Часть I

Основные сведения

Глава 1

Введение в базы данных

В ЭТОЙ ГЛАВЕ...

- Основные области применения систем с базами данных.
- Характеристики файловых систем.
- Проблемы использования файловых систем.
- Значение термина “база данных”.
- Значение термина “СУБД”.
- Типичные функции СУБД.
- Основные компоненты среды СУБД.
- Персонал, необходимый для обслуживания СУБД.
- История развития СУБД.
- Преимущества и недостатки СУБД.

“История исследований систем баз данных — это, по сути, история развития приложений, достигших исключительной производительности и оказавших потрясающее влияние на экономику. Если еще 20 лет назад эта сфера была всего лишь областью фундаментальных научных исследований, то теперь на исследованиях в области баз данных основана целая индустрия информационных услуг, ежегодный бюджет которой только в США составляет 10 миллиардов долларов. Достижения в исследованиях баз данных стали основой фундаментальных разработок коммуникационных систем, транспорта и логистики, финансового менеджмента, систем с базами знаний, методов доступа к научной литературе, а также большого количества гражданских и военных приложений. Они послужили также фундаментом значительного прогресса в ведущих областях науки — от информатики до биологии”.

Зильбершац ([276], [277])

Эта цитата взята из материалов семинаров по системам баз данных и, по сути, является достаточным основанием для того, чтобы приступить к изучению *систем с базами данных*. Можно утверждать, что появление баз данных стало самым важным достижением в области программного обеспечения. Базы данных лежат в основе информационных систем, и это коренным образом изменило характер работы многих организаций. С момента своего появления технология баз данных стала увлекательной областью деятельности, а также катализатором многих значительных достижений в области программного обеспечения. На упомянутом выше семинаре был подчеркнут тот факт, что развитие систем баз данных еще не завершено, вопреки тому, что может показаться на первый взгляд. В действительности, перефразируя древнее высказывание, можно сказать, что мы находимся только *в конце начального этапа* их развития. Приложения, ко-

торыми придется пользоваться в будущем, окажутся настолько сложными, что потребуется переосмыслить многие алгоритмы, например, используемые в настоящее время алгоритмы хранения файлов, а также доступа к ним. Развитие исходных алгоритмов сопровождалось различными нововведениями в области разработки программного обеспечения, и, несомненно, создание новых алгоритмов также будет иметь аналогичные последствия. В этой главе предлагается вводное описание систем с базами данных.

Структура этой главы

В разделе 1.1 рассматриваются некоторые встречающиеся в повседневной жизни способы применения систем баз данных, о существовании которых мы не всегда и подозреваем. В разделах 1.2 и 1.3 сравниваются ранние способы компьютеризации ручных картотек с помощью файловых систем с более удачными современными подходами на основе баз данных. В разделе 1.4 обсуждаются обязанности четырех типов специалистов, работающих с базами данных, а именно: администраторов данных и базы данных, проектировщиков базы данных, прикладных программистов и конечных пользователей. В разделе 1.5 излагается краткая история развития систем с базами данных, рассмотрение которой продолжается в разделе 1.6 обсуждением преимуществ и недостатков этих систем.

На протяжении всей этой книги любые рассматриваемые понятия иллюстрируются примерами конкретного учебного проекта, предназначенного для автоматизации деятельности воображаемого агентства *DreamHome*, занимающегося сдачей в аренду объектов недвижимости. Подробное описание этого проекта предлагается в разделе 10.4 и приложении А. В приложении Б представлен второй учебный проект, который является еще одним конкретным примером реального приложения. В конце большинства глав приведены упражнения, построенные на материале этих двух учебных проектов.

1.1. Введение

Базы данных стали неотъемлемой частью нашей повседневной жизни. Поэтому обсуждение баз данных в этом разделе мы начнем с изучения некоторых приложений систем баз данных. В дальнейших рассуждениях будем рассматривать базу данных как некий набор связанных данных, а систему управления базами данных, или СУБД (Database Management System — DBMS), — как программное обеспечение, которое управляет доступом к этой базе данных. Более строгие и точные определения будут даны в разделе 1.3.

Покупка в супермаркете

Например, доступ к базе данных необходим при оплате товаров в супермаркете, когда кассир считывает с покупок штрих-код. При этом ручной сканер передает полученный штрих-код в приложение базы данных, и эта информация используется для поиска цены конкретного товара в базе данных всех товаров. Затем программа вычитает количество всех только что проданных товаров из товарных запасов и распечатывает на кассовом аппарате их стоимость. Если запасы на складе станут ниже некоторого заранее определенного уровня, то система автоматически сформирует заказ на поставку дополнительного количества данного товара. Когда клиент делает покупки по телефону, кассир может проверить наличие того или иного товара на складе, также запустив некоторое приложение баз данных.

Расчеты с использованием кредитной карточки

Если при покупках используется кредитная карточка, кассир должен проверить наличие кредитных средств. Это можно сделать либо по телефону, либо автоматически, с помощью специального считывающего устройства, связанного с компьютером. В любом случае при этом используется база данных, которая содержит сведения о покупках, осуществляемых с помощью кредитной карточки. На основе номера кредитной карточки специальное приложение сверяет с кредитным лимитом суммарную стоимость товаров, приобретаемых в данный момент и купленных в течение текущего месяца. После подтверждения допустимости покупки все сведения о приобретенных товарах вводятся в базу данных. Однако еще до получения подтверждения допустимости покупки приложение базы данных должно проверить, что предъявленная клиентом карточка не находится в списке украденных или утерянных. Кроме того, должно существовать еще одно самостоятельное приложение баз данных, которое оплачивает счета после получения суммы платежа, а также ежемесячно отправляет полный отчет каждому владельцу кредитной карточки.

Заказ путевки в туристическом агентстве

Когда вы при планировании отпуска обращаетесь в туристическое агентство, работник этого агентства по вашему запросу просматривает базы данных со сведениями об имеющихся путевках и о расписании полетов. При бронировании какой-либо путевки система баз данных должна выполнить все необходимые для этого действия. В данном случае необходимо убедиться в том, что два разных сотрудника агентства не бронируют одну и ту же путевку или на данный рейс не забронированы места сверх предельно допустимого количества. Например, если в самолете некоторого рейса осталось только одно свободное место и два сотрудника туристического агентства попытаются его забронировать, то система должна корректно обработать эту ситуацию и разрешить забронировать последнее место только одному сотруднику, послав другому уведомление об отсутствии свободных мест. Кроме того, каждый из них может иметь другую, отдельную систему баз данных для выписки счетов.

Заказ книг в местной библиотеке

При посещении местной библиотеки, как правило, приходится обращаться к базе данных, содержащей сведения обо всех книгах, имеющихся в этой библиотеке, о ее читателях, заявках на бронирование книг и т.д. В ней обычно имеется компьютеризованный индекс, который позволяет читателям находить нужную им книгу по названию, фамилиям авторов или по тематике. Как правило, подобная система баз данных способна обрабатывать информацию о бронировании книг, что позволяет также зарезервировать книгу, взятую другим читателем. Когда эта книга будет возвращена, ждущему ее читателю по почте будет послано сообщение, что книга уже на месте и ее можно взять. Кроме того, такая система может посылать напоминания тем читателям, которые не вернули взятую книгу в указанный срок. Для ввода информации о книгах обычно используется устройство сканирования штрих-кода, аналогичное тому, которое применяется в супермаркетах. С его помощью организуется учет движения книг в библиотеке.

Оформление страхового полиса

При оформлении какого-либо страхового полиса (например, для страхования жизни, здоровья, строения, дома или автомобиля) страховой агент может обращаться к нескольким базам данных, содержащим сведения о различных страховых компаниях. После указания персональных сведений, например имени, адре-

са, возраста, а также информации о пристрастии к курению или спиртным напиткам, приложение системы баз данных использует их для определения стоимости страхового полиса. Страховой агент может просмотреть несколько баз данных с целью поиска страховой компании, которая предложит клиенту наилучшие условия страховки.

Работа в Internet

Приложения для баз данных лежат в основе функционирования многих узлов в Internet. В качестве примера рассмотрим оперативный книжный магазин, предоставляющий своим посетителям возможность просматривать и приобретать книги, такой как Amazon.com. Web-узел этого магазина позволяет проводить поиск книг по различным категориям, скажем, по компьютерным наукам или по экономике, или выбирать конкретную книгу по имени автора и по названию. И в том и в другом случае в процессе поиска используется база данных, размещенная на Web-сервере данного предприятия, которая содержит сведения о книге, позволяет узнать, имеется ли книга в продаже, сообщает информацию о поставке, уровне товарных запасов и требованиях по оформлению заказа. Сведения о книге включают ее название, ISBN, имена авторов, цену, динамику продаж, название издательства, перечень рецензентов, а также подробное описание. Преимущество такой базы данных заключается в том, что она позволяет вводить и накапливать дополнительную информацию о книгах. Скажем, одна и та же книга может быть отнесена к нескольким категориям (таким как компьютерные науки и языки программирования), указана в числе книг, пользующихся наибольшим спросом, или обозначена как учебная литература. Например, на узле компании Amazon дополнительная информация о книгах применяется для предоставления посетителю информации о том, какие книги обычно заказывают вместе с той, которая его интересует.

Как и в предыдущем примере, для покупки одной или нескольких книг в оперативном режиме можно воспользоваться кредитной карточкой. На узле Amazon.com предусмотрена персонализация предоставляемых услуг, поскольку в базе данных накапливается информация обо всех предыдущих посещениях каждого заказчика, в частности, о том, какие книги его интересовали, какой способ оплаты и доставки он предпочитает. При повторном посещении узла покупателя называют по имени и предоставляют ему список рекомендуемых новинок, с учетом предыдущих покупок.

Обучение в университете

В университете может существовать база данных с информацией о студентах, посещаемых ими курсах, выплачиваемых стипендиях, уже пройденных и изучаемых в настоящее время предметах, а также о результатах сдачи различных экзаменов. Кроме того, может также поддерживаться база данных с информацией о приеме студентов в следующем году, а также база данных персонала университета с личными данными сотрудников и сведениями об их зарплате, которые нужны для бухгалтерии.

1.2. Традиционные файловые системы

Стало почти традицией то, что любая книга с достаточно обширным описанием баз данных начинается с обзора их предшественниц — файловых систем (file-based system). Мы также последуем этой традиции. Несмотря на то что файловые системы давно устарели, все же есть несколько причин, по которым с ними следует познакомиться.

- Понимание проблем, присущих файловым системам, может предотвратить их повторение в СУБД. Иначе говоря, следует учесть опыт прошлых ошибок. На самом деле, слово “ошибки” в данном случае звучит несколько уничижительно и не дает никакого представления о той полезной работе, которая выполнялась этими системами в течение многих лет. Тем не менее оно дает понять, что существуют и другие, более эффективные способы управления данными.
- Знать принципы работы файловых систем не только очень полезно, но и необходимо при переходе от файловой системы к системе баз данных.

1.2.1. Подход, используемый в файловых системах

Файловые системы. Набор прикладных программ, которые выполняют для пользователей некоторые операции, например создание отчетов. Каждая программа хранит свои собственные данные и управляет ими.

Файловые системы были первой попыткой компьютеризировать известные всем ручные картотеки. Подобная картотека (или подшивка документов) в некоторой организации могла содержать всю внешнюю и внутреннюю документацию, связанную с каким-либо проектом, продуктом, задачей, клиентом или сотрудником. Обычно таких папок очень много, они нумеруются и хранятся в одном или нескольких шкафах. В целях безопасности шкафы могут закрываться на замок или находиться в охраняемых помещениях. У каждого из нас дома есть некое подобие такой картотеки, содержащее подшивки важных документов, например, счетов, гарантийных талонов, рецептов, страховых и банковских документов и т.п. Если нам понадобится какая-то информация, потребуется просмотреть картотеку от начала до конца, чтобы найти искомые сведения. Более продуманный подход предусматривает использование в такой системе некоторого алгоритма индексирования, позволяющего ускорить поиск нужных сведений. Например, можно использовать специальные разделители или отдельные папки для различных *логически связанных* типов документов.

Ручные картотеки позволяют успешно справляться с поставленными задачами, если количество хранимых информационных объектов невелико. Они также вполне подходят для работы с большим количеством объектов, которые нужно только хранить и извлекать. Однако они совершенно не подходят для тех случаев, когда нужно установить перекрестные связи или выполнить обработку сведений. Например, в типичном агентстве по сдаче в аренду объектов недвижимости может быть заведен отдельный файл для каждого сдаваемого в аренду или продаваемого объекта, для каждого потенциального покупателя или арендатора, а также для каждого сотрудника компании. Рассмотрим теперь, какие действия нужно выполнить, чтобы ответить на приведенные ниже вопросы.

- Какие из выставленных на продажу объектов недвижимости с тремя спальнями имеют сад и гараж?
- Какие из сдаваемых в аренду квартир расположены в пределах трех миль от центра города?
- Какова средняя арендная плата за квартиру с двумя спальнями?
- Чему равна общая годовая заработная плата всех сотрудников?
- Каким был оборот в прошлом месяце по сравнению с прогнозируемыми показателями в этом месяце?
- Каким будет ожидаемый ежемесячный оборот в следующем финансовом году?

В наше время клиентам, менеджерам и другим сотрудникам с каждым днем требуется все больше и больше информации. В некоторых областях деятельности существуют даже правовые нормы, регламентирующие оформление ежемесячных, ежеквартальных и годовых отчетов. Очевидно, что ручная картотека совершенно не подходит для выполнения работы подобного типа. Файловые системы были разработаны в ответ на потребность в получении более эффективных способов доступа к данным. Однако вместо организации централизованного хранилища всех данных предприятия был использован децентрализованный подход, при котором сотрудники каждого отдела при помощи специалистов по *обработке данных* (ОД) работают со своими собственными данными и хранят их в своем отделе. Чтобы проиллюстрировать основные принципы такого подхода, рассмотрим его на примере учебного проекта *DreamHome*.

Сотрудники отдела реализации отвечают за продажу и сдачу в аренду объектов недвижимости. Например, если клиент обращается в отдел реализации компании с предложением сдать в аренду принадлежащий ему объект недвижимости, то ему нужно заполнить форму, подобную представленной на рис. 1.1, *а*. В ней указываются такие сведения об объекте недвижимости, как адрес и количество комнат, а также информация о владельце. Кроме того, сотрудники отдела реализации обрабатывают запросы от потенциальных арендаторов, каждый из которых должен заполнить форму, подобную представленной на рис. 1.1, *б*. При помощи сотрудников отдела обработки данных (ОД) сотрудники отдела реализации создали информационную систему для управления данными об аренде недвижимости. Эта система состоит из трех показанных в табл. 1.1–1.3 файлов с данными о недвижимости (`PropertyForRent`), владельце (`PrivateOwner`) и арендаторе (`Client`). Для простоты изложения опустим детали, относящиеся к сотрудникам компании, различным ее отделениям и владельцам недвижимости, представляющим собой юридические лица.

Таблица 1.1. Информация в файле PropertyForRent отдела реализации

propertyNo	street	city	postcode	type	rooms	rent	ownerNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93

Таблица 1.2. Информация в файле PrivateOwner отдела реализации

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

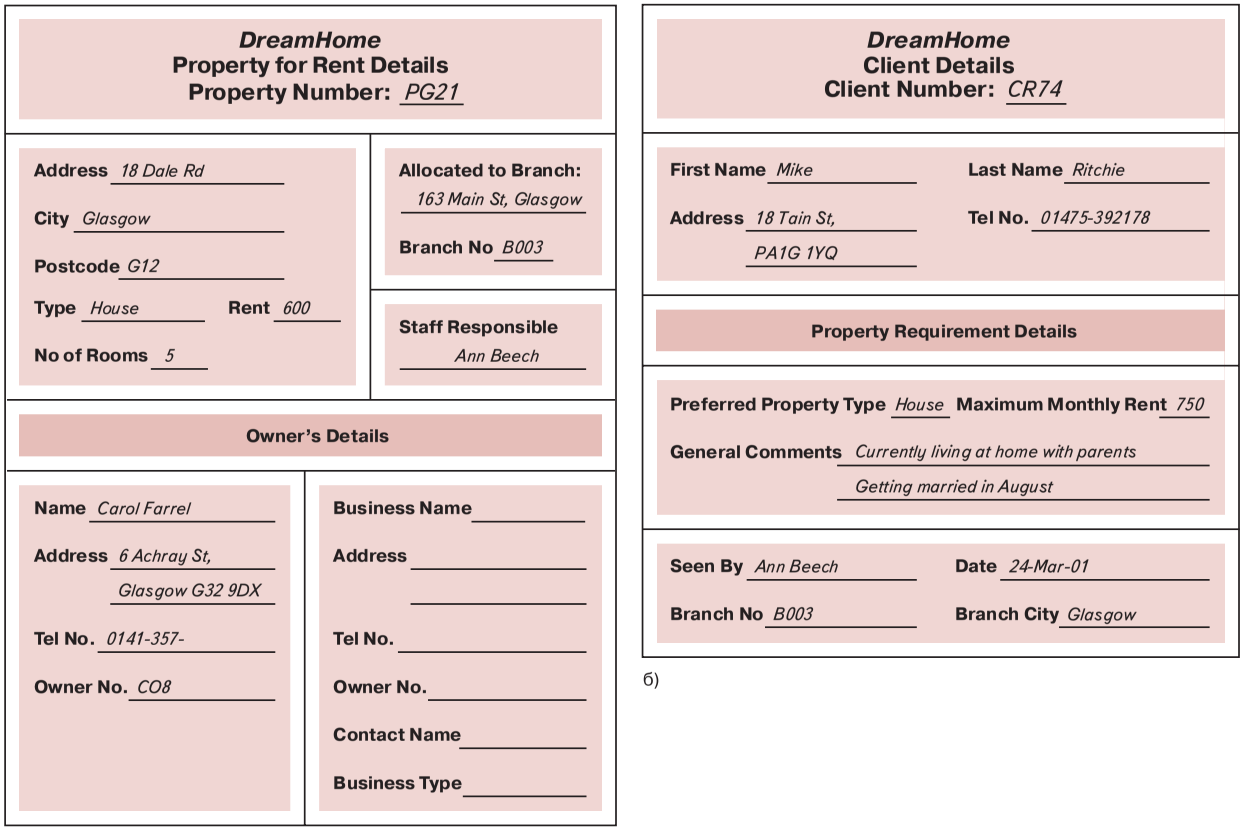


Рис. 1.1. Формы отдела реализации: а) форма со сведениями о сдаваемой в аренду недвижимости; б) форма с информацией об арендаторе

Таблица 1.3. Информация в файле Client отдела реализации

clientNo	fName	lName	address	telNo	prefType	maxRent
CR76	John	Kay	56 High St, London SW1 4EH	0207-774-5632	Flat	425
CR56	Aline	Stewart	64 Fern Dr, Glasgow G42 0BL	0141-848-1825	Flat	350
CR74	Mike	Ritchie	18 Tain St, PA1G 1YQ	01475-392178	House	750
CR62	Mary	Tregear	5 Tarbot Rd, Aberdeen AB9 3ST	01224-196720	Flat	600

Сотрудники отдела контрактов отвечают за оформление договоров об аренде недвижимости. Если клиент согласен арендовать некоторый сдаваемый в аренду объект, то одним из сотрудников отдела реализации заполняется форма, показанная на рис. 1.2. В ней указываются все необходимые сведения об арендаторе и сдаваемом в аренду объекте недвижимости. Эта форма передается в отдел контрактов, сотрудники которого присваивают договору номер и вносят дополнительные сведения об оплате и о продолжительности аренды. При помощи сотрудников отдела ОД сотрудники отдела контрактов создали для себя информационную систему учета договоров аренды. Эта система состоит из трех файлов, представленных в табл. 1.4–1.6. Файлы содержат сведения о договорах (`Lease`), объектах недвижимости (`PropertyForRent`) и арендаторах (`Client`), которые во многом сходны с данными в информационной системе отдела реализации.

Таблица 1.4. Информация в файле Lease отдела контрактов

lease No	property No	clientNo	rent	payment Method	deposit	paid	rentStart	rentFinish	duration
10024	PA14	CR62	650	Visa	1300	Y	1-Jun-01	31-May-02	12
10075	PL94	CR76	400	Cash	800	N	1-Aug-01	31-Jan-02	6
10012	PG21	CR74	600	Cheque	1200	Y	1-Jul-01	30-Jun-02	12

Таблица 1.5. Информация в файле PropertyForRent отдела контрактов

propertyNo	street	city	postcode	rent
PA14	16 Holhead	Aberdeen	AB7 5SU	650
PL94	6 Argyll St	London	NW2	400
PG21	18 Dale Rd	Glasgow	G12	600

Таблица 1.6. Информация в файле Client отдела контрактов

clientNo	fName	lName	address	telNo
CR76	John	Kay	56 High St, London SW1 4EH	0171-774-5632
CR74	Mike	Ritchie	18 Tain St, PA1G 1YQ	01475-392178
CR62	Mary	Tregear	5 Tarbot Rd, Aberdeen AB9 3ST	01224-196720

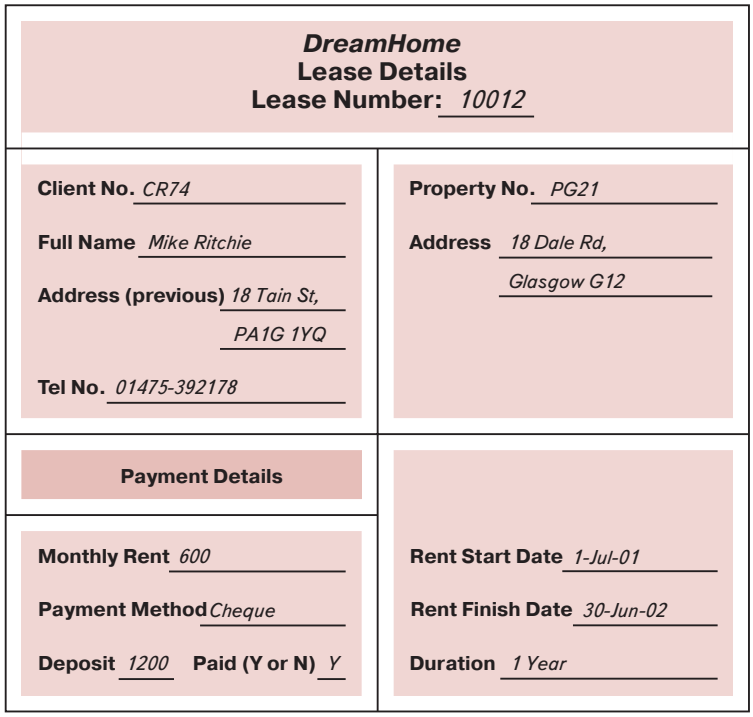


Рис. 1.2. Форма Lease Details — сведения о заключаемом договоре аренды

В целом эта ситуация схематически может быть представлена на рис. 1.3. На этой схеме показано, что каждый отдел обращается к своим собственным данным с помощью специализированных приложений. Набор приложений каждого отдела позволяет вводить данные, работать с файлами и генерировать некоторый фиксированный набор специализированных отчетов. Самым важным является то обстоятельство, что физическая структура и методы хранения записей файлов с данными жестко определены в коде программ приложений.

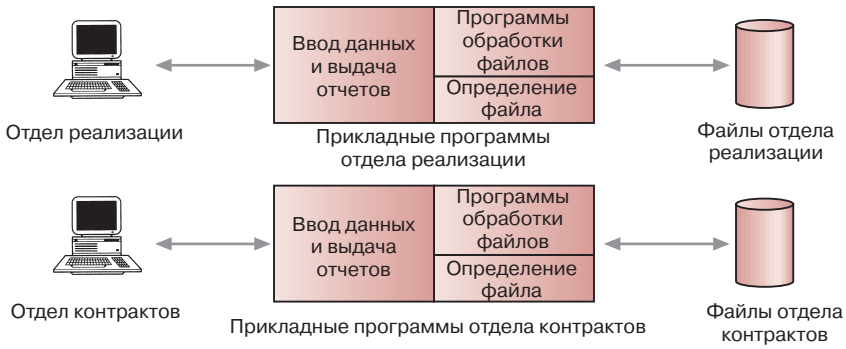
Аналогичные примеры можно найти и в других отделах. Например, в расчетном секторе бухгалтерии хранятся сведения о зарплате каждого сотрудника, помещенные в файл следующего формата:

```
StaffSalary(staffNo, fName, lName, sex, salary, branchNo)
```

В отделе кадров также хранится информация о персонале, но она представлена в файле иного формата:

```
Staff(staffNo, fName, lName, position, sex, dateOfBirth, salary,
branchNo)
```

Совершенно очевидно, что очень большое количество данных в отделах дублируется, и это весьма характерно для любых файловых систем. Прежде чем начать обсуждение недостатков этого подхода, было бы полезно познакомиться с терминологией, используемой в файловых системах. Файл является простым набором *записей* (record), которые содержат логически связанные данные. Например, файл `PropertyForRent`, содержимое которого представлено в табл. 1.1, содержит шесть записей, по одной для каждого сдаваемого в аренду объекта недвижимости. Каждая запись содержит логически связанный набор из одного или



Файлы отдела реализации

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Файлы отдела контрактов

Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)
PropertyForRent (propertyNo, street, city, postcode, rent)
Client (clientNo, fName, lName, address, telNo)

Рис. 1.3. Схема обработки данных в файловой системе

нескольких *полей* (field), каждое из которых представляет некоторую характеристику моделируемого объекта. Из табл. 1.1 видно, что поля файла `PropertyForRent` представляют такие характеристики сдаваемых в аренду объектов, как адрес, тип объекта и количество комнат.

1.2.2. Ограничения, присущие файловым системам

Такого краткого описания файловых систем вполне достаточно для того, чтобы понять суть присущих им ограничений, которые перечислены в табл. 1.7.

Таблица 1.7. Ограничения, присущие файловым системам

Ограничение
Разделение и изоляция данных
Дублирование данных
Зависимость от данных
Несовместимость файлов
Фиксированные запросы/быстрое увеличение количества приложений

Разделение и изоляция данных

Когда данные изолированы в отдельных файлах, доступ к ним весьма затруднителен. Например, для создания списка всех домов, отвечающих требованиям потенциальных арендаторов, предварительно нужно создать временный файл со списком арендаторов, желающих арендовать недвижимость типа “дом”. Затем в файле `PropertyForRent` следует осуществить поиск объектов недвижимости типа “дом” с арендной платой ниже установленного арендатором максимума. Выполнять подобную обработку данных в файловых системах достаточно сложно. Для извлече-

ния соответствующей поставленным условиям информации программист должен организовать синхронную обработку двух файлов. Трудности существенно возрастают, когда необходимо извлечь данные более чем из двух файлов.

Дублирование данных

Из-за децентрализованной работы с данными, проводимой в каждом отделе независимо от других отделов, в файловой системе фактически допускается бесконтрольное дублирование данных, и это, в принципе, неизбежно. Например, на рис. 1.3 ясно видно, что в отделе реализации и отделе контрактов дублируется информация об объектах недвижимости и арендаторах. Бесконтрольное дублирование данных нежелательно по следующим причинам.

- Дублирование данных сопровождается неэкономным расходованием ресурсов, поскольку на ввод избыточных данных требуется затрачивать дополнительное время и деньги.
- Более того, для их хранения необходимо дополнительное место во внешней памяти, что связано с дополнительными накладными расходами. Во многих случаях дублирования данных можно избежать за счет совместного использования файлов.
- Еще более важен тот факт, что дублирование данных может привести к нарушению их целостности. Иначе говоря, данные в разных отделах могут стать противоречивыми. Например, рассмотрим описанный выше случай дублирования данных в расчетном секторе бухгалтерии и отделе кадров. Если сотрудник переедет в другой дом и изменение адреса будет зафиксировано только в отделе кадров, то уведомление о зарплате будет послано ему по старому, т.е. ошибочному, адресу. Более серьезная проблема может возникнуть, если некий сотрудник получит повышение по службе с соответствующим увеличением заработной платы. И опять же, если это изменение будет зафиксировано только в информации отдела кадров, оставшись не проведенным в файлах расчетного сектора, то сотруднику будет ошибочно начисляться прежняя заработная плата. При возникновении подобной ошибки для ее исправления потребуется затратить дополнительное время и средства. Оба эти примера демонстрируют противоречия, которые могут возникнуть при дублировании данных. Поскольку не существует автоматического способа обновления данных одновременно и в файлах отдела кадров, и в файлах расчетного сектора, нетрудно предвидеть, что подобные противоречия время от времени будут неизбежно возникать. Даже если сотрудники расчетного сектора после получения уведомлений о подобных изменениях будут немедленно их вносить, все равно существует вероятность неправильного ввода измененных данных.

Зависимость от данных

Как уже упоминалось выше, физическая структура и способ хранения записей файлов данных жестко зафиксированы в коде приложений. Это значит, что изменить существующую структуру данных достаточно сложно. Например, увеличение в файле `PropertyForRent` длины поля адреса с 40 до 41 символа кажется совершенно незначительным изменением его структуры, но для воплощения этого изменения потребуется, как минимум, создать одноразовую программу специального назначения (т.е. программу, которая выполняется только один раз), преобразующую уже существующий файл `PropertyForRent` в новый формат. Она должна выполнять следующие действия:

- открыть исходный файл PropertyForRent для чтения;
- открыть временный файл с новой структурой записи;
- считать запись из исходного файла, преобразовать данные в новый формат и записать их во временный файл. Эти действия следует выполнить для всех записей исходного файла;
- удалить исходный файл PropertyForRent;
- присвоить временному файлу имя PropertyForRent.

Помимо этого, все обращающиеся к файлу PropertyForRent программы должны быть изменены с целью соответствия новой структуре файла. А таких программ может быть очень много. Следовательно, программист должен прежде всего выявить все программы, нуждающиеся в доработке, а затем их перепроверить и изменить. Обратите внимание, что многие подлежащие изменению программы могут обращаться к файлу PropertyForRent, но при этом вообще не использовать поле адреса. Ясно, что выполнение всех этих действий требует больших затрат времени и может явиться причиной появления ошибок. Данная особенность файловых систем называется *зависимостью программ от данных* (program-data dependence).

Несовместимость форматов файлов

Поскольку структура файлов определяется кодом приложений, она также зависит от языка программирования этого приложения. Например, структура файла, созданного программой на языке COBOL, может значительно отличаться от структуры файла, создаваемого программой на языке C. Прямая несовместимость таких файлов затрудняет процесс их совместной обработки.

Например, предположим, что сотрудникам отдела контрактов требуется найти имена и адреса всех владельцев, недвижимость которых в настоящее время сдана в аренду. К сожалению, в отделе контрактов нет сведений о владельцах недвижимости, так как они хранятся только в отделе реализации. Однако в файлах отдела контрактов имеется поле propertyNo с номерами объектов недвижимости, которые можно использовать для поиска соответствующих номеров в файле PropertyForRent отдела реализации. Этот файл также содержит поле ownerNo с номерами владельцев, которые можно использовать для поиска сведений о владельцах в файле PrivateOwner. Допустим, что программа отдела контрактов создана на языке COBOL, а программа отдела реализации — на языке C. Тогда с целью поиска соответствующих номеров объектов недвижимости в поле propertyNo в двух файлах PropertyForRent программисту потребуется создать программное обеспечение, предназначенное для преобразования этих полей в некоторый общий формат. Не вызывает сомнения, что этот процесс может оказаться весьма длительным и дорогим.

Фиксированные запросы/быстрое увеличение количества приложений

С точки зрения пользователя возможности файловых систем намного превосходят возможности ручных картотек. Соответственно возрастают и требования к реализации новых или модифицированных запросов. Однако файловые системы требуют больших затрат труда программиста, поскольку все необходимые запросы и отчеты должны быть созданы именно им. В результате события обычно развивались по одному из следующих двух сценариев. Во-первых, во многих организациях типы применяемых запросов и отчетов имели фиксированную форму, и не было никаких инструментов создания незапланированных или произ-

вольных (*ad hoc*) запросов как к самим данным, так и к сведениям о том, какие типы данных доступны.

Во-вторых, в других организациях наблюдалось быстрое увеличение количества файлов и приложений. В конечном счете наступал момент, когда сотрудники отдела обработки данных были просто не в состоянии справиться со всей работой с помощью имеющихся ресурсов. В этом случае нагрузка на сотрудников отдела настолько возрастала, что неизбежно наступал момент, когда программное обеспечение было неспособно адекватно отвечать запросам пользователей, эффективность его падала, а недостаточность документирования имела следствием дополнительное усложнение сопровождения программ. При этом часто игнорировались вопросы поддержки функционирования системы: не предусматривались меры по обеспечению безопасности или целостности данных; средства восстановления в случае сбоя аппаратного или программного обеспечения были крайне ограничены или вообще отсутствовали. Доступ к файлам часто ограничивался узким кругом пользователей, т.е. не предусматривалось их совместное использование даже сотрудниками одного и того же отдела.

В любом случае, подобная организация работы с течением времени изживает себя, и требуется искать другие решения.

1.3. Системы с использованием баз данных

Все перечисленные выше ограничения файловых систем являются следствием двух факторов.

1. Определение данных содержится внутри приложений, а не хранится отдельно и независимо от них.
2. Помимо приложений не предусмотрено никаких других инструментов доступа к данным и их обработки.

Для повышения эффективности работы необходимо использовать новый подход, а именно *базу данных* (database) и *систему управления базами данных*, или *СУБД* (Database Management System — DBMS). В этом разделе представлено формальное определение этих терминов, а также рассмотрены компоненты среды СУБД.

1.3.1. База данных

База данных. Совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

Чтобы глубже вникнуть в суть этого понятия, рассмотрим его определение более внимательно. База данных — это единое, большое хранилище данных, которое однократно определяется, а затем используется одновременно многими пользователями — представителями разных подразделений. Вместо разрозненных файлов с избыточными данными здесь все данные собраны вместе с минимальной долей избыточности. База данных уже не принадлежит какому-либо единственному отделу, а является общим корпоративным ресурсом. Причем база данных хранит не только рабочие данные этой организации, но и их описания. По этой причине базу данных еще называют *набором интегрированных записей с самоописанием*. В совокупности описание данных называется *системным каталогом* (system catalog), или *словарем данных* (data dictionary), а сами элементы описания принято называть *метаданными* (meta-data), т.е. “данными о дан-

ных". Именно наличие самоописания данных в базе данных обеспечивает в ней *независимость программ от данных* (program-data independence).

Подход, основанный на применении баз данных, где определение данных отделено от приложений, очень похож на подход, используемый при разработке современного программного обеспечения, когда наряду с внутренним определением объекта существует его внешнее определение. Пользователи объекта видят только его внешнее определение и не задумываются над тем, как он определяется и функционирует. Одно из преимуществ такого подхода, а именно *абстрагирования данных* (data abstraction), заключается в том, что можно изменить внутреннее определение объекта без каких-либо последствий для его пользователей, при условии, что внешнее определение объекта остается неизменным. Аналогичным образом, в подходе с использованием баз данных структура данных отделена от приложений и хранится в базе данных. Добавление новых структур данных или изменение существующих никак не влияет на приложения, при условии, что они не зависят непосредственно от изменяемых компонентов. Например, добавление нового поля в запись или создание нового файла никак не повлияет на работу имеющихся приложений. Однако удаление поля из используемого приложением файла повлияет на это приложение, а потому его также потребуется соответствующим образом модифицировать.

И, наконец, следует объяснить последний термин из определения базы данных, а именно понятие "логически связанный". При анализе информационных потребностей организации следует выделить сущности, атрибуты и связи. *Сущностью* (entity) называется отдельный тип объекта (человек, место или вещь, понятие или событие), который нужно представить в базе данных. *Атрибутом* (attribute) называется свойство, которое описывает некоторую характеристику рассматриваемого объекта; *связь* (relationship) — это то, что объединяет несколько сущностей. Например, на рис. 1.4 показана так называемая диаграмма "сущность–связь", или ER-диаграмма (Entity-Relationship — ER), для некоторой части учебного проекта *DreamHome*. Она состоит из следующих компонентов:

- шести сущностей (которые обозначены прямоугольниками): `Branch` (Отделение), `Staff` (Работник), `PropertyForRent` (Сдаваемый в аренду объект), `Client` (Клиент), `PrivateOwner` (Владелец объекта недвижимости) и `Lease` (Договор аренды);
- семи связей (которые обозначены стрелками): *`Has`* (Имеет), *`Offers`* (Предлагает), *`Oversees`* (Управляет), *`Views`* (Осматривает), *`Owns`* (Владеет), *`LeasedBy`* (Сдается в аренду) и *`Holds`* (Арендует);
- шести атрибутов, которые соответствуют каждой сущности: `branchNo` (Номер отделения), `staffNo` (Табельный номер работника), `propertyNo` (Номер сдаваемого в аренду объекта), `clientNo` (Номер клиента), `ownerNo` (Номер владельца) и `leaseNo` (Номер договора аренды).

Подобная база данных представляет сущности, атрибуты и логические связи между объектами. Иначе говоря, база данных содержит логически связанные данные. Более подробно модель типа "сущность–связь" рассматривается в главах 11 и 12.

1.3.2. Система управления базами данных — СУБД

СУБД. Программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

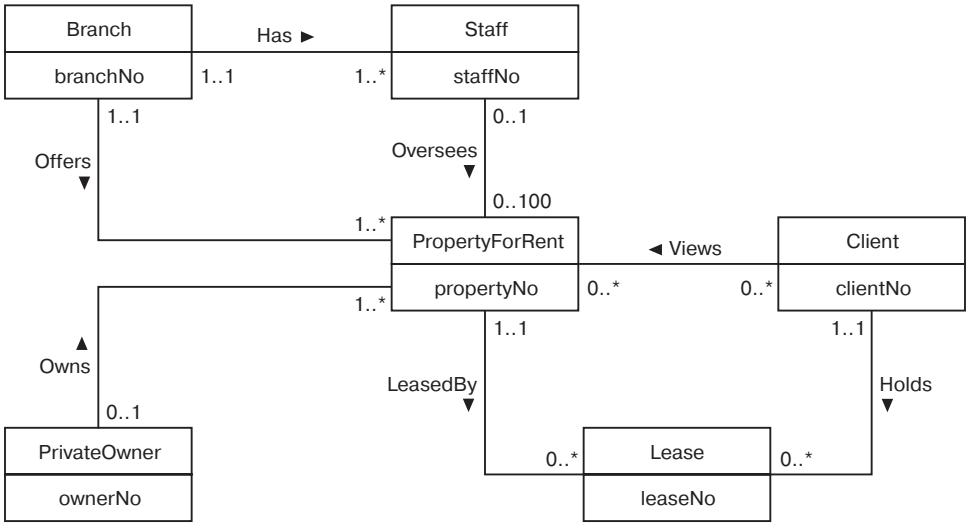


Рис. 1.4. Пример диаграммы "сущность–связь"

СУБД — это программное обеспечение, которое взаимодействует с прикладными программами пользователя и базой данных и обладает перечисленными ниже возможностями.

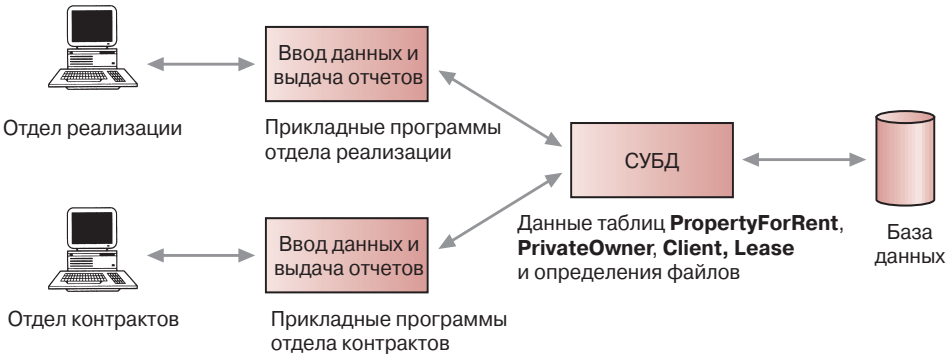
- Позволяет создать базу данных, что обычно осуществляется с помощью *языка определения данных* (DDL — Data Definition Language). Язык DDL предоставляет пользователям средства указания типа данных и их структуры, а также средства задания ограничений для информации, хранимой в базе данных.
- Позволяет вставлять, обновлять, удалять и извлекать информацию из базы данных, что обычно осуществляется с помощью *языка манипулирования данными* (DML — Data Manipulation Language). Наличие централизованного хранилища всех данных и их описаний позволяет использовать язык DML как общий инструмент организации запросов, который иногда называют *языком запросов* (query language). Наличие языка запросов позволяет устранить присущие файловым системам ограничения, при которых пользователям приходится иметь дело только с фиксированным набором запросов или постоянно возрастающим количеством программ, что порождает другие, более сложные проблемы управления программным обеспечением. Наиболее распространенным типом непроцедурного языка является язык структурированных запросов (Structured Query Language — SQL), который в настоящее время определяется специальным стандартом и *фактически* является обязательным языком для любых реляционных СУБД. (SQL произносится либо по буквам "S-Q-L", либо как мнемоническое имя "See-Quel".) В связи с огромной важностью языка SQL авторы этой книги посвятили ему три главы: 5, 6 и 21.
- Предоставляет контролируемый доступ к базе данных с помощью перечисленных ниже средств:
  - системы обеспечения защиты, предотвращающей несанкционированный доступ к базе данных со стороны пользователей;

- системы поддержки целостности данных, обеспечивающей непротиворечивое состояние хранимых данных;
- системы управления параллельной работой приложений, контролирующей процессы их совместного доступа к базе данных;
- системы восстановления, позволяющей восстановить базу данных до предыдущего непротиворечивого состояния, нарушенного в результате сбоя аппаратного или программного обеспечения;
- доступного пользователям каталога, содержащего описание хранимой в базе данных информации.

На рис. 1.5 показан пример реализации подхода с применением базы данных вместо рассмотренного ранее варианта с использованием файловой системы (см. рис. 1.3). В новом варианте отдел реализации и отдел контрактов используют собственные приложения для доступа к общей базе данных, организованной с помощью СУБД. Набор приложений каждого отдела обеспечивает ввод и корректировку данных, а также генерацию необходимых отчетов. Но в отличие от варианта с файловой системой физическая структура и способ хранения данных контролируются с помощью СУБД.

Представления

В связи с наличием указанных выше функциональных возможностей СУБД становится чрезвычайно полезным инструментом. Но поскольку для конечных пользователей не имеет значения, насколько проста или сложна внутренняя организация системы, можно услышать возражения, что СУБД затрудняет работу, предоставляя пользователям гораздо большее количество данных, чем им действительно требуется. Как показано на рис. 1.5, в подходе, основанном на использовании баз данных, необходимые сотрудникам отдела контрактов подробные сведения об объектах недвижимости организованы несколько иначе, чем в варианте с файловой системой, представленном на рис. 1.3. Теперь в базе данных содержатся также сведения о типе недвижимости, числе комнат и о владельце объекта, которые не всегда нужны сотрудникам компании. Для решения проблемы “устранения” излишних данных в СУБД предусмотрен механизм создания



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentFinish)

Рис. 1.5. Схема обработки данных с помощью СУБД

представлений (view), который позволяет любому пользователю иметь свой собственный "образ" базы данных (представление можно рассматривать как некоторое подмножество базы данных). Например, можно организовать представление, в котором сотрудникам отдела контрактов будут доступны только те данные, которые необходимы для оформления договоров аренды.

Помимо упрощения работы за счет предоставления пользователям только действительно нужных им данных, представления обладают несколькими другими достоинствами.

- Обеспечивают дополнительный уровень безопасности. Представления могут создаваться с целью исключения тех данных, которые не должны видеть некоторые пользователи. Например, можно создать некоторое представление, которое позволит менеджерам отделений и сотрудникам расчетного сектора бухгалтерии просматривать все данные о персонале, включая сведения об их зарплате. В то же время для организации доступа к данным других пользователей можно создать еще одно представление, из которого все сведения о зарплате будут исключены.
- Предоставляют механизм настройки внешнего интерфейса базы данных. Например, сотрудники отдела контрактов могут работать с полем `Monthly rent` (Ежемесячная арендная плата), используя для него более короткое и простое имя — `rent`.
- Позволяют сохранять внешний интерфейс базы данных непротиворечивым и неизменным даже при внесении изменений в ее структуру — например, при добавлении или удалении полей, изменении связей, разбиении файлов, их реорганизации или переименовании. Если в файл добавляются или из него удаляются поля, не используемые в некотором представлении, то все эти изменения никак не отразятся на данном представлении. Таким образом, представление обеспечивает полную независимость программ от реальной структуры данных, что позволяет устранить важнейший недостаток файловых систем.

Приведенные выше рассуждения имели несколько общий характер. В действительности реальный объем функциональных возможностей зависит от конкретной СУБД. Например, в СУБД для персонального компьютера может не поддерживаться параллельный совместный доступ, а управление режимом защиты, поддержанием целостности данных и восстановлением будет присутствовать только в очень ограниченной степени. Однако современные мощные многопользовательские СУБД предлагают все перечисленные выше функциональные возможности и многое другое. Современные системы представляют собой чрезвычайно сложное программное обеспечение, состоящее из миллионов строк кода и многих томов документации. Таков результат стремления получить программное обеспечение, которое могло бы удовлетворять требованиям все более общего характера. Более того, в настоящее время использование СУБД предполагает почти стопроцентную надежность и готовность даже при сбоях в аппаратном и программном обеспечении. Программное обеспечение СУБД постоянно совершенствуется и должно все больше и больше расширяться, чтобы удовлетворять все новым требованиям пользователей. Например, в некоторых приложениях теперь требуется хранить графику, видео, звук и т.д. Для охвата этой части рынка СУБД должна развиваться, причем со временем ей, вероятно, потребуется выполнять какие-то новые функции, а потому функциональная часть СУБД никогда не будет неизменной. Более подробно основные функции СУБД рассматриваются в последующих главах.

1.3.3. Компоненты среды СУБД

Как показано на рис. 1.6, в среде СУБД можно выделить следующие пять основных компонентов: аппаратное и программное обеспечение, данные, процедуры и пользователей.

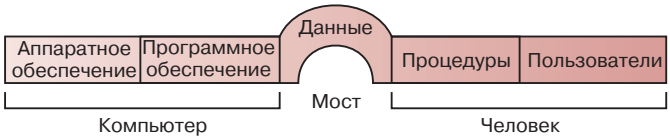


Рис. 1.6. Среда СУБД

Аппаратное обеспечение

Для работы СУБД и приложений необходимо некоторое аппаратное обеспечение. Оно может варьировать в очень широких пределах — от единственного персонального компьютера или одного мэйнфрейма до сети из многих компьютеров. Используемое аппаратное обеспечение зависит от требований данной организации и типа СУБД. Одни СУБД предназначены для работы только с конкретными типами операционных систем или оборудования, другие могут работать с широким кругом аппаратного обеспечения и различными операционными системами. Для работы СУБД обычно требуется некоторый минимум оперативной и дисковой памяти, но такой минимальной конфигурации может оказаться совершенно недостаточно для достижения приемлемой производительности системы. На рис. 1.7 показана упрощенная схема конфигурации аппаратного обеспечения для учебного проекта *DreamHome*. Она состоит из сети мини-компьютеров с центральным компьютером в Лондоне. На центральном компьютере работает *серверная часть* СУБД (backend), которая обслуживает и контролирует доступ к базе данных. На схеме также показано несколько компьютеров, расположенных в других регионах. На них работают *клиентские части* СУБД (frontend), которые осуществляют взаимодействие с пользователями. Подобная архитектура носит название *клиент/сервер* (client-server), где сервером является компьютер с серверной частью СУБД, а клиентами — компьютеры с клиентскими частями СУБД. Более подробно эта архитектура рассматривается в разделе 2.6.

Программное обеспечение

Этот компонент охватывает программное обеспечение самой СУБД и прикладных программ, вместе с операционной системой, включая и сетевое программное обеспечение, если СУБД используется в сети. Обычно приложения создаются на языках третьего поколения, таких как C, C++, Java, Visual Basic, COBOL, Fortran, Ada или Pascal, или на языках четвертого поколения, таких как SQL, операторы которых внедряются в программы на языках третьего поколения. Впрочем, СУБД может иметь свои собственные инструменты четвертого поколения, предназначенные для быстрой разработки приложений с использованием встроенных непроцедурных языков запросов, генераторов отчетов, форм, графических изображений и даже полномасштабных приложений. Использование инструментов четвертого поколения позволяет существенно повысить производительность системы и способствует созданию более удобных для обслуживания программ. Инструменты четвертого поколения рассматриваются в разделе 2.2.3.

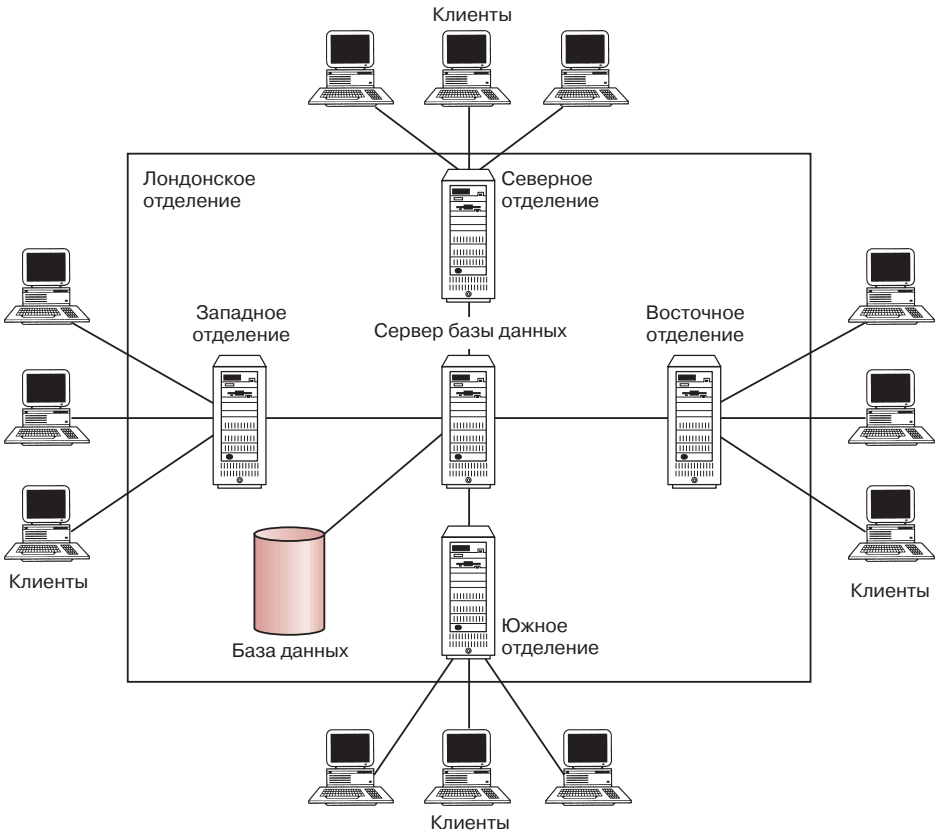


Рис. 1.7. Конфигурация аппаратного обеспечения для учебного проекта DreamHome

Данные

Вероятно, самым важным компонентом среды СУБД (с точки зрения конечных пользователей) являются данные. На рис. 1.6 показано, что данные играют роль моста между компьютером и человеком. База данных содержит как рабочие данные, так и метаданные, т.е. "данные о данных". Структура базы данных называется *схемой* (schema). Показанная на рис. 1.5 схема базы данных состоит из четырех файлов, или *таблиц* (table): `PropertyForRent` (Сдаваемый в аренду объект), `PrivateOwner` (Владелец собственности), `Client` (Клиент) и `Lease` (Договор аренды). Таблица `PropertyForRent` имеет восемь полей, или *атрибутов*: `propertyNo` (Номер объекта), `street` (Улица), `city` (Город), `postcode` (Почтовый индекс), `type` (Тип объекта), `rooms` (Количество комнат), `rent` (Ежемесячная арендная плата) и `ownerNo` (Номер владельца). Атрибут `ownerNo` моделирует связь между таблицей `PropertyForRent` и таблицей `PrivateOwner`, т.е. некий владелец *владеет* (`Owns`) некой сдаваемой в аренду недвижимостью — как показано на диаграмме "сущность–связь", представленной на рис. 1.4. В частности, из табл. 1.1 и 1.2 следует, что владелец под номером `CO46`, `Joe Keogh`, владеет недвижимостью под номером `PA14`.

В системном каталоге содержатся сведения, которые подробно рассматриваются в разделе 2.7.

Процедуры

К процедурам относятся инструкции и правила, которые должны учитываться при проектировании и использовании базы данных. Пользователям и обслуживающему персоналу базы данных необходимо предоставить документацию, содержащую подробное описание процедур использования и сопровождения данной системы, включая инструкции о правилах выполнения приведенных ниже действий.

- Регистрация в СУБД.
- Использование отдельного инструмента СУБД или приложения.
- Запуск и останов СУБД.
- Создание резервных копий СУБД.
- Обработка сбоев аппаратного и программного обеспечения, включая процедуры идентификации вышедшего из строя компонента, исправления отказавшего компонента (например, посредством вызова специалиста по ремонту аппаратного обеспечения), а также восстановления базы данных после устранения неисправности.
- Изменение структуры таблицы, реорганизация базы данных, размещенной на нескольких дисках, способы улучшения производительности и методы архивирования данных на вторичных устройствах хранения.

Пользователи

Последним, еще не рассмотренным нами компонентом среды СУБД являются пользователи системы. Этот компонент подробно обсуждается в разделе 1.4.

1.3.4. Разработка базы данных — смена принципов проектирования

До сих пор по умолчанию предполагалось, что данные в базе обладают некоторой структурой. Например, на рис. 1.5 показаны четыре таблицы: `PropertyForRent`, `PrivateOwner`, `Client` и `Lease`. Но как была получена такая структура? Ответ на этот вопрос достаточно прост: структура базы данных определяется во время ее проектирования. Однако сам процесс проектирования базы данных может оказаться чрезвычайно сложным. Для создания системы, которая удовлетворяла бы информационным потребностям некоторой организации, необходимо использовать подход, совершенно отличающийся от методов разработки обычных файловых систем, в которых вся работа заключается в разработке приложений, удовлетворяющих нуждам отдельных подразделений. Для успешной реализации системы на основе базы данных необходимо подумать прежде всего о данных и лишь потом о приложениях. Такая смена подхода вполне может расцениваться как *смена принципов проектирования*. Чтобы система полностью удовлетворяла запросам пользователей, необходимо очень внимательно отнестись к процессу проектирования базы данных. Плохо спроектированная база данных будет порождать ошибки, способные привести к принятию неправильных решений, которые повлекут за собой самые серьезные последствия для данной организации. С другой стороны, хорошо спроектированная база данных позволит создать систему, поставляющую корректную информацию, которая может успешно использоваться для принятия правильных и эффективных решений.

Цель этой книги заключается в том, чтобы помочь воплотить смену принципов проектирования в жизнь. В нескольких главах книги (главы 14–17) детально описывается методология проектирования баз данных, представленная в виде перечня

последовательно выполняемых этапов. Для каждого этапа предлагаются развернутые рекомендации по его выполнению. Например, на ER-диаграмме, представленной на рис. 1.4, показаны шесть сущностей, семь связей и шесть атрибутов. В предлагаемых рекомендациях будут даны советы, как идентифицировать все те сущности, атрибуты и связи, которые должны быть представлены в базе данных.

К сожалению, существующие методологии проектирования баз данных пока не получили широкого распространения. Большинство организаций или отдельных разработчиков при проектировании баз данных в очень незначительной степени полагается на какие-либо методологии. Именно это обстоятельство часто считается основной причиной неудач при разработке информационных систем. Из-за отсутствия структурированных подходов к проектированию баз данных необходимые для проведения разработки время и ресурсы обычно недооцениваются, а созданные базы данных часто неэффективны или не отвечают требованиям прикладных приложений. Предоставляемая документация часто бывает недостаточна, что чрезвычайно затрудняет сопровождение созданной базы данных.

1.4. Распределение обязанностей в системах с базами данных

В этом разделе мы рассмотрим упомянутый выше пятый компонент среды СУБД — ее пользователей. Среди них можно выделить четыре различные группы: администраторы данных и баз данных, разработчики баз данных, прикладные программисты и конечные пользователи.

1.4.1. Администраторы данных и администраторы баз данных

База данных и СУБД являются корпоративными ресурсами, которыми следует управлять так же, как и любыми другими ресурсами. Обычно управление данными и базой данных предусматривает управление и контроль за СУБД и помещенными в нее данными. *Администратор данных*, или *АД* (Data Administrator — DA), отвечает за управление данными, включая планирование базы данных, разработку и сопровождение стандартов, прикладных алгоритмов и деловых процедур, а также за концептуальное и логическое проектирование базы данных. АД консультирует и дает свои рекомендации руководству высшего звена, контролируя соответствие общего направления развития базы данных установленным корпоративным целям.

Администратор базы данных, или *АБД* (Database Administrator — DBA), отвечает за физическую реализацию базы данных, включая физическое проектирование и воплощение проекта, за обеспечение безопасности и целостности данных, за сопровождение операционной системы, а также за обеспечение максимальной производительности приложений и пользователей. По сравнению с АД обязанности АБД носят более технический характер, и для него необходимо знание конкретной СУБД и системного окружения. В одних организациях между этими ролями не делается различий, а в других важность корпоративных ресурсов отражена именно в выделении отдельных групп персонала с указанным кругом обязанностей. Более подробно администрирование данных и баз данных рассматривается в разделе 9.15.

1.4.2. Разработчики баз данных

В проектировании больших баз данных участвуют разработчики двух разных типов: разработчики логической базы данных и разработчики физической базы данных. *Разработчик логической базы данных* занимается идентификацией данных (т.е. сущностей и их атрибутов), связей между данными, и устанавливает ограничения, накладываемые на хранимые данные. Разработчик логической базы данных должен обладать всесторонним и полным пониманием структуры данных организации и ее *делового регламента*. Деловой регламент описывает основные требования к системе *с точки зрения организации*. Ниже приведен пример типичного делового регламента для проекта *DreamHome*.

- Любой сотрудник не может отвечать одновременно более чем за сто сдаваемых в аренду или продаваемых объектов недвижимости.
- Любой сотрудник не имеет права продавать или сдавать в аренду свою собственную недвижимость.
- Доверенное лицо не может выступать одновременно и как покупатель, и как продавец недвижимости.

Для эффективной работы разработчик логической базы данных должен как можно раньше вовлечь всех предполагаемых пользователей базы данных в процесс создания модели данных. В этой книге работа разработчика логической базы данных делится на два этапа.

- Концептуальное проектирование базы данных, которое совершенно не зависит от таких деталей ее воплощения, как конкретная целевая СУБД, приложения, языки программирования или любые другие физические характеристики.
- Логическое проектирование базы данных, которое проводится с учетом особенностей выбранной модели данных: реляционной, сетевой, иерархической или объектно-ориентированной.

Разработчик физической базы данных получает готовую логическую модель данных и занимается ее физической реализацией, в том числе:

- преобразованием логической модели данных в набор таблиц и ограничений целостности данных;
- выбором конкретных структур хранения и методов доступа к данным, обеспечивающих необходимый уровень производительности при работе с базой данных;
- проектированием любых требуемых мер защиты данных.

Многие этапы физического проектирования базы данных в значительной степени зависят от выбранной целевой СУБД, а потому может существовать несколько различных способов воплощения требуемой схемы. Следовательно, разработчик физической базы данных должен разбираться в функциональных возможностях целевой СУБД и понимать достоинства и недостатки каждого возможного варианта реализации. Разработчик физической базы данных должен уметь выбрать наиболее подходящую стратегию хранения данных с учетом всех существующих особенностей их использования. Если концептуальное и логическое проектирование базы данных отвечает на вопрос "*что?*", то физическое проектирование отвечает на вопрос "*как?*". Для решения этих задач требуются разные навыки работы, которыми чаще всего обладают разные люди. Методология концептуального проектирования базы данных рассматривается в главе 14, логического проектирования — в главе 15, а физического — в главах 16 и 17.

1.4.3. Прикладные программисты

Сразу после создания базы данных следует приступить к разработке приложений, предоставляющих пользователям необходимые им функциональные возможности. Именно эту работу и выполняют *прикладные программисты*. Обычно прикладные программисты работают на основе спецификаций, созданных системными аналитиками. Как правило, каждая программа содержит некоторые операторы, требующие от СУБД выполнения определенных действий с базой данных — например, таких как извлечение, вставка, обновление или удаление данных. Как уже упоминалось в предыдущем разделе, эти программы могут создаваться на различных языках программирования третьего или четвертого поколения.

1.4.4. Пользователи

Пользователи являются клиентами базы данных — она проектируется, создается и поддерживается для того, чтобы обслуживать их информационные потребности. Пользователей можно классифицировать по способу использования ими системы.

- **Рядовые пользователи.** Эти пользователи обычно даже и не подозревают о наличии СУБД. Они обращаются к базе данных с помощью специальных приложений, позволяющих в максимальной степени упростить выполняемые ими операции. Такие пользователи инициируют выполнение операций базы данных, вводя простейшие команды или выбирая команды меню. Это значит, что таким пользователям не нужно ничего знать о базе данных или СУБД. Например, чтобы узнать цену товара, кассир в супермаркете использует сканер для считывания нанесенного на него штрих-кода. В результате этого простейшего действия специальная программа не только считывает штрих-код, но и выбирает на основе его значения цену товара из базы данных, а также уменьшает значение в другом поле базы данных, обозначающем остаток таких товаров на складе, после чего выбивает цену и общую стоимость на кассовом аппарате.
- **Опытные пользователи.** С другой стороны спектра находятся опытные конечные пользователи, которые знакомы со структурой базы данных и возможностями СУБД. Для выполнения требуемых операций они могут использовать такой язык запросов высокого уровня, как SQL. А некоторые опытные пользователи могут даже создавать собственные прикладные программы.

1.5. История развития СУБД

Как уже упоминалось выше, предшественницами СУБД были файловые системы. Однако появление СУБД не привело к полному исчезновению файловых систем. Для выполнения некоторых специализированных задач файловые системы используются до сих пор. Считается, что развитие СУБД началось еще в 1960-е годы, когда разрабатывался проект запуска корабля Apollo на Луну. Этот проект был начат по инициативе президента США Кеннеди, поставившего задачу осуществить пилотируемый полет и высадку человека на Луну к концу десятилетия. В то время не существовало никаких систем, способных обрабатывать или как-либо управлять тем огромным количеством данных, которое было необходимо для реализации этого проекта.

В результате специалисты основного подрядчика — компании North American Aviation (NAA) (которая теперь называется Rockwell International) — разработали программное обеспечение под названием GUAM (Generalized Update Access Method). Основная идея GUAM была построена на том, что малые компоненты

объединяются вместе как части более крупных компонентов до тех пор, пока не будет собран воедино весь проект. Применяемую при этом структуру, напоминающую перевернутое дерево, часто называют *иерархической структурой* (hierarchical structure). В середине 1960-х годов корпорация IBM присоединилась к фирме NAA для совместной работы над GUAM, в результате чего была создана система IMS (Information Management System). Причина, по которой корпорация IBM ограничила функциональные возможности IMS только управлением иерархиями записей, заключалась в том, что необходимо было обеспечить работу с устройствами хранения с последовательным доступом, а именно с магнитными лентами, которые были в то время основным типом носителя. Спустя некоторое время это ограничение удалось преодолеть. Несмотря на то что IMS является самой первой из всех коммерческих СУБД, она до сих пор остается основной иерархической СУБД, используемой на большинстве крупных мэйнфреймов.

Другим заметным достижением середины 1960-х годов было появление системы IDS (Integrated Data Store) фирмы General Electric. Работу над ней возглавлял один из пионеров исследований в области систем управления базами данных — Чарльз Бачман (Charles Bachmann). Развитие этой системы привело к созданию нового типа систем управления базами данных — *сетевых* (network) СУБД, — что оказало существенное влияние на информационные системы того поколения. Сетевая СУБД создавалась для представления более сложных взаимосвязей между данными, чем те, которые можно было моделировать с помощью иерархических структур, а также для формирования стандарта баз данных. Для создания таких стандартов в 1965 году на конференции организации CODASYL (Conference on Data Systems Languages), проходившей при участии представителей правительства США и бизнесменов, была сформирована рабочая группа List Processing Task Force, переименованная в 1967 году в группу Data Base Task Group (DBTG). В компетенцию группы DBTG входило определение спецификаций среды, которая допускала бы разработку баз данных и управление данными. Предварительный вариант отчета этой группы был опубликован в 1969 году, а первый полный вариант — в 1971 году. Предложения группы DBTG содержали три компонента.

- *Сетевая схема* — это логическая организация всей базы данных в целом (с точки зрения АБД), которая включает определение имени базы данных, типа каждой записи и компонентов записей каждого типа.
- *Подсхема* — это часть базы данных, как она видится пользователям или приложениям.
- *Язык управления данными* — инструмент для определения характеристик и структуры данных, а также для управления ими.

Группа DBTG также предложила стандартизировать три языка.

- *Язык определения данных для схемы* (Data Definition Language — DDL), который позволяет АБД ее описать.
- *Язык определения данных для подсхемы* (также DDL), который позволяет определять в приложениях те части базы данных, доступ к которым будет необходим.
- *Язык манипулирования данными* (Data Manipulation Language — DML), предназначенный для управления данными.

Несмотря на то что этот отчет официально не был утвержден Национальным институтом стандартизации США (American National Standards Institute — ANSI), большое количество систем было разработано в полном соответствии с этими предложениями группы DBTG. Теперь они называются *CODASYL-системами*, или *DBTG-системами*. CODASYL-системы и системы на основе ие-

рархических подходов представляют собой СУБД *первого поколения*. Более подробно они рассматриваются в материалах, представленных на сопровождающем Web-узле (URL которого приведен во введении к данной книге). Однако этим двум моделям присущи перечисленные ниже недостатки.

- Даже для выполнения простых запросов с использованием переходов и доступом к определенным записям необходимо создавать достаточно сложные программы.
- Независимость от данных существует лишь в минимальной степени.
- Отсутствие общепризнанных теоретических основ.

В 1970 году Э. Ф. Кодд (E. F. Codd), работавший в исследовательской лаборатории корпорации IBM, опубликовал очень важную и весьма своевременную статью о реляционной модели данных, позволявшей устранить недостатки прежних моделей. Вслед за этим появилось множество экспериментальных реляционных СУБД, а первые коммерческие продукты появились в 1970–1980-х годах. Особенно следует отметить проект System R, разработанный в исследовательской лаборатории корпорации IBM, расположенной в городе Сан-Хосе, штат Калифорния, созданный в конце 1970-х годов [9]. Этот проект был задуман с целью доказать практичность реляционной модели, что достигалось посредством реализации предусмотренных ею структур данных и требуемых функциональных возможностей. На основе этого проекта были получены важнейшие результаты.

- Был разработан структурированный язык запросов SQL, который с тех пор стал стандартным языком любых реляционных СУБД.
- В 1980-х годах были созданы различные коммерческие реляционные СУБД — например, DB2 или SQL/DS корпорации IBM или Oracle корпорации Oracle Corporation.

В настоящее время существует несколько сотен различных реляционных СУБД для мэйнфреймов и персональных компьютеров, хотя во многих из них определение реляционной модели трактуется слишком широко. В качестве примеров многопользовательских СУБД могут служить система INGRES II фирмы Computer Associates и система Informix фирмы Informix Software, Inc. Примерами реляционных СУБД для персональных компьютеров являются Access и FoxPro фирмы Microsoft, Paradox фирмы Corel Corporation, InterBase и BDE фирмы Borland, а также R:Base фирмы R:Base Technologies. Реляционные СУБД относятся к СУБД *второго поколения*. Более подробно реляционная модель данных рассматривается в главе 3.

Однако реляционная модель обладает также некоторыми недостатками — в частности, ограниченными возможностями моделирования. Для решения этой проблемы был выполнен большой объем исследовательской работы. В 1976 году Чен (Chen) предложил модель “сущность–связь” (Entity-Relationship model — ER-модель), которая в настоящее время стала самой распространенной технологией проектирования баз данных и является основой методологии, описанной в главах 14 и 15. В 1979 году Кодд сделал попытку устранить недостатки собственной основополагающей работы и опубликовал расширенную версию реляционной модели — RM/T (1979), затем еще одну версию — RM/V2 (1990). Попытки создания модели данных, позволяющей более точно описывать реальный мир, неформально называют *семантическим моделированием данных* (semantic data modeling).

В ответ на все возрастающую сложность приложений баз данных появились две новые системы: *объектно-ориентированные СУБД*, или ООСУБД (Object-Oriented DBMS — OODBMS), и *объектно-реляционные СУБД*, или ОРСУБД (Object-Relational DBMS — ORDBMS). Однако, в отличие от предыдущих моделей, действительная структура этих моделей не совсем ясна. Попытки реализации подобных моделей представляют собой СУБД *третьего поколения*, которые более подробно будут рассмотрены в главах 24–27.

1.6. Преимущества и недостатки СУБД

СУБД обладают как многообещающими потенциальными преимуществами, так и недостатками, которые мы кратко рассмотрим в этом разделе.

Преимущества

Преимущества систем управления базами данных перечислены в табл. 1.8.

Таблица 1.8. Преимущества систем управления базами данных

Преимущество
Контроль за избыточностью данных
Непротиворечивость данных
Больше полезной информации при том же объеме хранимых данных
Совместное использование данных
Поддержка целостности данных
Повышенная безопасность
Применение стандартов
Повышение эффективности с ростом масштабов системы
Возможность нахождения компромисса при противоречивых требованиях
Повышение доступности данных и их готовности к работе
Улучшение показателей производительности
Упрощение сопровождения системы за счет независимости от данных
Улучшенное управление параллельной работой
Развитые службы резервного копирования и восстановления

Контроль за избыточностью данных

Как уже говорилось в разделе 1.2, традиционные файловые системы неэкономно расходуют внешнюю память, сохраняя одни и те же данные в нескольких файлах. Например, на рис. 1.3 в файлах отдела реализации и файлах отдела контрактов хранятся одинаковые сведения об арендуемой недвижимости и арендаторах. При использовании базы данных, наоборот, предпринимается попытка исключить избыточность данных за счет интеграции файлов, что позволяет исключить необходимость хранения нескольких копий одного и того же элемента информации. Однако полностью избыточность информации в базах данных не исключается, а лишь ограничивается ее степень. В одних случаях ключевые элементы данных необходимо дублировать для моделирования связей, а в других случаях некоторые данные требуется дублировать из соображений повышения производительности системы. Причины введения в базу контролируемой избыточности данных станут понятны при чтении следующих глав.

Непротиворечивость данных

Устранение избыточности данных или контроль над ней позволяет уменьшить риск возникновения противоречивых состояний. Если элемент данных хранится в базе только в одном экземпляре, то для изменения его значения потребуется выполнить только одну операцию обновления, причем новое значение станет

доступным сразу всем пользователям базы данных. А если этот элемент данных с ведома системы хранится в базе данных в нескольких экземплярах, то такая система сможет следить за тем, чтобы копии не противоречили друг другу. К сожалению, во многих современных СУБД такой способ обеспечения непротиворечивости данных не поддерживается автоматически.

Больше полезной информации при том же объеме хранимых данных

Благодаря интеграции рабочих данных организации на основе тех же данных можно получать дополнительную информацию. Например, в показанной на рис. 1.3 файловой системе сотрудникам отдела контрактов неизвестны владельцы сданных в аренду объектов. Аналогично, сотрудники отдела реализации не имеют полных сведений о договорах аренды. При интеграции этих файлов в общей базе сотрудники отдела контрактов получают доступ к сведениям о владельцах, а сотрудники отдела реализации — к сведениям о договорах аренды. Теперь на основе тех же данных пользователи смогут получать больше информации.

Совместное использование данных

Файлы обычно принадлежат отдельным лицам или целым отделам, которые используют их в своей работе. В то же время база данных принадлежит всей организации в целом и может совместно использоваться всеми зарегистрированными пользователями. При такой организации работы большее количество пользователей может работать с бо́льшим объемом данных. Более того, при этом можно создавать новые приложения на основе уже существующей в базе данных информации и добавлять в нее только те данные, которые в настоящий момент еще не хранятся в ней, а не определять заново требования ко всем данным, необходимым новому приложению. Новые приложения могут также использовать такие предоставляемые типичными СУБД функциональные возможности, как определение структур данных и управление доступом к данным, организация параллельной обработки и обеспечение средств копирования/восстановления, исключив необходимость реализации этих функций со своей стороны.

Поддержка целостности данных

Целостность базы данных означает корректность и непротиворечивость хранимых в ней данных. Целостность обычно описывается с помощью *ограничений*, т.е. правил поддержки непротиворечивости, которые не должны нарушаться в базе данных. Ограничения можно применять к элементам данных внутри одной записи или к связям между записями. Например, ограничение целостности может гласить, что зарплата сотрудника не должна превышать 40 000 фунтов стерлингов в год или же что в записи с данными о сотруднике номер отделения, в котором он работает, должен соответствовать реально существующему отделению компании. Таким образом, интеграция данных позволяет АБД задавать требования по поддержке целостности данных, а СУБД применять их.

Повышенная безопасность

Безопасность базы данных заключается в защите базы данных от несанкционированного доступа со стороны пользователей. Без привлечения соответствующих мер безопасности интегрированные данные становятся более уязвимыми, чем данные в файловой системе. Однако интеграция позволяет АБД определить требуемую систему безопасности базы данных, а СУБД привести ее в действие. Система обеспечения безопасности может быть выражена в форме имен и паролей для идентификации пользователей, которые зарегистрированы в этой базе данных. Доступ к данным со стороны зарегистрированного пользователя может

быть ограничен только некоторыми операциями (извлечением, вставкой, обновлением и удалением). Например, АБД может быть предоставлено право доступа ко всем данным в базе данных, менеджеру отделения компании — ко всем данным, которые относятся к его отделению, а инспектору отдела реализации — лишь ко всем данным о недвижимости, в результате чего он не будет иметь доступа к конфиденциальным данным, таким как, зарплата сотрудников.

Применение стандартов

Интеграция позволяет АБД определять и применять необходимые стандарты. Например, стандарты отдела и организации, государственные и международные стандарты могут регламентировать формат данных при обмене ими между системами, соглашения об именах, форму представления документации, процедуры обновления и правила доступа.

Повышение эффективности с увеличением масштабов системы

Комбинируя все рабочие данные организации в одной базе данных и создавая набор приложений, которые работают с одним источником данных, можно добиться существенной экономии средств. В этом случае бюджет, который обычно выделялся каждому отделу для разработки и поддержки их собственных файловых систем, можно объединить с бюджетами других отделов (с более низкой общей стоимостью), что позволит добиться повышения эффективности при росте масштабов производства. Теперь объединенный бюджет можно будет использовать для приобретения оборудования в той конфигурации, которая в большей степени отвечает потребностям организации. Например, она может состоять из одного мощного компьютера или сети из небольших компьютеров.

Возможность нахождения компромисса для противоречивых требований

Потребности одних пользователей или отделов могут противоречить потребностям других пользователей. Но поскольку база данных контролируется АБД, он может принимать решения о проектировании и способе использования базы данных, при которых имеющиеся ресурсы всей организации в целом будут использоваться наилучшим образом. Эти решения обеспечивают оптимальную производительность для самых важных приложений, причем чаще всего за счет менее критичных.

Повышение доступности данных и их готовности к работе

Данные, которые пересекают границы отделов, в результате интеграции становятся непосредственно доступными конечным пользователям. Потенциально это повышает функциональность системы, что, например, может быть использовано для более качественного обслуживания конечных пользователей или клиентов организации. Во многих СУБД предусмотрены языки запросов или инструменты для создания отчетов, которые позволяют пользователям вводить не предусмотренные заранее запросы и почти немедленно получать требуемую информацию на своих терминалах, не прибегая к помощи программиста, который для извлечения этой информации из базы данных должен был бы создать специальное программное обеспечение. Например, менеджер отделения компании может получить перечень всех сдаваемых в аренду квартир с месячной арендной платой свыше 400 фунтов стерлингов, введя на своем терминале следующий оператор SQL:

```
SELECT *
FROM PropertyForRent
WHERE type='Flat' AND rent>400;
```

Улучшение показателей производительности

Как уже упоминалось выше, в СУБД предусмотрено много стандартных функций, которые программист обычно должен самостоятельно реализовать в приложениях для файловых систем. На базовом уровне СУБД обеспечивает все низкоуровневые процедуры работы с файлами, которую обычно выполняют приложения. Наличие этих процедур позволяет программисту сконцентрироваться на разработке более специальных, необходимых пользователям функций, не заботясь о подробностях их воплощения на более низком уровне. Во многих СУБД предусмотрена также среда разработки четвертого поколения с инструментами, упрощающими создание приложений баз данных. Результатом является повышение производительности работы программистов и сокращение времени разработки новых приложений (с соответствующей экономией средств).

Упрощение сопровождения системы за счет независимости от данных

В файловых системах описания данных и логика доступа к данным встроены в каждое приложение, поэтому программы становятся зависимыми от данных. Для изменения структуры данных (например, для увеличения длины поля с адресом с 40 символов до 41 символа) или для изменения способа хранения данных на диске может потребоваться существенно преобразовать все программы, на которые эти изменения способны оказать влияние. В СУБД подход иной: описания данных отделены от приложений, а потому приложения защищены от изменений в описаниях данных. Эта особенность называется *независимостью от данных*. Подробнее речь о ней пойдет в разделе 2.1.5. Наличие независимости программ от данных значительно упрощает обслуживание и сопровождение приложений, работающих с базой данных.

Улучшенное управление параллельной работой

В некоторых файловых системах при одновременном доступе к одному и тому же файлу двух пользователей может возникнуть конфликт двух запросов, результатом которого будет потеря информации или утрата ее целостности. В свою очередь, во многих СУБД предусмотрена возможность параллельного доступа к базе данных и гарантируется отсутствие подобных проблем. Более подробно управление параллельным доступом к данным обсуждается в главе 19.

Развитые службы резервного копирования и восстановления

Ответственность за обеспечение защиты данных от сбоев аппаратного и программного обеспечения в файловых системах возлагается на пользователя. Так, может потребоваться каждую ночь выполнять резервное копирование данных. При этом в случае сбоя может быть восстановлена резервная копия, но результаты работы, выполненной после резервного копирования, будут утрачены, и данную работу потребуется выполнить заново. В современных СУБД предусмотрены средства снижения вероятности потерь информации при возникновении различных сбоев. Подробнее методы восстановления баз данных после сбоя рассматриваются в разделе 19.3.

Недостатки

Недостатки подхода, связанного с применением баз данных, перечислены в табл. 1.9.

Таблица 1.9. Недостатки систем управления базами данных

Недостаток
Сложность
Размер
Стоимость СУБД
Дополнительные затраты на аппаратное обеспечение
Затраты на преобразование
Производительность
Более серьезные последствия при выходе системы из строя

Сложность

Обеспечение функциональности, которой должна обладать каждая хорошая СУБД, сопровождается значительным усложнением программного обеспечения СУБД. Чтобы воспользоваться всеми преимуществами СУБД, проектировщики и разработчики баз данных, администраторы данных и администраторы баз данных, а также конечные пользователи должны хорошо понимать функциональные возможности СУБД. Непонимание принципов работы системы может привести к неудачным результатам проектирования, что будет иметь самые серьезные последствия для всей организации.

Размер

Сложность и широта функциональных возможностей приводит к тому, что СУБД становится чрезвычайно сложным программным продуктом, который может потребовать много места на диске и нуждаться в большом объеме оперативной памяти для эффективной работы.

Стоимость СУБД

В зависимости от имеющейся вычислительной среды и требуемых функциональных возможностей стоимость СУБД может изменяться в очень широких пределах. Например, однопользовательская СУБД для персонального компьютера может стоить около 100 долларов. Однако большая многопользовательская СУБД для мэйнфрейма, обслуживающая сотни пользователей, может быть чрезвычайно дорогостоящей: от 100 000 до 1 000 000 долларов. Кроме того, следует учесть ежегодные расходы на сопровождение системы, которые составляют некоторый процент от ее общей стоимости.

Дополнительные затраты на аппаратное обеспечение

Для удовлетворения требований, предъявляемых к дисковым накопителям со стороны СУБД и базы данных, может понадобиться приобрести дополнительные устройства хранения информации. Более того, для достижения требуемой производительности может понадобиться более мощный компьютер, который, возможно, будет работать только с СУБД. Приобретение другого дополнительного аппаратного обеспечения приведет к дальнейшему росту затрат.

Затраты на преобразование

В некоторых ситуациях стоимость СУБД и дополнительного аппаратного обеспечения может оказаться несущественной по сравнению со стоимостью преобразования существующих приложений для работы с новой СУБД и новым аппаратным обеспечением. Эти затраты включают также стоимость подготовки персонала для

работы с новой системой, а также оплату услуг специалистов, которые будут оказывать помощь в преобразовании и запуске новой системы. Все это является одной из основных причин, по которой некоторые организации остаются сторонниками прежних систем и не хотят переходить к более современным технологиям управления базами данных. Термин *традиционная система* иногда используется для обозначения устаревших и, как правило, не самых лучших систем.

Производительность

Обычно файловая система создается для некоторых специализированных приложений, например для оформления счетов, а потому ее производительность может быть весьма высока. Однако СУБД предназначены для решения более общих задач и обслуживания сразу нескольких приложений, а не какого-то одного из них. В результате многие приложения в новой среде будут работать не так быстро, как прежде.

Более серьезные последствия при выходе системы из строя

Централизация ресурсов повышает уязвимость системы. Поскольку работа всех пользователей и приложений зависит от готовности к работе СУБД, выход из строя одного из ее компонентов может привести к полному прекращению всей работы организации.

Резюме

- **Система управления базами данных (СУБД)** является базовой структурой информационной системы, в корне изменившей методы работы многих организаций. СУБД все еще остается объектом интенсивных научных исследований, и для многих важных задач все еще не удалось найти удовлетворительное решение.
- Предшественником СУБД была **файловая система**, т.е. набор приложений, которые выполняли отдельные необходимые для пользователя операции, такие как создание отчетов. Каждая программа определяла и управляла своими собственными данными. Хотя файловая система была значительным достижением по сравнению с ручной картотекой, ее использование все еще было сопряжено с большими проблемами, которые в основном были связаны с избыточностью данных и зависимостью программ от данных.
- Появление СУБД было вызвано необходимостью разрешить проблемы, характерные для файловых систем. База данных — это совместно используемый набор логически связанных данных (и описание этих данных), который предназначен для удовлетворения информационных потребностей организации. **СУБД** — это программное обеспечение, которое позволяет пользователям определять, создавать и обслуживать базу данных, а также управлять доступом к ней.
- Доступ к базе данных осуществляется с помощью СУБД. Для этого предусмотрен язык определения данных (Data Definition Language — **DDL**), с помощью которого пользователи могут определять структуру базы данных, а также язык управления данными (Data Manipulation Language — **DML**), с помощью которого пользователи могут вставлять, удалять и извлекать данные из базы.
- СУБД позволяет организовать контроль за доступом пользователей к базе данных. Она предоставляет средства поддержки безопасности и целостности данных, обеспечивает параллельную работу многих приложений, средства копирования/восстановления, а также позволяет организовать доступный пользователям каталог. В типичной СУБД также предусмотрен механизм создания представлений, предназначенных для упрощения вида данных, с которыми имеют дело пользователи.

- **Среда СУБД** состоит из аппаратного обеспечения (компьютеров), программного обеспечения (СУБД, операционной системы и приложений), данных, процедур и пользователей. В данном контексте к пользователям относятся администраторы данных и баз данных, проектировщики баз данных, прикладные программисты и конечные пользователи.
- Корни СУБД лежат в файловых системах. Иерархические и CODASYL-системы представляют собой первое поколение СУБД. Типичным представителем **иерархической модели** является система IMS (Information Management System), а **сетевой (CODASYL-модели)** — система IDS (Integrated Data System). Обе они появились в середине 1960-х годов. **Реляционная модель**, впервые предложенная Э. Ф. Коддом в 1970 году, представляет собой второе поколение СУБД. Она оказала значительное влияние на сообщество разработчиков СУБД, и в настоящее время существует более 100 различных типов реляционных СУБД. Третье поколение СУБД представляют **объектно-реляционные** СУБД и **объектно-ориентированные** СУБД.
- Среди преимуществ подхода, основанного на использовании баз данных, следует отметить контролируемую избыточность данных, непротиворечивость данных, совместное использование данных, повышенную безопасность и целостность. А среди недостатков можно указать сложность, высокую стоимость и снижение производительности приложений, а также возможность весьма серьезных последствий при выходе системы из строя.

Вопросы

1.1. Приведите четыре примера СУБД, помимо тех, которые перечислены в разделе 1.1.

1.2. Объясните значение следующих терминов:

а) данные;

б) база данных;

в) система управления базами данных;

г) независимость от данных;

д) безопасность;

е) целостность;

ж) представления.

1.3. Опишите подход, используемый для обработки данных в файловых системах. В чем состоят основные недостатки этого подхода?

1.4. Опишите основные характеристики подхода, основанного на использовании базы данных, и сравните их с характеристиками обычных файловых систем.

1.5. Опишите пять компонентов среды СУБД. Как они связаны друг с другом?

1.6. Объясните роли следующих групп пользователей базы данных:

а) администратор данных;

б) администратор базы данных;

в) проектировщик логической части базы данных;

г) проектировщик физической части базы данных;

д) прикладной программист;

е) обычные пользователи.

1.7. Каковы основные достоинства и недостатки систем управления базами данных?