САМОУЧИТЕЛЬ

bhv

УРОКИ MySQL

Анатолий Мотев

- Установка и настройка СУБД MySQL
- Проектирование и создание баз данных
- Основы языка SQL
- Создание приложений на PHP

+ CD

Создай базу данных своими руками!

Анатолий Мотев

УРОКИ MySQL

САМОУЧИТЕЛЬ

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.2
М85

Мотев А. А.

М85 Уроки MySQL. Самоучитель. — СПб.: БХВ-Петербург, 2006. — 208 с.: ил.

ISBN 5-94157-658-7

Книга посвящена использованию СУБД MySQL для разработки интернет-проектов. В виде уроков рассмотрены все необходимые этапы работы с базами данных: от проектирования структуры до реализации приложений на языке PHP, позволяющих манипулировать данными. Изложенный материал сопровождается многочисленными примерами, комментариями и упражнениями. Показано, как создать гостевую книгу, форум, регистрацию пользователей, интернет-магазин и другие сложные элементы web-сайта. К книге прилагается компакт-диск, который содержит учебную базу данных и скрипты, описанные в книге, а также дистрибутивы MySQL и другие программы, распространяемые по лицензии GNU/GPL.

Для 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	*Татьяна Темкина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.03.06.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 16,77.
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 5-94157-658-7

Оглавление

Предисловие

Большим компаниям необходимы базы данных, а также сложное и дорогое программное обеспечение для работы с ними. Такие программные продукты позволяют управлять огромными объемами информации целой компании, поскольку обладают полным набором функций для работы с данными.

Напротив, пользователи домашних компьютеров обычно не нуждаются в базах данных. Они хранят свою информацию (телефоны, адреса и т. д.) в небольших файлах, создаваемых с помощью специальных программ вроде записных книжек, электронных таблиц или телефонных справочников.

Но есть и промежуточная категория пользователей, которым требуется хранение и управление информацией средних объемов. К этой категории относятся небольшие предприятия и организации, филиалы больших компаний, а также пользователи домашних компьютеров, которым нужно поддерживать сложные персональные данные (например, размещаемый в Интернете каталог любимых фильмов с дополнительной информацией о сюжетах, актерах, съемочных группах и т. д.).

Таким пользователям как раз и адресована книга.

О чем эта книга

В данной книге рассматриваются возможности системы управления базами данных (СУБД) MySQL, вопросы создания баз данных в этой среде и их применения для реализации полноценных приложений в сети Интернет. Она познакомит читателя с основами баз данных, научит проектировать и создавать их с использованием MySQL, а также настраивать базу данных и управлять ею. СУБД MySQL достаточно проста в освоении и использовании, а множество примеров и практических заданий помогут вам лучше и быстрее усвоить материал.

Также в этой книге большое внимание уделено взаимодействию MySQL с программами на языке PHP, используемом для создания динамических сайтов в Интернете. Это позволит вам применить полученные знания и навыки на практике и создать гостевую книгу, форум и многие другие полезные и интересные приложения на основе баз данных для своего сайта или сайта вашей фирмы.

Примечание

При описании интерфейса для языка PHP предполагается наличие у читателя основных навыков работы с рассматриваемым языком.

Как пользоваться книгой

Книга разделена на четыре части.

Во *введении* я расскажу о том, что такое базы данных и где их можно использовать. Также в нем описаны модели построения базы данных.

Часть I "Проектирование базы данных" может сначала показаться не представляющей ценности, но на самом деле это одна из самых важных частей книги. Правильное проектирование важно для разработчика, если ставится задача создания приложения для работы с базами данных, достаточно легко масштабируемого в случае необходимости внесения изменений. Правильное проектирование базы данных также позволяет обеспечить высокую производительность.

Часть II "MySQL" описывает процесс инсталляции ядра MyQSL и использование установленных утилит для обращения к базе данных.

В *части III "Формирование запросов к БД. Язык SQL"* приведены важнейшие конструкции языка SQL, используемого для формирования структуры базы данных и манипулирования информацией, хранящейся в ней.

Часть IV "PHP и MySQL" предназначена для тех, кто уже имеет базовый опыт создания программ на языке PHP и хочет узнать, как "заставить" свои приложения общаться с базами данных MySQL.

Материал книги изложен в виде уроков, последовательно описывающих принципы работы с СУБД MySQL. В некоторых уроках есть контрольные задания, выполнение которых поможет вам оценить, насколько хорошо усвоен данный урок.

Приложения содержат справочные материалы — таблицу зарезервированных слов СУБД MySQL и список функций языка PHP, используемых для работы с MySQL, а также ответы на контрольные вопросы и примеры выполнения практических заданий, приведенных в книге, и описание сопроводительного компакт-диска книги.

Введение

Эта книга вводит вас в мир разработки малых и средних баз данных с помощью MySQL. MySQL *не является* базой данных. Это компьютерная программа, позволяющая пользователю создавать, поддерживать базы данных и управлять ими. Такой тип программного обеспечения называется *системой управления базами данных (СУБД)*. СУБД действует как посредник между физической базой данных и ее пользователями.

Создателем СУБД MySQL является Майкл Видениус (Michael Widenius, Monty) из шведской компании TcX. В 1979 г. он разработал для внутрифирменного использования средство управления базами данных и назвал его UNIREG. Впоследствии программа UNIREG была переписана на нескольких разных языках и расширена для поддержки больших баз данных.

В 1994 г. компания TcX начала разрабатывать приложения для Интернета, применяя для поддержки этого проекта программу UNIREG. К сожалению, динамическая генерация web-страниц с помощью этой программы приводила к большим накладным расходам. Поэтому разработчики из TcX, взяв UNIREG за основу, использовали утилиты сторонних разработчиков для mSQL и к маю 1995 г. у TcX имелась СУБД, удовлетворявшая внутренним потребностям компании, — MySQL 1.0.

Что же касается названия, то сам разработчик говорит об этом так: "До конца неясно, откуда идет название MySQL. В TcX базовый каталог, а также значительное число библиотек и утилит в течение десятка лет имели префикс "my". Вместе с тем мою дочь тоже зовут Май (My). Поэтому остается тайной, какой из двух источников дал название MySQL".

С момента публикации СУБД MySQL в Интернете она была перенесена на многие системы под управлением ОС UNIX, а также Win32 и OS/2. TcX считает, что сейчас MySQL используется примерно на 500 000 серверов.

Если у вас есть опыт работы с реляционными базами данных и их проектирования, вы можете сразу перейти к *части II* данной книги. Но если вы только

начали осваивать данную область, то информация, приведенная здесь и в *части I*, будет весьма полезной и интересной.

Что такое базы данных и где они используются

Для начала нужно выяснить, что представляют собой базы данных и для каких задач их можно, а иногда и необходимо применять.

Определение базы данных может звучать так:

Определение

База данных (БД) — это именованная совокупность данных, отражающая совокупность объектов и их отношения в предметной области.

Наверное, читателю не совсем понятно данное определение. Давайте же разберемся. *Предметной областью* для БД является то, что она описывает. Например, предметом для описания может быть магазин по продаже детских игрушек, склад товаров или список сотрудников большой компании, хранящийся в отделе кадров. То есть предметной областью для базы данных может быть все, что требует хранения довольно больших по объему и сложных по структуре данных.

Объекты — это структурные единицы, из которых состоит предметная область. Если в качестве предметной области взять магазин, то объектами будут являться товары, покупатели, продавцы, поставщики, заказы и т. д.

У каждого объекта есть набор *свойств*, который необходимо описать. Например, у товара могут быть следующие свойства:

1. Код (артикул).
2. Наименование.
3. Фирма-производитель.
4. Описание.
5. Цена.
6. Количество единиц на складе.

Все объекты, описанные в предметной области, связаны между собой отношениями (рис. В1).

То есть поставщики поставляют в магазин различные товары, покупатели делают заказы, заказы производятся на определенный товар и т. д.

И, наконец, совокупность (собрание) данных является именованной просто потому, что любая база данных имеет имя (название).

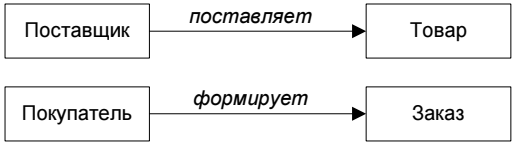


Рис. В1. Примеры отношений между объектами

Базы данных и приложения

Каждая база данных служит определенной цели. Простого наличия СУБД не достаточно, чтобы у вашей базы данных появилось назначение. Все зависит от того, как вы будете использовать хранимые данные. Если магазин не будет продавать товары, то какой смысл в организации и хранении товаров, ведении складского учета и т. д.? Кроме этого, нужна возможность изменения информации об имеющихся товарах и добавления информации о новых. Представьте себе компьютерный магазин, продающий в течение многих лет одни и те же процессоры, видеокарты и другие комплектующие. Я думаю, вскоре его станут называть "Антикварным".

Базы данных существуют для того, чтобы мы могли с ними взаимодействовать. Но взаимодействие происходит не непосредственно с базой данных, а косвенно — с помощью специально разработанного программного обеспечения. До появления Интернета базы данных обычно использовались в больших компаниях для поддержки всевозможных функций — бухгалтерии, финансов, контроля поставок, складского учета, учета персонала и т. п. Развитие Интернета и усложнение задач домашних вычислений способствовали перемещению потребностей в использовании БД за пределы больших компаний.

Базы данных и Интернет

Наиболее популярная область применения MySQL — разработка приложений для Интернета. В настоящее время спрос на сложные и надежные приложения достаточно велик. Соответственно вырос спрос и на базы данных. С их помощью можно реализовывать множество полезных функций web-сайта. Практически любым содержимым страницы можно управлять с помощью базы данных.

Возьмем в качестве примера интернет-магазин. Заходя на такой сайт, мы должны видеть список всех доступных товаров с их описанием и фотографиями. Если опубликовать каталог товаров в виде простой HTML-страницы, то придется вручную редактировать ее каждый раз, когда требуется добавить новый товар или изменить цену. Если же вместо этого хранить данные о товарах в базе данных, то появится возможность изменения web-страницы в

режиме реального времени, сразу же после внесения в базу данных сведений о новом товаре или изменений для имеющегося товара. Кроме этого, можно будет интегрировать сайт с какой-либо системой электронной коммерции.

Так как же web-страница взаимодействует с базой данных? База данных находится на вашем web-сервере. На web-странице вы размещаете форму, в которую пользователь вводит свой запрос (например, для поиска чего-либо) или те данные, которые нужно передать. После отправки данных из формы на сервер последний запускает написанную вами программу, которая извлекает данные, переданные пользователем. Далее программа формирует запрос на языке SQL *(см. часть III)* для выборки или изменения данных, а СУБД чудесным образом делает все остальное. Если пользователь запрашивал какие-то данные из БД, то ваша программа может оформить результат запроса в виде новой HTML-страницы и отправить обратно пользователю.

Обычно такие программы создаются в виде CGI-сценариев или серверных приложений на языке Java. Возможно также встраивание программы прямо в HTML-страницу.

Таким образом, использование базы данных для управления подобным сайтом дает очевидные преимущества как продавцу, так и покупателю.

Примечание

Подробнее взаимодействие базы данных и приложений описано в *части IV*. В ней я познакомлю вас с базовым синтаксисом языка PHP, а также расскажу о возможностях, предоставляемых данным языком для работы с базами данных.

Другие СУБД среднего масштаба

Первой СУБД среднего масштаба с поддержкой SQL была mSQL. Она недолго оставалась в одиночестве — в настоящий момент ее коллегами являются такие СУБД, как PostgreSQL, GNU SQL, Beagle и уже известная нам MySQL. Все эти продукты относятся к категории Open Source, т. е. поставляются с открытым исходным кодом (даже если за использование этого продукта необходимо заплатить).

PostgreSQL

В начале 1980-х доктор Майкл Стоунбрейкер (Michael Stonebreaker) из Калифорнийского университета в Беркли (University of California at Berkeley) создал систему управления базами данных Ingres, которая предвосхитила многие концепции, реализованные в современных СУБД. Ingres была некоммерческим проектом, который финансировался университетом.

Одна из компаний обратила внимание на коммерческий потенциал этого продукта и, зарегистрировав торговую марку Ingres, выпустила коммерческий

продукт. Исходная некоммерческая версия Ingres была переименована в University Ingres и в дальнейшем развивалась независимо от коммерческой версии.

Через некоторое время доктор Стоунбреикер пошел в своих исследованиях дальше того, что предполагалось в начальных целях проекта Ingres. Он решил разработать совершенно новую систему управления базами данных, которая бы развила идеи, заложенные в Ingres. Эта СУБД стала известна как Postgres.

Postgres, как и Ingres, была открытым для всех проектом (он также финансировался университетом). Сейчас Postgres соперничает по популярности с MySQL и mSQL среди СУБД среднего масштаба.

Подробнее узнать о PostgreSQL можно на сайте **http://www.postgresql.org**.

GNU SQL

Проект GNU является для многих разработчиков символом свободы. Официальная лицензия на продукты GNU гарантирует свободный доступ и полную свободу для изменений исходного кода. До недавнего времени большим пробелом было отсутствие СУБД среди таких продуктов.

Институт системного программирования Российской академии наук работает над устранением этого недостатка. Недавно он выпустил первую открытую бета-версию GNU SQL — полнофункциональную СУБД с поддержкой языка SQL.

В настоящее время GNU SQL поддерживает многие развитые возможности — транзакции, вложенные запросы.

Подробнее узнать о GNU SQL можно на сайте **http://www.ispras.ru**.

Beagle

Проект Beagle был разработан и реализован Робертом Клейном (Robert Klein). Так же как и GNU SQL, Beagle был задуман как полностью SQL-совместимый сервер со всеми необходимыми функциями.

Домашняя страница Beagle расположена на сайте **http://www.beaglesql.org**.

Модели данных

Различают три модели, по которым можно построить БД:

- ❒ иерархическая;
- ❒ сетевая;
- ❒ реляционная.

В СУБД MySQL используется реляционная модель данных. Но все же про другие модели следует сказать несколько слов, чтобы читатель имел о них представление.

Иерархическая модель

В иерархической модели отношения между объектами строятся в виде дерева. Данная модель характеризуется такими параметрами, как уровни, узлы, связи. Принцип работы модели таков: несколько узлов более низкого уровня соединяются при помощи связи с одним узлом более высокого уровня. *Узел* (сегмент дерева) — информационная модель элемента, находящегося на данном уровне иерархии.

В качестве примера можно привести базу данных университета (рис. В2).

Рис. В2. Иерархическая модель

Обучение в университете разделено на курсы, на каждом курсе есть какое-то количество групп, в состав каждой группы входят конкретные студенты. Рассмотрев данный пример, мы можем выделить следующие свойства иерархической модели:

- несколько узлов низшего уровня могут быть связаны только с одним узлом высшего уровня;
- каждый узел имеет свое имя;
- у иерархического дерева имеется только одна вершина (корень), не подчиненная никакой другой вершине.

Сетевая модель

Сетевая модель базы данных похожа на иерархическую. Она имеет те же основные составляющие (узел, уровень, связь), однако характер их отношений принципиально иной. Сетевая модель основана на связях между *наборами данных (агрегатами)*.

В качестве примера можно привести базу данных преподавательского состава университета (рис. В3).

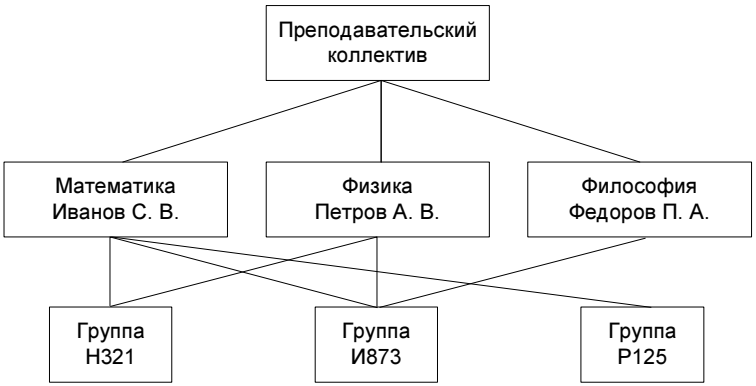


Рис. В3. Сетевая модель

Связь можно осуществить между элементами разных уровней.

Реляционная модель

Реляционная база данных представляется как совокупность таблиц, связанных друг с другом. Для наглядности приведу пример таблицы, которая может появиться в базе данных книжного магазина (табл. В1).

Таблица В1. *Книги*

Код	Название	Автор
00001	Основы JavaScript	Пол Уилтон
00002	Веб-дизайн	Дмитрий Кирсанов
00003	Введение в XML	Дэвид Хантер

Другая таблица (табл. В2) вполне может существовать в базе данных какого-нибудь компьютерного магазина.

Таблица В2. *Товары*

Артикул	Наименование	Модель	Цена (руб.)
M-0001	Модем ZyXEL	Omni 56K	1500
V-0033	Монитор LG	Flatron F700P	5000
A-0242	Колонки SVEN	SPS-611	800

В *части I* мы более подробно рассмотрим специфику таблиц, а пока обратим внимание на их некоторые особенности. Любая таблица состоит из строки заголовков столбцов и одной или более строк с данными. Эти столбцы и строки должны иметь следующие свойства:

- каждому столбцу таблицы присваивается имя, которое должно быть уникальным для этой таблицы;
- столбцы таблицы упорядочиваются слева направо (т. е. самый первый слева столбец таблицы — это столбец 1, второй слева — столбец 2, ..., самый правый — столбец *n*), хотя упорядоченными они являются только с точки зрения пользователя. Порядок, в котором определены имена столбцов, становится порядком, в котором в них должны вводиться данные;
- строки таблицы не упорядочены (их последовательность определяется лишь последовательностью ввода в таблицу);
- при выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их информационному содержанию;
- в поле на пересечении строки и столбца любой таблицы всегда имеется только одно значение данных и никогда не должно быть множества значений;
- всем строкам таблицы соответствует один и тот же набор столбцов, хотя любая строка может содержать пустые значения в некоторых столбцах;
- все строки таблицы обязательно отличаются друг от друга хотя бы одним значением, что позволяет однозначно определить любую строку такой таблицы.

Так почему же модель называется реляционной? Все просто, *отношение* (англ. relation) — это математический термин, используемый для обозначения неупорядоченного набора (совокупности) однотипных записей или таблиц определенного вида. Реляционные системы берут свое начало в математической теории множеств. Они были предложены сотрудником компании IBM Э. Ф. Коддом (E. F. Codd) в 1968 г.

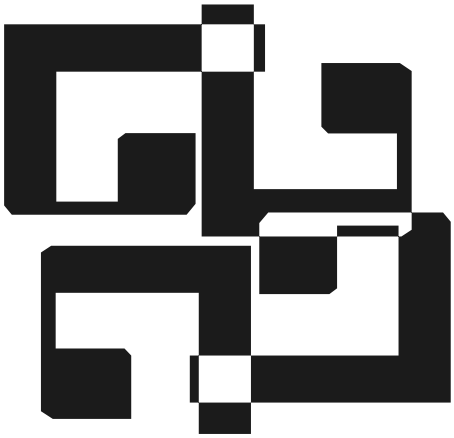
Немного терминологии

Настало время определиться с терминами, используемыми при разработке БД. Основные термины и их описание:

- *сущность* (англ. entity) — то, что описано конкретной таблицей (пример: сущность "Книга");

- *поле* (англ. field) — свойство описываемой сущности (примеры: поле "Название", поле "Автор");
- *запись* (англ. record) — одна строка таблицы;
- *связь* (англ. relationship) — логическое отношение между двумя сущностями.

Итак, таблица описывает отдельную сущность. Сущность описывается полями. Между сущностями бывает организована связь (такие сущности называют связанными).



ЧАСТЬ I

Проектирование базы данных

Урок 1. Анализ предметной области, проблемы и пути их решения

Урок 2. Физическое проектирование таблиц, виды связей, нормализация

Урок 3. Типы данных и типы таблиц

Итак, теперь мы знаем, что такое база данных (БД) и для чего она может понадобиться. Настало время приступить к проектированию БД.

Рассмотрим этапы, которые нам необходимо пройти, прежде чем будут созданы БД и приложения, которые смогут использовать ее в своей работе:

1. Анализ предметной области.
2. Выбор модели данных (таблицы, связи).
3. Выбор СУБД.
4. Проектирование таблиц.
5. Проектирование приложений (модулей/программ), управляющих БД.
6. Реализация БД на компьютере.
7. Разработка средств администрирования БД (т. е. добавления, удаления, редактирования данных).
8. Эксплуатация БД.

Пункты 2 и 3 нами уже продуманы — мы используем реляционную модель данных, реализуемую в СУБД MySQL.

УРОК 1

Анализ предметной области, проблемы и пути их решения

Анализ предметной области — это анализ исходного набора данных. Предположим, перед нами стоит задача построения БД для какого-то магазина. Прежде всего, необходимо определить, какие данные нам понадобится хранить.

Допустим, нам нужно знать:

- дату продажи;
- фамилию, имя и отчество продавца;
- фамилию, имя и отчество покупателя;
- адрес покупателя;
- товар;
- сумму покупки (в рублях).

Представим все эти данные в виде таблицы (табл. 1.1).

На первый взгляд, данная таблица вполне нам подходит, т. к. содержит все необходимые данные. Если же рассмотреть ее более тщательно, то можно заметить, что такой способ хранения данных повлечет за собой ряд проблем и сложностей.

Во-первых, покупатель Петров встречается в таблице несколько раз — он постоянный клиент и довольно часто заходит в наш магазин. В итоге, при каждом его посещении нам приходится вносить одни и те же данные, например адрес его проживания. Следовательно, мы получаем избыточность данных — повторный ввод информации. Кроме этого, при очередном вводе данных о покупателе мы можем сделать ошибку, указав не тот номер дома или номер квартиры. Соответственно, у нас получится два разных покупателя.

Таблица 1.1. *База данных магазина, состоящая из одной таблицы*

Дата продажи	ФИО продавца	ФИО покупателя	Адрес покупателя	Товар	Сумма покупки (в рублях)
12.12.2004	Иванов Иван Иванович	Сергеев Андрей Васильевич	ул. Мира, д. 8, кв. 15	Телевизор SMK321	12000
12.12.2004	Васин Петр Сергеевич	Петров Иван Иванович	ул. Марата, д. 32, кв. 3	Лампа накаливания (60 Вт), удлинитель (10 м)	25
13.12.2004	Кузьмин Владимир Владимирович	Мухина Анна Викторовна	ул. Ленина, д. 21, кв. 14	Аудиосистема PS-911	4800
14.12.2004	Задорнов Виталий Петрович	Петров Иван Иванович	ул. Марата, д. 32, кв. 3	Чайник Tefal, сковорода Tefal	2340
14.12.2004	Соколов Сергей Александрович	Ванин Герасим Андреевич	пр. Науки, д. 45, кв. 26	Телефон Dect S115	1200
...	...	...	...	...	...

Во-вторых, если нам понадобится изменить данные, то придется менять их в нескольких местах. Допустим, в адрес закралась орфографическая ошибка или господин Петров решил его сменить. В этом случае нам придется изменить адрес столько раз, сколько упоминаний о нем есть в таблице. Напомню, что господин Петров заглядывает к нам довольно часто, чтобы сделать очередную покупку. И чем чаще это происходит, тем больше работы нам приходится делать, чтобы поддерживать непротиворечивость данных. Таким образом, получаем проблему обновления данных.

На этом список проблем не заканчивается — есть сложности и при удалении данных. Если мы захотим удалить из таблицы какого-нибудь покупателя, то вместе с покупателем будет удалено все, что с ним было связано. Например, будут удалены сведения о товарах, которые он когда-то приобретал.

Из всего этого следует, что структура хранения данных, представленная в табл. 1.1, нас совершенно не устраивает. Чтобы избавиться от всех этих сложностей, мы используем прием, называемый *нормализацией*.

УРОК 2

Физическое проектирование таблиц, виды связей, нормализация

Прежде чем приступить к нормализации, необходимо подробнее обсудить некоторые фундаментальные понятия реляционных баз данных. Данная модель состоит из трех основных элементов:

- сущность;
- атрибут;
- связь.

Сущности — это те вещи или объекты, данные о которых необходимо хранить. В модели данных сущность представляется в виде прямоугольника с заголовком. Заголовок является именем сущности или, если сказать проще, это название таблицы, хранящей данные. То есть сущность в БД — это таблица.

Атрибуты (поля таблицы) описывают те данные, которые нам нужно знать о сущности. Значение атрибута может быть числом, строкой символов, датой, временем или другим базовым значением данных.

Связями, как вы помните, называются логические взаимоотношения между сущностями. Но о связях мы поговорим позже.

В нашем примере база данных содержит ряд объектов: покупатель, продавец, наименование товара, дата продажи и т. д. Какие из них являются сущностями? Обратим внимание, что мы определили несколько видов данных (имя, адрес), относящихся к каждому покупателю. Без них невозможно описать покупателя. Поэтому покупатель является одним из объектов, которые мы хотели бы описать, т. е. сущностью. Давайте приступим к разработке модели данных с сущностью "Покупатель" (табл. 2.1).

Таблица 2.1. *Сущность "Покупатель"*

Покупатель

Примечание

Почему мы назвали нашу сущность "Покупатель", а не "Покупатели"? По общепринятому соглашению имя сущности должно быть в единственном числе, т. к. каждая сущность дает имя ее экземпляру. Например, "Петров Иван Иванович" является экземпляром сущности "Покупатель", а не "Покупатели".

У каждой сущности есть атрибуты, которые ее описывают. О покупателе нам могут понадобиться подробные сведения (табл. 2.2), например, если он покупает что-то в кредит.

Таблица 2.2. *Сущность "Покупатель" с атрибутами*

Покупатель		
ФИО покупателя	**Адрес**	**Телефон**
Сергеев Андрей Васильевич	ул. Мира, д. 8, кв. 15	362-23-32
Петров Иван Иванович	ул. Марата, д. 32, кв. 3	352-48-69
Ванин Герасим Андреевич	пр. Науки, д. 45, кв. 26	236-88-00
...	...	...

Вот теперь пришло время нормализации.

Впервые понятие "нормализация" ввел Е. Ф. Кодд, занимавшийся исследованиями в компании IBM. Целью нормализации является устранение из БД некоторых нежелательных характеристик. В частности, ставится задача избежать избыточности данных, приводящей к сложностям при операциях добавления, изменения и удаления данных.

Понятие нормализации включает пять нормальных форм. Мы рассмотрим три из них — этого достаточно, чтобы сделать структуру БД вполне работоспособной.

Первая нормальная форма (1НФ)

Сущность приведена к первой нормальной форме (1НФ), если все ее атрибуты имеют единственное значение. Если в каком-либо атрибуте есть повторяющиеся значения, то сущность не приведена к 1НФ. Посмотрев на нашу

базу данных (см. табл. 1.1), можно заметить, что в атрибуте "Товар" встречается больше одного значения, т. е. БД не находится в 1НФ. Это означает, что мы упустили, по крайней мере, еще одну сущность. Атрибут "Товар" описывает сведения о купленном товаре. Возможно, он тоже является сущностью. Давайте внесем его в нашу модель и добавим другие атрибуты (табл. 2.3).

Таблица 2.3. *Сущность "Товар" с атрибутами*

Товар		
Наименование	**Производитель**	**Цена (в рублях)**
Чайник	Tefal	1145
Телевизор SMK321	Sony	12 000
Телефон Dect S115	Dialon	1200
...	...	...

Теперь у нас есть сущность, приведенная к 1НФ.

Прежде чем переходить к рассмотрению второй нормальной формы, необходимо подробнее поговорить о связях между сущностями.

Ключи и связи

У каждого экземпляра сущности должен быть уникальный идентификатор, который будет однозначно определять каждую запись. Какой же из атрибутов может быть таким идентификатором? Нам необходимо выбрать атрибут, который подчиняется следующим правилам:

- он уникален для каждой записи (экземпляра сущности);
- для каждой записи он имеет значение, отличное от `NULL` (отсутствие данных);
- для каждой записи его значение не изменяется.

Такой атрибут называется *первичным ключом* (primary key). Если ни один из атрибутов не удовлетворяет этим правилам, то нужно ввести новый атрибут или создать первичный ключ из нескольких атрибутов. Рассмотрим в качестве примера таблицу, описывающую сущность "Покупка" (табл. 2.4).

В данной таблице ни один из атрибутов нельзя назначить ключевым, т. к. во всех полях данные могут повторяться. В каждом заказе может быть несколько товаров, каждый товар может присутствовать во многих заказах (исключение составляют магазины, торгующие уникальными товарами, но наш магазин таким не является). В этом случае можно задать *составной ключ*

(composite primary key), состоящий из полей "Номер заказа" и "Номер товара". Комбинация значений этих полей будет уникальной.

Таблица 2.4. *Сущность "Покупка" с атрибутами*

Покупка		
Номер заказа	**Номер товара**	**Количество единиц товара**
1	23	1
1	12	1
2	25	2
3	12	10
4	23	2

Выбор ключевого атрибута сущности играет очень важную роль при проектировании БД, т. к. он используется для моделирования связей. Если атрибут не удовлетворяет хотя бы одному из перечисленных правил, это может повлиять на всю модель данных.

Рассмотрим таблицу, описывающую покупателя (см. табл. 2.2). Можно попытаться выбрать в качестве ключевого поле "ФИО". Но что если покупатель изменит фамилию (это вполне возможно)? В этом случае нарушается третье правило для ключевых атрибутов. Кроме того, значения могут оказаться не уникальными — в каждом большом городе найдется несколько Петровых Иванов Ивановичей. Тогда нарушится первое правило. Наконец, возможно, что, вводя данные о покупателе, вы не будете знать его фамилию, имя и отчество. Тогда нарушается второе правило, которое гласит, что значение ключа должно быть отличным от `NULL`.

Исходя из этого, для таблицы "Покупатель" лучше задать новые поля (табл. 2.5).

Примечание

Для наглядности мы будем подчеркивать имя ключевого атрибута в каждой таблице.

Таблица 2.5. *Сущность "Покупатель" с ключевым атрибутом*

Покупатель			
Номер покупателя	**ФИО покупателя**	**Адрес**	**Телефон**

В эту таблицу мы ввели новое поле "Номер покупателя", которое будет уникально определять каждого конкретного покупателя нашего магазина.

Ключевые атрибуты сущностей (таблиц) позволяют моделировать связи, описывающие взаимоотношения между ними. Есть три типа связей:

- *один к одному* (1:1) — устанавливается между таблицами, если запись в первой таблице соответствует только одной записи во второй таблице;
- *один ко многим* (1:M) — устанавливается между таблицами, если запись в первой таблице соответствует одной или нескольким записям во второй таблице;
- *многие ко многим* (M:M) — устанавливается между таблицами, если одной записи в первой таблице соответствует несколько записей во второй таблице, а одной записи во второй таблице соответствует несколько записей в первой таблице.

Последний вид связи в реляционных таблицах не реализуется. Связь "1:1" встречается довольно редко. Если при проектировании таблиц вы столкнетесь с такой связью, следует еще раз внимательно рассмотреть свой проект. Возможно, сущности, между которыми установлена такая связь, являются на самом деле одной. В этом случае их следует объединить.

Рис. 2.1. Связь между таблицами "Заказ" и "Продавец"

Связь между таблицами, показанными на рис. 2.1, определена как "1:M". По номеру продавца мы можем узнать, какие заказы он обслуживал. Итак, таблицы "Продавец" и "Заказ" связаны по полю "Номер продавца".

Поле, которое указывает на запись в другой таблице, связанную с данной записью, называется *внешним ключом* (foreign key). Например, в таблице "Заказ" внешним ключом является поле "Номер продавца".

Иными словами, внешний ключ — это поле (или набор полей), значения которого совпадают с имеющимися значениями первичного ключа другой таблицы.

Ссылочная целостность

Итак, мы уже знаем, что значения в ключевом поле должны быть уникальными. Это является одним из правил *ссылочной целостности*. Некоторые СУБД могут контролировать уникальность первичных ключей — при попытке присвоить первичному ключу значение, уже имеющееся в другой записи, СУБД сгенерирует диагностическое сообщение. Это сообщение в дальнейшем может быть передано в приложение, при помощи которого пользователь управляет данными.

Если две таблицы являются связанными, то внешний ключ одной таблицы должен содержать только значения, имеющиеся среди значений первичного ключа другой таблицы. Допустим, если удалить запись из таблицы "Продавец", то в связанной с ней таблице "Заказ" будут присутствовать заказы, обслуженные несуществующим продавцом.

Вторая нормальная форма (2НФ)

Прочно связав таблицы, мы можем продолжить нормализацию.

Реляционная таблица приведена ко второй нормальной форме (2НФ), если она приведена к первой нормальной форме и ее неключевые поля полностью зависят от первичного ключа.

Таблица "Покупатель" (см. табл. 2.5) приведена ко второй нормальной форме. Но этого недостаточно — в этой таблице есть дополнительные зависимости. Например, если в таблице есть несколько покупателей с одним адресом (члены одной семьи), то при смене этого адреса потребуется изменить несколько записей.

Третья нормальная форма (3НФ)

Таблица приведена к третьей нормальной форме (3НФ), если она приведена ко второй нормальной форме и ни одно неключевое поле не зависит от других неключевых полей.

Чтобы перейти от второй нормальной формы к третьей, нужно выполнить следующие шаги:

1. Определить все поля (или группы полей), от которых зависят другие поля.
2. Создать новую таблицу для каждого такого поля (или группы полей) и переместить группы зависящих от него полей в эту таблицу. Поле (или группа полей), от которого зависят все остальные перемещенные поля, станет при этом первичным ключом новой таблицы.

3. Удалить перемещенные поля из исходной таблицы, оставив лишь те из них, которые станут внешними ключами.

Мы должны создать для адреса покупателя новую таблицу и переместить в нее поля из исходной таблицы (табл. 2.6 и 2.7).

Таблица 2.6. *Сущность "Покупатель" в 3НФ*

Покупатель		
Номер покупателя	**ФИО покупателя**	**Номер адреса**
1	Петров Иван Иванович	1
2	Сидоров Андрей Анатольевич	2
3	Ванин Алексей Сергеевич	3
4	Ванин Иван Алексеевич	3
...	...	...

Таблица 2.7. *Сущность "Адрес покупателя" в 3НФ*

Адрес покупателя		
Номер адреса	**Адрес**	**Телефон**
1	ул. Мира, д. 34, кв. 40	954-87-23
2	ул. Ленина, д. 123, кв. 23	941-85-25
3	ул. Московская, д. 12	654-78-22
...	...	...

Получившиеся таблицы связаны по полю "Номер адреса" (рис. 2.2).

У нас есть два покупателя, живущих по одному адресу (например, отец и сын). Если их адрес изменится, то сразу у обоих.

Итак, теперь мы знаем, как проектируются таблицы и устанавливаются связи между ними. Настало время для создания реальной БД.

В качестве предметной области возьмем "библиотеку". Нам необходимо провести анализ данной предметной области и разработать БД, которую мы будем использовать на протяжении всей книги. Для начала давайте подумаем, что нам необходимо хранить в нашей базе. Во-первых, мы должны хранить сведения о книге (табл. 2.8).