А. ХОМОНЕНКО, С. АДАДУРОВ

bhv®

РАБОТА С БАЗАМИ ДАННЫХ В C++ BUILDER

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
Сергей Ададуров

РАБОТА С БАЗАМИ ДАННЫХ В C++ BUILDER

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.1
Х76

Хомоненко А. Д., Ададуров С. Е.

Х76 Работа с базами данных в C++ Builder. — СПб.: БХВ-Петербург, 2006. — 496 с.: ил.

ISBN 5-94157-690-0

Рассматривается использование средств C++ Builder для разработки приложений баз данных. Даются понятия баз данных, характеризуются элементы и описываются этапы проектирования реляционных баз данных, изложена технология разработки информационных систем. Показаны основные приемы работы с данными при создании таблиц, подготовке SQL-запросов, использовании триггеров и хранимых процедур. Подробно описаны основные визуальные компоненты для разработки приложений, а также инструменты для администрирования локальных и удаленных данных. Рассматриваются навигационный и реляционный способы доступа к данным с помощью BDE, ADO, dbExpress и Interbase Express, основы программирования на SQL. Показывается использование локальных и удаленных баз данных, включая создание многоуровневых информационных систем. Благодаря подробному изложению тем и большому числу примеров книга может служить практическим руководством по работе с базами данных.

Для разработчиков БД

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Татьяна Лапина*
Компьютерная верстка	*Екатерины Трубниковой*
Корректор	*Татьяна Кошелева*
Дизайн серии	*Инна Тачина*
Дизайн обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.11.05.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 39,99.
Тираж 3000 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 5-94157-690-0

Содержание

Предисловие ... **1**

ЧАСТЬ I. ОСНОВЫ РАБОТЫ С БАЗАМИ ДАННЫХ ... **3**

Глава 1. Основные понятия баз данных ... **5**

Банки данных ... 5
Модели данных ... 6
Базы данных и приложения ... 7
Механизмы доступа приложений ... 8
BDE ... 9
ADO ... 9
dbExpress ... 9
InterBase Express ... 9
Варианты архитектуры для BDE ... 10

Глава 2. Реляционные базы данных и средства работы с ними ... **15**

Элементы реляционной базы данных ... 15
Таблицы баз данных ... 15
Ключи и индексы ... 18
Методы и способы доступа к данным ... 21
Связи между таблицами ... 23
Механизм транзакций ... 27
Бизнес-правила ... 28
Словарь данных ... 29
Таблицы форматов dBase и Paradox ... 29
Средства для работы с базами данных ... 34
Инструменты ... 35
Компоненты ... 36
Исключения баз данных ... 41

Глава 3. Проектирование баз данных ... **45**

Проблемы и подходы к проектированию ... 45
Функциональные зависимости атрибутов ... 49
Нормализация баз данных ... 50
Средства CASE ... 54

Глава 4. Технология создания информационной системы ... **57**
Варианты создания таблиц ... 57
Создание таблицы с помощью Database Desktop ... 59
Описание полей ... 62
Задание индексов ... 64
Задание ограничений на значения полей ... 66
Задание ссылочной целостности ... 69
Задание паролей ... 71
Задание языкового драйвера ... 74
Задание таблицы для выбора значений ... 74
Просмотр списка подчиненных таблиц ... 78
Изменение структуры таблицы ... 78
Создание приложения BDE ... 79
Использование модуля данных ... 82

Глава 5. Компоненты доступа к данным с помощью BDE ... **87**
Наборы данных ... 87
Состояния наборов данных ... 90
Режимы наборов данных ... 95
Доступ к полям ... 98
Особенности набора данных *Table* ... 100
Особенности набора данных *Query* ... 109
Объекты поля ... 115
Редактор полей ... 117
Операции с полями ... 126
Доступ к значению поля ... 127
Проверка типа и значения поля ... 131
Форматирование отображаемого значения поля ... 137
Источник данных ... 140

ЧАСТЬ II. ТЕХНОЛОГИИ ДОСТУПА К ДАННЫМ ... **143**

Глава 6. Визуальные компоненты для работы с данными ... **145**
Отображение и редактирование значения логического поля ... 147
Отображение и выбор значения поля ... 148
Отображение и выбор значения поля в списке ... 150
Простой и комбинированный списки ... 151
Списки, сформированные по значениям поля набора данных ... 151
Представление записей в табличном виде с помощью сетки ... 152
Характеристики сетки ... 152
Столбцы сетки ... 156
Компонент "модифицированная сетка" ... 161
Использование навигационного интерфейса ... 165
Компонент "графическое изображение" ... 167
Построение диаграмм ... 171

Глава 7. Навигационный доступ к данным ... **179**
Операции с таблицей БД ... 180
Создание, удаление и переименование ... 180
Установка уровня доступа ... 181

Сортировка набора данных ... 183
Навигация по набору данных ... 185
Перемещение по записям ... 185
Переход по закладкам ... 188
Фильтрация записей ... 192
Фильтрация по выражению ... 192
Фильтрация по диапазону ... 197
Навигация с псевдофильтрацией ... 200
Поиск записей ... 200
Поиск в наборах данных ... 201
Поиск по индексным полям ... 204
Модификация набора данных ... 206
Редактирование записей ... 208
Добавление записей ... 212
Удаление записей ... 214
Пример формы приложения ... 215
Связывание таблиц ... 223

Глава 8. Доступ к данным с помощью запросов ... 227

Основные сведения о языке SQL ... 228
Функции языка ... 229
Определение данных ... 230
Создание и удаление таблицы ... 230
Изменение состава полей таблицы ... 233
Создание и удаление индекса ... 234
Отбор данных из таблиц ... 235
Описание оператора *SELECT* ... 235
Управление полями ... 236
Простое условие отбора записей ... 240
Сложные критерии отбора записей ... 244
Группирование записей ... 245
Сортировка записей ... 246
Соединение таблиц ... 248
Модификация записей ... 250
Редактирование записей ... 251
Вставка записей ... 252
Удаление записей ... 253
Статический и динамический запросы ... 254
Запросы с параметрами ... 255

Глава 9. Технология dbExpress ... 259

Общая характеристика ... 259
Установление соединения с сервером ... 260
Компоненты доступа к данным ... 266
Универсальный доступ к данным ... 266
Просмотр таблиц ... 273
Выполнение SQL-запроса ... 274
Выполнение хранимых процедур ... 276
Компонент редактирования набора данных ... 277
Отладка соединения с сервером ... 281

Глава 10. Технология ADO ... **283**
Общая характеристика ... 283
Установление соединения ... 285
Управление соединением и транзакциями ... 289
Компоненты доступа к данным ... 291
Доступ к таблицам ... 294
Выполнение запросов ... 294
Вызов хранимых процедур ... 295
Компонент ADODataSet ... 295
Команды ADO ... 296
Пример приложения ... 298

Глава 11. Создание и просмотр отчетов с помощью QuickReport ... **303**
Компоненты отчета ... 303
Компонент-отчет ... 303
Полоса отчета ... 311
Компоненты, размещаемые в полосе ... 313
Простой отчет ... 316
Заголовок отчета ... 317
Заголовки столбцов и данные ... 317
Итоговая полоса ... 319
Колонтитулы ... 319

Глава 12. Инструменты ... **321**
Программа BDE Administrator ... 321
Работа с псевдонимами ... 323
Параметры драйвера ... 325
Системные установки ... 329
Использование конфигурационных файлов ... 331
Программа Database Desktop ... 331
Редактирование записей таблиц ... 333
Работа с псевдонимами ... 333
Работа с SQL-запросами ... 334
Визуальное конструирование запросов ... 335
Отбор записей из таблицы ... 336
Связывание таблиц ... 337

ЧАСТЬ III. УДАЛЕННЫЕ БАЗЫ ДАННЫХ ... **339**

Глава 13. Введение в работу с удаленными базами данных ... **341**
Основные понятия ... 341
Архитектура "клиент-сервер" ... 342
Сервер и удаленная БД ... 343
Средства работы с удаленными БД ... 343
Сервер InterBase ... 345
Бизнес-правила ... 346
Организация данных ... 347
Запуск сервера ... 349
Особенности приложения ... 350
Соединение с базой данных ... 350

Глава 14. Работа с удаленными базами данных InterBase ... **353**

Создание базы данных ... 353
Управление структурой таблиц ... 357
Описание столбца ... 359
Ограничения столбца ... 360
Описание ключей ... 363
Определение ограничений ссылочной целостности ... 365
Использование индексов ... 366
Домены ... 368
Представления ... 369
Хранимые процедуры ... 370
Создание и изменение ... 371
Виды хранимых процедур ... 372
Язык хранимых процедур ... 372
Триггеры ... 379
Создание и изменение ... 379
Примеры использования ... 380
Создание генераторов ... 383
Механизм событий сервера ... 384
Управление привилегиями ... 385
Манипулирование данными ... 387

Глава 15. Доступ к удаленным БД с помощью BDE ... **391**

Управление соединениями с базой данных ... 391
Компонент *Database* ... 391
Компонент *Session* ... 395
Соединение с базой данных ... 399
Вызов хранимых процедур ... 400
Вызов процедуры выбора ... 400
Вызов исполняемой процедуры ... 402
Механизм транзакций ... 404
Механизм кэшированных изменений ... 407
Компонент *UpdateSQL* ... 408
Компоненты *Database* и *Query* ... 412

Глава 16. Технология InterBase Express ... **417**

Общая характеристика ... 417
Установление соединения с сервером ... 418
Управление транзакциями ... 420
Компоненты доступа к данным ... 422
Генераторы для автоинкрементных полей ... 423
Доступ к таблицам ... 424
Выполнение запросов ... 424
Получение и редактирование данных ... 425
Компонент *IBSQL* ... 430
Пример приложения ... 430

Глава 17. Инструменты для работы с удаленными базами данных ... **435**

Программа IBConsole ... 435
Управление сервером ... 436

Подключение к серверу ... 436
Регистрация сервера ... 438
Просмотр протокола работы сервера ... 439
Операции с сертификатами ... 439
Управление пользователями ... 440
Управление БД ... 440
Регистрация базы данных ... 441
Подключение базы данных ... 442
Создание базы данных ... 442
Просмотр метаданных ... 442
Сбор мусора ... 443
Проверка состояния базы данных ... 443
Анализ статистики ... 444
Сохранение и восстановление базы данных ... 445
Интерактивное выполнение SQL-запросов ... 449
Программа SQL Monitor ... 452

Глава 18. Трехуровневые приложения ... 455

Принципы построения трехуровневых приложений ... 455
Сервер приложений ... 457
Приложение клиента ... 463

Предметный указатель ... 475

Предисловие

Система программирования Borland C++ Builder 6 завоевала достаточно прочные позиции среди профессиональных и начинающих программистов. Здесь можно отметить ряд причин: большую популярность языка программирования C++, удобство визуального конструирования приложений, развитые возможности доступных средств системы, эффективность генерируемого кода и др.

Несмотря на появление современных технологий типа .NET и соответствующих систем программирования, таких как Visual C++ .NET, система C++ Builder будет устойчиво занимать свою нишу. Это обусловлено меньшей требовательностью к аппаратным ресурсам при разработке приложений, большей легкостью в освоении и применении средств системы для разработки приложений различной степени сложности.

В книге рассматриваются технологии применения C++ Builder 6 для разработки приложений баз данных. C++ Builder 6 не является системой управления базами данных (СУБД), строго ориентированной на разработку приложений для работы с ними. Тем не менее ее возможности практически ни в чем не уступают возможностям таких СУБД, как Visual FoxPro или Access. Она позволяет создавать приложения с помощью инструментальных программных средств, визуально подготавливать, а также непосредственно писать SQL-запросы к базам данных. С ее помощью можно создавать приложения для работы с локальными и удаленными базами данных.

Для полноты представления работы с информационными системами в книге затрагиваются вопросы проектирования реляционных баз данных, для работы с которыми и используется система C++ Builder 6. Дается описание языка структурированных запросов SQL.

Описываются основные компоненты, свойства, методы и события для работы с локальными и распределенными базами данных. По традиции подробное описание и примеры приводятся для компонентов, используемых для доступа к данным с помощью механизма BDE и других механизмов (dbExpress, ADO

и InterBase Express). Приводятся отличительные особенности и даются примеры применения.

Дается описание технологии применения различных инструментальных средств, используемых для создания приложений по работе с локальными и удаленными базами данных. В частности, рассматриваются BDE Administrator, IBConsole и SQL Monitor. Приводятся примеры реально работающих программ, которые читатель может использовать в своих разработках.

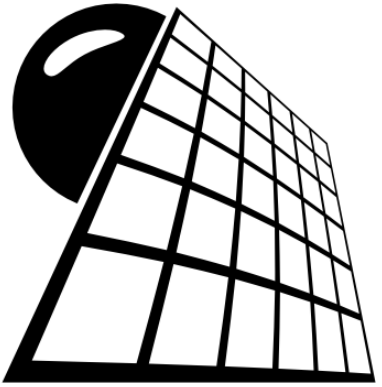
При написании книги использовалась версия Enterprise системы C++ Builder 6, как предоставляющая наибольшие возможности. В книге получили освещение сравнительно новые средства и технологии, появившиеся или получившие дальнейшее развитие в системе C++ Builder 6 (технологии dbExpress и ADO работы с базами данных, технологии InterBase Express работы с сервером баз данных InterBase и др.).

Несмотря на относительную простоту построения приложений в среде C++ Builder, имеются определенные трудности в правильном использовании свойств и методов компонентов системы. Приведенные в справочной помощи системы примеры, как выясняется, не всегда верны. Все это свидетельствует о необходимости создания методически отработанных материалов, где в структурированном виде с наглядными примерами дается описание технологии построения приложений по работе с базами данных. Именно этого и хотелось достичь при подготовке материалов книги. Насколько это удалось, судить читателю.

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

Выражаем признательность В. Э. Гофману, совместная работа с которым над книгами по Delphi во многом способствовала появлению этой книги.

Авторы



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

Система C++ Builder не является СУБД в буквальном смысле этого слова, тем не менее, она обладает вполне развитыми возможностями СУБД. Предоставляемые C++ Builder средства обеспечивают создание и ведение локальных и клиент-серверных БД, а также разработку приложений для работы практически с любыми БД. Назвать систему C++ Builder обычной СУБД мешает, наверное, только то, что 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 C++ Builder 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.

Механизмы доступа приложений

Одно- и двухуровневые приложения C++ Builder 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 Express реализует непосредственный доступ к базам данных сервера 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.

Трехуровневые приложения C++ Builder 6 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 приложения C++ Builder, осуществляющие работу с БД. Приложение через BDE передает запрос к базе данных, а обратно получает требуемые данные. Механизм BDE до 6-й версии системы C++ Builder получил самое широкое распространение ввиду широкого спектра предоставляемых им возможностей. Идеологи фирмы Borland планируют более широкое применение других механизмов как более эффективных. 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.

InterBase Express

Механизм доступа InterBase Express ориентирован строго на работу с сервером InterBase и основан на прямом применении функций API этого сервера. Отсюда следуют все достоинства и недостатки использования этого механизма доступа. Он обеспечивает высокую скорость работы компонентов

доступа к данным. Очевидным недостатком механизма доступа InterBase является невозможность применения его для серверов баз данных, отличных от сервера InterBase SQL Server.

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.

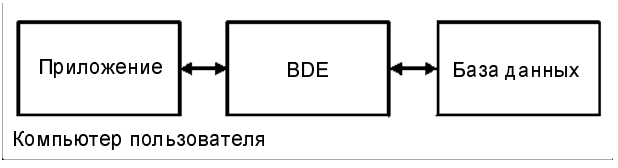


Рис. 1.1. Локальная архитектура

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).

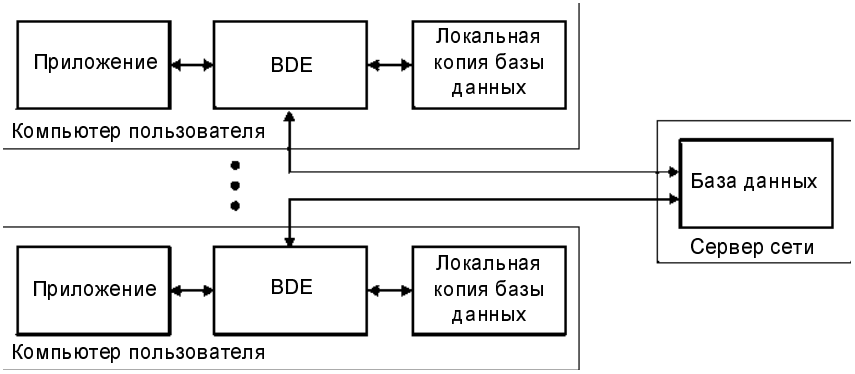


Рис. 1.2. Архитектура "файл-сервер"

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 называют также *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.

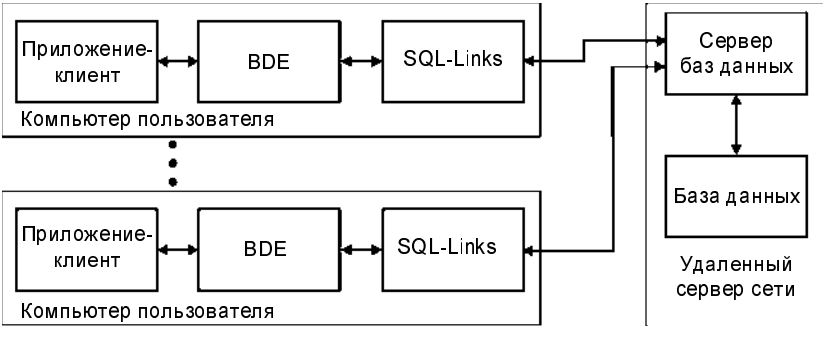


Рис. 1.3. Архитектура "клиент-сервер" ("толстый" клиент)

Для реализации архитектуры "клиент-сервер" обычно используются многопользовательские СУБД, например, Oracle или Microsoft SQL Server. Подобные СУБД называют также *промышленными*, так как 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 приложения C++ Builder к промышленным СУБД осуществляется через драйверы SQL-Links. Отметим, что при работе с "родной" для C++ Builder СУБД InterBase можно обойтись без драйверов SQL-Links.

Описанная архитектура является двухуровневой (уровень приложения-клиента и уровень сервера БД). Клиентское приложение называют также *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).

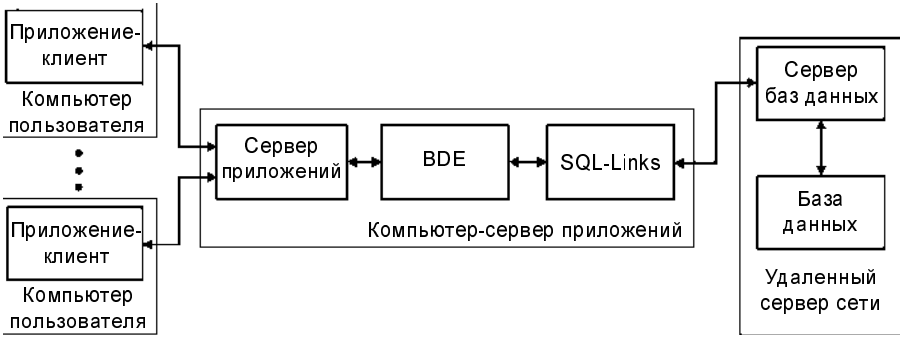


Рис. 1.4. Трехуровневая архитектура "клиент-сервер" ("тонкий" клиент)

В трехуровневой архитектуре" часть средств и кода, предназначенных для организации доступа к данным и их обработки, 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, его имя 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 — 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 приведены далее в *разд. "Таблицы форматов dBase и Paradox"* данной главы.

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.

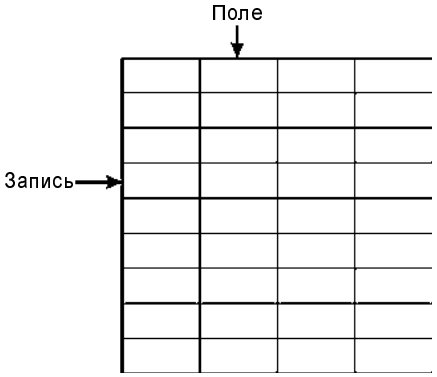


Рис. 2.1. Схема таблицы базы данных

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. Если они 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 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 является использование для него поля соответствующего типа, которое автоматически обеспечивает поддержку уникальности значений. Например, для таблиц Paradox таким является поле автоинкрементного типа, 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 автоинкрементных полей.

Отметим, что при создании и ведении БД правильным подходом считается задание в каждой таблице ключа даже в том 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.

Индекс, как и ключ, 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 C++ Builder записи набора `Table` можно сортировать только по индексным полям. Набор данных `Query` позволяет выполнить средствами SQL сортировку по любым полям, однако и в этом случае для индексированных полей упорядочивание записей выполняется быстрее.

Для одной таблицы можно создать несколько индексов. В каждый момент времени один из них можно сделать текущим, т. е. активным. Даже при существовании нескольких индексов таблица может не иметь текущего индекса (текущий индекс важен, например, при выполнении поиска и сортировки записей набора данных `Table`).

Ключевые поля обычно индексируются автоматически. В таблицах Paradox ключ также является главным (первичным) индексом, который не именуется. Для таблиц dBase ключ не создается, и его роль выполняет один из индексов.

Замечание

Создание ключа может привести к побочным эффектам. Так, если в таблице Paradox определить ключ, то записи автоматически упорядочиваются по его значениям, что в ряде случаев является нежелательным.

Таким образом, использование ключей и индексов позволяет:

- однозначно идентифицировать записи;
- избегать дублирования значений в ключевых полях;
- выполнять сортировку таблиц;
- ускорять операции поиска в таблицах;
- устанавливать связи между отдельными таблицами БД;
- использовать ограничения ссылочной целостности.

Одной из основных задач БД является обеспечение *быстрого доступа к данным* (поиска данных). Время доступа к данным в значительной степени зависит от используемых для поиска данных методов и способов.

Методы и способы доступа к данным

Выделяют следующие методы доступа к данным таблиц:

- последовательный;
- прямой;
- индексно-последовательный.

При *последовательном* методе выполняется последовательный просмотр всех записей таблицы и поиск нужных из них. Этот метод доступа является крайне неэффективным и приводит к значительным временным затратам на поиск, прямо пропорциональным размеру таблицы (числу ее записей). Поэтому его рекомендуется использовать только для относительно небольших таблиц.

При *прямом* доступе нужная запись выбирается в таблице на основании ключа или индекса. При этом просмотр других записей не выполняется. Напомним, что значения ключей и индексов располагаются в упорядоченном виде и содержат ссылку, указывающую на расположение соответствующей записи в таблице. При поиске записи выполняется не последовательный просмотр всей таблицы, а непосредственный доступ к записи на основании ссылки.

Индексно-последовательный метод доступа включает в себя элементы последовательного и прямого методов доступа и используется при поиске группы записей. Этот метод реализуется только при наличии индекса, построенного по полям, значения которых должны быть найдены. Суть его заключается в том, что находится индекс первой записи, удовлетворяющей заданным условиям, и соответствующая запись выбирается из таблицы на основании ссылки. Это является прямым доступом к данным. После обработки первой найденной записи осуществляется переход к следующему значению индекса и в таблице выбирается запись, соответствующая значению этого индекса. Так последовательно перебираются индексы всех записей, удовлетворяющих заданным условиям, что является последовательным доступом.

Достоинством прямого и индексно-последовательного методов является максимально возможная скорость доступа к данным, плата за которую — расход памяти на хранение информации о ключах и индексах.

Указанные методы доступа реализуются СУБД и не требуют специального программирования. Задачей разработчика является определение соответствующей структуры БД, в данном случае — определение ключей и индексов.

Так, если для поля создан индекс, то при поиске записей по этому полю автоматически используется индексно-последовательный метод доступа, в противном случае — последовательный метод.

Замечание

При создании составного индекса важен порядок составляющих его полей. Например, если индекс создан по полям `фамилия`, `номер отдела` и `дата рождения`, а поиск ведется одновременно по полям `фамилия`, `дата рождения` и `номер отдела`, то такой индекс использован не будет. В результате доступ к таблице осуществляется последовательным методом. В подобной ситуации (для таблицы с большим числом записей) разработчик должен создать также индекс, построенный по полям `фамилия`, `дата рождения` и `номер отдела` именно в таком порядке!

При выполнении операций с таблицами используется один из следующих *способов доступа к данным*:

- навигационный;
- реляционный (SQL-ориентированный).

Навигационный способ доступа заключается в обработке каждой отдельной записи таблицы. Этот способ обычно используется в локальных БД или в удаленных БД небольшого размера. Если необходимо обработать несколько записей, то все они обрабатываются поочередно.

Реляционный способ доступа основан на обработке сразу *группы записей*, при этом, если необходимо обработать одну запись, то обрабатывается группа, состоящая из одной записи. Так как реляционный способ доступа основывается на SQL-запросах, его также называют *SQL-ориентированным*. Этот способ доступа ориентирован на выполнение операций с удаленными БД и является предпочтительным при работе с ними, хотя его можно использовать и для локальных БД.

Способ доступа к данным выбирается программистом и зависит от средств доступа к БД, используемых при разработке приложения. Например, в приложениях, создаваемых в C++ Builder, реализацию навигационного способа доступа для механизма BDE можно осуществить с применением компонентов `Table` или `Query`, а реляционного — с помощью компонента `Query`. Для других механизмов доступа к данным также можно использовать навигационный и реляционный способы доступа, естественно с применением соответствующих аналогов компонентов `Table` и `Query` (например, `ADOTable` и `ADOQuery` для механизма ADO).

Таким образом, методы доступа к данным определяются структурой БД, а способы доступа — приложением.