АНАТОЛИЙ ХОМОНЕНКО
ВЛАДИМИР ГОФМАН

РАБОТА С БАЗАМИ ДАННЫХ В DELPHI

3-Е ИЗДАНИЕ

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

ТЕХНОЛОГИИ BDE, ADO, dbExpress И INTERBASE EXPRESS

РАБОТА С ЛОКАЛЬНЫМИ И УДАЛЕННЫМИ БАЗАМИ ДАННЫХ

ПРОГРАММИРОВАНИЕ НА SQL

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Анатолий Хомоненко
Владимир Гофман

РАБОТА С БАЗАМИ ДАННЫХ В DELPHI

3-Е ИЗДАНИЕ

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.06
ББК 32.973.26-18.2
Х76

Хомоненко А. Д., Гофман В. Э.

Х76 Работа с базами данных в Delphi. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2005. — 640 с.: ил.

ISBN 5-94157-361-8

Рассматривается использование средств Delphi 7 для разработки приложений баз данных. Даются понятия баз данных, характеризуются элементы и описываются этапы проектирования реляционных баз данных, изложена технология разработки информационных систем, освещаются приемы работы с данными, создание таблиц и приложений баз данных, подготовка отчетов. Рассматриваются навигационный и реляционный способы доступа к данным с помощью технологий BDE, ADO, dbExpress и Interbase Express, основы программирования на SQL. Показывается использование локальных и удаленных баз данных, включая создание многоуровневых информационных систем и публикацию баз данных в Интернете. Также рассматриваются возможности Delphi 2005 по работе с базами данных. Благодаря подробному изложению тем и большому числу примеров книга может служить практическим руководством по работе с базами данных.

Для программистов

УДК 681.3.06
ББК 32.973.26-18.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.02.05.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 51,6.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

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 5-94157-361-8

Оглавление

Предисловие

Система визуального программирования Delphi обладает большой популярностью среди широкого круга пользователей: от неспециалистов до системных программистов, занимающихся разработкой сложных приложений и информационных систем.

Delphi позволяет быстро и удобно разрабатывать эффективные приложения, включая приложения для работы с базами данных. Система имеет развитые возможности по созданию пользовательского интерфейса, широкий набор функций, методов и свойств для решения прикладных расчетно-вычислительных задач. В системе имеются развитые средства отладки, облегчающие разработку приложений.

Традиционно Delphi относят к системам быстрой разработки приложений. Вместе с тем эта система обладает практически всеми возможностями современных СУБД, таких как Microsoft Access и Visual FoxPro. Она позволяет удобно создавать приложения с помощью инструментальных программных средств, визуально подготавливать запросы к базам данных, а также непосредственно писать SQL-запросы к базам данных.

Delphi позволяет создавать приложения для работы с локальными и удаленными базами данных, включая публикацию баз данных в Интернете. Применительно к работе с базами данных Delphi обеспечивает широкий набор инструментальных средств, поддерживает современные технологии, в том числе многоуровневую технологию "клиент-сервер".

Книга посвящена рассмотрению технологий разработки приложений для работы с базами данных и Интернетом в Delphi 7. В книге описываются основные компоненты, свойства, методы и события для работы с базами данных. Приводятся примеры реально работающих программ, которые читатель может использовать в своих разработках.

При написании книги мы и использовали версию Enterprise системы Delphi 7, как предоставляющую наибольшие возможности. Другие версии (Professional и Persona) могут отличаться, например, составом Хранилища объектов или значками объектов, в том числе форм.

По существу, книга представляет собой исправленное и существенно дополненное (по отношению ко 2-му изданию тех же авторов) произведение по работе с базами данных, которое доработано применительно к работе в среде Delphi 7 и снабжено материалами, характеризующими разработку приложений для баз данных в Delphi 2005.

Ее отличия от предыдущего издания в основном заключаются в освещении ряда новых средств и технологий, появившихся или получивших дальнейшее развитие в системе (технологий dbExpress и ADO работы с базами данных, технологии InterBase Express работы с сервером баз данных InterBase и др.), описании

техники работы с новым генератором отчетов Rave 5.0, знакомстве с основами построения и использования Web-служб, а также в уточнении состава визуальных компонентов системы и др.

Книга ориентирована на широкий круг читателей — от подготовленных пользователей до специалистов по программированию.

Ниже перечислены наиболее важные, на наш взгляд, новые средства и возможности системы Delphi 7 по работе с базами данных и Интернетом.

- Технологии работы с базами данных:
  - драйверы dbExpress обновлены для поддержки работы с Informix SE, Oracle 9i, DB2 7.2, InterBase 6.5 и MySQL 3.23.49; имеется новый драйвер для MS SQL 2000;
  - имеется несколько новых и измененных компонентов для работы с базами данных; вместо SQL Links для доступа к базам данных SQL Server рекомендуется использовать dbExpress;
  - можно задать специальный модуль данных SOAP в сервере приложений, имеющий многочисленные модули данных (свойство `SOAPServerIID` или добавление интерфейса модулей данных в конец URL); использовать компонент-соединение SOAP для вызова расширений интерфейса серверов приложения (свойство `SOAPServerIID` и метод `GetSOAPServer`).
- Технология CORBA: появилась новая версия комилятора IDL2PAS для разработки CORBA-приложений, рекомендуемая для использования вместо старых средств, интегрированных с поддержкой технологии COM.
- Web-технологии:
  - в Delphi входит IntraWeb из состава AtoZed Software; его можно использовать для разработки серверных Web-приложений с помощью стандартных средств формы, а также для разработки страниц приложений Web Broker и WebSnap;
  - Delphi поддерживает Apache 2 как целевой тип для Web Broker, WebSnap и SOAP (Simple Object Access Protocol); вместо интерфейса Win-CGI рекомендуется использование CGI, ISAPI/NSAPI или Apache в качестве целевого типа;
  - новый UDDI-обозреватель позволяет размещать и импортировать WSDL-документы, учитываемые в журналах регистрации UDDI;
  - новые классы и интерфейсы позволяют читать или вставлять заголовки в конверты SOAP, передающие сообщения между клиентами и серверами; клиентские и серверные приложения Web-служб получили возможность управлять прикрепленными сообщениями;
  - предоставляются большие возможности по управлению взаимодействием между вызываемым интерфейсом и документом WSDL на основе использования публикуемых событий;

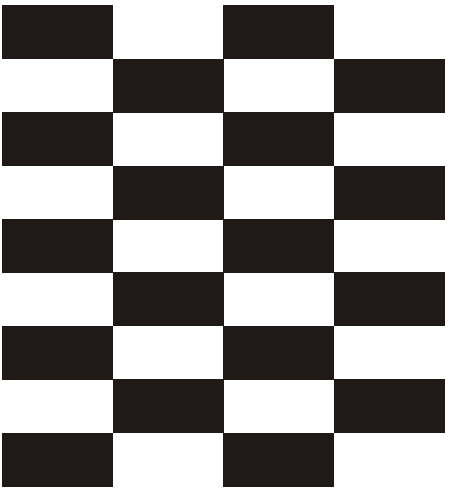
- новый интерфейс IRIOAccess позволяет осуществлять доступ к удаляемому объекту интерфейса, который реализует вызываемый интерфейс.

Для полноты картины приводим также новые средства и возможности Delphi 2005 по работе с базами данных.

- в части провайдеров доступа к данным BDP.NET для обеспечения новых свойств и улучшения их поддержки:
  - добавлен диалог для хранимых процедур;
  - теперь BDP.NET включает пространство имен для поддержки Sybase как провайдера;
  - многие обновления сделаны для улучшения надежности и производительности провайдеров;
  - обеспечивается возможность программно перемещать данные от одного провайдера к другому;
  - добавлены классы, позволяющие связывать приложение БД с многими провайдерами;
  - для `BdpDataAdapter` добавлена поддержка схемы таблицы и полей.
- применительно к VCL для поддержки .NET по работе с базами данных:
  - BDE для .NET обновлен для обеспечения динамической загрузки DLL без указания пути, для улучшения производительности управления объектами типа BLOB и для включения классов, ранее отсутствовавших в .NET;
  - обновления для технологии dbExpress;
  - для DataSnap импортирован ряд компонентов .NET, связанных с установлением соединений.
- в интересах упрощения проектирования и разработки BDP.NET-приложений для БД усилены возможности Проводника кода (Data Explorer):
  - поддерживается перенос данных от одного провайдера к другому;
  - хранимые процедуры можно перетаскивать (Drag-and-drop) от одного провайдера в Проводнике кода на форму в Конструкторе;
  - обеспечивается множество сервисов метаданных, что позволяет просматривать и модифицировать схему базы данных.

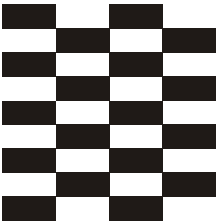
Авторы[1]

[1] Материалы 9—11, 15, 22 и 23 глав подготовлены А. Хомоненко; 18—20 глав — Е. Мещеряковым и А. Хомоненко при участии В. Гофмана. Остальные главы подготовлены В. Гофманом и А. Хомоненко совместно. Исправление и дополнение материалов книги применительно к версии Delphi 7/2005 выполнил А. Хомоненко.



ЧАСТЬ I

Основы работы с базами данных



Глава 1

Основные понятия баз данных

Часто для успешного функционирования различным организациям требуется развитая информационная система, реализующая автоматизированный процесс сбора, манипулирования и обработки данных.

Банки данных

Современной формой информационных систем являются *банки данных*, имеющие в своем составе:

- вычислительную систему;
- систему управления базами данных (СУБД);
- одну или несколько баз данных (БД);
- набор прикладных программ (приложений БД).

База данных (БД) обеспечивает хранение информации, а также удобный и быстрый доступ к данным. Она представляет собой совокупность данных различного характера, организованных по определенным правилам. Информация в БД должна быть:

- непротиворечивой;
- неизбыточной;
- целостной.

Система управления базой данных (СУБД) — это совокупность языковых и программных средств, предназначенных для создания, ведения и использования БД. По характеру применения СУБД разделяют на персональные и многопользовательские.

Персональные СУБД обеспечивают возможность создания локальных БД, работающих на одном компьютере. К персональным СУБД относятся Paradox, dBase, FoxPro, Access и др.

Замечание

СУБД Access 97/2000/2002/2003 также обеспечивают возможность многопользовательского доступа к данным.

Многопользовательские СУБД позволяют создавать информационные системы, функционирующие в архитектуре "клиент-сервер". Наиболее известными многопользовательскими СУБД являются Oracle, Informix, SyBase, Microsoft SQL Server, InterBase.

В состав языковых средств современных СУБД входят:

- язык описания данных, предназначенный для описания логической структуры данных;
- язык манипулирования данными, обеспечивающий выполнение основных операций над данными — ввод, модификацию и выборку;
- язык структурированных запросов (SQL, Structured Query Language), обеспечивающий управление структурой БД и манипулирование данными, а также являющийся стандартным средством доступа к удаленным БД;
- язык запросов по образцу (QBE, Query By Example), обеспечивающий визуальное конструирование запросов к БД.

Прикладные программы, или *приложения*, служат для обработки данных, содержащихся в БД. Пользователь осуществляет управление БД и работу с ее данными именно с помощью приложений, которые также называют *приложениями БД*.

Иногда термин "база данных" трактуют в более широком смысле и обозначают им не только саму БД, но и приложения, обрабатывающие ее данные.

Замечание

Хотя система Delphi и не является СУБД в буквальном смысле этого слова, тем не менее, она обладает вполне развитыми возможностями СУБД. Предоставляемые Delphi средства обеспечивают создание и ведение локальных и клиент-серверных БД, а также разработку приложений для работы практически с любыми БД. Назвать систему Delphi обычной СУБД мешает, наверное, только то, что, с одной стороны, у нее нет "своего" формата таблиц (языка описания данных), поэтому она использует форматы таблиц других СУБД, например, dBase, Paradox или InterBase (это вряд ли является недостатком, поскольку названные форматы хорошо себя зарекомендовали); с другой стороны, в плане создания приложений различного назначения, в том числе приложений БД, возможности Delphi не уступают возможностям специализированных СУБД, а зачастую и превосходят их.

Модели данных

База данных содержит данные, используемые какой-либо прикладной информационной системой (например, системами "Сирена" или "Экспресс" продажи авиа- и железнодорожных билетов).

В зависимости от вида организации данных различают следующие основные модели представления данных в базе:

- ❒ иерархическую;
- ❒ сетевую;
- ❒ реляционную;
- ❒ объектно-ориентированную.

В *иерархической* модели данные представляются в виде древовидной (иерархической) структуры. Подобная организация данных удобна для работы с иерархически упорядоченной информацией, однако при оперировании данными со сложными логическими связями иерархическая модель оказывается слишком громоздкой.

В *сетевой* модели данные организуются в виде произвольного графа. Недостатком сетевой модели является жесткость структуры и высокая сложность ее реализации.

Кроме того, значительным недостатком иерархической и сетевой моделей является то, что структура данных задается на этапе проектирования БД и не может быть изменена при организации доступа к данным.

В *объектно-ориентированной* модели отдельные записи базы данных представляются в виде объектов. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам объектно-ориентированных языков программирования. Объектно-ориентированные модели сочетают особенности сетевой и реляционной моделей и используются для создания крупных БД со сложными структурами данных.

Реляционная модель, предложенная в 70-х годах XX века сотрудником фирмы IBM Эдгаром Коддом, получила название от английского термина relation (отношение). Реляционная БД представляет собой совокупность таблиц, *связанных отношениями.* Достоинствами реляционной модели данных являются простота, гибкость структуры, удобство реализации на компьютере, наличие теоретического описания. Большинство современных БД для персональных компьютеров являются реляционными. При последующем изложении материала речь пойдет именно о реляционных БД.

Базы данных и приложения

В зависимости от взаимного расположения приложения и БД можно выделить:

- ❒ локальные БД;
- ❒ удаленные БД.

Для выполнения операций с локальными БД разрабатываются и используются так называемые *локальные приложения*, а для операций с удаленными БД — *клиент-серверные приложения.*

Расположение БД в значительной степени влияет на разработку приложения, обрабатывающего содержащиеся в этой базе данные.

Так, различают следующие виды приложений:

- приложения, использующие локальные базы данных, называют *одноуровневыми* (однозвенными) приложениями, поскольку приложение и базы данных образуют единую файловую систему;
- приложения, использующие удаленные базы данных, разделяют на двухуровневые (двухзвенные) и многоуровневые (многозвенные). *Двухуровневые* приложения содержат клиентскую и серверную части;
- *многоуровневые* (обычно трехуровневые) приложения кроме клиентской и серверной частей имеют дополнительные части. К примеру, в трехуровневых приложениях имеются клиентская часть, сервер приложений и сервер базы данных.

Одно- и двухуровневые приложения Delphi могут осуществлять доступ к локальным и удаленным БД с использованием следующих механизмов:

- BDE (Borland Database Engine — процессор баз данных фирмы Borland), предоставляющий развитый интерфейс API для взаимодействия с базами данных;
- ADO (ActiveX Data Objects — объекты данных ActiveX) осуществляет доступ к информации с помощью OLE DB (Object Linking and Embedding Data Base — связывание и внедрение объектов баз данных);
- dbExpress обеспечивает быстрый доступ к информации в базах данных с помощью набора драйверов;
- InterBase реализует непосредственный доступ к базам данных InterBase.

Выбор варианта технологии доступа к информации в базах данных, кроме прочих соображений, определяется с учетом удобства подготовки разработанного приложения к распространению, а также дополнительного расхода ресурсов памяти. К примеру, инсталляция для BDE требует примерно 15 Мбайт внешней памяти на диске и настройки псевдонимов используемых баз данных. Вариант InterBase вряд ли можно назвать конкурентоспособным, поскольку он ориентирован строго на работу с одноименным сервером баз данных.

Трехуровневые приложения Delphi 7 можно создавать с помощью механизма DataSnap. Используемые при создании трехуровневых (многоуровневых) приложений баз данных компоненты расположены на страницах **DataSnap** и **Data Access** Палитры компонентов.

BDE

BDE представляет собой совокупность динамических библиотек и драйверов, обеспечивающих доступ к данным. Процессор BDE должен устанавливаться на всех компьютерах, на которых выполняются Delphi-приложения, осуществляющие работу с БД. Приложение через BDE передает запрос к базе данных, а обратно получает требуемые данные. Механизм BDE до 7-й версии системы Del-

phi получил самое широкое распространение ввиду широкого спектра предоставляемых им возможностей. Идеологи фирмы Borland планируют отказаться от его поддержки, заменив его другими механизмами. Мы приводим множество примеров и описание технологии применения BDE для работы с базами данных в связи с тем, что накоплено большое количество приложений с использованием этого подхода.

ADO

Механизм ADO доступа к информации базы данных является стандартом фирмы Microsoft. Использование этой технологии подразумевает использование настраиваемых провайдеров данных. Технология ADO обеспечивает универсальный механизм доступа из приложений к информации источников данных. Эта технология основана на стандартных интерфейсах COM, являющихся системным механизмом Windows. Это позволяет удобно распространять приложения баз данных без вспомогательных библиотек.

dbExpress

Механизм доступа dbExpress подразумевает использование совокупности драйверов, компонентов, инкапсулирующих соединения, транзакций, запросов, наборов данных и интерфейсов, с помощью которых обеспечивается универсальный доступ к функциям этого механизма. Обеспечение взаимодействия с серверами баз данных по технологии dbExpress основано на использовании специализированных драйверов. Последние для получения данных применяют запросы SQL. На стороне клиента при этом нет кэширования данных, здесь применяются только однонаправленные курсоры и не обеспечивается возможность прямого редактирования наборов данных.

Варианты архитектуры для BDE

Здесь мы рассмотрим различные варианты архитектуры информационной системы на примере технологии BDE. Варианты архитектуры для других технологий доступа к данным рассмотрим позже при непосредственном их описании.

Локальные БД располагаются на том же компьютере, что и работающие с ними приложения. В этом случае говорят, что информационная система имеет локальную архитектуру (рис. 1.1). Работа с БД происходит, как правило, в *однопользовательском* режиме. При необходимости можно запустить на компьютере другое приложение, одновременно осуществляющее доступ к этим же данным. Для управления совместным доступом к БД необходимы специальные средства контроля и защиты. Эти средства могут понадобиться, например, в случае, когда приложение пытается изменить запись, которую редактирует другое приложение. Каждая разновидность БД осуществляет подобный контроль своими способами и обычно имеет встроенные средства разграничения доступа.

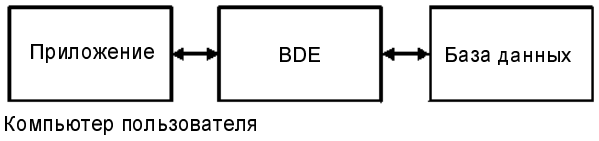


Рис. 1.1. Локальная архитектура

Для доступа к локальной БД процессор баз данных BDE использует стандартные драйверы, которые позволяют работать с форматами БД dBase, Paradox, FoxPro, а также с текстовыми файлами.

При использовании локальной БД в сети можно организовать многопользовательский доступ. В этом случае файлы БД и предназначенное для работы с ней приложение располагаются на сервере сети. Каждый пользователь запускает со своего компьютера это расположенное на сервере приложение, при этом у него запускается копия приложения. Такой сетевой вариант использования локальной БД соответствует архитектуре "файл-сервер". Приложение при использовании архитектуры "файл-сервер" также может быть записано на каждый компьютер сети, в этом случае приложению отдельного компьютера должно быть известно местонахождение общей БД (рис. 1.2).

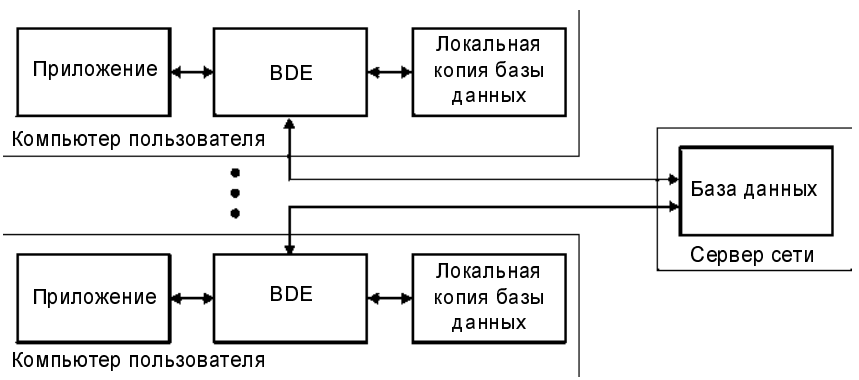


Рис. 1.2. Архитектура "файл-сервер"

При работе с данными на каждом пользовательском компьютере сети используется локальная копия БД. Эта копия периодически обновляется данными, содержащимися в БД на сервере.

Архитектура "файл-сервер" обычно применяется в сетях с небольшим количеством пользователей, для ее реализации подходят персональные СУБД, например, Paradox или dBase. Достоинствами этой архитектуры являются простота реализации, а также то, что приложение фактически разрабатывается в расчете на одного пользователя и не зависит от компьютера сети, на который оно устанавливается.

Однако архитектура "файл-сервер" имеет и существенные недостатки.

- Пользователь работает со своей локальной копией БД, данные в которой обновляются при каждом запросе к какой-либо из таблиц. При этом с сервера пересылается новая копия всей таблицы, данные из которой затребованы. Таким образом, если пользователю необходимо несколько записей таблицы, с сервера по сети пересылается вся таблица. В результате циркуляции в сети больших объемов избыточной информации *резко возрастает нагрузка на сеть*, что приводит к соответствующему снижению ее быстродействия и производительности информационной системы в целом.
- В связи с тем, что на каждом компьютере имеется своя копия БД, изменения, сделанные в ней одним пользователем, в течение некоторого времени являются неизвестными другим пользователям. Поэтому требуется постоянное обновление БД. Кроме того, возникает *необходимость синхронизации* работы отдельных пользователей, связанная с блокировкой в таблицах записей, которые в данный момент редактирует другой пользователь.
- Управление БД осуществляется с разных компьютеров, поэтому в значительной степени затруднена *организация управления доступом*, соблюдения *конфиденциальности* и поддержания *целостности* БД.

Удаленная БД размещается на компьютере-сервере сети, а приложение, осуществляющее работу с этой БД, находится на компьютере пользователя. В этом случае мы имеем дело с архитектурой "клиент-сервер" (рис. 1.3), когда информационная система делится на неоднородные части — сервер и клиент БД. В связи с тем, что компьютер-сервер отделен от клиента, его также называют *удаленным сервером.*

Клиент — это приложение пользователя. Для получения данных клиент формирует и отсылает запрос удаленному серверу, на котором помещена БД. Запрос формулируется на языке SQL, который является стандартным средством доступа к серверу при использовании реляционных моделей данных. После получения запроса удаленный сервер направляет его программе SQL Server (серверу баз данных) — специальной программе, управляющей удаленной БД и обеспечивающей выполнение запроса и выдачу его результатов клиенту.

Таким образом, в архитектуре "клиент-сервер" клиент посылает запрос на предоставление данных и получает только те данные, которые действительно были затребованы. Вся обработка запроса выполняется на удаленном сервере. Такая архитектура обладает следующими достоинствами.

- Снижение нагрузки на сеть, поскольку теперь в ней циркулирует только нужная информация.
- Повышение безопасности информации, связанное с тем, что обработка запросов всех клиентов выполняется единой программой, расположенной на сервере. Сервер устанавливает общие для всех пользователей правила использования БД, управляет режимами доступа клиентов к данным, запрещая, в частности, одновременное изменение одной записи различными пользователями.

- Уменьшение сложности клиентских приложений за счет отсутствия в них кода, связанного с контролем БД и разграничением доступа к ней.

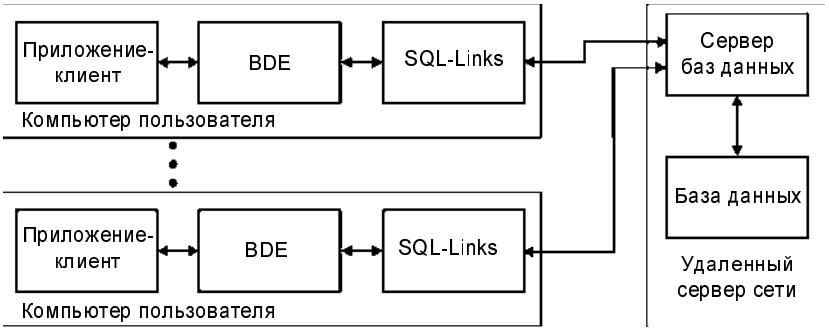


Рис. 1.3. Архитектура "клиент-сервер" ("толстый" клиент)

Для реализации архитектуры "клиент-сервер" обычно используются многопользовательские СУБД, например, Oracle или Microsoft SQL Server. Подобные СУБД также называют *промышленными*, т. к. они позволяют создать информационную систему организации или предприятия с большим числом пользователей. Промышленные СУБД являются сложными системами и требуют мощной вычислительной техники и соответствующего обслуживания. Обслуживание выполняет специалист (или группа специалистов), называемый *системным администратором БД* (администратором). Основные задачи системного администратора:

- защита БД;
- поддержание целостности БД;
- обучение и подготовка пользователей;
- загрузка данных из других БД;
- тестирование данных;
- резервное копирование и восстановление;
- внесение изменений в информационную систему.

Доступ Delphi-приложения к промышленным СУБД осуществляется через драйверы SQL-Links. Отметим, что при работе с "родной" для Delphi СУБД InterBase можно обойтись без драйверов SQL-Links.

Описанная архитектура является двухуровневой (уровень приложения-клиента и уровень сервера БД). Клиентское приложение также называют *сильным*, или "толстым", клиентом. Дальнейшее развитие данной архитектуры привело к появлению трехуровневого варианта архитектуры "клиент-сервер" (приложение-клиент, сервер приложений и сервер БД) (рис. 1.4).

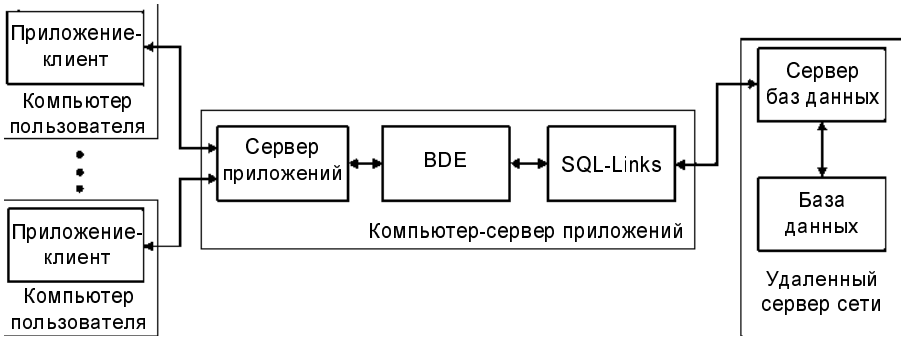


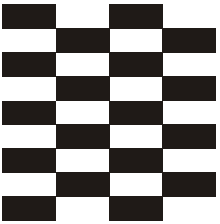
Рис. 1.4. Трехуровневая архитектура "клиент-сервер" ("тонкий" клиент)

В трехуровневой архитектуре часть средств и кода, предназначенных для организации доступа к данным и их обработке, из приложения-клиента выделяется в сервер приложений. Само клиентское приложение при этом называют *слабым*, или "тонким", клиентом. В сервере приложений удобно располагать средства и код, общие для всех клиентских приложений, например, средства доступа к БД.

Основные достоинства трехуровневой архитектуры "клиент-сервер" состоят в следующем:

- разгрузка сервера от выполнения части операций, перенесенных на сервер приложений;
- уменьшение размера клиентских приложений за счет разгрузки их от лишнего кода;
- единое поведение всех клиентов;
- упрощение настройки клиентов — при изменении общего кода сервера приложений автоматически изменяется поведение приложений-клиентов.

Напомним, что локальные приложения БД называют *одноуровневыми*, а клиент-серверные приложения БД — *многоуровневыми*.



Глава 2

Реляционные базы данных и средства работы с ними

Большинство современных баз данных для персональных компьютеров являются реляционными. Достоинства реляционной модели организации данных — простота, гибкость структуры, удобство реализации на компьютере, наличие теоретического описания.

Элементы реляционной базы данных

Реляционная база данных (БД) состоит из взаимосвязанных таблиц. Каждая таблица содержит информацию об объектах одного типа, а совокупность всех таблиц образует единую БД.

Таблицы баз данных

Таблицы, образующие БД, находятся в каталоге (папке) на жестком диске. Таблицы хранятся в файлах и похожи на отдельные документы или электронные таблицы (например, табличного процессора Microsoft Excel), их можно перемещать и копировать обычным способом, скажем, с помощью Проводника Windows. Однако в отличие от документов, таблицы БД поддерживают *многопользовательский* режим доступа, это означает, что их могут одновременно использовать несколько приложений.

Для одной таблицы создается несколько файлов, содержащих данные, индексы, ключи и т. п. Главным из них является файл с данными, имя этого файла совпадает с именем таблицы, которое задается при ее создании. В некотором смысле понятия таблицы и ее главного файла являются синонимами, при выборе таблицы выбирается именно ее главный файл: для таблицы dBase это файл с расширением dbf, а для таблицы Paradox — файл с расширением db. Имена остальных файлов таблицы назначаются автоматически — все файлы имеют одинаковые имена, совпадающие с именами таблиц, и разные расширения, указывающие на содержимое соответствующего файла. Расширения файлов приведены ниже в данной главе *в разд. "Таблицы форматов dBase и Paradox"*.

Каждая таблица БД состоит из строк и столбцов и предназначена для хранения данных об однотипных объектах информационной системы. Строка таблицы называется *записью*, столбец таблицы — *полем* (рис. 2.1). Каждое поле должно иметь уникальное в пределах таблицы имя.

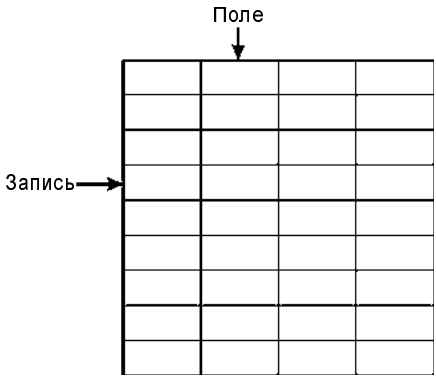


Рис. 2.1. Схема таблицы базы данных

Поле содержит данные одного из допустимых типов, например, строкового, целочисленного или типа "дата". При вводе значения в поле таблицы автоматически производится проверка соответствия типа значения и типа поля. В случае, когда эти типы не совпадают, а преобразование типа значения невозможно, генерируется исключение.

Особенности организации таблиц зависят от конкретной СУБД, используемой для создания и ведения БД. Например, в локальной таблице dBase и в таблице сервера InterBase нет поля автоинкрементного типа (с автоматически наращиваемым значением), а в таблице dBase нельзя определить ключ. Подобные особенности необходимо учитывать при выборе типа (формата) таблицы, т. к. они влияют не только на организацию БД, но и на построение приложения для работы с этой БД. Однако, несмотря на все различия таблиц, существуют общие правила создания и ведения БД, а также разработки приложений, которые и будут далее рассмотрены.

Замечание

В зависимости от типа таблиц и системы разработки приложений может использоваться различная терминология. Например, в InterBase поле таблицы называется столбцом.

Основу таблицы составляет описание ее полей, каждая таблица должна иметь хотя бы одно поле. Понятие структуры таблицы является более широким и включает:

- описание полей;
- ключ;

- индексы;
- ограничения на значения полей;
- ограничения ссылочной целостности между таблицами;
- пароли.

Иногда ограничения на значения полей, ограничения ссылочной целостности между таблицами, а также права доступа называют одним общим термином "ограничения".

Отметим, что отдельные элементы структуры зависят от формата таблиц, например, для таблиц dBase нельзя задать ограничения ссылочной целостности (т. к. у них нет ключей). Все элементы структуры задаются на физическом уровне (уровне таблицы) и действуют для всех программ, выполняющих операции с БД, включая средства разработки и ведения БД (например, программу Database Desktop). Многие из этих элементов (например, ограничения на значения полей или поля просмотра) можно также реализовать в приложении программно, однако в этом случае они действуют только в пределах своего приложения.

С таблицей в целом можно выполнять следующие операции:

- создание (определение структуры);
- изменение структуры (реструктуризация);
- переименование;
- удаление.

При *создании* таблицы задаются структура и имя таблицы. При сохранении на диске создаются все необходимые файлы, относящиеся к таблице. Их имена совпадают с именем таблицы.

При *изменении структуры* таблицы в ней могут измениться имена и характеристики полей, состав и наименования ключа и индексов, ограничения. Однако имена таблицы и ее файлов остаются прежними.

При *переименовании* таблица получает новое имя, в результате чего новое имя также получают все ее файлы. Для этого используются соответствующие программы (утилиты), предназначенные для работы с таблицами БД, например, Database Desktop или Data Pump.

Замечание

Таблицу нельзя переименовать, просто изменив названия всех ее файлов, например, с помощью Проводника Windows.

При *удалении* таблицы с диска удаляются все ее файлы. В отличие от переименования, удаление таблицы можно выполнить посредством любой программы (в том числе и с помощью Проводника Windows).

Ключи и индексы

Ключ представляет собой комбинацию полей, данные в которых однозначно определяют каждую запись в таблице. Простой ключ состоит из одного поля, а составной (сложный) — из нескольких полей. Поля, по которым построен ключ, называют *ключевыми*. В таблице может быть определен только один ключ. Ключ обеспечивает:

- однозначную идентификацию записей таблицы;
- ускорение выполнения запросов к БД;
- установление связи между отдельными таблицами БД;
- использование ограничений ссылочной целостности.

Ключ также называют *первичным ключом* или *первичным (главным) индексом*.

Информация о ключе может храниться в отдельном файле или совместно с данными таблицы. Например, в БД Paradox для этой цели используется отдельный файл (ключевой файл или файл главного индекса) с расширением px. В БД Access вся информация содержится в одном общем файле с расширением mdb. Значения ключа располагаются в определенном порядке. Для каждого значения ключа имеется уникальная ссылка, указывающая на расположение соответствующей записи в таблице (в главном ее файле). Поэтому при поиске записи выполняется не последовательный просмотр всей таблицы, а прямой доступ к записи на основании упорядоченных значений ключа.

Ценой, которую разработчик и пользователь платят за использование такой технологии, является увеличение размера БД вследствие необходимости хранения значений ключа, например, в отдельном файле. Размер этого файла зависит не только от числа записей таблицы (что достаточно очевидно), но и от полей, составляющих ключ. В ключевом файле, кроме ссылок на соответствующие записи таблицы, сохраняются и значения самих ключевых полей. Поэтому при вхождении в состав ключа длинных строковых полей размер ключевого файла может оказаться соизмеримым с размером файла с данными таблицы.

Таблицы различных форматов имеют свои особенности построения ключей. Вместе с тем существуют и общие правила.

- Ключ должен быть уникальным. У составного ключа значения отдельных полей (но не всех одновременно) могут повторяться.
- Ключ должен быть достаточным и не избыточным, т. е. не содержать поля, которые можно удалить без нарушения уникальности ключа.
- В состав ключа не могут входить поля некоторых типов, например, графическое поле или поле комментария.

Выбор ключевых полей не всегда является простой и очевидной задачей, особенно для таблиц с большим количеством полей. Нежелательно выбирать в качестве ключевых поля, содержащие фамилии людей в таблице сотрудников организации или названия товаров в таблице данных склада. В этом случае высока

вероятность существования двух и более однофамильцев, а также товаров с одинаковыми названиями, которые различаются, к примеру, цветом (значение другого поля). Для указанных таблиц можно использовать, например, поле кода сотрудника и поле артикула товара. При этом предполагается, что указанные значения являются уникальными.

Удобным вариантом создания ключа будет использование для него поля соответствующего типа, которое автоматически обеспечивает поддержку уникальности значений. Для таблиц Paradox таким является поле автоинкрементного типа, еще одним достоинством которого является небольшой размер (4 байта). В то же время в таблицах dBase и InterBase поле подобного типа отсутствует, и программист должен обеспечивать уникальность значений ключа самостоятельно, например, используя специальные генераторы.

Отметим, что при создании и ведении БД правильным подходом считается задание в каждой таблице ключа даже в случае, если на первый взгляд он не нужен. В примерах таблиц, которые приводятся при изложении материала, как правило, ключ создается, и для него вводится специальное автоинкрементное поле с именем `Code` или `Number`.

Индекс, как и ключ, строится по полям таблицы, однако он может допускать повторение значений составляющих его полей — в этом и состоит его основное отличие от ключа. Поля, по которым построен индекс, называют *индексными*. Простой индекс состоит из одного поля, а составной (сложный) — из нескольких полей.

Индексы при их создании именуются. Как и в случае с ключом, в зависимости от СУБД индексы могут храниться в отдельных файлах или совместно с данными. Создание индекса называют *индексированием таблицы*.

Использование индекса обеспечивает:

- ❒ увеличение скорости доступа к данным (поиска);
- ❒ сортировку записей;
- ❒ установление связи между отдельными таблицами БД;
- ❒ использование ограничений ссылочной целостности.

В двух последних случаях индекс применяется совместно с ключом второй таблицы.

Как и ключ, индекс представляет собой своеобразное оглавление таблицы, просмотр которого выполняется перед обращением к ее записям. Таким образом, использование индекса повышает *скорость доступа* к данным в таблице за счет того, что доступ выполняется не последовательным, а индексно-последовательным методом.

Сортировка представляет собой упорядочивание записей по полю или группе полей в порядке возрастания или убывания их значений. Можно сказать, что индекс служит для сортировки таблиц по индексным полям. В частности, в Delphi записи набора `Table` можно сортировать только по индексным полям.