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!

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

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.

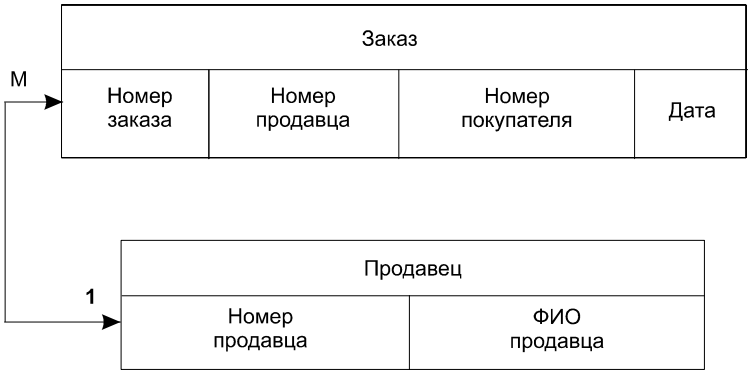

Рис. 2.1. Связь между таблицами "Заказ" и "Продавец"

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).

Рис. 2.2. Связь между таблицами "Покупатель" и "Адрес покупателя"

Таблица 2.8. *Сущность "Книга"*

Книга					
Номер книги	**Название**	**Год издания**	**Номер автора**	**Номер издательства**	**Номер раздела**
...	...	...	...	...	...

Номер книги будет уникально определять совокупность всех остальных ее атрибутов. Например, в нашу библиотеку поступила партия книг:

- автор: Томас Уилтон;
- название: "HTML 4.0. Справочник программиста";
- год издания: 2004;
- издательство: "Мир книг";
- раздел: "Компьютерная литература".

Допустим, партия включает 25 книг. Если бы номер присваивался каждому экземпляру книги, то нам пришлось бы вводить в таблицу "Книга" все эти данные двадцать пять раз. Поэтому номер книги будет определен непосредственно для партии этих книг. Но когда книга будет выдаваться на руки читателю, необходимо фиксировать номер конкретного экземпляра книги, а не номер партии. Поэтому необходимо хранить сведения о каждом экземпляре каждой книги (табл. 2.9).

Связь между этими таблицами будет осуществляться по номеру книги. То есть, зная номер экземпляра, мы сможем определить все остальные атрибуты книги.

Таблица 2.9. *Сущность "Экземпляр"*

Экземпляр	
Номер экземпляра	**Номер книги**
...	...

Поскольку наши писатели и поэты обычно не ограничиваются одним произведением, нам пришлось бы вводить сведения об авторе в таблицу "Книга" столько раз, сколько его книг есть в библиотеке. Поэтому в таблице "Книга" в качестве сведений об авторе зафиксирован номер автора (это куда лучше, чем вводить множество раз "Пушкин Александр Сергеевич"), а для хранения сведений об авторе мы создадим отдельную таблицу (табл. 2.10).

Таблица 2.10. *Сущность "Автор"*

Автор	
Номер автора	**ФИО автора**
...	...

Поле "ФИО автора" будет включать в себя его фамилию, имя и отчество. С таблицей "Книга" связь будет осуществляться по полю "Номер автора".

Кроме этого, нам нужно создать еще две таблицы, в которых будут храниться сведения об издательствах (табл. 2.11) и разделах (жанрах) (табл. 2.12).

Таблица 2.11. *Сущность "Издательство"*

Издательство	
Номер издательства	**Название издательства**
...	...

Таблица 2.12. *Сущность "Раздел"*

Раздел	
Номер раздела	**Название раздела**
...	...

Таблицы "Издательство" и "Раздел" будут связаны с таблицей "Книга" по полям "Номер издательства" и "Номер раздела" соответственно.

Итак, чтобы полностью описать книгу, нам потребовалось создать пять таблиц. Благодаря этому мы избежим избыточности данных. Но это еще не все. Нам нужно знать данные о читателях, посещающих библиотеку. Давайте создадим таблицу "Читатель" (табл. 2.13).

Таблица 2.13. *Сущность "Читатель"*

Читатель		
Номер читателя	**ФИО читателя**	**Номер адреса**
...	...	...

Также нам понадобится таблица с адресами читателей, чтобы избежать проблем в том случае, если будет несколько читателей, живущих по одному адресу (табл. 2.14).

Таблица 2.14. *Сущность "Адрес читателя"*

Адрес читателя		
Номер адреса	**Адрес**	**Телефон**
...	...	...

И, наконец, мы должны знать, какие книги находятся на руках у читателей. Для этого создадим новую таблицу и назовем ее "Абонемент" (табл. 2.15).

Таблица 2.15. *Сущность "Абонемент"*

Абонемент				
Номер записи	**Номер читателя**	**Номер экземпляра**	**Дата выдачи**	**Дата возврата**
...	...	...	...	...

Итак, мы полностью описали нашу БД, осталось только проставить связи между всеми таблицами (рис. 2.3).

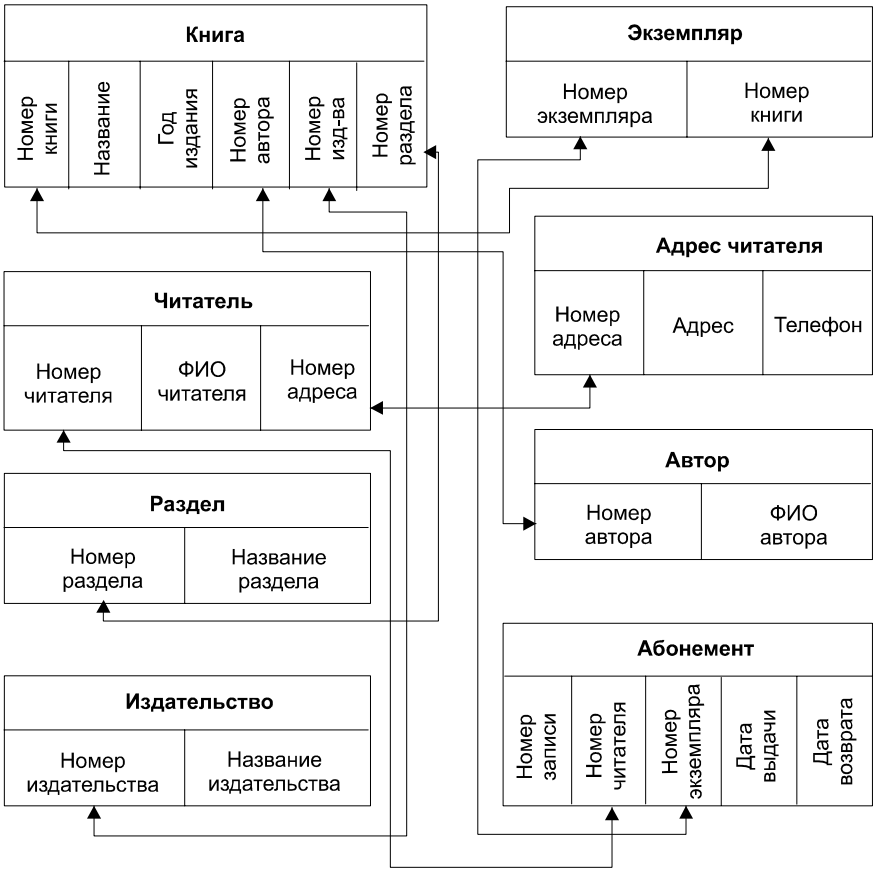

Рис. 2.3. Связи между сущностями в БД "Библиотека"