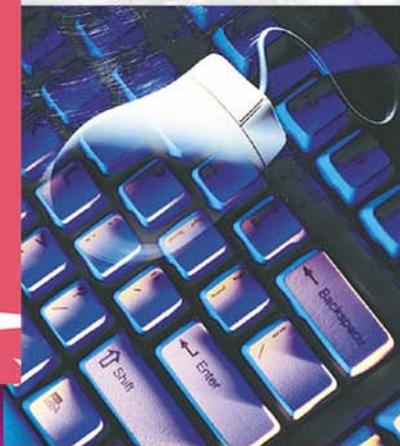


РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ В DELPHI



СУБД: MS SQL SERVER 2000,
Firebird И InterBase

РАБОТА С ГЕНЕРАТОРАМИ
ОТЧЕТОВ: QReport,
RaveReports, FastReport

АВТОМАТИЗАЦИЯ
МОДЕЛИРОВАНИЯ,
РАЗРАБОТКИ
И ПОДДЕРЖКИ БАЗЫ
ДАННЫХ В СРЕДЕ ERwin

ОСОБЕННОСТИ
РАЗРАБОТКИ
ПРИЛОЖЕНИЙ В DELPHI 7
И DELPHI 2005

ОБЗОР ДОПОЛНИТЕЛЬНЫХ
КОМПОНЕНТОВ

PRO

ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

+ ВИДЕОКУРС 

Андрей Шкрыль

**РАЗРАБОТКА
КЛИЕНТ-СЕРВЕРНЫХ
ПРИЛОЖЕНИЙ В
DELPHI**

Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.06
ББК 32.973.26-018.2
Ш66

Шкрыль А. А.

Ш66 Разработка клиент-серверных приложений в Delphi. —
СПб.: БХВ-Петербург, 2006. — 480 с.: ил.

ISBN 5-94157-761-3

Рассмотрены практические вопросы по разработке клиент-серверных приложений в среде Delphi 7 и Delphi 2005 с использованием СУБД MS SQL Server 2000, InterBase и Firebird. Приведена информация о теории построения реляционных баз данных и языке SQL. Освещены вопросы эксплуатации и администрирования СУБД. Большое внимание уделено различным генераторам отчетов QReport, RaveReports и FastReport. Описано использование системы проектирования, разработки и поддержки баз данных ERwin. Рассмотрены дополнительные компоненты для разработки клиент-серверных приложений, а также даны ответы на часто задаваемые вопросы. Материал излагается по принципу "от простого к сложному" и сопровождается иллюстрациями, практическими примерами и видеороликами.

На компакт-диске содержатся исходные коды, видеоролики по созданию приложений, а также дополнительные компоненты и инструменты для работы с БД.

Для разработчиков клиент-серверных приложений

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Руслан Абдрахманов</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн обложки	<i>Игоря Цырульникова</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 19.10.05.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 38,7.

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 5-94157-761-3

© Шкрыль А. А., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Благодарности	1
Введение	3
Глава 1. Введение в реляционные базы данных	5
1.1. Вступление.....	5
1.2. База данных и ее составные части	5
1.3. Создание связей между таблицами	10
1.4. Служебные элементы базы данных	15
1.5. Основные принципы разработки базы данных.....	18
1.6. Транзакции	19
1.7. Типы данных	19
1.8. Нормализация	20
1.9. Системы управления базами данных (СУБД)	24
1.10. Целостность данных	25
Глава 2. Язык SQL.....	27
2.1. Вступление.....	27
2.2. Принцип построения запросов.....	28
2.3. Команды управления данными (DML)	28
2.4. Команды определения данных (DDL)	46
2.5. Несколько полезных запросов	48
Глава 3. InterBase.....	51
3.1. Вступление.....	51
3.2. Повторение материала.....	51
3.3. Постановка задачи	52
3.4. Установка InterBase.....	52
3.5. Создание базы данных в InterBase	61
3.6. Резервное копирование и восстановление базы данных.....	74
3.7. Разработка приложения "клиент-сервер" в Delphi.....	78
Глава 4. Microsoft SQL Server 2000	107
4.1. Вступление.....	107
4.2. Установка	107
4.3. Постановка задачи	115

4.4. Создание базы данных в MS SQL Server.....	116
4.5. Целостность данных.....	122
4.6. Визуализация структуры базы данных.....	129
4.7. Работа с хранимыми процедурами.....	133
4.8. Резервное копирование и восстановление базы данных.....	139
4.9. Разработка приложения "клиент-сервер" в Delphi.....	145
Глава 5. FireBird.....	179
5.1. Вступление.....	179
5.2. Установка FireBird.....	179
5.3. Постановка задачи.....	188
5.4. Разработка базы данных в FireBird.....	189
5.5. Работа в IBExpert.....	189
5.6. Работа с хранимыми процедурами.....	210
5.7. Резервное копирование и восстановление базы данных.....	222
5.8. Разработка приложения "клиент-сервер" в Delphi.....	224
Глава 6. QReport.....	267
6.1. Вступление.....	267
6.2. Установка QReport в Delphi 7.....	267
6.3. Установка QReport в Delphi 2005.....	269
6.4. Использование QReport.....	273
6.5. Принцип построения отчета.....	276
6.6. Усложняем пример.....	278
6.7. Фильтры.....	283
6.8. Подстановочные (Lookup) поля.....	285
Глава 7. RaveReports.....	291
7.1. Вступление.....	291
7.2. Установка RaveReports.....	291
7.3. Использование RaveReports.....	292
7.4. Принцип построения отчета.....	301
7.5. Усложняем пример.....	302
7.6. Подстановочные (Lookup) поля.....	308
7.7. Многостраничный отчет.....	310
7.8. Фильтры.....	317
Глава 8. FastReport.....	321
8.1. Вступление.....	321
8.2. Установка FastReport.....	321
8.3. Использование FastReport.....	324
8.4. Принцип построения отчета.....	329
8.5. Усложняем пример.....	334
8.6. Фильтры.....	345

Глава 9. XML	347
9.1. Вступление.....	347
9.2. Структура XML-файла	348
9.3. Практика.....	351
9.4. Полезный пример	358
Глава 10. Проектирование базы данных в среде ERwin	363
10.1. Вступление.....	363
10.2. Основные понятия	364
10.3. Установка ERwin.....	368
10.4. Анализ предметной области.....	369
10.5. От объектов к сущностям	378
10.6. Работа в ERwin.....	382
10.7. Определение связей в модели.....	417
10.8. Индексы	425
10.9. Subject Areas.....	426
10.10. Stored Display	430
10.11. ERwin — MS SQL Server 2000.....	431
10.12. Печать модели средствами ERwin.....	436
Глава 11. Обзор дополнительных компонентов	439
11.1. Ehlib	439
11.2. scExcelExport.....	443
11.3. Trend v.2.03.	446
11.4. DBGridBelang v.1.0.	449
11.5. JanH DBGrid v.1.0	450
11.6. Little Report v.1.0	451
Глава 12. Ответы на часто задаваемые вопросы	453
Заключение	465
Приложение. Описание компакт-диска	466
Список используемой литературы	469
Предметный указатель	470

Благодарности

Хочу сказать спасибо своим родителям, благодаря которым я появился на свет, которые привили мне любовь к получению новых знаний и упорство в достижении целей. Брату Антону и сестре Ольге за то, что позволяли пользоваться своим компьютером для написания отдельных глав книги.

Выражаю большую признательность своей жене Наташе, самой умной, доброй и красивой женщине на свете, за ее поддержку и помощь в поиске новых идей. А также благодарю ее за терпение.

Огромная благодарность Чубаркину Артуру Владимировичу, который помог мне не затеряться в этой жизни, благодаря которому я открыл в себе силы написать эту книгу.

Особая благодарность Фленову Михаилу, при содействии которого я совершенствовался как специалист, который помогал мне своими советами и делился опытом. Спасибо тебе, Михаил, без тебя даже не зародилась бы идея написания этой книги.

Хочу поблагодарить отделение Сбербанка России, Автозаводское отделение № 8213 города Тольятти. Особенно отдел информатизации и автоматизации банковских работ, где начиналась моя практика и где я впервые познакомился с MS SQL Server 2000.

Признательность Плахову Сергею, благодаря которому я узнал, про IVExpert и возможности написания серьезных баз данных при помощи FireBird.

Благодарность Кропотову Максиму и Обенко Николаю за их моральную поддержку в процессе моей работы.

Особая благодарность Шишигину Игорю Владимировичу — специалисту с большой буквы — за его оперативные ответы на мои вопросы, профессиональное ведение проекта по написанию книги и просто человеческую поддержку. Также хочется поблагодарить Абдрахманова Руслана за помощь на завершающем этапе работы по созданию книги и всех сотрудников издательства "БХВ-Петербург", которые приняли участие в данном проекте.

Введение

Идея написания данной книги зародилась внезапно. Я давно увлекаюсь темой разработки базы данных, и меня всегда удивлял тот факт, что по данной теме очень мало качественной литературы, где бы доступным и понятным языком объяснялись основы и давались конкретные и, что очень важно, рабочие примеры.

Очень часто в книгах по Delphi в разделе "Работа с базами данных" даются типовые примеры работы с BDE — технологией, которая в настоящий момент уже не поддерживается Borland, — и работы с InterBase. Я не понимаю, зачем продолжать выпускать такие книги, когда в сети очень много информации по этим темам, и к тому же она бесплатная. А как же все остальное, ведь базы данных это не только Paradox и InterBase.

Совершенно случайно, путешествуя по просторам Сети, я нашел информацию о книге Михаила Фленова "Программирование в Delphi глазами хакера", почитал отзывы, и решил ее заказать. И именно с этой книги началась моя карьера программиста и появилась вера в себя как специалиста. Первой моей мыслью было: "Наконец-то появился автор, который объясняет вещи простым и доступным языком". Из этой же книги я узнал, что у Михаила есть собственный сайт <http://www.vr-online.ru>, на котором он выкладывает свои статьи, посвященные компьютерной тематике. И у меня зародилась идея попробовать свои силы и написать собственную статью, так как к тому времени я уже неплохо владел InterBase и MS SQL Server. Первая моя статьей была посвящена работе с InterBase. Причем, что очень меня удивило, статью скачали очень большое количество людей, она стала даже более популярна, чем статьи, которые появились раньше, чем моя. Это натолкнуло меня на мысль, что я не один блуждаю в поисках информации. И я решил продолжить и написал вторую статью, посвященную работе с MS SQL Server.

Вот таким образом, можно сказать, появилась идея написания этой книги.

Я отдаю себе отчет, что многих вопросов не удалось коснуться. Но в книге нельзя рассмотреть абсолютно все, а тем более в книге, охватывающей широкий круг тем, таких как работа с различными СУБД, построение отчетов, проектирование баз данных.

После прочтения данной книги у читателя могут возникнуть дополнительные вопросы, посвященные темам, рассмотренным в ней. Проявление интереса к этим темам, поиск ответов на вопросы и новой информации в рамках этих тем создаст отличные условия для развития читателя как

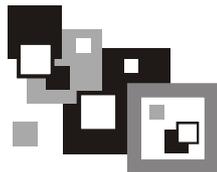
специалиста в области программирования и разработки баз данных. Приобретение дополнительных знаний — необходимое условие для перехода на новый уровень развития с возможностью в дальнейшем самостоятельно отвечать на собственные вопросы и на вопросы других, начинающих.

Хорошая книга в моем понимании — это книга, после прочтения которой читатель понял что-то новое и смог применить эти знания на практике. К тому же, если у читателя возникли конкретные вопросы — это отлично. Значит, он будет развиваться в данном направлении, и время на прочтение книги было потрачено не зря. Часто читателям приходится, как настоящим археологам, сидеть над каждой строчкой кода и сравнивать то, что написано в книге, с теми примерами, поставляемыми на диске к книге, искать ошибки. А все дело оказывается в том, что упущены какие-то мелочи, несущественные для автора, но для читателя имеющие огромное значение. Предусматривая это, я решил для каждого примера и для каждой версии Delphi, рассмотренных в книге (Delphi 7 и Delphi 2005), записать видеоролик, чтобы читатель мог просмотреть, если ему вдруг непонятен какой-то момент, а ни в книге, ни в исходных текстах нет объяснений.

Вообще меня поражает тот факт, что в наш век высоких технологий, когда мультимедиа давно используется во всех сферах деятельности, к книгам это до сих пор не применяется. Я считаю, что большое преимущество этой книги в том, что есть наглядные материалы (видеоролики), а также существует форум, куда читатель может обратиться, и ему обязательно помогут. Таким образом, диалог автора с читателем после прочтения книги не заканчивается, он может продолжиться, если у читателя появится желание сделать это во Всемирной сети, где я и остальные участники форума, а также члены команды VR-TEAM, с радостью поделятся своими знаниями и опытом.

Приятного прочтения.

Глава 1



Введение в реляционные базы данных

1.1. Вступление

Реляционные базы данных имеют мощный теоретический фундамент, основанный на математической теории отношений. Появление теории реляционных баз данных дало толчок к разработке языка запросов SQL.

В реляционной модели объекты реального мира и взаимосвязи между ними представляются с помощью совокупности связанных между собой таблиц или отношений.

Свое название реляционная теория заимствовала из математики. Основатель реляционной теории — доктор Э. Ф. Кодд.

1.2. База данных и ее составные части

Базы данных используются в тех случаях, когда возникает необходимость манипулировать большими объемами данных. У базы данных есть имя. Она состоит из таблиц, в которых хранятся данные, и из дополнительных элементов, поддерживающих работоспособность базы данных, а также доступность содержащейся в ней информации (рис. 1.1).

Рассмотрим более подробно элементы базы данных.

- *Таблицы* — это набор обычных двумерных таблиц, знакомых всем из школьного курса. У таблицы есть два параметра — *строки* и *столбцы*. Строки по реляционной теории называются *кортежами*, но, как правило, в настоящее время используют более распространенный термин — *запись*. Столбцы называются *атрибутами*. У каждого поля (атрибута) есть свой *тип данных*. *Тип данных* — набор значений, который может принимать атрибут. Например, числовой тип позволяет использовать только числа; логический — да, нет; текстовый — символы и т. д.

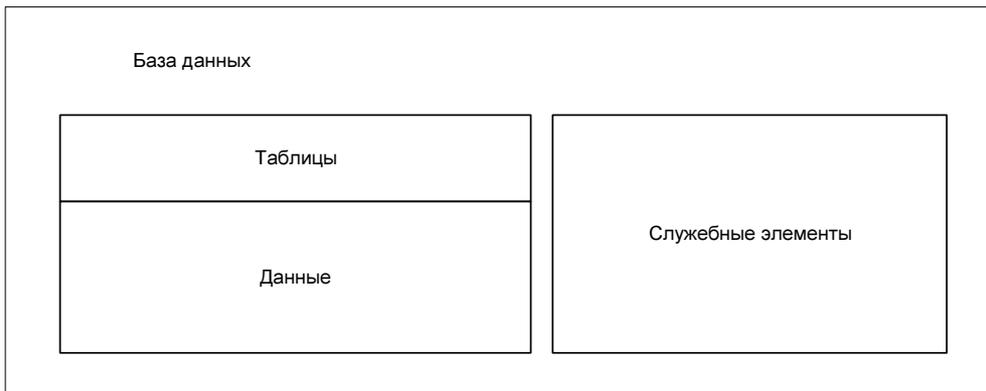


Рис. 1.1. Структура базы данных

У каждой таблицы есть имя, также имя есть и у каждого атрибута таблицы. Таким образом, можно легко обратиться к нужному элементу базы данных, как показано на рис. 1.2 и 1.3.

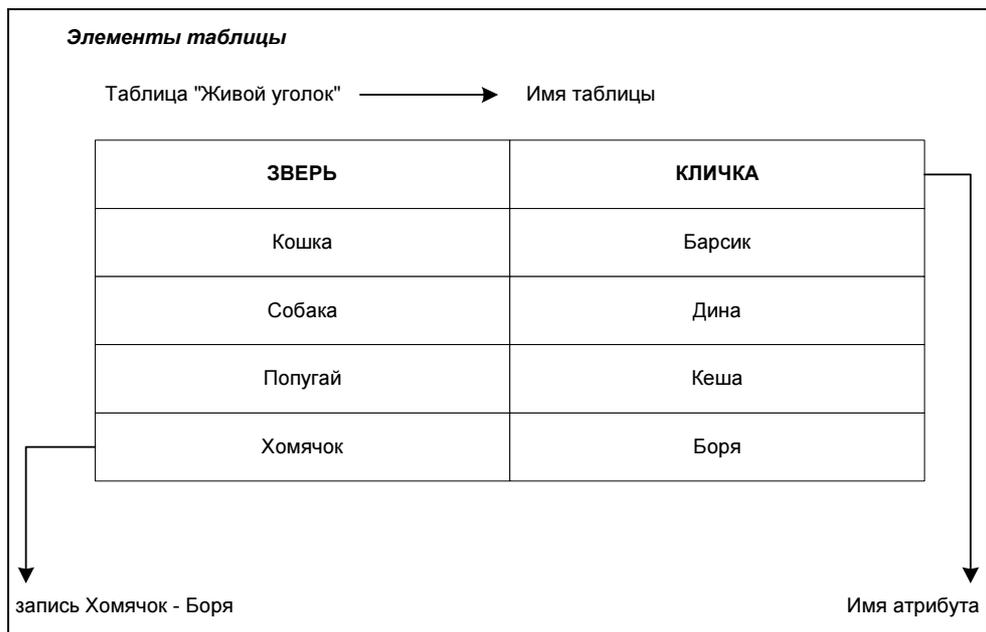


Рис. 1.2. Элементы таблицы

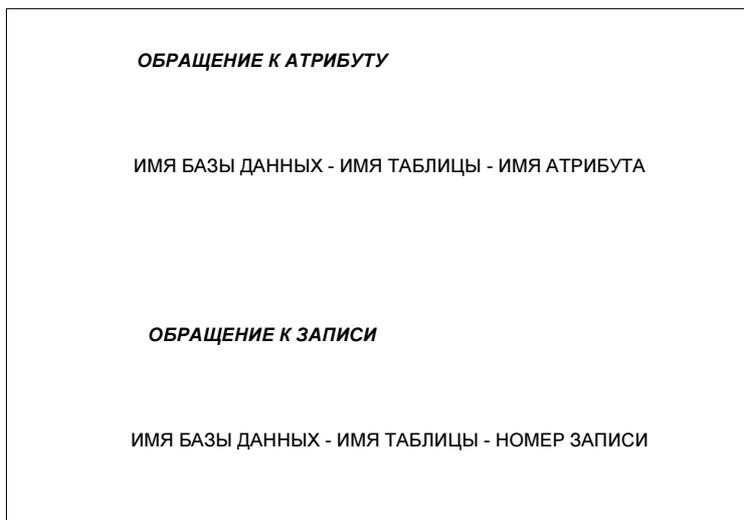


Рис. 1.3. Обращение к элементам таблицы

Вывод

Таблицы и атрибуты одной и той же таблицы не должны иметь одинаковые имена, иначе возникнет неоднозначность.

- *Данные* — это та информация, которую внес пользователь и которая хранится в таблицах. При проектировании таблиц важно обеспечить непротиворечивость хранящейся в ней информации. Рассмотрим это на примере (табл. 1.1).

Таблица 1.1. Таблица "Персонал"

Имя	Отчество	Фамилия
Вася	Васильевич	Васечкин
Вася	Васильевич	Васечкин
Миша	Андреевич	Кузькин

Существует некая таблица "Персонал", в ней содержатся имена людей, работающих в небольшой фирме. Предположим, что в этой фирме работают отец и сын с одинаковыми именами. Как определить, глядя на эту таблицу, какая запись кому соответствует?

Для решения этой проблемы в реляционной теории введено понятие *первичного ключа* — атрибута или комбинации атрибутов, являющейся уникальной

для всех записей таблицы. Если ключ содержит только одно поле, то это поле называется *ключевым*. Предназначением ключевого поля для данных в табл. 1.2 является обеспечение уникальности каждой записи.

Таблица 1.2. Таблица "Персонал" с ключевым полем

Ключевое поле	Имя	Отчество	Фамилия
1	Вася	Васильевич	Васечкин
2	Вася	Васильевич	Васечкин
3	Миша	Андреевич	Кузькин

В данном примере информацию уже можно различить, так как было добавлено дополнительное поле. Например, можно сказать, что запись с номером 1 соответствует отцу, а запись с номером 2 — сыну.

Таблица 1.3. Таблица "Персонал" с новыми записями

Ключевое поле	Имя	Отчество	Фамилия
1	Вася	Васильевич	Васечкин
2	Вася	Васильевич	Васечкин
3	Миша	Андреевич	Кузькин
4	Андрей	Сергеевич	Иванов

Когда данные табл. 1.3 будут отображаться пользователю, ему будет намного удобней, если данные предварительно будут упорядочены, например, по имени сотрудника.

В реляционной теории существует понятие *индексов*. Индекс является промежуточным звеном между пользователем и таблицей и обеспечивает более быстрый доступ к данным. Один из самых распространенных вариантов использования индексов при разработке клиентского приложения в Delphi — это упорядочивание информации, то есть для того поля, значения которого будут упорядочиваться (производиться сортировка), создается индекс (рис. 1.4).

Индексы могут быть *первичными* и *вторичными*.

□ Первичный индекс — это ключевое поле, оно автоматически индексируется. Первичный индекс может быть:

- простым — первичный индекс состоит из одного атрибута (ключевое поле — яркий пример простого первичного индекса);

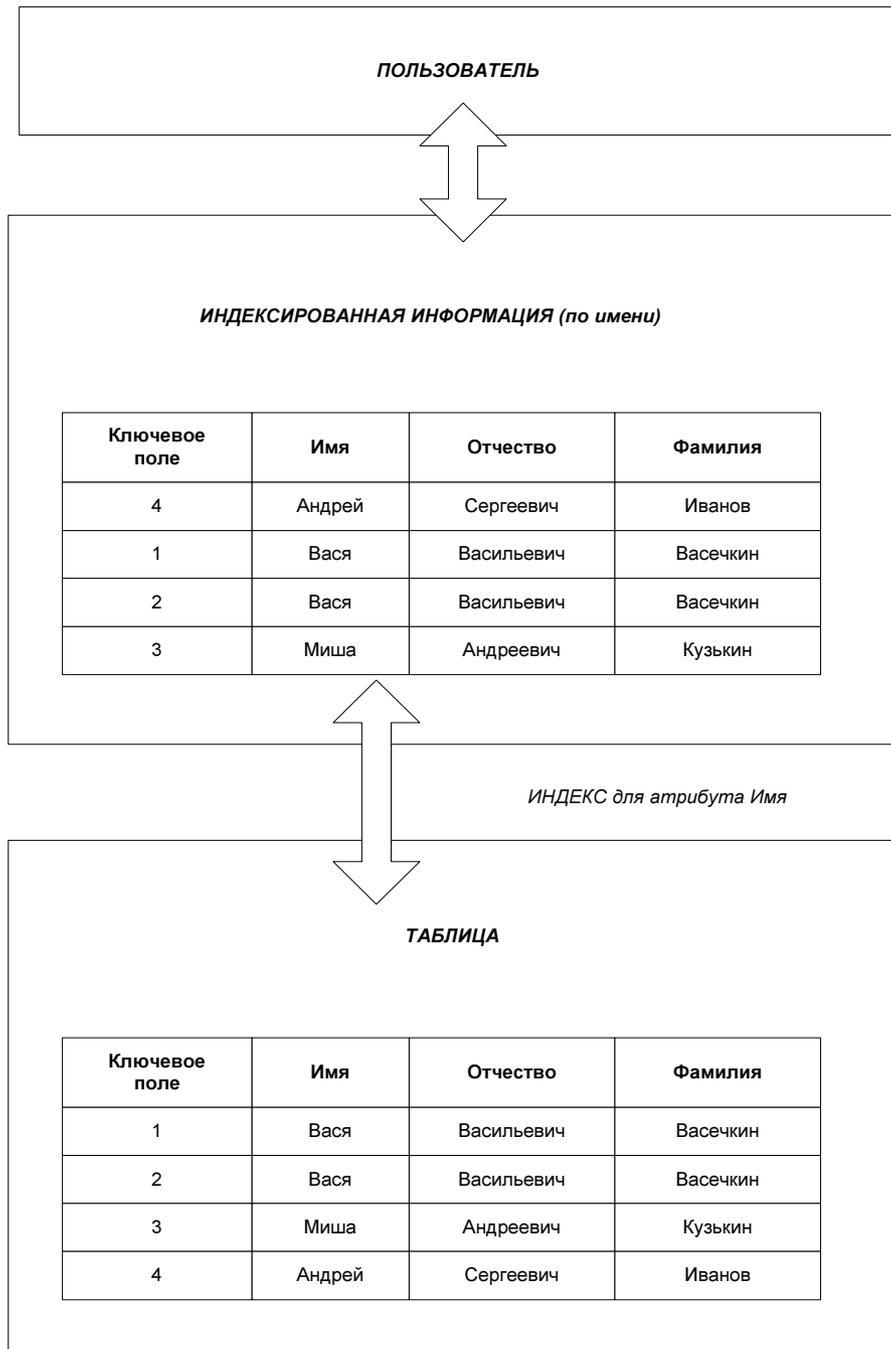


Рис. 1.4. Принцип работы индекса при использовании его для сортировки данных

- составным — первичный индекс состоит из нескольких атрибутов (можно создать составной первичный индекс, состоящий из атрибутов Имя — Отчество — Фамилия).
- Вторичный индекс — это любое неключевое поле (набор полей), для которого создан индекс. Вторичный индекс, так же как и поле, которому он принадлежит, имеет имя.

1.3. Создание связей между таблицами

Уже упоминалось о том, что база данных это набор таблиц. Так вот, сами по себе таблицы представляют небольшой интерес. Вся мощь использования баз данных можно оценить, если извлекать информацию из нескольких таблиц одновременно. Как правило, для этого таблицы предварительно *связывают*, хотя это не является обязательным условием.

При организации связи одна из таблиц будет главной, а другая — второстепенной. Для того чтобы связать две таблицы, используется ключевое поле (его еще называют ключом) и индекс.

Рассмотрим таблицу "Сотрудники" (табл. 1.4) и таблицу "Зарплата" (табл. 1.5).

Таблица 1.4. Таблица "Сотрудники"

Ключевое поле	Имя	Отчество	Фамилия
1	Вася	Васильевич	Васечкин
2	Вася	Васильевич	Васечкин
3	Миша	Андреевич	Кузькин
4	Андрей	Сергеевич	Иванов

Таблица 1.5. Таблица "Зарплата"

Ключевое поле	Дата получения	Сумма	Кто получил (код сотрудника)
1	10.10.2005	10 000	2
2	13.10.2005	5000	1
3	15.10.2005	7000	2
4	11.10.2005	18 000	3

Зарплата не может существовать сама по себе, она всегда выплачивается кому-то, а ситуация, когда сотрудник не получает зарплату, вполне реальная.

На основании этого можно сделать вывод, что таблица "Сотрудники" — главная, а таблица "Зарплата" — второстепенная.

Для организации связи в таблицу "Зарплата" добавляется атрибут "Кто получил", который содержит код сотрудника, полученный из атрибута "Ключевое поле" таблицы "Сотрудники". Таким образом, можно сказать, что некий сотрудник Вася Васильевич Васечкин с кодом 2 получал зарплату дважды, а сотрудник с кодом 4 — Андрей Сергеевич Иванов — не получил еще ни разу (рис. 1.5).

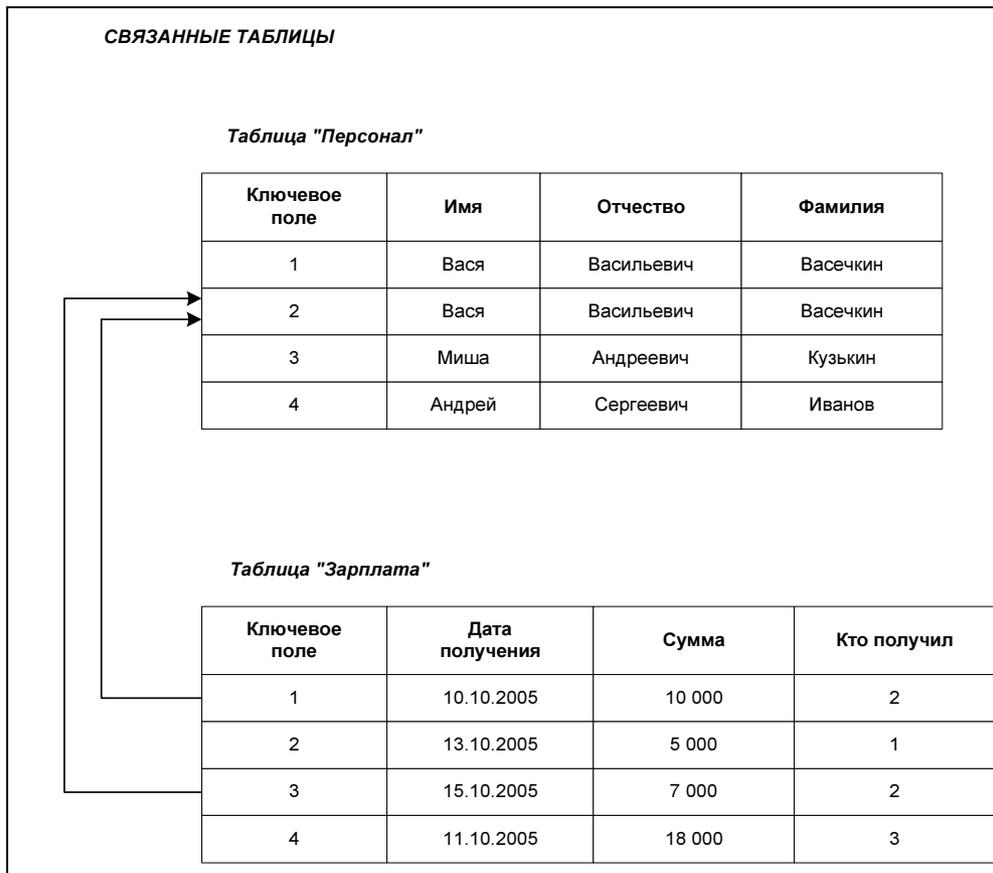


Рис. 1.5. Связанные таблицы

Так как таблицы отражают объекты реального мира, а, как известно, объекты могут взаимодействовать друг с другом, то можно сказать, что между таблицами есть *отношения* (связь).

В реляционной теории существует три типа отношений (связей):

- Один-к-одному — наиболее редко встречающийся тип, когда одной записи в главной таблице соответствует одна запись в подчиненной (рис. 1.6).

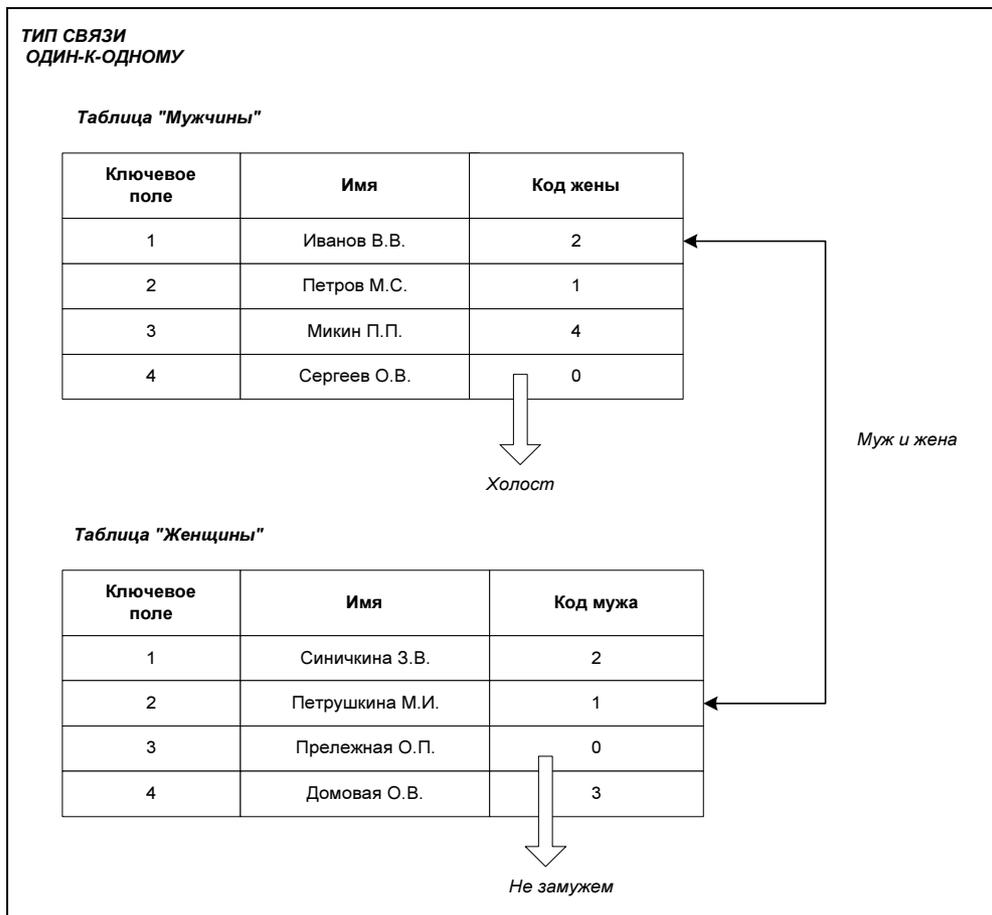


Рис. 1.6. Тип связи один-к-одному

В России многоженство запрещено, поэтому у одного мужчины может быть в один момент времени только одна жена, также и у женщины в один момент может быть только один муж. Связь между таблицами реализуется добавлением в обе таблицы по дополнительному атрибуту. В таблицу "Мужчины" — атрибута "Код жены", в таблицу "Женщины" — атрибута "Код мужа". Также данный тип связи можно реализовать, доба-

вив дополнительную таблицу "Отношения", в которой будут присутствовать два атрибута: "Код мужчины" и "Код женщины".

- Один-ко-многим — наиболее распространенный тип связи, когда одной записи в главной таблице соответствуют одна или несколько записей во второстепенной таблице (рис. 1.7).

**ТИП СВЯЗИ
ОДИН-КО-МНОГИМ**

Таблица "Люди"

Ключевое поле	Имя	Код группы крови
1	Иванов В.В.	2
2	Петров М.С.	1
3	Микин П.П.	2
4	Сергеев О.В.	3

*У Иванова В.В. и Микина П.П.
одинаковая группа крови*

Таблица "Группа крови"

Ключевое поле	Группа крови
1	Первая отрицательная
2	Вторая отрицательная
3	Вторая положительная
4	Третья положительная

Рис. 1.7. Тип связи один-ко-многим

У человека может быть только одна группа крови, но одна группа крови может одновременно принадлежать множеству людей. Для организации данного типа связи в таблицу "Люди" добавляется дополнительный атрибут "Код группы крови", содержащий ссылку на запись в таблице "Группа крови".

- Многие-ко-многим — тип связи, когда множеству записей в главной таблице соответствует множество записей из второстепенной таблицы. Этот тип связи самый сложный, и для его реализации вводят дополнительную таблицу. Для примера рассмотрим отношения, складывающиеся между пациентом и врачом в больнице (рис. 1.8).

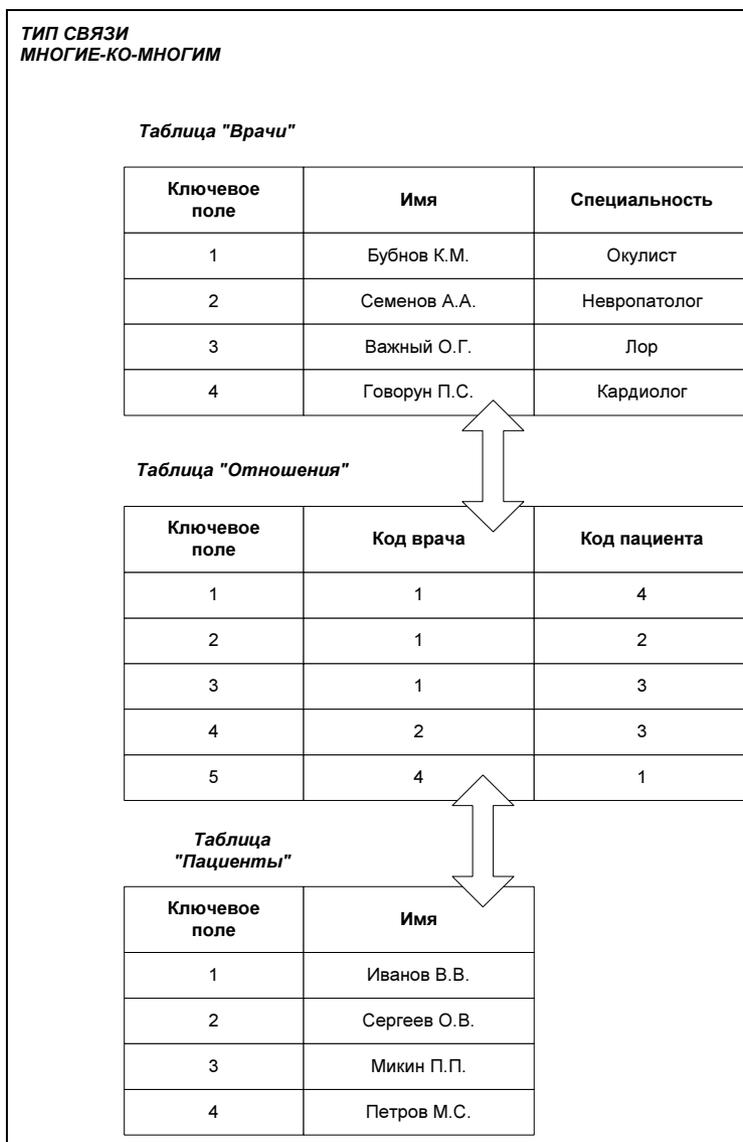


Рис. 1.8. Тип связи многие-ко-многим

Один пациент может записаться на прием к нескольким врачам, также аналогичная ситуация и с врачами. К одному врачу может быть записано множество пациентов.

Таким образом, получается, что множество пациентов может быть записано к множеству врачей, или у множества врачей может быть множество пациентов.

Для того чтобы реализовать эти отношения, нужно ввести дополнительную таблицу "Отношения", в которой будут прописываться отношения между пациентом и врачом. Например, по поводу рис. 1.8 можно сказать следующее:

- к окулисту записано трое пациентов: Сергеев О. В., Петров М. С., Микин П. П.;
- к невропатологу записан один пациент: Микин П. П., записанный также и к окулисту;
- к кардиологу записан один пациент: Иванов В. В.;
- к лору не записан никто.

1.4. Служебные элементы базы данных

Теперь рассмотрим *служебные элементы* базы данных (у всех этих элементов, так же как у таблиц и индексов, есть имя).

- Представление — это виртуальная таблица, сама по себе не хранит информацию, а является результатом выполнения некоторого запроса (о запросах мы поговорим во *главе 2*). В основном используется для обеспечения безопасности (пользователю показывается только то, что нужно) и для формирования отчетов (рис. 1.9).
- Хранимая процедура — этот объект представлен набором команд (об этом более подробно в следующей главе). Хранимая процедура не содержит никакой информации из базы данных, вместо этого она содержит команды о том, что нужно сделать с данными (добавить, удалить, произвести обработку) или в каком виде представить их пользователю (рис. 1.10).
- Триггер — это та же самая процедура, только выполнение ее происходит автоматически при выполнении некоего события, связанного с таблицей. Например, при использовании СУБД InterBase (определение понятия СУБД будет дано ниже), когда происходит добавление записи в таблицу, возникнет событие `Before_Insert`.

По аналогии с триггером работают продавцы в магазинах. Они продадут вам сигареты и спиртные напитки, если вам больше 18 лет, в противном случае в продаже вам будет отказано. В данном случае в роли события выступает желание покупателя сделать покупку, в роли триггера — продавец.

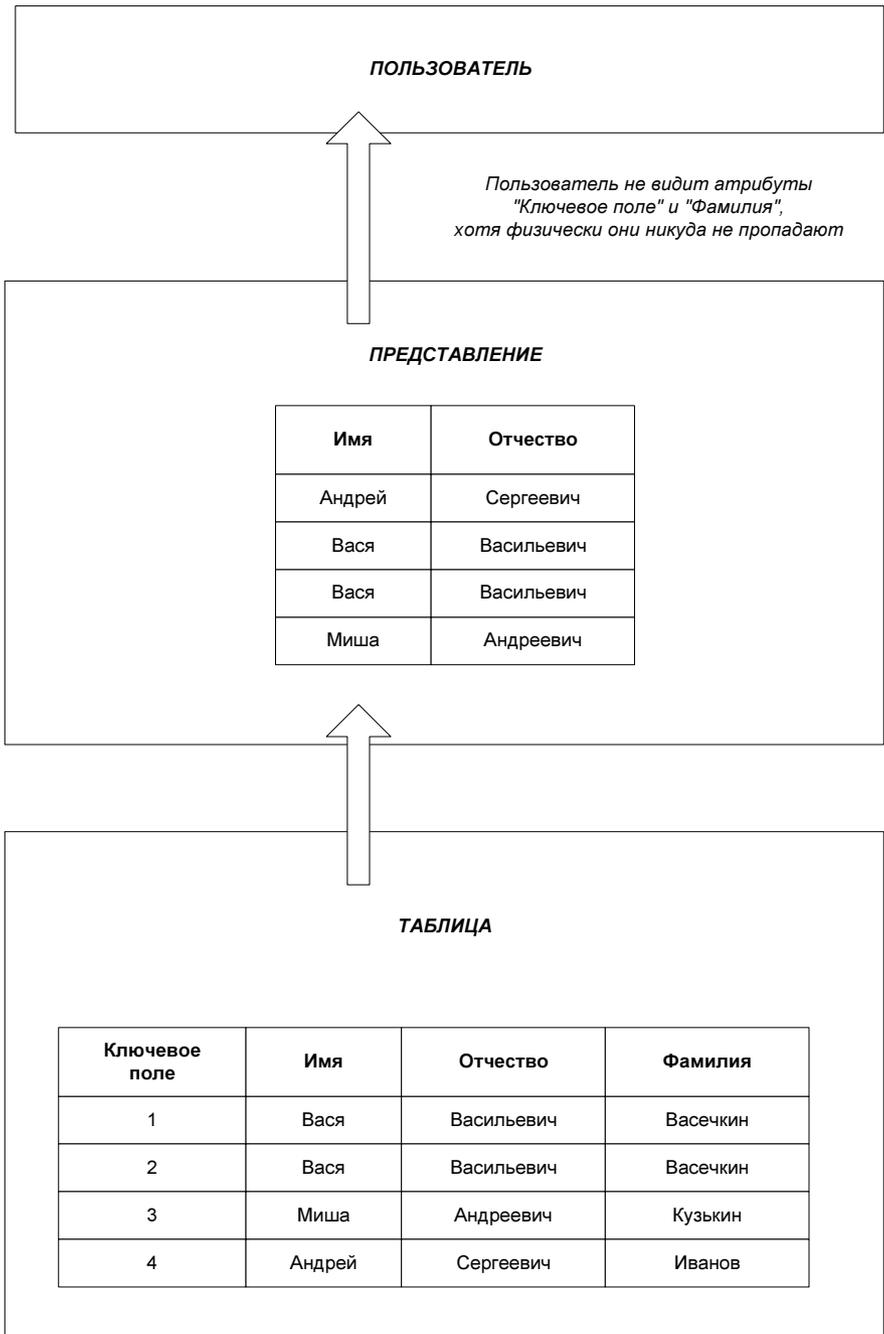


Рис. 1.9. Принцип работы представления

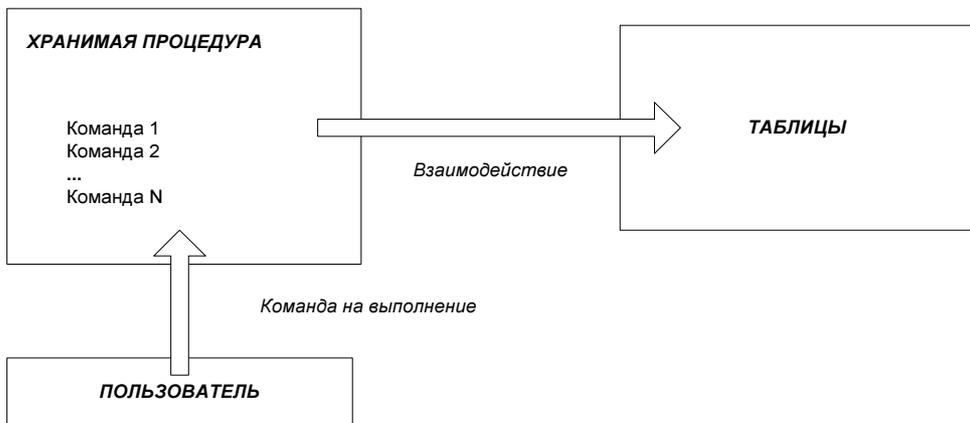


Рис. 1.10. Принцип работы хранимой процедуры

Опишем эту ситуацию в виде в виде *алгоритма*.

1. Покупатель хочет совершить покупку сигарет.
2. Возникает событие — покупка.
3. Управление передается продавцу (триггеру), который смотрит на покупателя и пытается определить, исполнилось ли ему 18 лет или нет.
4. Если исполнилось, то покупка осуществляется.
5. Если покупатель слишком молод, то продавец (триггер) ничего не продает.

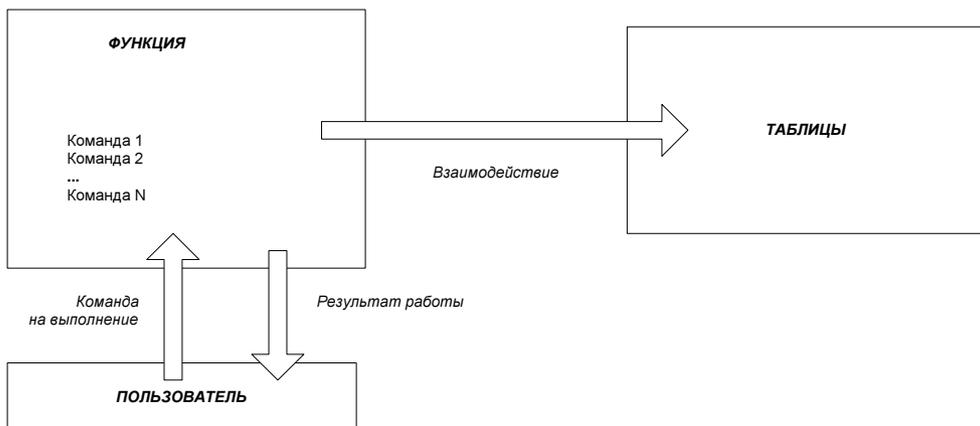


Рис. 1.11. Принцип работы функции

- Функция — предоставляет один из способов манипулирования данными. Функция — это, так же как и процедура, набор команд. Главное отличие, что функция всегда возвращает результат (рис. 1.11).
- Генератор — это элемент, предназначенный для построения *первичных ключей*. Генератор сначала создается, потом ему задается начальное значение, далее он присваивается атрибуту таблицы. Далее, чтобы генератор заработал, создается триггер для события "Вставка новой записи" (при использовании СУБД InterBase этим событием будет `Before_Insert`). И каждый раз, когда в таблицу добавляется новая запись, предварительно идет обращение к генератору, который в свою очередь выдает записи уникальный номер.

1.5. Основные принципы разработки базы данных

Теперь мы рассмотрим к чему разработчик или проектировщик базы данных должен стремиться в своей работе, это *основные правила* построения качественной базы данных.

- *Доступность*. База данных должна быть спроектирована так, чтобы данные, хранящиеся в ней, были доступны пользователю в любое время. База данных должна быть защищена от сбоев и непредвиденных ситуаций. Если в работе базы данных все же случился сбой, то он должен быть устранен в минимальные сроки и посредством специалиста или специалистов, обслуживающих данную базу.
- *Расширяемость*. База данных, как правило, отражает отношения чего-либо (товарно-денежные отношения людей) либо хранит информацию о каких-то объектах (данные по продажам, паспортные данные людей и т. д.). Со временем могут потребоваться изменения в базе данных, либо расширилось количество данных, которые требуется хранить, либо изменились правила отношений между людьми. Можно приводить кучу примеров причин, из-за которых возникает необходимость в изменении структуры базы данных. Принцип расширяемости регламентирует процесс внесения изменений в структуру базы данных. По времени он должен быть минимальным, а качество отраженных изменений — максимально.
- *Непротиворечивость*. База данных должна быть спроектирована таким образом, чтобы была исключена возможность возникновения в ней противоречивой информации. Например, если существует таблица "Персонал" (см. табл. 1.1) и в ней содержатся две абсолютно одинаковые записи (Вася Васильевич Васечкин), то можно утверждать, что условие непротиворечивости нарушено.

1.6. Транзакции

Транзакция — это неделимый блок действий (совокупность команд) с базой данных, который можно либо целиком выполнить, либо не выполнить вообще. Как правило, говорят, что транзакцию можно либо подтвердить, либо откатить.

Принцип работы транзакции можно описать следующим образом.

1. Необходимо внести данные в базу данных.
2. Начинаем транзакцию.
3. Вносим изменения.
4. Пытаемся завершить транзакцию.
 - Если завершить удалось, то все изменения успешно сохраняются.
 - Если произошла ошибка, то отказываемся от всех изменений.
5. Если передумали вносить изменения, то пропускаем пункт 4 и сразу откатываем транзакцию.

Например, у человека на кредитной карточке имеется 1000 рублей и карточка не может иметь отрицательный баланс (количество денежных средств с минусом). Предположим, что он хочет оплатить покупку в магазине на сумму 700 рублей и в это же время у него с карточки снимается 500 рублей за квартплату. Если не использовать транзакции, то баланс будет отрицательным (−200 рублей), а это не допустимо. Транзакции помогают защитить базу данных от ошибок. Предположим, что первой снялась сумма 500 рублей. Тогда человек не сможет оплатить покупку в магазине, потому что транзакция будет откатываться, так как условие отрицательного баланса будет вызывать ошибку.

1.7. Типы данных

По реляционной теории необходимо, чтобы типы используемых данных были простыми. Простые типы данных — это такие, которые не обладают внутренней структурой. К простым типам относятся:

- Логический
- Строковый
- Числовой

1.8. Нормализация

Нормализация — это процесс улучшения качества отношений между таблицами и, как следствие, свойств базы данных в целом. Основопологающим принципом является наложение некоторых ограничений и правил на структуру таблиц в базе данных.

В теории реляционных баз данных выделяются следующие *нормальные формы*:

1. Первая нормальная форма (1NF).
2. Вторая нормальная форма (2NF).
3. Третья нормальная форма (3NF).
4. Нормальная форма Бойса—Кодда (BCNF).
5. Четвертая нормальная форма (4NF).
6. Пятая нормальная форма (5NF).

Мы рассмотрим только первые три. Вызвано это тем, что приведение базы данных к третьей нормальной форме является достаточным условием для того, чтобы база данных соответствовала основным правилам построения качественной базы данных, которые были рассмотрены в *разд. 1.5* (доступность, расширяемость, непротиворечивость). Четвертая и пятая нормальные формы применяются редко и на данном этапе изучения для нас не представляют интерес.

Таблица находится в 1НФ, если все ее атрибуты содержат только данные простых типов, т. е. атрибут не может хранить данные структурированного типа (рис. 1.12).

ПЕРВАЯ НОРМАЛЬНАЯ ФОРМА		
Имя	Отчество	Фамилия
Вася	Васильевич	Васечкин
Вася	Васильевич	Васечкин
Миша	Андреевич	Кузькин
Андрей	Сергеевич	Иванов

Рис. 1.12. Таблица "Персонал", приведенная к первой нормальной форме

В таблице "Персонал" (см. рис. 1.12) каждый атрибут содержит данные только простого типа (все атрибуты имеют строковый или символьный тип данных).

Таблица находится во 2НФ, если она находится в 1НФ и каждый ее *неключевой* атрибут (тот, который не входит в состав первичного ключа) полностью зависит от первичного ключа (рис. 1.13).

ВТОРАЯ НОРМАЛЬНАЯ ФОРМА			
Ключевое поле	Имя	Отчество	Фамилия
1	Вася	Васильевич	Васечкин
2	Вася	Васильевич	Васечкин
3	Миша	Андреевич	Кузькин
4	Андрей	Сергеевич	Иванов

Рис. 1.13. Таблица "Персонал", приведенная ко второй нормальной форме

На рис. 1.13 видно, что в таблицу "Персонал" добавлен дополнительный атрибут "Ключевое поле"; таким образом, мы исключаем ситуацию, когда в таблице будут присутствовать одинаковые записи и теперь каждый атрибут таблицы зависит от первичного ключа (атрибут "Ключевое поле"). Таким образом, 2НФ накладывает условие, что в каждой таблице должен быть первичный ключ.

Таблица находится в 3НФ, если она находится во 2НФ и каждый неключевой атрибут *нетранзитивно* зависит от первичного ключа.

Сейчас все разберем на примере. Имеется 2 таблицы (рис. 1.14). В таблице "Персонал" хранятся данные людей: фамилия, имя и отчество. В таблице "Отделы" хранятся данные по отделам и по персоналу, который в них работает.

Поговорим о недостатках этой базы данных (рис. 1.14). Во-первых, в таблице "Отделы" явно содержится избыточная информация. Получается, что имя, отчество и фамилия о человеке хранятся сразу в двух таблицах. Во-вторых, если возникнет ситуация, что у сотрудника изменилась фамилия, например, сменил он ее, то данные придется изменять в обеих таблицах. В-третьих, если создать запись в таблице "Отделы" без указания конкретного сотрудника, то атрибуты "Имя", "Отчество", "Фамилия" окажутся пустыми и

таким образом получится, что в отделе работает сотрудник, которого нет в таблице "Персонал". Другим словами, несмотря на то, что *неключевые атрибуты* обеих таблиц полностью зависят от первичного ключа (то есть таблицы приведены к 2НФ), атрибуты таблицы "Отделы" еще к тому же находятся в зависимости от атрибутов таблицы "Персонал". Такая зависимость называется транзитивной.

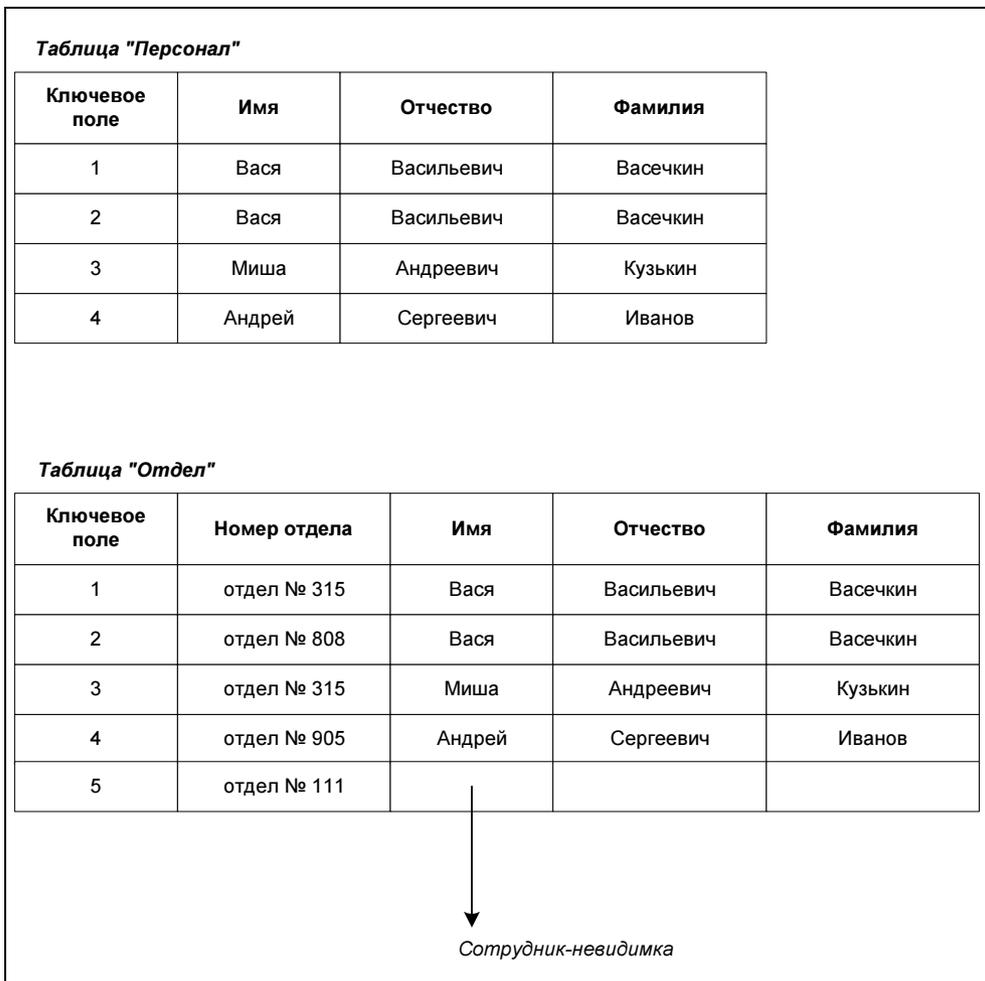


Рис. 1.14. Таблицы "Персонал" и "Отделы"

Приведем таблицы к 3НФ (рис. 1.15).

ТРЕТЬЯ НОРМАЛЬНАЯ ФОРМА**Таблица "Персонал"**

Ключевое поле	Имя	Отчество	Фамилия
1	Вася	Васильевич	Васечкин
2	Вася	Васильевич	Васечкин
3	Миша	Андреевич	Кузькин
4	Андрей	Сергеевич	Иванов

Таблица "Размещение сотрудников"

Ключевое поле	Код сотрудника	Код отдела
1	1	1
2	2	2
3	3	1
4	4	3

**Таблица
"Отделы"**

Ключевое поле	Номер отдела
1	отдел № 315
2	отдел № 808
3	отдел № 905

Рис. 1.15. Пример ЗНФ

В результате небольших изменений преодолены ограничения, накладываемые предыдущим вариантом базы данных (рис. 1.15). Могут быть как отделы без сотрудников, например, вновь отремонтированные помещения или