Л. В. Рудикова

БАЗЫ ДАННЫХ
РАЗРАБОТКА ПРИЛОЖЕНИЙ

ДЛЯ СТУДЕНТА

SELECT

bhv

Л. В. Рудикова

Базы данных
Разработка приложений
для студента

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.2
Р83

Рудикова Л. В.

Р83 Базы данных. Разработка приложений. — СПб.: БХВ-Петербург, 2006. — 496 с.: ил.

ISBN 5-94157-805-9

Книга является практическим руководством по созданию баз данных и приложений, использующих базы данных. Материал тщательно подобран с целью максимального удовлетворения запросов студенческой аудитории при сохранении компактного объема. Рассматриваются: реляционная модель данных, реляционная алгебра, язык SQL, создание пользовательских приложений средствами Microsoft Access, разработка клиент-серверных приложений с использованием InterBase и Delphi, новые направления в развитии баз данных и т. д. В книге более 110-ти разобранных примеров с пошаговыми инструкциями по их выполнению и свыше 230-ти задач для самостоятельного решения.

Для широкого круга пользователей и разработчиков баз данных

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.03.06.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 31.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

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

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

ISBN 5-94157-805-9

Оглавление

Введение

Организация автоматизированных систем занимает значительное место в становлении единого информационного пространства и компьютеризации многих аспектов промышленной, хозяйственной и учебной деятельности, различные направления которых связаны с хранением и обработкой больших массивов данных. Обращение к массивам данных лежит в основе практически любой компьютерной системы, построение которой в той или иной мере связано с использованием базы данных.

Книга является практическим руководством по реализации баз данных. Она написана с учетом современных требований, предъявляемых к правильной организации баз данных и охватывает обширный теоретический и практический материал: реляционная модель данных, реляционная алгебра, углубленное знакомство с языком SQL, создание пользовательских приложений средствами Microsoft Access, разработка клиент-серверных приложений средствами InterBase и т. д.

В книге использованы следующие программные продукты:

- Microsoft Access — система управления реляционными базами данных для создания различных автоматизированных приложений;
- InterBase — промышленный сервер баз данных для создания приложений типа "клиент-сервер";
- Delphi — объектно-ориентированная среда программирования для разработки клиентских приложений пользователя.

Представленные в книге примеры и задачи, а также большое количество индивидуальных заданий предназначены для

углубленного освоения различных практических аспектов, связанных с базами данных.

Книга состоит из 7 глав, каждая из которых посвящена определенной тематике реализации либо использования баз данных. Материал книги может быть полезен:

- в качестве учебного пособия для студентов, изучающих базы данных в различных курсах систем обработки данных, систем управления базами данных и т. д.;
- преподавателям при подготовке лекций и проведении практических и лабораторных работ;
- пользователям, разработчикам и программистам — для расширения профессиональных возможностей при использовании средств проектирования и реализации баз данных.

Автор выражает благодарность Зайцу Юрию Эдуардовичу, Горчичко Сергею Алексеевичу и Щегловой Ольге Леонидовне за помощь, оказанную при подготовке рукописи к изданию, а также всему издательскому коллективу группы "БХВ-Петербург".

Глава 1

Основы баз данных

В середине 60-х годов развитие вычислительной техники, а также потребности общества, прежде всего, потребности менеджмента, привели к появлению первых коммерческих информационных систем, которые позволили хранить длительное время и обрабатывать необходимую информацию, представляющую собой большие массивы данных.

Первые коммерческие информационные системы использовались, прежде всего, для ведения бухгалтерии: составление различных отчетов, ведомостей, сводов и т. д. Как правило, эти системы выполняли основные функции по обработке документов, в силу чего они получили название *систем обработки данных*.

С точки зрения организации данных это были файловые системы, позволявшие хранить большое количество информации в течение продолжительного времени. Однако первые информационные системы выполняли в основном лишь канцелярскую работу и обладали рядом существенных недостатков. В общем случае эти системы не давали гарантии, что данные не будут потеряны, если они не скопированы. Они не поддерживали также эффективного доступа к данным, расположенным в неизвестном файле, язык запросов на данные в файлах и т. д. Допуская параллельный доступ к файлам множества пользователей или процессов, они не предотвращают

ситуации изменения одного и того же файла одновременно многими пользователями, поэтому изменения одного из пользователей в файле могут вообще не появиться.

Более поздние системы перешли к накоплению и управлению информацией, которая на сегодняшний момент является важнейшим фактором существования любой организации.

Первые системы управления базами данных (СУБД) возникли из систем, которые обеспечивали пользователю возможность визуально воспринимать данные в основном так, как они хранились. В этих системах применялись различные модели данных для описания структуры хранимой информации в базе данных. Главные из них — иерархическая модель, основанная на деревьях, и сетевая модель, основанная на графах.

Недостаток первых моделей и систем состоял в том, что они не поддерживали языки запросов высокого уровня.

В 1970 году появилась статья Эдгара Кодда о представлении данных, организованных в виде двумерных таблиц, называемых отношениями (Codd E. F., "A relational model for large shared data banks", Comm. ACM, 13:6, pp. 377—387). С этого момента реляционная модель широко используется при создании различных баз данных. Следует отметить, что пользователь реляционной системы не связан со структурой памяти, в отличие от пользователя прежних систем БД. Запросы можно выражать на языке очень высокого уровня. Поставщиком первых реляционных и репрезентационных СУБД была фирма IBM.

Примерами систем, в которых используются базы данных, могут быть различные банковские системы, системы резервирования авиационных либо железнодорожных билетов, различные системы автоматизированного управления предприятием и т. д.

В настоящее время системы управления базами данных должны служить, прежде всего, основой информационных систем корпоративного управления и поддерживать принцип открытой системы, основу которого составляет согласованность

объединенного вместе различного оборудования и программного обеспечения.

Понятие базы данных

Под *системой с базой данных* обычно понимается любая информационная система на базе компьютера, в которой данные могут совместно использоваться многими приложениями.

Поясним также некоторые основные понятия, связанные с данным определением.

Данные — разрозненные факты предметной области (описания, опросные и анкетные данные, ведомости и т. д.).

Информация — организованные и обработанные данные, как правило, информация представляет собой структурированные факты предметной области.

Замечание

Часто при рассмотрении различных вопросов, связанных с обработкой информации, понятия "данные" и "информация" используются как синонимы. При этом имеется в виду, что обработка касается всегда только структурированных фактов предметной области, т. е. реально подразумевается обработка информации.

Информационная система — некоторая автоматизированная система на базе компьютера либо на базе иной вычислительной техники, которая организует данные и выдает требуемую информацию.

Информационно-управляющая система — некоторая автоматизированная система на базе компьютера либо на базе иной вычислительной техники, обеспечивающая информационную поддержку менеджмента.

Под *базой данных* понимается множество взаимосвязанных элементарных групп данных (информации), которые могут обрабатываться одной или несколькими прикладными системами.

Каждая СУБД должна удовлетворять следующим требованиям:

- обеспечивать пользователю возможность создавать новые базы данных и определять их *схему (логическую структуру данных)* с помощью специального языка — *языка определения данных*; поддерживать разнообразные представления одних и тех же данных;
- позволять *"запрашивать"* данные (информацию из базы) и изменять их с помощью *языка запросов*, или *языка манипулирования данными*; допускать интеграцию и совместное использование данных различными приложениями;
- поддерживать хранение очень больших массивов данных, измеряемых гигабайтами и более, в течение длительного времени, защищая их от случайной порчи и неавторизованного использования, а также обеспечивать модификацию базы данных в случае необходимости и доступ к данным путем запросов, т. е. гарантировать безопасность и целостность данных;
- контролировать доступ к данным одновременно для многих пользователей; исключать влияние запроса одного пользователя на запрос другого и не допускать одновременный доступ, который может испортить данные, т. е. гарантировать управление параллельным доступом к данным.

Таким образом, в системе с базой данных можно выделить несколько компонентов.

- Пользователи — люди, которые используют информацию, находящуюся в базе данных. Здесь можно выделить следующие группы пользователей:
  - системные администраторы — отвечают за основные операции системы;
  - администраторы базы данных — управляют работой СУБД и обеспечивают функционирование базы данных;
  - проектировщики базы данных — разрабатывают структуру базы данных;

- системные аналитики — определяют основные функции системы базы данных и проектируют формы ввода данных, отчеты и процедуры, с помощью которых обеспечиваются доступ к данным и манипулирование (добавление, изменение, удаление) данными;
- программисты — создают программный код;
- непосредственные пользователи — используют прикладные программы для выполнения необходимых операций по автоматизации деятельности некоторого подразделения и т. д.

- Приложения — программы пользователей, которым необходима информация из системы.
- СУБД — программное обеспечение, которое управляет доступом к данным и обеспечивает указанные функциональные возможности системы с базой данных.
- Информация — обработанные данные (строки, хранящиеся в файлах).
- Хост-система — компьютерная система, в которой хранятся файлы. Доступ к строкам данных осуществляется хост-системой. Роль СУБД состоит в том, чтобы генерировать запросы, позволяющие использовать функциональные возможности системы управления файлами хост-системы для обслуживания различных приложений. СУБД представляет собой дополнительный уровень программного обеспечения, надстроенный над программным обеспечением хост-системы.
- Оборудование — все системные программные средства (универсальный компьютер (mainframe), персональный компьютер (ПК), ноутбук, карманный компьютер).
- Периферийное оборудование — физические устройства, обеспечивающие ввод/вывод, а также электронные устройства для подключения дополнительных компьютеров и организации сети.

Графически систему с базой данных можно представить в виде логической последовательности уровней, представленной на рис. 1.1.

Рис. 1.1. Уровни системы с базой данных

На самом нижнем уровне находятся данные, хранящиеся в физических файлах (физическая память базы данных). На верхнем уровне располагаются приложения, у которых имеется собственное представление одних и тех же физических данных. Каждое представление базы данных предполагает определенную логическую структуру, построенную из лежащих в основе физических данных. Чтобы обеспечить интерфейс между физической памятью базы данных и ее разнообразными логическими версиями (множеством поддерживаемых представлений), СУБД, в свою очередь, также состоит из нескольких уровней.

Трехуровневая архитектура СУБД

Первые попытки стандартизации общей архитектуры СУБД относятся к 1971 году, в котором предложен двухуровневый подход к архитектуре СУБД на основе использования системного представления, включающего понятие схемы базы данных и пользовательских представлений, т. е. подсхем.

В 1978 году комитетом ANSI/SPARC (ANSI, American National Standard Institute — Национальный институт стандартизации США; SPARC, Standart Planning and Requirements Committee — Комитет планирования стандартов и норм) официально зафиксировано различие между логическим и физическим представлением данных. В частности, была предложена обобщенная структура систем с базой данных. Эта структура получила название *трехуровневой архитектуры*, включающей в себя внутренний, концептуальный и внешний уровни (рис. 1.2).

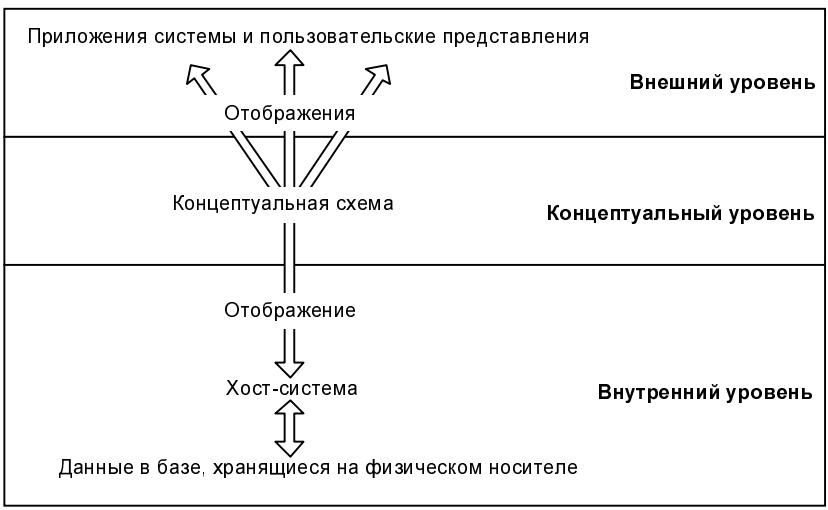

Рис. 1.2. Трехуровневая архитектура системы управления базой данных

Введение трехуровневой архитектуры базы данных позволило отделить пользовательское представление базы данных от ее физического представления. Оно обусловлено, прежде всего, следующими причинами:

- каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя собственное представление об этих данных, а также изменять его при необходимости, что не должно оказывать влияние на представления о данных других пользователей;
- обращение пользователя к базе данных не должно зависеть от особенностей хранения в ней данных;
- администратор базы данных может при необходимости изменять структуру хранения данных в базе, включая концептуальную структуру базы данных, причем данные действия не должны влиять на пользовательские представления данных;
- внутренняя структура хранения данных не должна зависеть от изменения физических устройств хранения информации.

Сформулируем различия между соответствующими уровнями архитектуры базы данных.

Внутренний уровень — уровень, определяющий физический вид базы данных, наиболее близкий к физическому хранению. Он связан со способами сохранения информации на физических устройствах. К нему имеют отношение дисководы, физические адреса, индексы, указатели и т. д. За этот уровень отвечают проектировщики физической базы данных, которые решают, какие физические устройства будут хранить данные, какие методы доступа будут использоваться для извлечения и обновления данных и какие меры следует принять для поддержания или повышения быстродействия системы управления базами данных. Пользователи не касаются данного уровня.

Концептуальный уровень — структурный уровень, который дает представление о логической схеме базы данных. На данном уровне выполняется концептуальное проектирование ба-

зы данных, которое включает анализ информационных потребностей пользователей и определение нужных им элементов данных. Результатом концептуального проектирования является концептуальная схема, логическое описание всех элементов данных и отношений между ними.

Внешний уровень — структурный уровень базы данных, определяющий пользовательские представления данных. Каждая пользовательская группа (либо пользователь) получает свое собственное представление данных в базе данных. Каждое такое представление данных дает ориентированное на пользователя описание элементов данных, из которых состоит представление данных и отношений между ними. Его можно напрямую вывести из концептуальной схемы. Совокупность таких пользовательских представлений данных и образует внешний уровень.

Под *схемой базы данных* понимается общее описание базы данных. В соответствии с трехуровневой архитектурой различают три различных типа схем базы данных.

Внешнему уровню представления базы данных соответствует, как правило, несколько *внешних схем (подсхем)*, которые соответствуют различным представлениям данных пользователей СУБД.

Концептуальная схема описывает все элементы данных, связи между ними, а также необходимые ограничения для поддержки целостности данных. Для каждой базы данных имеется только одна концептуальная схема данных.

Внутренняя схема является полным описанием внутренней модели данных и содержит определения хранимых записей, методы представления, описания полей данных, сведения об индексах и выбранных схемах хеширования. По аналогии с концептуальной схемой, для каждой базы данных имеется также только одна внутренняя схема.

Исходя из трехуровневого представления, а также из различных схем базы данных, следует, что СУБД должна устанавливать соответствие и следить за непротиворечивостью типов схем: внешними, концептуальной и внутренней. Концепту-

альная схема является центральным связующим звеном между каждой внешней схемой и внутренней схемой базы данных. Концептуальная схема связана с внутренней схемой посредством внутреннего концептуального отображения, в свою очередь, каждая внешняя схема связана с концептуальной схемой с помощью внешнего концептуального отображения, которое позволяет отображать пользовательское представление на соответствующую часть концептуальной схемы.

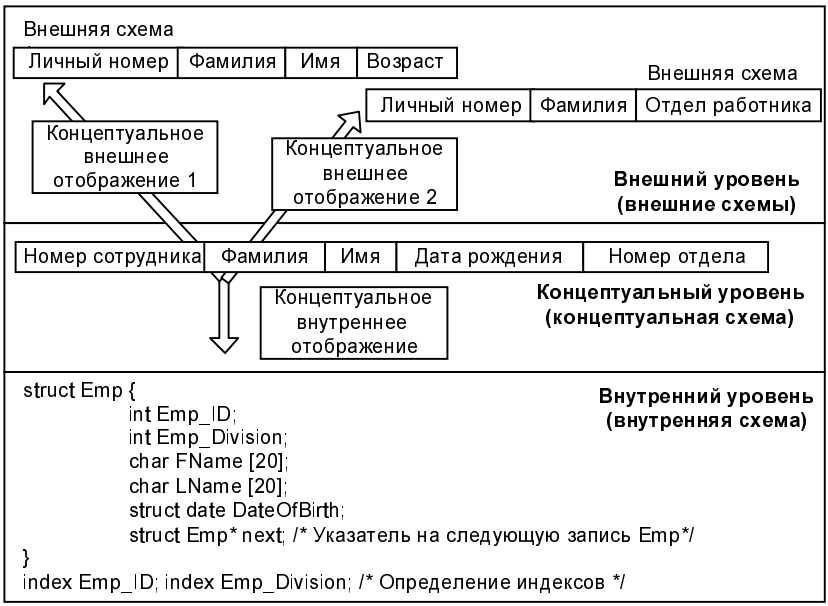

Рис. 1.3. Различия в схемах базы данных

Так (рис. 1.3), `Внешняя схема 1` предоставляет следующую информацию о сотрудниках учреждения: личный номер, фамилия, имя, возраст; `Внешняя схема 2`: личный номер, фамилия, отдел работника. Данные схемы получены на основе единой концептуальной схемы базы данных, причем поле `Возраст` получается путем вычисления на основе поля `Дата рождения`. В свою очередь, концептуальная схема преобразует-

ся в СУБД с помощью концептуального внутреннего отображения во внутреннюю схему, содержащую физическое описание структуры записи концептуального представления.

В теории и практике баз данных следует различать также описание базы данных и саму базу данных. Под *описанием базы данных* понимается схема базы данных, которая создается в процессе проектирования базы данных, изменение которой предполагается в исключительных случаях. Понятие *базы данных* предполагает всю информацию, содержащуюся в базе, которая может изменяться с течением времени. Совокупность информации, хранящейся в базе данных в любой определенный момент времени, называется *состоянием базы данных*. Таким образом, одной и той же схеме базы данных может соответствовать множество различных состояний базы данных.

Часто встречаются также следующие понятия для схемы и состояния базы данных. Схема базы данных называется *содержанием* базы данных, а ее состояние — *детализацией*.

Главное назначение трехуровневой архитектуры — обеспечение *независимости от данных*, т. е. любые изменения на нижних уровнях базы данных не должны влиять на верхние уровни. Независимость от данных бывает двух типов:

- логическая — полная защищенность внешних схем от изменений, которые вносятся в концептуальную схему;
- физическая — защищенность концептуальной схемы от изменений, которые вносятся во внутреннюю схему.

Архитектура типичной СУБД

На рис. 1.4 представлены главные компоненты архитектуры типичной СУБД. Рассмотрим назначение каждого компонента.

Компонент `Данные, Метаданные` включает не только данные, но также информацию о структуре данных *(метаданные)*. В реляционной СУБД метаданные включают в себя системные таблицы (отношения), имена отношений, имена атрибутов этих отношений и типы данных этих атрибутов. Как пра-

вило, СУБД поддерживает индексы данных. *Индекс* — это структура данных, которая помогает быстро найти элементы данных при наличии части их значения. Индексы представляют собой часть хранимых данных, а описания, указывающие, какие данные имеют индексы, — часть метаданных.

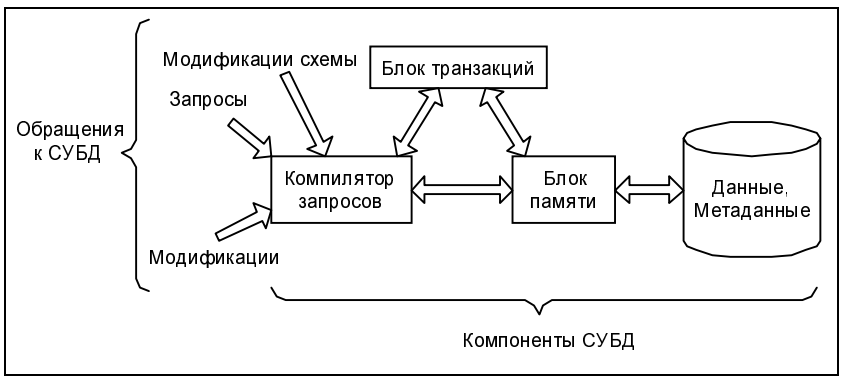

Рис. 1.4. Главные компоненты СУБД

Компонент `Блок памяти` получает требуемую информацию из места хранения данных и изменяет в нем соответствующую информацию по требованию расположенных выше уровней системы.

В простых системах баз данных блоком памяти может служить система файлов операционной системы. Однако для повышения эффективности СУБД обычно осуществляет прямой контроль памяти. Блок памяти состоит из двух компонентов.

- *Блок файлов* контролирует расположение файлов на диске и получает блок или блоки, содержащие файлы, по запросу блока буфера (диск в общем случае делится на *дисковые блоки* — смежные области памяти, содержащие от 4000 до 16 000 байт).
- *Блок буфера* управляет основной памятью. Он получает блоки данных с диска через блок файлов и выбирает стра-

ницу основной памяти для хранения конкретного блока. Блок буфера может временно сохранять дисковый блок в основной памяти, но возвращает его на диск, когда страница основной памяти нужна для другого блока. Страницы также возвращаются на диск по требованию блока транзакций.

Компонент `Компилятор запросов` — обрабатывает различные обращения к СУБД (запросы) и запрашивает изменения данных или метаданных. Он предлагает лучший способ выполнения необходимой операции и выдает соответствующие команды блоку памяти.

Компилятор запросов преобразует запрос или действие с базой данных, которые могут быть выполнены на очень высоком уровне (например, в виде запроса SQL), в последовательность более простых запросов. Часто самой трудной частью обработки запроса является его *организация*, т. е. выбор хорошего *плана запроса* или последовательности запросов к системе памяти, отвечающей на запрос.

Как правило, компилятором запросов обрабатываются три типа обращений к СУБД.

- *Запросы* — вопросы, касающиеся данных, находящихся в базе. Запросы могут генерироваться двумя способами:
  - с помощью общего интерфейса запросов (например, запросы, сформулированные на языке запросов высокого уровня — SQL, которые передаются компилятору запросов, он также получает ответы на них; как правило, реляционная СУБД поддерживает SQL-запросы);
  - с помощью интерфейсов прикладных программ (запросы передаются через специальный интерфейс, который предполагает генерацию только заданных запросов к базе, например, посредством полей ввода, полей со списком и т. д.; через данный интерфейс нельзя передавать произвольные запросы).
- *Модификации (модифицирующие запросы)* — операции по изменению данных (удаление, изменение, добавление). Они

также могут выполняться либо с помощью общего интерфейса, либо через интерфейс прикладной программы.

- *Модификации схемы базы данных* — это команды администраторов базы данных, которые имеют право изменять схему базы данных либо создавать новую базу данных.

Компонент `Блок транзакций` отвечает за целостность системы и должен обеспечить одновременную обработку многих запросов, отсутствие интерференции запросов (интерференция — сложение, в данном случае необходимо исключить наложение запросов и их взаимовлияние и т. д.) и защиту данных в случае выхода системы из строя. Блок транзакций взаимодействует с компилятором запросов, т. к. для разрешения конфликтных ситуаций должен учитывать, на какие данные воздействуют текущие запросы. В силу этого некоторые запросы могут быть отложены и установлена очередность их выполнения. Блок транзакций взаимодействует также с блоком памяти, т. к. схемы защиты данных обычно включают в себя хранение файла регистрации изменений данных. При правильном порядке выполнения операции файл *регистрации* содержит запись изменений, поэтому можно заново выполнить даже те изменения в базе данных, которые из-за сбоя в системе были прерваны и не внесены в физическое пространство диска.

Типичные СУБД позволяют пользователю выполнить несколько запросов и/или изменений в одной транзакции. Под *транзакцией* понимается совокупность действий (группа операций), которые необходимо выполнить последовательно, но которые будут восприниматься как единое целое.

Как правило, система базы данных поддерживает одновременно множество транзакций. Именно правильное выполнение всех таких транзакций и обеспечивает на схеме (рис. 1.4) `Блок транзакций`. Правильное выполнение транзакций обеспечивается ACID-свойствами (Atomicity, Consistency, Isolation, Durability):

- *атомарность* — требование выполнения либо всех транзакций, либо ни одной из них (например, изъятие денег со счета в банке данного клиента и внесение соответствую-

щих изменений должны быть единственной атомарной транзакцией, не допускающей выполнение каждой из этих операций по отдельности);

- *непротиворечивость* — состояние, при котором данные соответствуют всем возможным ожиданиям (например, условие непротиворечивости для базы данных бронирования авиационных билетов состоит в том, что ни одно из мест в самолете не бронируется для двух пассажиров);
- *изоляция* — предполагает, что при параллельном выполнении двух или более транзакций их результаты должны быть изолированы друг от друга. Одновременное выполнение двух транзакций не должно приводить к результату, которого не было, если бы они выполнялись последовательно (например, при продаже авиационных билетов на один и тот же рейс в случае последнего свободного места при одновременном запросе-обращении к базе двух служащих запрос одного должен быть выполнен, другого — нет);
- *долговременность* — предполагает, что после завершения транзакции результат не должен быть утрачен при сбоях в системе (даже если сбой происходит сразу после окончания транзакции).

Следует также отметить, что современное программное обеспечение поддерживает архитектуру *"клиент-сервер"*. Данная концепция предполагает, что один процесс (*клиент*) посылает запрос для выполнения другому процессу (*серверу*). Как правило, база данных часто разделяется на процесс сервера и несколько процессов клиента (рис. 1.5).

В простейшей архитектуре "клиент-сервер" вся СУБД является сервером, за исключением интерфейсов запроса, которые взаимодействуют с пользователем и посылают запросы или другие команды на сервер. Так, реляционная СУБД часто использует язык SQL для представления запросов от клиента к серверу. Затем сервер базы данных предоставляет клиенту ответ в виде соответствующей таблицы, в которой отображает-

ся интересующая информация. Существует тенденция увеличения нагрузки на клиентах, т. к. при наличии множества одновременно работающих пользователей базы данных могут возникнуть проблемы с сервером.

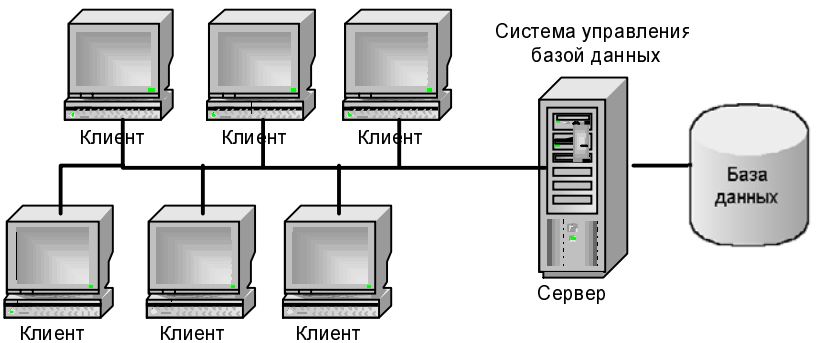

Рис. 1.5. Главные компоненты архитектуры клиент-сервер

В настоящее время СУБД получили широкое распространение в различных сферах деятельности и выполняют основную часть процедур, связанных с накоплением и обработкой информации. В связи с этим в табл. 1.1 приведем классификацию типов СУБД, зависящую от количества пользователей, местоположения базы данных и сферы использования.

Таблица 1.1. *Типы систем управления базами данных*

Способ классификации	Тип СУБД	Ключевые признаки	
По количеству пользователей	Однопользовательская	В конкретный момент времени с базой данных работает только один пользователь	
		Настольная база данных	На базе персонального компьютера

Таблица 1.1 *(окончание)*

Способ классификации	Тип СУБД	Ключевые признаки	
	Много-пользова-тельская	База данных рабочей группы	Число пользова-телей менее 50 человек
		База данных предприятия	Большое число пользователей (более 50 человек)
По месту разме-щения базы данных	Централи-зованная	СУБД работает с базой данных, размещенной на одном сервере	
	Распреде-ленная	СУБД работает с базой данных, размещенной на нескольких серверах	
По способу при-менения и сфере использования	Транзак-ционная (рабочая или опера-ционная)	СУБД работает с базой данных, в которой для транзакций отводит-ся минимальное время и результа-ты запросов к базе должны отобра-жаться в наикратчайшие сроки	
	База дан-ных под-держки ре-шений (хранилище данных)	СУБД работает с базой данных, предназначенной для получения необходимой информации при вы-работке стратегических или такти-ческих решений с целью оптимиза-ции деятельности предприятия	

Обзор направлений, лежащих в основе современных СУБД

Теория и практика баз данных в настоящее время развиваются в различных направлениях. Это, прежде всего, новые технологии, объектно-ориентированное программирование, ограничения и триггеры, мультимедийные данные, Интернет, новые приложения типа хранилища или интеграции данных.

Архитектура "клиент-сервер" составляет основу достаточно широкого класса современного программного обеспечения. В системах с данной архитектурой могут происходить два типа независимых процессов: процесс клиента и процесс сервера, которые могут выполняться как на одном компьютере, так и на различных компьютерах, подключенных к сети. В общей модели "клиент-сервер" предполагается четкое разделение процессов клиента и сервера, которые не зависят друг от друга. Серверы и клиенты могут находиться друг с другом в связи "многие-ко-многим": один сервер может предоставлять сервисы для многих клиентов, в свою очередь, каждый клиент может обращаться с запросами на многие серверы.

Объектно-ориентированное программирование предлагает для систем баз данных следующие возможности:

- оперирование данными с помощью развитой системы типов данных в более естественных формах, чем в классических моделях баз данных;
- с помощью классов и иерархии классов возможно более простое совместное либо повторное использование программного обеспечения;
- использование абстрактных типов данных предотвращает неправильное использование данных (в том случае, если разрешить доступ к ним только через точно спроектированные функции, использующие указанные данные правильно).

Ограничения и триггеры представляют собой активные элементы базы данных, которые выполняют указанные в них действия в подходящие моменты. Такие элементы особенно актуальны в различных коммерческих системах.

Ограничения — булевы функции, значения которых должны быть истинными. Например, в базу данных бухгалтерии вводится некоторое ограничение, запрещающее отрицательный баланс счета. Любое изменение в такой базе данных, нарушающее данное ограничение и приводящее к отрицательному балансу, отвергается СУБД.

Триггеры — части программы, реагирующие на определенное событие. Событиями могут быть различные действия: добав-

ление, удаление либо изменение элементов данных определенного типа. Если происходит данное событие, то выполняется заданная последовательность действий. Например, если изменяется должность работающего в организации, то происходит автоматический перерасчет базовых начислений на зарплату данного сотрудника.

С активными элементами в более ранних СУБД возникали технические трудности из-за их неэффективной реализации в случае большого объема данных, на которые они воздействуют. Однако в современных СУБД данные активные элементы применяются достаточно широко.

Поддержка *данных мультимедиа*, т. е. данных большого объема, способных изменяться в широких пределах. Это, например, такие типы данных, как аудио- и видеозаписи, сигналы, полученные от радаров, спутниковые изображения, графика различных форматов и т. д. Для хранения подобных данных создаются различные средства расширения СУБД. Операции, выполняемые с данными мультимедиа, не являются простыми и не подходят для традиционных форм данных. СУБД должна включать в себя возможность ввода пользователями по собственному выбору функций, применимых к данным мультимедиа. Часто в системах, использующих данные мультимедиа, применяются объектно-ориентированные методы.

Интеграция данных предполагает получение необходимой информации из различных баз и источников данных, а также ее обработку. Концепция *хранилищ данных*, в которые копируется информация из множества наследственных баз данных и соответствующим образом переводится в центральную базу данных, а также система оперативного анализа данных OLAP (On-Line Analytical Processing) используются в настоящее время в системах поддержки решений (DSS). Это несомненно важно для: планирования и анализа производства, оперативного складского учета, изучения информационных потребностей в торговой сфере, в сфере рекламного бизнеса различных уровней, в электронных каталогах распространения через Интернет и т. д.

Наиболее важные модели, технологии и системы управления базами данных приведены в табл. 1.2.