКУПКА будет выглядеть следующим образом: ПОКУПКА (ДатаПокупки, СерийныйНомер, ЭлектронныйАдрес). Скопируйте данные из таблицы ПОКУПКА и добавьте к ним столбец ЭлектронныйАдрес из таблицы КЛИЕНТ. Назовите получившуюся таблицу ПОКУПКА1.
7. Теперь у вас имеются три возможные структуры: КЛИЕНТ1+ПОКУПКА, КЛИЕНТ+ПОКУПКА1 и КЛИЕНТ1+ПОКУПКА1. При каких условиях вы могли бы рекомендовать первую структуру? Вторую структуру? Третью структуру?

Вопросы к проекту Twigs Tree

Саманта Грин владеет и управляет фирмой Twigs Tree. Она закончила лесоводческие курсы и работала в большой фирме по ландшафтному дизайну, которая занималась подрезкой деревьев. Через несколько лет она купила грузовик, машину для корчевания пней и другое оборудование и затем открыла в Сент-Луисе, штат Миссури, собственное дело.

Хотя большинство выполняемых ею заказов представляют собой одноразовые операции по спиливанию дерева или корчеванию пня, есть и повторяющиеся — например, подрезка деревьев каждый год или раз в два года. Когда дело идет медленно, она звонит старым клиентам, чтобы напомнить им о себе и о необходимости произвести очередное подрезание деревьев.

Саманта хранит о своих заказах следующие данные: ИмяКлиента, Телефон, Улица, Город, Штат, Индекс, ДатаОбслуживания, Описание, Затраты, Плата, ДатаПлатежа.

- ✦ Предположим, Саманта хранит свои данные в виде списка. Опишите пять потенциальных проблем, с которыми она может при этом столкнуться.
- ✦ Допустим, Саманта решила вместо списка создать персональную базу данных со следующими таблицами: КЛИЕНТ (Имя, Телефон, Улица, Город, Штат, Индекс) и ЗАКАЗ (ДатаОбслуживания, Описание, Затраты, Плата, ДатаПлатежа).

1. Постройте таблицу, заполнив ее данными, соответствующими структуре таблицы КЛИЕНТ. В таблице должно быть по меньшей мере четыре строки. (Для ответа на первые вопросы достаточно ввести данные в текстовом редакторе.)

2. Какой из столбцов таблицы КЛИЕНТ можно использовать для однозначной идентификации строки в таблице (то есть в качестве ключа)?
3. Постройте таблицу данных, которая соответствует структуре ЗАКАЗ. В ней должно быть по меньшей мере четыре строки.
4. Объясните, почему ни один из столбцов таблицы ЗАКАЗ нельзя использовать в качестве ключа.
5. Добавьте в таблицу КЛИЕНТ столбец идентификатора, как к таблицам на рис. 1.3. Дополните соответствующим образом ваши данные.
6. Имея определенные выше таблицы, невозможно связать конкретного клиента с его заказом. Один из способов сделать это — добавить столбец идентификатора из таблицы ЗАКАЗ в таблицу КЛИЕНТ. Таблица КЛИЕНТ при этом будет содержать следующее: КЛИЕНТ (Имя, Телефон, Улица, Город, Штат, Индекс, ИдентЗаказа).

   Скопируйте ваши данные из таблицы КЛИЕНТ и добавьте к ним ИдентЗаказа. Назовите новую таблицу КЛИЕНТ1.
7. Еще один способ создать связь между таблицами — поместить столбец Телефон из таблицы КЛИЕНТ в таблицу ЗАКАЗ. При этом таблица ЗАКАЗ будет выглядеть так: ЗАКАЗ (ДатаОбслуживания, Описание, Затраты, Плата, ДатаПлатежа, Телефон).

   Скопируйте ваши данные из таблицы ЗАКАЗ и добавьте к ним Телефон. Назовите новую таблицу ЗАКАЗ1.
8. Теперь у вас есть три возможных структуры базы данных: КЛИЕНТ1+ЗАКАЗ, КЛИЕНТ+ЗАКАЗ1 и КЛИЕНТ1+ЗАКАЗ1. При каких условиях вы могли бы рекомендовать первую структуру? Вторую структуру? Третью структуру?