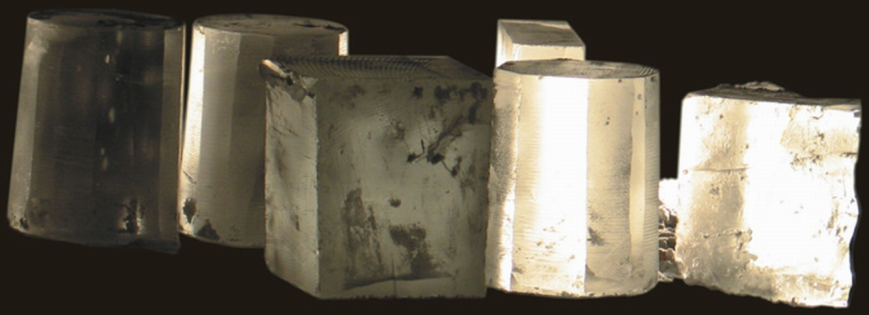
АЛЕКСАНДР БОНДАРЬ

InterBase и Firebird

практическое руководство для умных пользователей и начинающих разработчиков

bhv



Подробное описание языков DDL и DML серверов InterBase и Firebird

Практические рекомендации по созданию структур баз данных

Детальное рассмотрение аспектов манипулирования данными и управления транзакциями

Реальные примеры клиентских приложений на Delphi с использованием FIBPlus и IBX

+ CD

Александр Бондарь

InterBase и Firebird

практическое руководство для умных пользователей и начинающих разработчиков

Санкт-Петербург
«БХВ-Петербург»
2007

УДК 681.3.06
ББК 32.973.26-018.2
Б81

Бондарь А. Г.

Б81 InterBase и Firebird. Практическое руководство для умных пользователей и начинающих разработчиков. — СПб.: БХВ-Петербург, 2007. — 592 с.: ил. + CD-ROM

ISBN 978-5-9775-0098-2

Рассматриваются возможности серверов баз данных InterBase и Firebird. Описываются объекты базы данных (таблицы, домены, индексы, представления и т. д.), синтаксис и семантика операторов SQL, используемых для работы с метаданными и с данными базы данных. Рассматриваются вопросы проектирования реляционных баз данных для решения реальных задач предметной области, на которой проводятся всевозможные исследования. Приводится множество примеров использования операторов манипулирования данными. Детально описаны транзакции, взаимодействие параллельных процессов с различными характеристиками и уровнями изоляции транзакций. Создается учебная база данных и множество программ в среде Delphi, иллюстрирующих возможности серверов базы данных. Компакт-диск содержит примеры из книги, а также программное обеспечение для разработчиков БД: InterBase 2007 Developer Edition, Firebird 2.0 и др.

Для разработчиков баз данных и программистов

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	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Инны Тачиной*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.04.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 47,73.
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 978-5-9775-0098-2

Оглавление

Введение 10

Для кого эта книга? 10
Содержание книги 12
Где найти нужные сведения и программные средства? 16
Благодарности 17
Что там за перевалом? 18

Глава 1. Реляционные базы данных InterBase и Firebird 19

1.1. Основные объекты и понятия реляционной базы данных 21
1.2. Реализация отношений в реляционной модели 28
1.2.1. Отношение "один-к-одному" 28
1.2.2. Отношение "один-ко-многим" 29
1.2.3. Отношение "многие-ко-многим" 30
1.3. Нормализация таблиц 32
1.3.1. Первая нормальная форма 32
1.3.2. Вторая нормальная форма 33
1.3.3. Третья нормальная форма 34
1.3.4. Четвертая и пятая нормальные формы 35
1.3.5. Контрольное задание 36
1.4. 12 правил Кодда 37
1.5. Язык SQL 40
1.5.1. Назначение языка SQL 40
1.5.2. Синтаксические конструкции 40
1.5.3. Основные операторы SQL 43
1.6. Обращение к программам системы управления базами данных 44
Что там за перевалом? 46

Глава 2. Создание базы данных 47

2.1. Учетные записи пользователя 47
2.2. Создание новой учетной записи пользователя 48
2.2.1. Использование gsec 48
2.2.2. Использование IBExpert 53
2.2.3. Для особо одаренных. Создание собственной программы для работы с учетными записями пользователей 55
2.3. Создание учебной базы данных 71
2.3.1. Использование скриптов 73
2.3.2. Несколько слов о транзакциях 73

2.3.3. Создание базы данных с использованием isql 74
2.3.4. Создание базы данных с использованием IBExpert 76
2.3.5. Для особо одаренных. Собственная программа для создания базы данных 80
2.4. Соединение с базой данных 84
2.5. Удаление базы данных 85
2.6. Регистрация базы данных в IBExpert 87
2.7. Копирование и восстановление базы данных 89
2.7.1. Копирование и восстановление базы данных утилитой командной строки 89
2.7.2. Копирование и восстановление базы данных в IBExpert 91
2.7.3. Для особо одаренных. Собственная программа копирования базы данных 94
2.7.4. Для особо одаренных. Собственная программа восстановления базы данных 99
Что там за перевалом? 101

Глава 3. Работа с доменами 103

3.1. Типы данных 103
3.1.1. Числовые типы данных 104
3.1.2. Типы данных даты и времени 108
3.1.3. Строковые типы данных 112
3.1.4. Тип данных *BLOB* 112
3.1.5. Тип данных *BOOLEAN* 113
3.2. Синтаксис оператора создания домена 113
3.3. Создание доменов 120
3.4. Изменение домена 122
3.5. Удаление домена 123
3.6. Создание доменов для учебной базы данных 123
Что там за перевалом? 125

Глава 4. Работа с таблицами 126

4.1. Первичные ключи 126
4.2. Проектирование таблиц 128
4.2.1. Общая постановка задачи 129
4.2.2. Справочные таблицы 129
4.2.3. Оперативные таблицы 132
4.2.4. Промежуточные итоги 137
4.3. Контрольная работа 138
4.3.1. Учебные дисциплины 138
4.3.2. Сотрудники организации 138
4.3.3. Движение товаров на складах 139
4.4. Синтаксические конструкции 139
4.4.1. Работа с генераторами 139
4.4.2. Синтаксис создания таблицы 141

4.5. Создание таблиц и генераторов 147
4.6. Удаление и изменение таблиц 162
4.7. Создание таблиц для учебной базы данных 164
4.8. Использование индексов 164
4.9. Правильные ответы 166
4.9.1. Дисциплины, разделы, темы 166
4.9.2. Организации, отделы, сотрудники 168
4.9.3. Движение материалов на складах 169
Что там за перевалом? 172

Глава 5. Добавление, изменение, удаление и выборка данных 173

5.1. Добавление данных в базу данных 174
5.1.1. Добавление в таблицу одной строки 174
5.1.2. Добавление в таблицу нескольких строк 177
5.2. Удаление данных 179
5.3. Изменение данных 180
5.4. Выборка данных 181
5.5. Использование выражений и функций в операторах 182
5.6. Выполнение скриптов добавления и изменения данных 184
Что там за перевалом? 194

Глава 6. Для особо одаренных. Написание программ работы с базой данных 195

6.1. Программа просмотра и удаления стран 196
6.1.1. Использование компонентов FIBPlus 196
6.1.2. Использование компонентов IBX 213
6.2. Полнофункциональная программа работы со справочником стран 219
6.2.1. Использование компонентов FIBPlus 219
6.2.2. Использование компонентов IBX 232
6.3. Программа работы со справочниками стран и регионов 235
6.3.1. Использование компонентов FIBPlus 235
6.3.2. Использование компонентов IBX 240
6.4. Изменение упорядоченности наборов данных 241
6.4.1. Использование компонентов FIBPlus 242
6.4.2. Использование компонентов IBX 247
Что там за перевалом? 249

Глава 7. Поиск данных 250

7.1. Синтаксис оператора *SELECT* 250
7.2. Простые варианты поиска данных 252
7.2.1. Задание отображаемых столбцов 253
7.2.2. Упорядочение результата запроса. Предложение *ORDER BY* 256
7.2.3. Использование ключевого слова *DISTINCT* 260
7.2.4. Задание условий выборки. Предложение *WHERE* 263
7.3. Соединение таблиц 286

7.3.1. Внешние соединения 287
7.3.2. Более сложные примеры соединений 293
7.3.3. Внутреннее соединение 298
7.3.4. Замечания по синтаксису 301
7.4. Группировка результатов выборки 303
7.5. Использование подзапросов в операторах SQL 308
7.6. Использование представлений 312
7.6.1. Создание представлений 313
7.6.2. Примеры представлений 313
7.6.3. Типы представлений: только для чтения или изменяемые 317
7.6.4. Удаление представлений 317
Что там за перевалом? 318

Глава 8. Транзакции 319

8.1. Синтаксис оператора *SET TRANSACTION* 320
8.2. Характеристики транзакций 320
8.2.1. Режим доступа 321
8.2.2. Уровень изоляции 321
8.2.3. Режим разрешения блокировок 322
8.3. Для особо одаренных. Написание исследовательской программы 323
8.3.1. Использование компонентов FIBPlus 323
8.3.2. Использование компонентов IBX 333
8.3.3. Числовые значения параметров TPB 334
8.4. Исследование характеристик транзакций 335
8.4.1. Уровень изоляции *READ COMMITTED* 336
8.4.2. Уровень изоляции *SNAPSHOT* 340
8.4.3. Уровень изоляции *SNAPSHOT TABLE STABILITY* 341
8.4.4. Средства резервирования таблиц 342
8.4.5. Использование разделенных транзакций 345
8.4.6. Преимущества использования компонентов FIBPlus 347
8.4.7. Вложенные транзакции 349
8.5. Краткие итоги 354
Что там за перевалом? 356

Глава 9. Хранимые процедуры и триггеры 357

9.1. Язык хранимых процедур и триггеров 358
9.1.1. Использование оператора *SET TERM* 358
9.1.2. Использование переменных 359
9.1.3. Выполняемые операции 360
9.2. Исключения базы данных 365
9.2.1. Пользовательские исключения 366
9.2.2. Обработка исключений и ошибок базы данных в триггерах и хранимых процедурах 367
9.3. События базы данных 368
9.4. Триггеры 368

9.4.1. Создание триггеров 369
9.4.2. Удаление и изменение триггеров 370
9.4.3. Примеры триггеров 371
9.5. Хранимые процедуры 389
9.5.1. Создание, изменение и удаление хранимых процедур 390
9.5.2. Выполняемые хранимые процедуры 391
9.5.3. Хранимые процедуры выбора 394
Что там за перевалом? 406

Глава 10. Для особо одаренных. Полезные программы работы с базой данных 408

10.1. Программа работы со списком людей 408
10.1.1. Использование компонентов FIBPlus 410
10.1.2. Использование компонентов IBX 435
10.2. Программа работы с двумя базами данных 440
10.3. Программа просмотра и печати справочника стран 449
10.3.1. Использование компонентов FIBPlus 449
10.3.2. Использование компонентов IBX 457
10.4. Программа, включающая средство редактирования полей *BLOB* 457
Что там за перевалом? 458

Приложения

Приложение 1. Установка необходимых программ и компонентов 459

П1.1. Установка Firebird 2.0 459
П1.2. Установка InterBase 2007 Developer Edition 468
П1.3. Установка компонентов FIBPlus 474
П1.3.1. Установка полной версии 475
П1.3.2. Установка пробной версии 477
П1.4. Установка компонентов RichView 486
П1.5. Установка компонентов FastReport 486
П1.6. Установка IBExpert 490

Приложение 2. Наборы символов и порядок сортировки 491

П2.1. Полный список наборов символов и порядков сортировки 491
П2.2. Структура системных таблиц 498
П2.3. Для особо одаренных. Программа просмотра набора символов и порядка сортировки 499
П2.3.1. Использование компонентов FIBPlus 499
П2.3.2. Использование компонентов IBX 504

Приложение 3. Коды ошибок 507

Приложение 4. Структура некоторых системных таблиц 590

П4.1. База данных, файлы, страницы 590
П4.1.1. Структура системных таблиц 590
П4.1.2. Программа с использованием компонентов FIBPlus 592
П4.1.3. Программа с использованием компонентов IBX 594
П4.2. Отношения (таблицы, представления) 594
П4.2.1. Структура системных таблиц 594
П4.2.2. Программы просмотра отношений 595
П4.3. Триггеры 598
П4.3.1. Структура системной таблицы 598
П4.3.2. Программы просмотра описаний триггеров 599
П4.4. Хранимые процедуры 601
П4.4.1. Структура системных таблиц 602
П4.4.2. Программы просмотра описаний хранимых процедур и их параметров 603

Приложение 5. Дополнительные материалы 606

Предметный указатель 608

Эту книгу я посвящаю
памяти моего очень близкого друга
Владимира Васильевича Рыбина.
Мне страшно жаль, Володя,
что ты не можешь прочесть эти строки.

Введение

Говорят, что мало кто читает введения, предисловия. Если это действительно так, то не стану здесь много рассказывать, какие же замечательные системы управления базами данных InterBase и Firebird. Поскольку вы держите в руках эту книгу, то наверняка знаете, что это компактные и мощные серверы баз данных. К тому же Firebird — абсолютно бесплатная система, а версия InterBase 2007 Developer Edition — бесплатная для разработчиков.

Для кого эта книга?

У нас в стране очень много книг серии "для чайников". В оригинале, правда, это звучит несколько иначе — "for dummies", что все же более правильно следует перевести как "для придурков". Появилась еще одна серия, в названии которой присутствуют слова complete, idiot's, guide. В зависимости от допустимой перестановки слов в названии этой серии может получиться либо "полное руководство", либо "для полных идиотов"[1].

Я же считаю, что человек, ринувшийся заниматься программированием, не может быть ни dummy, ни тем более idiot. По этой причине в название этой книги я поместил слово "умный".

Основная задача книги — дать возможность "потрогать руками" различные средства этих серверов баз данных по принципу "делай раз, делай два". Выполняя предложенные действия, вы действительно освоите множество полезных вещей в работе с базами данных. Я, по крайней мере, освоил многое. В том числе и благодаря написанию этой книги.

Чтение книги не требует предварительных знаний в области реляционных баз данных. Откровенно говоря, это не совсем так. Человек, вообще не

[1] Я настоятельно рекомендую некоторым моим студентам тщательно изучать книги именно этих серий.

знакомый с программированием, его основными принципами, вряд ли получит настоящее удовольствие от использования данной книги. Конечно же, для подобной деятельности нужны определенные знания, некая "программерская" культура.

При этом книга, конечно же, в первую очередь предназначена для начинающих использовать эти замечательные серверы баз данных, хотя — бьюсь об заклад — мало кто из корифеев InterBase и Firebird так же тщательно выполнял все те работы, которые описаны в книге. Уверен, что книга будет полезна и тем, кто много и плодотворно работает с этими серверами.

Основную целевую аудиторию этого издания я бы разделил на три категории.

- Люди, желающие естественным, структурированным, путем получить необходимые практические знания о системах управления базами данных вообще и о системах InterBase и Firebird в частности. Их будет в первую очередь интересовать язык SQL, его возможности, операторы, а также существующие программные средства для выполнения запросов SQL. Написание собственных программ работы с базами данных их, похоже, не очень в настоящий момент интересует. Им следует пока смело пролистывать страницы, посвященные разработке программ, и внимательно изучать материал, посвященный серверам баз данных и конструкциям языка SQL. К написанию программ можно приступить и потом. Уверен, что этим они со временем обязательно займутся.
- Те, кого интересует именно разработка программ. Про SQL они слышали, а может быть, и довольно активно использовали его возможности. Они могут лишь бегло просматривать все описания, относящиеся к реляционным базам данных, операторам SQL, и выполнять подробно описанные действия по созданию собственных программ. Со временем они могут вернуться к начальным главам и более глубоко освоить действия с базой данных с помощью средств SQL, тем более что эти действия того стоят.
- Люди, которые, доверившись автору, выполняют все предлагаемые в книге действия — используют утилиты командной строки и программу графического интерфейса, выполняя операторы SQL для получения желаемого результата, запускают среду Delphi, пишут программы, предлагаемые автором, находят более элегантные решения, чем те, которые описаны в книге, получают от этого большое удовольствие и намекают автору, что следовало бы быть повнимательнее при выборе вариантов решения задач. Наверное, нет необходимости говорить, что такая категория людей всегда добьется в жизни много больше, чем другие.

Вообще говоря, существует еще одна категория читателей. Это люди, которые просто из любопытства начинают просматривать книгу, потом вдруг по-

нимают, что реляционные базы — это действительно очень интересно, и со временем становятся крупными специалистами в этой области деятельности. Удачи!

Содержание книги

В *главе 1 "Реляционные базы данных InterBase и Firebird"* очень кратко рассказывается о работе с внешними данными, с базами данных. Рассматриваются объекты реляционных баз данных. Разбираются отношения в базах данных и их реализация применительно к реляционным базам данных. Здесь же мы познакомимся с такой важной деятельностью, как нормализация таблиц, выполним небольшую работу по созданию нескольких простых таблиц. В конце главы рассказывается о средствах описания синтаксиса формальных языков, дается формальное описание некоторых базовых конструкций языка SQL.

Если вам захочется поподробнее узнать об истории, принципах и — страшно сказать — математических основах реляционных баз данных, вы всегда сможете найти подходящую литературу.

Сама наша работа с базами данных начинается в *главе 2 "Создание базы данных"*. Мы начнем с создания и изменения учетных записей пользователя. После этого различными способами будем создавать нашу учебную базу данных, с которой будем работать на протяжении всей этой книги. Научимся регистрировать базу данных в программе IBExpert. Освоим различные средства копирования и восстановления баз данных. Там же мы создадим собственные программы для работы с учетными записями пользователя, для создания базы данных и для выполнения копирования и восстановления баз данных.

Затем в *главе 3 "Работа с доменами"* рассмотрим домены, типы данных, используемые в серверах InterBase и Firebird. Проведем множество экспериментов с различными типами данных. Научимся создавать, изменять и удалять домены. Наконец, создадим несколько доменов, которые будем использовать при описании таблиц нашей базы данных.

Достаточно объемная и важная *глава 4 "Работа с таблицами"*. В ней рассматривается основной и самый сложный объект реляционной базы данных — таблицы. Даются рекомендации по выбору первичного ключа. Рассматривается весьма сложный синтаксис оператора создания таблиц. Проектируются и создаются все таблицы, используемые в нашей учебной базе данных. Кратко рассматриваются индексы. Предлагается для решения несколько задач по проектированию и созданию таблиц. Там же я даю и свои варианты решений этих задач. Ваши могут быть лучше.

В *главе 5 "Добавление, изменение, удаление и выборка данных"* подробно рассматриваются операторы, выполняющие соответствующие действия. Не-

сколько предварительных слов сказано об операторе выборки данных. Здесь мы добавляем и изменяем данные нашей базы данных.

Глава 6 называется *"Для особо одаренных. Написание программ работы с базой данных"*. Здесь мы с вами сделаем небольшой перерыв в рассмотрении средств и языковых конструкций систем управления базами данных и разом напишем множество программ для решения различных полезных задач работы с базами данных.

Самый сложный и интересный оператор — оператор `SELECT`, позволяющий осуществлять выборку данных из базы. Ему полностью посвящена весьма объемная *глава 7 "Поиск данных"*. Мы достаточно подробно рассмотрим богатый синтаксис и необыкновенные по своей мощности возможности оператора `SELECT`, выполним много различных операций поиска данных в нашей учебной базе данных. Будем использовать в операторе различные функции, выполним операции соединения (`JOIN`). Там же исследуются представления (view), создается некоторое количество довольно простых (и не очень) полезных представлений.

В *главе 8 "Транзакции"* рассматривается такой замечательный и на самом деле не слишком сложный механизм, как транзакции — различные их характеристики и варианты использования. Мы напишем тестовую программу, или несколько программ, которые позволят нам детально исследовать отдельные характеристики транзакций и их влияние на параллельные процессы. Если у вас есть возможность работать в локальной сети, вы сможете в полном объеме оценить достоинства транзакций.

В *главе 9 "Хранимые процедуры и триггеры"* мы рассмотрим процедурное расширение языка SQL, используемое для написания хранимых процедур и триггеров. Вы сами или с моей помощью напишете несколько полезных триггеров и хранимых процедур. Исследование возможностей этих объектов базы данных займет достаточно много времени, и это время не будет потрачено впустую.

Завершает книгу *глава 10 "Для особо одаренных. Полезные программы работы с базой данных"*, где мы с вами выполним совместную разработку некоторого количества довольно сложных и, главное, интересных программ. Для их создания мы будем использовать все полученные в процессе работы над книгой знания и умения. Поверьте, результат будет замечательным.

В приложениях содержатся дополнительные полезные данные.

В *приложении 1* приведено подробное описание инсталляции всех программ и компонентов, которые вам понадобятся для выполнения заданий этой книги и вообще в вашей профессиональной деятельности за исключением Delphi.

Приложение 2 содержит полное описание наборов символов и порядка сортировки, используемых в системах InterBase и Firebird. Поясняется структура

соответствующих системных таблиц. Мы с вами напишем программы, отображающие состав наборов символов и порядка сортировки, фактически присутствующие в вашей реальной базе данных.

Приложение 3 содержит описание кодов ошибок, с которыми вы можете столкнуться в вашей работе.

В *приложении 4* приведена структура некоторых системных таблиц. Создаются программы для отображения в удобном виде данных из этих таблиц.

Приложение 5 описывает содержимое сайта, где расположены необходимые материалы для работы с книгой.

Многие ваши работы при использовании этой книги выполняются в три этапа.

1. Сначала выполняются необходимые действия с использованием утилит командной строки, входящих в комплект поставки серверов базы данных. В любом случае это следует проделать на начальных этапах, чтобы оценить все удобства (или неудобства) таких утилит. Потом использование командной строки вам, конечно, надоест, и вы перейдете к работе только с программой графического интерфейса, тем более что при попытках отобразить русский текст из строковых полей базы данных в командной строке вы получите не совсем то, что ожидали.
2. Затем те же действия выполняются с использованием программы графического интерфейса. Здесь и в повседневной программистской жизни я использую IBExpert. Практически с тем же успехом можно использовать и любую другую программу, например, BlazeTop.
3. Третий этап (главы и разделы начинаются со слов "для особо одаренных") дает возможность написать собственную программу, выполняющую те же самые действия. Для этого нужна среда Delphi версии не ниже 7 и компоненты FIBPlus или IBX для работы с базами данных.

Разработка программ описывается очень подробно, временами даже слишком подробно. В первую очередь это касается разработки начальных программ. У некоторых очень внимательных читателей и продвинутых программистов это может вызвать раздражение, однако менее внимательные и начинающие программисты будут мне благодарны за многочисленные повторы.

Примерно 80% материала книги относится к основным сведениям по серверам баз данных и языковым средствам, остальная часть связана с написанием качественных программ по работе с базами данных.

Если же вам не захочется сразу же проявить свою одаренность, вы можете отложить написание программ на более позднее время или же вовсе скрыть

от окружающих и от себя самого (самой) свои необыкновенные способности по написанию полезных программ.

Примеры не содержат ошибок. Последнее время это стало весьма распространенной программистской шуткой. Однако должен сказать, что *все* примеры и программы проверены мною в Firebird 1.5.3, Firebird 2.0 и InterBase 2007 Developer Edition на локальном компьютере в операционной системе Windows XP и в локальной сети, включающей 15 компьютеров, под управлением Windows 2003.

Данная книга, с моей точки зрения, содержит большое количество достоинств. Вот лишь некоторые из них.

- **Практический подход к освоению систем управления базами данных.** Я стараюсь избегать специфической терминологии, наукообразия в изложении материала. Это дает возможность, не вдаваясь в философские и математические детали, лежащие в основе реляционных баз данных, научиться за короткий промежуток времени использовать описываемые СУБД в своей практической деятельности. Такой подход, я надеюсь, позволит вам создавать качественные базы данных и столь же хорошие программы работы с этими базами данных.
- **Последовательность в изложении материала.** Это не справочник и не избранные темы относительно реляционных баз данных вообще и конкретных СУБД в частности. Последовательно изучая материал книги и выполняя предложенные действия, вы, в конце концов, приобретете глубокие и твердые знания, научитесь на практике применять эти знания.
- **Доступность изложения.** Книга написана нормальным человеческим языком для нормальных людей. Любой желающий при некоторых интеллектуальных усилиях освоит предлагаемый материал.

При этом книга, разумеется, не свободна и от некоторых мелких недостатков. Собственно говоря, их ровно три.

- **Практический подход к освоению систем управления базами данных.** Это не позволяет читателю получить глубокие знания в области реляционной алгебры, истории возникновения и развития реляционных баз данных.
- **Последовательность в изложении материала.** Это не дает возможности читателю рассматривать данную книгу только как справочник, позволяющий быстро найти ответ на интересующий его частный вопрос.
- **Доступность изложения.** Материал в книге представлен очень свободно, без надлежащей математической строгости. По прочтении такой книги нет причин невзначай сообщить знакомым, что вот, мол, освоил я таки столь сложную книгу.

Если вас устраивают достоинства и не слишком огорчают недостатки — приступайте к этой интереснейшей деятельности!

Надеюсь, вы понимаете, что освоение нового материала или упорядочение знаний по уже известной тематике всегда требует определенных усилий, временами довольно больших. Мне нравится сравнение такой деятельности с прохождением горных маршрутов. Есть периоды, когда нужно напрячь все силы и подниматься по камням, по снегу, по фирну все выше и выше к перевалу. Сердце сильно бьется в груди. Воздух разряжен, дышать трудно. Каждый шаг дается с трудом. Такое напряжение всегда вознаграждается, когда в состоянии необыкновенного восторга ты стоишь на высокой точке своего маршрута, понимая, какой сложный путь ты преодолел. После этого идет плавный быстрый спуск. Вот ты уже на равнине, ласково светит солнце, кругом зеленая травка, цветочки, птички поют, пчелки летают. И ты готовишься к выходу на следующий перевал, прекрасно понимая, что там будет еще сложнее, еще лучше, еще интереснее...

Я с трудом скрываю свою зависть, зависть белую, к тем людям, которые впервые пойдут по этому маршруту.

Где найти нужные сведения и программные средства?

Скрипты, используемые для создания базы данных, всех ее объектов, помещения в таблицы данных, а также тексты программ находятся на сайте издательства по адресу: ftp://ftp.bhv.ru/9785977500982.zip.

Знающим английский язык (*настоящий* программист должен знать этот язык) можно порекомендовать использовать полный комплект документации по InterBase 2007. Эта документация станет доступной только после инсталляции сервера базы данных. С Firebird 2.0 вы получаете три документа в формате PDF. Если вы хотите серьезно заниматься этими базами данных, имеет смысл изучить или хотя бы просмотреть все перечисленные документы. На русском языке существует объемная книга Хелен Борри "Firebird: руководство разработчика баз данных", а также книга А. Ковязина и С. Вострикова "Мир InterBase". Книгу Хелен Борри можно было бы назвать энциклопедией Firebird. Вообще по реляционным базам данных существует море литературы как на английском, так и на русском языках. Порекомендовать какую-либо одну из них я не решусь.

Если вы хотите выполнять предлагаемые примеры, вам, по меньшей мере, понадобится сервер базы данных не менее InterBase 6 — это бесплатная версия, правда, содержащая большое количество ошибок.

На сайтах **sourceforge.net** и **firebirdsql.org** можно найти бесплатный Firebird последних версий. Самые последние сведения об InterBase (русскоязычный вариант сайта) находятся на **embarcadero.com/ru/**.

Я также использую программу графического интерфейса для работы с базами данных — IBExpert. Получить бесплатную для стран бывшего СССР (граждане которых не делают вид, что не знают русского языка) новую версию можно на сайте **www.ibexpert.com**.

Бесплатную пробную, "триальную" версию компонентов работы с базами данных FIBPlus вы найдете на сайте **devrace.com**.

Компоненты FastReports на сайте **fast-report.com/**.

Компоненты RichView находятся на сайте **trichview.com/**.

Для выполнения всех работ, описанных в этой книге, вам понадобится еще и среда разработки Delphi версии не менее 7. Это платная система и по нашим меркам довольно дорогая, но она того стоит.

Благодарности

Это, пожалуй, самая приятная часть введения. Автор испытывает чувство признательности к очень многим людям и организациям, благодаря которым оказалась написанной эта книга.

В первую очередь мне хотелось бы поблагодарить ребят (и девчат), которые создавали для нас с вами такие замечательные СУБД, как InterBase и Firebird.

Спасибо ребятам, разработавшим очень удобную программу работы с этими СУБД — IBExpert. Именно ее я использую в своей повседневной работе. И другим советую.

Большая благодарность разработчику компонентов FIBPlus для работы с базами данных, созданных в InterBase и Firebird, Сергею Бузаджи. Спасибо тебе, Сергей, за прекрасную разработку и постоянное совершенствование этого, казалось бы и без того совершенного продукта. Спасибо тебе также и за твое долготерпение и подробные объяснения в то время, когда я заваливал тебя различными вопросами (признаюсь — не всегда самыми умными) по тонкостям использования этих компонентов.

Спасибо Сергею Ткаченко за довольно сложные, но очень мощные полнофункциональные компоненты RichView для работы с текстами в обогащенном формате RTF. Эти компоненты называются автором средствами для отображения, редактирования и печати гипертекста.

Спасибо компании FastReports во главе с Михаилом Филиппенко за компоненты FastReport, очень удобные и красивые, позволяющие довольно легко создавать практически любые отчеты.

Огромная благодарность Сергею Вострикову, научному редактору этой книги, который внимательно перечитывал постоянно меняющиеся главы и давал ценные советы. Спасибо, Сережа, а те грубые слова, которые ты мне говорил по поводу моей привычки обрабатывать ошибочные ситуации базы данных, я уже почти забыл.

Хочу поблагодарить коллектив издательства "БХВ-Петербург" за их высокий уровень профессионализма. По своему предыдущему опыту сотрудничества с этим издательством могу отметить действительно профессиональную работу, в первую очередь, корректоров и редакторов.

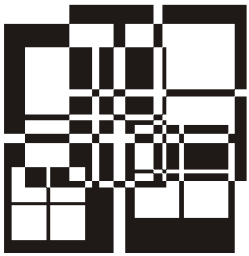
И, наконец, самая большая благодарность всем моим студентам. За их поддержку, за их попытки изобразить полное непонимание отдельных вопросов работы с базами данных. В результате книга стала еще более доступной для большинства людей, проявляющих в той или иной мере интерес к реляционным базам данных. Я очень надеюсь, что читателя не будут так уж сильно раздражать многочисленные повторы, встречающиеся в книге.

Что там за перевалом?

В следующей главе мы начнем рассмотрение основ реляционных баз данных вообще и СУБД InterBase и Firebird в частности.

Мы рассмотрим объекты базы данных, реализацию отношений в реляционных базах данных, способы и практическую необходимость нормализации таблиц, а также основные синтаксические конструкции языка SQL.

ГЛАВА 1



Реляционные базы данных InterBase и Firebird

За сравнительно короткий промежуток времени идеи и технологии использования баз данных прошли большой и, надо сказать, впечатляющий путь.

Сначала все обрабатываемые данные носили исключительно последовательный характер. Это были колоды перфокарт, печатающие устройства. Позже появились магнитные ленты.

Технические предпосылки сложной организации данных возникли только после появления внешних запоминающих устройств прямого доступа — магнитных барабанов, пакетов магнитных дисков. В это время и появились первые идеи создания баз данных и систем управления базами данных.

В настоящее время большинство прикладных программ связано с обработкой большого объема данных, имеющих сложную структуру. Примерами таких программ являются поисковые системы (библиографические и так называемые "базы знаний"), программные системы организационного управления (обработка данных для управления предприятиями, для решения учетных задач, например, существует одна из наиболее емких в плане количества обрабатываемых данных подсистем — бухгалтерский учет), системы управления космическими объектами и др. Для хранения данных при решении подобных задач используются базы данных.

База данных — это один или несколько связанных между собой файлов, в которых хранятся все обрабатываемые данные, а также описания этих данных — метаданные. Программные средства работы с данными в базе данных называются системой управления базой данных (сокращенно СУБД) или иногда, не слишком удачно, сервером базы данных. Мы будем использовать оба этих термина. Есть еще один термин, связанный с такими программами.

Ядро СУБД называют engine, дословно двигатель, движок. Языковые средства, используемые в базах данных, дают возможность, во-первых, описывать данные в базе данных, а во-вторых, осуществлять манипулирование данными — выполнять добавление, изменение, удаление и выборку данных.

Понятно, что сама по себе база данных просто так никому не нужна. Она используется вместе с соответствующими программами, обрабатывающими эти данные. Если подняться с программистского уровня рассмотрения баз данных на общечеловеческий, то можно сказать, что данные в базе данных описывают состояние какой-то конкретной предметной области человеческой деятельности, а пользовательские, клиентские программы решают задачи из этой предметной области.

Одним из важных принципов баз данных является то, что описание данных (метаданные) хранится также в самой базе данных. Это позволяет уменьшить зависимость программ от структуры хранимых данных.

В *реляционных базах данных* все обрабатываемые данные хранятся в виде однородных таблиц. Эти таблицы часто называют плоскими, двумерными, поскольку все строки одной таблицы имеют одинаковую структуру. Программные средства работы с реляционными базами данных называются реляционной СУБД. В качестве языковых средств используется SQL — Structured Query Language, язык структурированных запросов. Этот язык включает в себя две части — язык описания данных (DDL, Data Definition Language) и язык манипулирования данными (DML, Data Manipulation Language).

Несмотря на то, что существуют международные стандарты SQL, реальные системы управления базами данных временами довольно сильно отличаются друг от друга как по языковым средствам, так и по выполняемым функциям. Мы будем рассматривать семейство серверов баз данных InterBase и Firebird. Они очень похожи. Это компактные и довольно мощные серверы баз данных. Большинство описываемых здесь возможностей существует в InterBase, начиная с версии 6.0, и в Firebird версии 1.5 и выше.

При написании программ работы с базами данных, поддерживаемыми этими СУБД, а также при выполнении таких программ, нам практически все равно, где находится сама база данных — на нашем, локальном, компьютере или на сервере в локальной сети. Иными словами, поддержка архитектуры "клиент-сервер" в этих СУБД осуществляется легко и непринужденно. От нас почти не требуется никаких дополнительных затрат сил и времени для того, чтобы наши программы работали как на локальном компьютере, так и в архитектуре "клиент-сервер", когда с одной базой данных одновременно работает несколько пользовательских программ. Единственная задача — так организовать использование транзакций и их характеристик, чтобы минимизировать блокировки при одновременной работе многих пользователей с одной базой данных. Транзакциям мы посвятим отдельную главу.

В реляционных базах данных существует своя система строгих понятий и принципов, которые и делают подобные СУБД очень удобными в работе.

Давайте договоримся о самой начальной терминологии — в базах данных хранятся, а программами обрабатываются только *данные*. Термин же *информация*, и в плане хранения, и в плане обработки, я предпочитаю применять только к человеку.

1.1. Основные объекты и понятия реляционной базы данных

Рассмотрим некоторые понятия и объекты реляционной базы данных, а также отдельные их характеристики. В соответствующих главах мы рассмотрим эти объекты подробнее.

Все данные в реляционной базе данных хранятся в виде таблиц — плоских двумерных структур. В реляционной базе данных хранятся собственно данные, организованные в виде таблиц, и *метаданные* — описания этих данных. Метаданные хранятся в *системных таблицах*.

Каждая *таблица (table)* содержит произвольное количество *строк (row)*. Другие названия для строки: запись (record), реже — кортеж. В русскоязычной литературе практически одинаково часто используются термины "строка" и "запись", равно как и в англоязычной — аналогичные английские варианты. Все строки одной таблицы имеют одинаковую структуру. Они состоят из одного или нескольких *столбцов (column)*, иногда называемых полями (field), элементами данных (data element), колонками или атрибутами (attribute). У нас обычно используются термины "столбец" и "поле". Каждый столбец имеет конкретный тип данных.

Таблицу любители реляционной алгебры часто называют отношением. С математической точки зрения это правильно, однако у большинства программистов такой термин вызывает негативную реакцию.

Тип данных (datatype) — важнейшая характеристика столбца, определяющая множество возможных значений, которые может принимать данный столбец, и операции, допустимые для данного столбца. Например, тип данных `INTEGER` позволяет хранить в столбце целые числа в определенном диапазоне; для такого столбца можно использовать четыре арифметические операции — сложение, вычитание, умножение и деление. Для строковых типов данных множеством допустимых значений являются строки, содержащие произвольные символы. К ним применяется одна операция конкатенации — соединения двух строк в одну. У логического типа данных множество допустимых значений состоит из двух элементов, истина (`TRUE`) и ложь (`FALSE`). В реляционных базах данных для логического типа данных используется еще и неопределен-

ное, или неизвестное, (NULL) значение. Множеством допустимых операций являются, как минимум, отрицание, дизъюнкция (логическое ИЛИ) и конъюнкция (логическое И). Существуют также типы данных даты, времени, чисел с плавающей точкой и др.

Домен (domain) — в математической терминологии это множество допустимых значений и множество допустимых операций для элемента данных. В реляционных базах данных существует объект домен, который описывает некоторые характеристики элемента данных (как минимум, тип данных) и на который можно ссылаться при описании столбцов таблицы.

Пустое значение (NULL) — в реляционных базах данных столбец и локальная переменная в хранимых процедурах и триггерах помимо "нормального" значения может иметь также пустое, или неизвестное значение. Интересным и даже на первый взгляд несколько странным свойством этого значения является то, что два поля, имеющие своим значением NULL, не равны друг другу. Любая операция сравнения, в которой принимают участие такие поля, не даст значений ни TRUE (истина), ни FALSE (ложь). Результатом сравнения в этом случае будет NULL (неизвестный результат). До сих пор еще ведутся споры относительно необходимости использования и правильности интерпретации пустого значения. Мы же с вами не станем тратить на это время и примем сложившуюся практику так, как она есть. Тем более что способы работы с пустыми значениями хорошо отработаны. При этом только нужно постоянно помнить, что прежде чем сравнивать соответствующие объекты базы данных, для которых допустимо пустое значение, требуется обязательная проверка их на значение NULL. Лично я об этом иногда забываю.

Индекс (index). Для таблицы может быть задано любое количество индексов[2]. Индекс состоит из одного или нескольких столбцов таблицы. В базе данных он представлен упорядоченным списком значений и адресов. В каждом элементе списка находится адрес строки таблицы, содержащей это значение. По этому адресу нужная строка быстро отыскивается на внешнем носителе. Индексы используются для ускорения выборки данных по запросу и упорядочения результатов запроса, а также иногда для обеспечения уникальности значений (unique index — уникальный индекс). В отличие от простых "настольных" (desktop) систем управления базами данных, где без индексов просто нельзя было обойтись, в реляционных СУБД индексы не играют столь боль-

[2] Вообще-то ограничение существует. Для одной таблицы может быть создано не более 65 536 индексов. Я плохо представляю себе таблицу, для которой кому-то захочется создавать такое количество индексов. Мне также трудно представить и человека, который решит создавать столько индексов. Лучше заняться чем-нибудь действительно полезным, например, написанием книг по базам данных.

шой роли. Здесь основное их назначение — повышение производительности работы системы.

В создаваемой нами учебной базе данных мы не станем использовать индексы.

Первичный ключ (`PRIMARY KEY`) — столбец или несколько столбцов таблицы, значение которых однозначно определяет конкретную строку таблицы. Основным требованием к первичному ключу является его уникальность — в таблице не может быть двух разных строк, имеющих одинаковое значение первичного ключа. Ни один столбец, входящий в состав первичного ключа, не может иметь пустого значения (`NULL`). Таблица может иметь только один первичный ключ. Первичный ключ — один из важнейших элементов в системе реляционных баз данных. Помимо однозначной идентификации конкретной строки в таблице он позволяет в связке с внешними ключами реализовать все возможные отношения между данными в базе данных. Для каждого первичного ключа система автоматически создает индекс, позволяющий ускорить выборку необходимых записей. Ни в коем случае нельзя явно создавать для таблицы индекс, соответствующий по структуре первичному ключу.

Часто бывает сложно выбрать существующие столбцы таблицы в качестве первичного ключа. В таких случаях используется искусственный первичный ключ. В таблицу добавляется числовой столбец, он будет использован в качестве первичного ключа. Значение для него выбирается из генератора (см. далее). Более подробно о первичных ключах мы поговорим позже.

Уникальный (`UNIQUE`) ключ. Во многом схож с первичным ключом. Он также однозначно определяет строку таблицы. Может принимать участие в реализации отношений в базе данных — на него могут ссылаться внешние ключи других таблиц или той же самой таблицы. Кроме того, в таблице может существовать несколько уникальных ключей. Для уникального ключа системой также автоматически создается индекс. Вы не должны явно создавать индекс, имеющий аналогичную структуру. Лично в моей практике работы с базами данных я не припомню случая, когда бы мне приходилось использовать уникальные ключи. У меня есть смутные подозрения, что наличие в таблице уникальных ключей, которые к тому же используются в связке с внешними ключами, говорит, скорее всего, об ошибках при проектировании базы данных. Впрочем, я могу ошибаться.

Внешний ключ (`FOREIGN KEY`) — столбец или несколько столбцов таблицы (дочерней таблицы), которые ссылаются на первичный или уникальный ключ другой или той же самой таблицы (родительской таблицы). Либо внешний ключ весь должен быть пустым (иметь значения `NULL` для всех столбцов, входящих в состав ключа), либо в таблице, на которую ссылается внешний ключ, должна существовать строка с соответствующим значением первичного (или

уникального) ключа. Понятно, что количество столбцов и их типы данных для внешнего ключа и первичного (уникального) ключа, на который ссылается внешний ключ, должны совпадать. Для внешнего ключа система также автоматически создает индекс. Индекс такой же структуры для таблицы вы создавать не должны.

Связка "внешний ключ/первичный (уникальный) ключ" используется для поддержания *ссылочной целостности* (referential integrity) базы данных и для реализации отношений между строками различных таблиц.

Здесь следует отдавать себе отчет, что никаких *структурных* связей между внешними ключами дочерней таблицы и соответствующими им первичными ключами родительской таблицы не существует. Внешний ключ никоим образом не ссылается на первичный с помощью, например, каких-то указателей, как подобные вещи были реализованы в предыдущих поколениях СУБД. Просто существуют индексы и некоторые программные средства (автоматически создаваемые триггеры — см. далее), которые позволяют поддерживать нужные соответствия внешних ключей первичным. Эти вопросы мы рассмотрим позже довольно подробно.

Сама идея реализации отношений таким простым способом изумляет, во-первых, именно своей простотой и, во-вторых, необыкновенной гибкостью, т. е. возможностью добавлять в существующую базу данных новые отношения, практически ничего не меняя в существующих данных.

Генератор (generator) — наиболее простой объект базы данных, используемый для генерации уникального числового значения. Для получения очередного значения из генератора используется функция `GEN_ID`. При первом обращении к функции возвращается единица, при последующих обращениях возвращается значение, обычно на единицу больше предыдущего значения. Используется для получения уникальных значений искусственного первичного ключа. Об использовании искусственных первичных ключей мы поговорим более подробно, когда займемся проектированием наших таблиц.

Хранимая процедура (stored procedure) — программа, обычно выполняющая какие-то действия с данными из базы данных (на самом деле требование работы с данными из базы данных не является обязательным; процедура может выполнять любые действия с любыми доступными ей данными). Хранится в области метаданных базы данных, в системных таблицах. К хранимой процедуре могут обращаться любые программы, работающие с этой базой данных: обычные клиентские программы, хранимые процедуры и триггеры. Допустима также рекурсия — хранимая процедура может обращаться и сама к себе.

Хранимые процедуры выполняются на стороне сервера (серверной машины, где расположена база данных), а не на стороне клиента. В этом проявляется высокая эффективность использования хранимых процедур, когда они обра-

батывают большой объем данных именно на стороне сервера, минимизируя сетевой трафик, т. е. пересылку данных по локальной сети.

Хранимые процедуры могут получать от вызывающей программы параметры и возвращать произвольное количество значений.

Одно из наиболее простых действий, которое можно доверить хранимым процедурам, — получение из генератора значения искусственного первичного ключа. При формировании программой значений столбцов новой строки таблицы, помещаемой в базу данных, выполняется обращение к хранимой процедуре за новым значением первичного ключа. Такие процедуры называются выполняемыми (executable) хранимыми процедурами. Второй вид — хранимые процедуры выбора (selectable), которые позволяют на основании сколь угодно сложных алгоритмов осуществлять выборку данных из произвольных таблиц базы данных.

Триггер (trigger) — как и хранимая процедура является программой, выполняющей действия с данными базы данных. Однако обращение напрямую к триггеру невозможно. Он автоматически вызывается (по-английски trigger fires, т. е. вспыхивает, загорается) при наступлении какого-то события базы данных — удаление записи, помещение новой записи в базу данных, модификация записи. Триггер может вызываться как до наступления такого события, так и сразу после него. Это называется фазой события. Выполнив не слишком сложное умножение двух чисел, получаем шесть различных событий, при которых может срабатывать триггер. Триггер, как и хранимая процедура, выполняется на стороне сервера. Если для его работы требуется обращение к большому количеству данных, то в этом случае, как и при хранимых процедурах, мы получаем большой выигрыш в плане уменьшения сетевого трафика. Триггер может обращаться к любой хранимой процедуре этой базы данных. Триггер не может получать параметров и возвращать какие-либо значения.

Основная прелесть триггеров в том, что программисту не нужно заботиться о выполнении некоторых важных действий в процессе модификации данных в базе данных. Такие действия можно поручить триггеру, который выполнит все необходимое, независимо от того, помнит программист об этих действиях или нет. Например, получение значения искусственного ключа из генератора можно поручить и триггеру, который будет автоматически вызываться перед добавлением новой строки в таблицу. Если программист забыл получить из хранимой процедуры значение такого ключа, то триггер молча исправит такую невнимательность[3].

---

[3] На самом деле мы с вами будем использовать еще более умные и надежные средства работы с искусственными ключами. Но об этом чуть позже, когда мы начнем писать серьезные программы работы с нашей учебной базой данных.

Правда, существует и обратная сторона такой прелести. Приходилось наблюдать случаи, когда в результате внесения изменений в существующую программную систему программист получал совсем не те результаты, которые ожидал увидеть. Часто программисты забывают о существовании триггеров, которые, выполняясь автоматически, вносят непонятные корректировки в данные базы данных.

Для поддержания отношений с помощью связки "внешний ключ/первичный (уникальный) ключ" система автоматически создает системные триггеры. На них мы с вами со временем посмотрим внимательно.

Пользовательские исключения (exception). Для реализации эффективной обработки ошибочных ситуаций можно создавать в конкретной базе данных исключения, задавая в них текст, объясняющий ошибочную ситуацию. В хранимых процедурах и триггерах на основании каких-то проверок данных при выявлении ошибок вы можете выдавать соответствующее исключение, текст которого возвращается вызвавшей программе или программе, инициировавшей вызов триггера. Это весьма удобное средство обработки ошибок. Позже мы с вами создадим несколько исключений в нашей базе данных и используем их по назначению.

События базы данных (event). Из хранимых процедур и триггеров существует возможность отправлять события всем клиентам, подключенным к базе данных и "прослушивающим" конкретные события. Очень интересное средство, позволяющее выполнить, например, синхронизацию состояния текущих наборов данных у всех клиентов при изменении таблиц одним из клиентов.

Язык PSQL. Мы говорили, что в составе SQL выделяются язык описания данных (DDL) и язык манипулирования данными (DML). Существует еще и так называемое расширение языка SQL — процедурный SQL, PSQL. Он используется для написания хранимых процедур и триггеров. Основной особенностью языков DDL и DML является их *декларативность* — на этих языках только описывается, *что* должно быть сделано, но ни при каких обстоятельствах не указывается, *как* это должно быть сделано. Язык PSQL добавляет в реляционные базы данных *императивность*, функциональность — он позволяет описывать именно то, *как* должны быть выполнены действия с данными. В этом языке представлены (хотя и довольно скромно) все основные конструкции, характерные для языков программирования — объявление внутренних, локальных переменных, входных и выходных параметров, присваивание, вычисления, проверка условий, выполнение различных циклов и т. д. Кроме этого, для триггеров допустимо использование так называемых контекстных переменных, позволяющих обращаться к значениям любых столбцов записи до их изменения пользователем (вариант `OLD`) и после изменения (вариант `NEW`).

Представление (view) — динамический результат одной или нескольких реляционных операций над базовыми таблицами с целью создания новой таблицы. Представление является *виртуальной* таблицей, которая в базе данных не хранится, но создается по требованию отдельного пользователя. То, что сейчас было написано про представление, правильно, но, на мой взгляд, для нормального человека (не работавшего с представлениями) не очень понятно. Мы займемся представлениями ближе к концу этой книги. Рассмотрим примеры.

Транзакция (transaction) — механизм базы данных, позволяющий либо подтвердить группу выполняемых операций с базой данных, либо отменить все действия этой группы операций. Любые действия с базой данных — как с данными, так и с метаданными — выполняются в контексте (иными словами — под управлением) какой-либо транзакции. Транзакция стартует перед началом любого действия с базой данных — изменения данных базы данных или выборки существующих данных. По завершении группы операций все действия можно или подтвердить (оператор `COMMIT`), или отменить (оператор `ROLLBACK`). Рассмотрению транзакций мы посвятим отдельную главу.

Функции, определенные пользователем (User Defined Functions, UDF) — функции, написанные на любом языке программирования и хранящиеся вне базы данных, но описанные в базе. Могут использоваться для расширения возможностей языка SQL. В стандартных комплектах поставки серверов базы данных содержится довольно большое количество UDF. В основном это математические функции.

Предметная область — реальная область человеческой деятельности, для решения задач которой мы и создаем наши базы данных и пишем клиентские программы графического интерфейса. Примерами предметной области могут служить система решения задач бухгалтерского учета на предприятии, библиографическая поисковая система, очень сложная система управления ресурсами предприятия (ERP) и многое другое. Именно для решения задач конкретных предметных областей и создаются базы данных и соответствующие клиентские приложения.

Клиентское приложение — программа, использующая данные базы данных для решения задач предметной области. Имеет графический интерфейс, удобный для пользователя. Выполняется на стороне клиента — клиентском компьютере в локальной вычислительной сети. На самом деле нам совершенно все равно, где выполняется клиентское приложение — на сервере или на клиентском компьютере. Необходимые согласования выполнит система управления базами данных.

1.2. Реализация отношений в реляционной модели

Существует три вида отношений в любых базах данных: "один-к-одному", "один-ко-многим" и "многие-ко-многим". Рассмотрим реализацию этих отношений в реляционной модели. При любом отношении все вопросы решаются связкой "внешний ключ/первичный (уникальный) ключ".

1.2.1. Отношение "один-к-одному"

Вообще говоря, если между двумя таблицами базы данных появляется такое отношение, то лучше объединить эти таблицы в одну. Основной причиной использования этого отношения является экономия памяти и увеличение скорости выполнения запросов.

Такое отношение используется в том случае, если связь не является обязательной.

Рассмотрим пример, к сожалению, довольно глупый и надуманный. Не могу припомнить, чтобы в реальной жизни у меня была потребность в реализации отношения "один-к-одному".

Правда, в нашей учебной базе данных будет одно такое отношение внутри одной и той же таблицы. Случай довольно специфический. Это связь между супругами в таблице людей. Вообще, связь, которую мы используем, обычно называют циклической. Далеко не все модели данных ее поддерживают. Реляционные базы данных поддерживают.

Итак, пусть есть две таблицы. Первая содержит некоторые сведения о человеке. Вторая — сведения о его автомобиле: марка, год выпуска и т. д. Считается, что человек может иметь не более одного автомобиля.

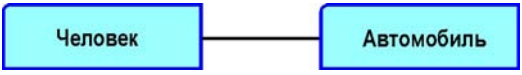


Рис. 1.1. Отношение "один-к-одному"

Здесь связь между человеком и автомобилем не является обязательной. Очень много людей не имеют личного транспорта.

Можно было бы сведения об автомобиле поместить в первую таблицу, однако, поскольку далеко не каждый человек имеет машину, это привело бы к большому количеству пустых полей в таблице и некоторому перерасходу внешней памяти. Правда, перерасход памяти не будет слишком большим, поскольку наши серверы баз данных хранят пустые значения (как и все другие) весьма компактно.

Для реализации отношения "один-к-одному" в данном случае сделаем первичный ключ во второй таблице таким же, как и в первой. Этот первичный ключ также будет и внешним ключом, ссылающимся на первичный ключ первой таблицы.

1.2.2. Отношение "один-ко-многим"

Отношение "один-ко-многим" реализуется связкой "внешний ключ/первичный (уникальный) ключ" (рис. 1.2).

Рис. 1.2. Отношение "один-ко-многим"

Рассмотрим вполне реальный пример. Пусть у нас есть таблица, содержащая список стран (табл. 1.1). Вторая таблица содержит список регионов каждой страны (республики, края, области, в некоторых странах — штаты, графства, земли) (табл. 1.2). Первичным ключом будет некоторый код страны. Во второй таблице нужно ввести поле внешнего ключа (код страны), которое будет ссылаться на первичный ключ первой таблицы.

Таблица 1.1. *Страны*

Краткое название	Полное название	Код страны
РОССИЯ	Российская Федерация	558
США	Соединенные Штаты Америки	604

"Код страны" здесь является первичным ключом.

Таблица 1.2. *Регионы*

Код страны	Код региона	Центр региона	Регион
558	32	Брянск	Брянская область
558	25	Владивосток	Приморский край
558	15	Владикавказ	Республика Северная Осетия
558	33	Владимир	Владимирская область
558	34	Волгоград	Волгоградская область
558	35	Вологда	Вологодская область