7

Работа с MySQL

Настало время узнать, как выполнить подключение к базе данных MySQL с помощью клиентского инструментария из состава MySQL. Для этих целей можно использовать инструмент с веб-интерфейсом, который называется phpMyAdmin. Кроме того, в этой главе мы рассмотрим порядок использования SQL для создания баз данных, учетных записей пользователей, таблиц, а также приемы модификации объектов, имеющихся в базе данных.

База данных MySQL

У MySQL есть собственный интерфейс для организации взаимодействия с клиентами, с помощью которого можно перемещать данные и изменять параметры базы данных. Обратите внимание: чтобы иметь возможность работать с базой данных, у вас должна быть учетная запись и пароль. Назначение *пользователей* базы данных позволяет ограничить круг пользователей, обладающих правом доступа к таблицам на сервере. Каждый сервер MySQL может содержать несколько баз данных, где группируются таблицы. Веб-приложения, работающие на стороне сервера, могут использовать как свои собственные или как единую, общую для всех приложений базу данных.

Вы можете установить MySQL у себя или воспользоваться предложением своего интернет-провайдера. Большинство провайдеров, предоставляющих поддержку PHP, обеспечивает и возможность пользования базой данных MySQL. Если вы испытываете какие-либо затруднения, посетите страницу технической поддержки своего провайдера или обратитесь к нему напрямую, чтобы узнать следующие технические подробности, касающиеся подключения к серверу:

- IP-адрес сервера баз данных;
- имя базы данных;
- имя пользователя;
- пароль.

Если вы установили MySQL на своем компьютере, то сможете использовать значения параметров по умолчанию, принятые во время установки, и пароль, который вы назначили. В этой главе мы рассматриваем два способа взаимодействия с базой данных MySQL: из командной строки и с помощью phpMyAdmin – инструмента с веб-интерфейсом.

Доступ к базе данных из командной строки

Один из способов взаимодействия с MySQL основан на использовании клиента командной строки MySQL (MySQL command line client). В зависимости от используемой операционной системы вам понадобится либо открыть командную оболочку в Windows (введя команду `cmd` в диалоге Выполнить, как показано на рис. 7.1), либо открыть окно терминала в таких средах, как Mac OS X и UNIX.

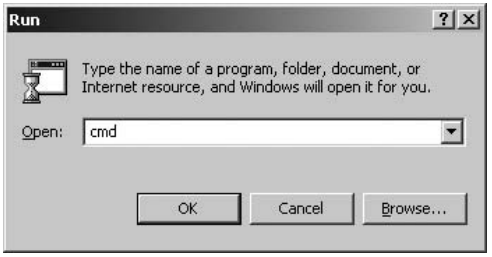

Рис. 7.1. *Диалоговое окно* Выполнить *в операционной системе Windows*

Как только вы доберетесь до командной строки, введите команду `mysql` и нажмите клавишу Enter. Синтаксис команды `mysql`:

```
mysql -h имя_хоста -u пользователь -p
```

Если вы установили MySQL на своем компьютере, имя пользователя по умолчанию будет `root`. В этом случае вы можете опустить ключ `-h` с параметром *имя_хоста*. Когда MySQL выведет приглашение «Enter password», введите пароль. Если имя хоста, имя пользователя и пароль были указаны корректно, вы увидите приветствие (рис. 7.2).

Сразу после установки MySQL пароль пользователя `root` – пустая строка.

Пусть вас не пугает интерфейс командной строки, в действительности он не так сложен в использовании.

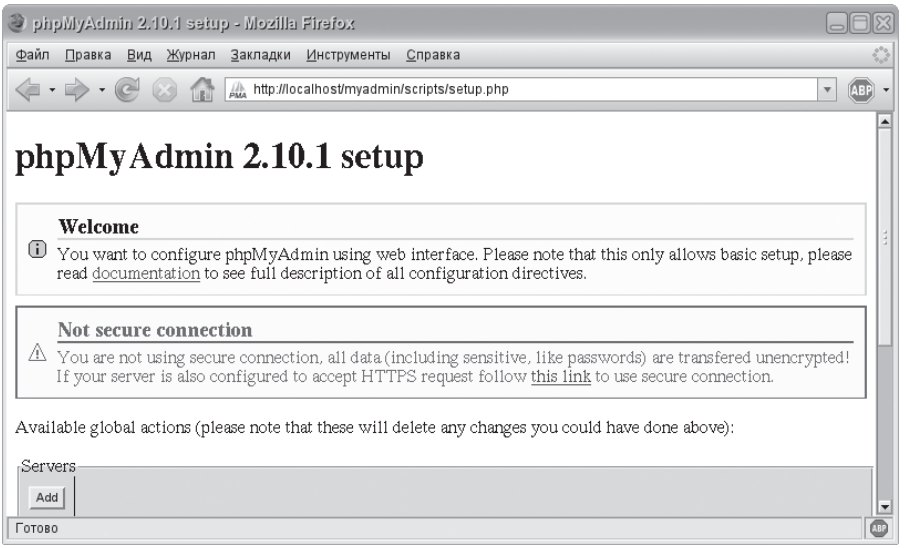

***Рис. 7.2.** Регистрация на сервере MySQL выполнена успешно*

Приглашения к вводу

В строке с приглашением к вводу вы можете вводить команды, отправляемые базе данных, и завершать их нажатием клавиши Enter. Помимо этого, есть ряд команд, интерпретируемых непосредственно сервером MySQL. Чтобы вывести список этих команд, напечатайте слово `help` или `\h` после приглашения `mysql>`. В табл. 7.1 описаны некоторые разновидности приглашения к вводу.

Таблица 7.1. Назначение различных приглашений к вводу

Приглашение к вводу	Назначение
`mysql>`	Ожидание ввода команды
`->`	Ожидание ввода следующей строки команды
`'>`	Ожидание продолжения строки, которая начинается с открывающей одиночной кавычки
`">`	Ожидание продолжения строки, которая начинается с открывающей двойной кавычки

Команды

В табл. 7.2 перечислены команды, которые можно вводить в строке приглашения MySQL.

Эти команды позволяют вам выполнять различные действия, например с помощью команды `source` можно запустить команды SQL из файла сценария.

Таблица 7.2. Команды клиента MySQL

Команда	Параметры	Назначение
`quit`		Завершение работы утилиты командной строки
`use`	Имя базы данных	Переход к использованию указанной базы данных
`show`	`tables` или `databases`	Вывод списка доступных таблиц или баз данных
`describe`	Имя таблицы	Вывод описания столбцов таблицы
`status`		Вывод номера версии и сведений о состоянии базы данных
`source`	Имя файла	Исполнение команд указанного файла как сценария

Чтобы вывести список доступных баз данных, введите команду:

```
mysql> SHOW DATABASES;
```

В нашем случае она вернет:

```
+----------+
| Database |
+----------+
| mysql    |
+----------+
1 rows in set (0.00 sec)
```

Чтобы вернуться к командам MySQL, исполнявшимся ранее, достаточно просто воспользоваться клавишей стрелка вверх.

По умолчанию сразу после установки доступна база данных с именем `mysql`. В этой базе данных уже есть информация об учетной записи пользователя. Не удаляйте ее! Если при запуске команде `mysql` не было передано имя базы данных, сделать это можно будет с помощью команды `USE`.

Чтобы подключиться к базе данных `mysql`, введите следующую команду в строке приглашения к вводу:

```
USE mysql;
```

Она должна вернуть сообщение:

```
Database changed
```

Если интернет-провайдер предоставил вам базу данных с иным именем, используйте его вместо `mysql`.

Управление базой данных

Теперь, когда мы подключились к базе данных, можно создавать учетные записи пользователей, свои базы данных и таблицы. Если вы пользуетесь услугами хостинга, то вам может не потребоваться создавать свою базу данных и учетную запись, так как интернет-провайдер, скорее всего, сделает это за вас.

Создание учетных записей

Для создания дополнительных учетных записей, помимо учетной записи по умолчанию для привилегированного пользователя `root`, следует использовать команду `grant`. Синтаксис команды `grant`:

```
GRANT PRIVILEGES ON DATABASE.OBJECTS TO 'имяпользователя'@'имяхоста' IDENTIFIED
BY 'пароль';
```

Например:

```
GRANT ALL PRIVILEGES ON *.* TO 'michele'@'localhost' IDENTIFIED BY 'secret';
```

Эта команда создаст учетную запись пользователя `michele`, обладающего полным доступом к локальным базам данных. Чтобы зарегистрироваться на сервере с привилегиями пользователя `michele`, нужно ввести команду:

```
exit
```

После этого необьходимо перезапустить MySQL из командной строки с новым именем пользователя и паролем. Для запуска MySQL с привилегиями определенного пользователя и паролем нужно выполнить команду:

```
mysql -h имяхоста -u имяпользователя -ппароль
```

Если вы не хотите предоставлять пользователям доступ к таблицам, кроме их собственных, замените символ `*` в строке `GRANT ALL PRIVILEGES ON *.* TO 'michele'` именем базы данных, принадлежащей пользователю, например:

```
GRANT ALL PRIVILEGES ON store.* TO 'michele'@'localhost' IDENTIFIED BY
'secret';
```

Эта команда должна запускаться с привилегиями пользователя `root` или любого другого, обладающего соответствующими правами. В вышеприведенной строке слово `store` соответствует имени базы данных, которую мы создадим в следующем разделе, и правом доступа к которой наделим пользователя.

Создание базы данных MySQL

Приступим к созданию базы данных с именем `store`. Создание баз данных выполняется с помощью команды `CREATE DATABASE`, которая выглядит примерно так:

```
CREATE DATABASE store;
```

Если все в порядке, вы получите примерно такой результат:

```
Query OK, 1 row affected (0.03 sec)
```

Имена баз данных не должны содержать пробелов. Кроме того, на таких серверах UNIX, как Linux и Mac OS X, имена баз данных чувствительны к регистру букв.

Чтобы перейти к использованию этой базы данных, введите:

```
USE store;
```

Вы должны получить следующий результат:

```
Database changed.
```

Если все сделано правильно, будет создана и выбрана для работы новая база данных. Очень важный этап – создание таблиц с данными, вскоре мы его рассмотрим.

Использование phpMyAdmin

Инструмент phpMyAdmin (*http://www.phpmyadmin.net/*) позволяет администрировать базу данных MySQL с помощью обычного веб-броузера. Все, что требуется для работы с этим инструментом, – это веб-сервер с установленным PHP и база данных MySQL, которую вы собираетесь администрировать.

Чтобы установить MySQL, нужно выполнить следующие действия:

1. Щелкните по ссылке Downloads (загрузить) на главной странице.
2. Загрузите файл архива, например *all-languages.tar.gz* (UNIX) или *all-languages.zip* (Windows).
3. Распакуйте архив (включая все подкаталоги) в каталог на вашем компьютере.
4. Перепишите их на сервер интернет-провайдера, где находится ваша учетная запись и смогут исполняться файлы PHP. Или, если вы развернули локальный сервер, перепишите их в подкаталог с «говорящим» именем, например *myadmin*, корневого каталога с документами веб-сервера.
5. Для хранения файлов с настройками phpMyAdmin создайте в каталоге *myadmin* подкаталог *config*. Чтобы установить права доступа, которые позволят программе изменять файлы с настройками, в операционной системе Linux вместо создания каталогов выполните следующие команды:

```
cd myadmin
mkdir config
chmod o+rw config
cp config.inc.php config/
chmod o+w config/config.inc.php
```

6. В своем броузере откройте страницу *http://localhost/myadmin/scripts/setup.php*. После этого вы увидите примерно такое окно, как на рис. 7.3.

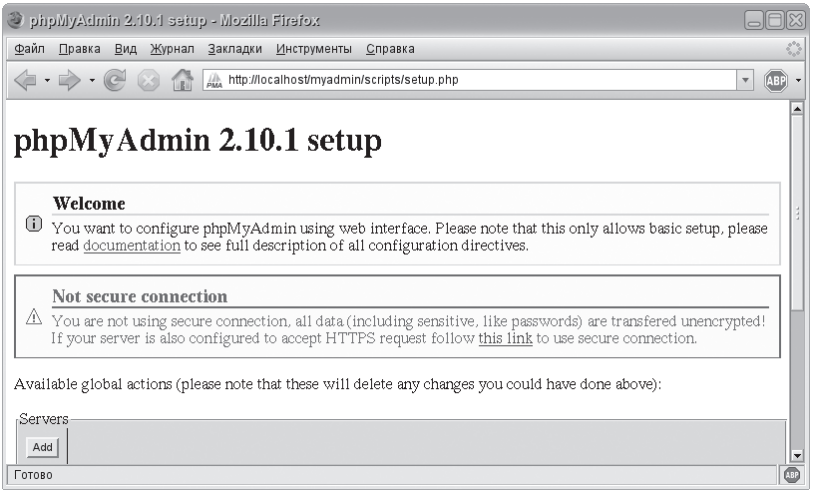

Рис. 7.3. *Программа phpMyAdmin создаст файл с настройками для phpMyAdmin*

7. Щелкните по кнопке Add (добавить) в разделе Servers (серверы). В окне броузера появится страница настройки сервера (рис. 7.4).
8. В большинстве параметров настройки можно оставить значения по умолчанию. Вы должны ввести в поле Password for config auth (пароль для аутентификации типа config) пароль пользователя root для MySQL.
9. В поле Authentication type (тип аутентификации) выберите значение cookie, чтобы разрешить доступ к MySQL только тем пользователям, которые имеют учетную запись MySQL.
10. Щелкните по кнопке Add (добавить).
11. Щелкните по кнопке Save (сохранить) в разделе Configuration (параметры настройки), чтобы сохранить изменения в файле настроек.
12. Скопируйте файл *config.inc.php* в каталог *myadmin*.
13. Удалите каталог *config*.
14. Откройте в броузере страницу *http://localhost/myadmin/index.php*. Перед вами должна появиться страница регистрации (рис. 7.5).
15. Чтобы подключиться к серверу MySQL, введите имя пользователя root и пароль.

Если вы устанавливали пакет **XAMPP и при попытке соединения получили сообщение об ошибке «The configuration file now needs a secret passphrase (blowfish_secret)» – в файле настроек** нужно указать пароль (параметр blowfish_secret), то вам следует в файле *phpmyadmin/config.inc.php* заменить строку `$cfg['blow-fish_secret'] = ''` **сторокой** `$cfg['blowfish_secret'] = 'value'`.

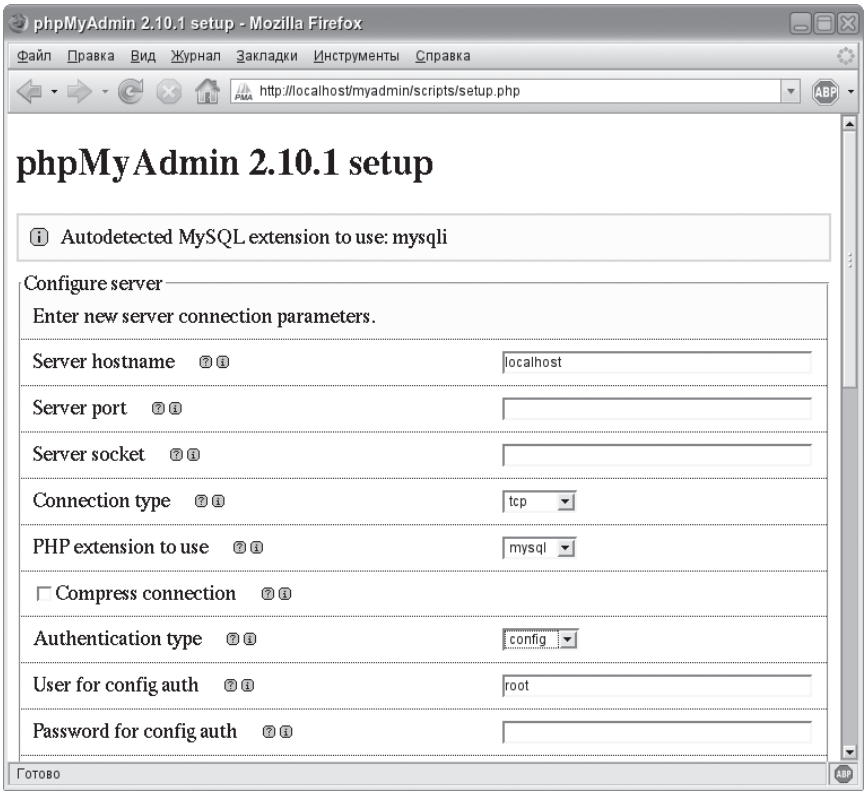

Рис. 7.4. *Определение параметров настройки соединения с сервером MySQL*

После установки и подключения к базе данных главная страница инструмента phpMyAdmin должна выглядеть примерно так, как показано на рис. 7.6, за исключением номера версии.

В раскрывающемся списке Database (База данных) можно выбрать любую из доступных баз данных. Инструмент администрирования позволяет увидеть параметры настройки базы данных и имеющиеся в ней объекты (например, таблицы), а также добавлять новые таблицы с помощью графического интерфейса. С помощью phpMyAdmin можно создавать новые базы данных и таблицы, запускать запросы и просматривать статистику работы сервера.

На рис. 7.7 показан список таблиц в тестовой базе данных test, которая будет создана нами далее в этой же главе. Если для своей базы данных вы выбрали другое имя, подставьте его вместо названия «test». Щелкнув, например, по имени таблицы `authors` слева, вы получите ее подробное описание.

После щелчка по имени таблицы `authors` появится описание структуры этой таблицы. Эта страница предоставляет возможность просматривать структуры базы данных, что особенно ценно, если база данных была создана не вами.

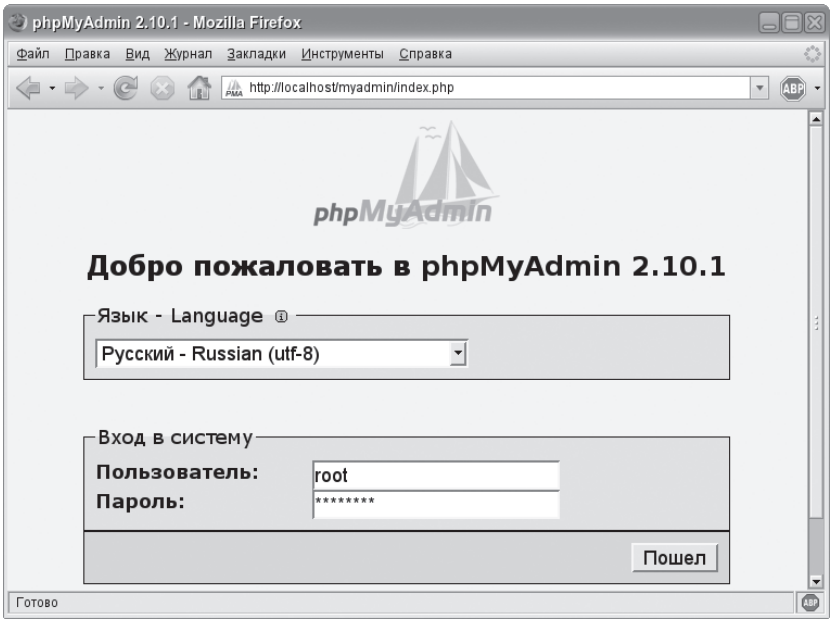

Рис. 7.5. *Страница регистрации, ограничивающая доступ к базе данных*

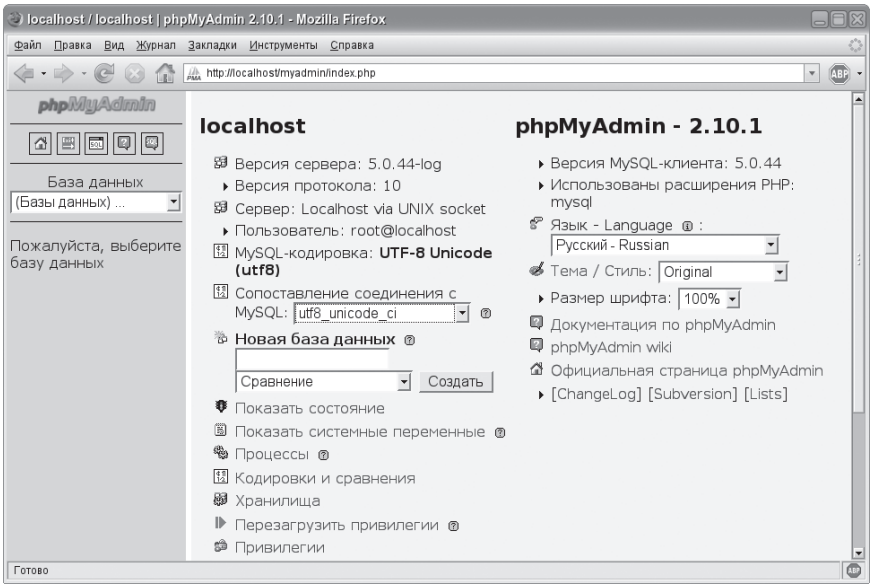

Рис. 7.6. *Выбор базы данных для администрирования в phpMyAdmin*

Чтобы просмотреть содержимое таблицы, щелкните по вкладке Browse (Обзор). На рис. 7.8 показано содержимое этой вкладки для таблицы `authors`.

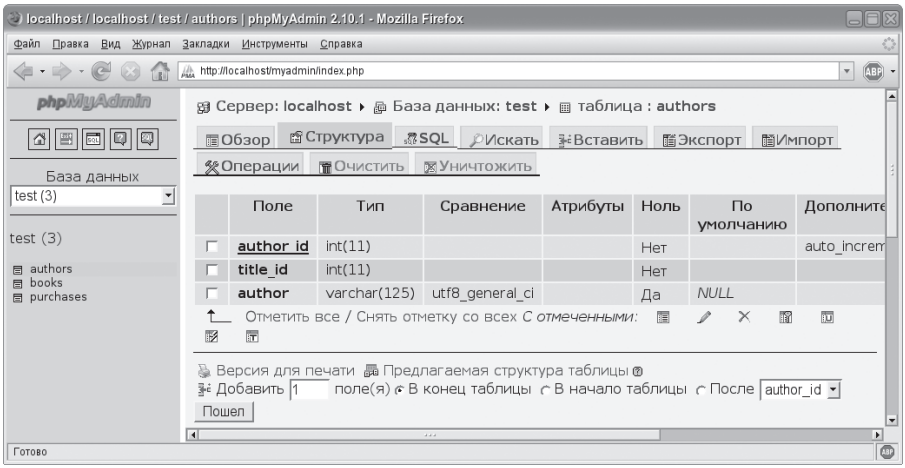

Рис. 7.7. *Объекты базы данных test и структура таблицы authors*

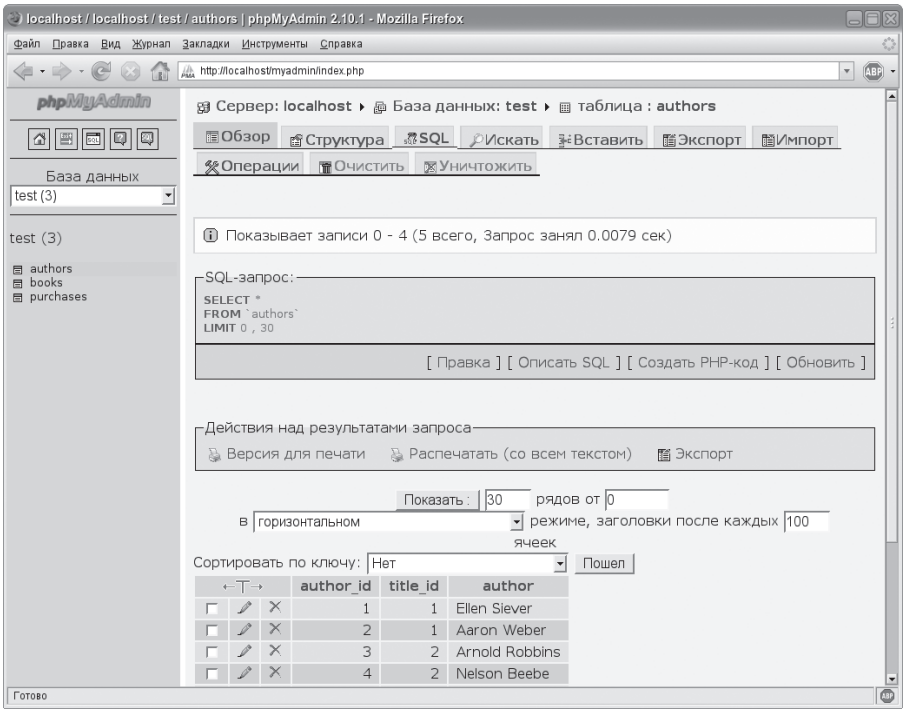

Рис. 7.8. *Данные из таблицы authors вместе с запросом, использованным для их получения*

Инструмент администрирования с веб-интерфейсом предоставляет несложный интерфейс как для просмотра содержимого базы данных, так и для создания новых объектов или модификации данных. Возможно,

графический интерфейс покажется вам удобнее текстового клиента командной строки `mysql`.

Теперь, чтобы ввести вас в курс дела, рассмотрим основную структуру базы данных. Начнем с знакомства с языком запросов SQL, предназначенным для организации взаимодействия с базой данных. Первый шаг на пути к созданию нашей базы данных – создание нескольких таблиц. После этого вы узнаете, как добавлять, просматривать и изменять данные.

Основные сведения о базах данных

Базы данных – это хранилища информации. Они обладают непревзойденными возможностями в управлении и манипулировании информацией. *Структурированная информация* – это способ организации взаимосвязанных данных, уже упоминавшийся в главах 3–6. К основным типам структурированной информации, которые также называются *структурами данных*, относятся:

- файлы;
- списки;
- массивы;
- записи;
- деревья;
- таблицы.

Каждая из этих базовых структур имеет массу разновидностей и позволяет выполнять разнообразные операции над данными. Чтобы проще было понять эту концепцию, можно представить себе телефонный справочник. Это самая распространенная база данных, которая содержит различные элементы информации – имя абонента, адрес и номер телефона, а в некоторых районах и другую информацию. Некоторые имена в телефонном справочнике могут быть выделены шрифтом, но записи, в основном, оформлены одинаково.

Если попытаться описать телефонный справочник в терминах базы данных, то вся книга представляет собой таблицу, которая содержит записи о каждом из абонентов. Каждая запись состоит из трех полей (также называемых столбцами, или атрибутами) – абонента, адреса и номера телефона. Записи идентифицируются по полю абонента, которое называется *ключевым полем*. Справочник отсортирован по фамилиям абонентов. На рис. 7.9 показано, как выглядят типичные записи и поля в базе данных, построенной по аналогии с телефонным справочником. Хотя записи в базе данных MySQL хранятся в случайном порядке, в запросе можно указать порядок сортировки.

Если взять эти данные из справочника и поместить их в базу данных, можно строить запросы, например, чтобы узнать, кому принадлежит номер

Абонент	Адрес	Номер телефона
Davis, Michele	7505 N. Linksway FxPnt 53217	414-352-4818
Meyer, Simon	5802 Beard Avenue S 55419	612-925-6897
Phillips, Jon	4204 Zenith Avenue S 55416	612-924-8020
Phillips, Peter	6200 Bayard Avenue HgldPk 55411	651-668-2251

Рис. 7.9. *Поля и записи в телефонном справочнике*

651-668-2251, или отыскать всех абонентов с именем Davis по заданному почтовому индексу. Этот тип баз данных напоминает большую таблицу, потому они называются *одноуровневыми* (flat-file) базами данных; это означает, что вся база данных содержится в единственной таблице. Начиная с 1970-х годов, одноуровневые базы данных были вытеснены *реляционными* базами данных, которые поддерживают возможность организации взаимоотношений между таблицами в зависимости от предъявляемых требований.

Язык структурированных запросов (SQL)

Теперь, определив таблицу, мы можем приступать к заполнению ее данными. MySQL будет следовать командам. Для манипулирования данными используются команды языка структурированных запросов (Structured Query Language, SQL). Этот язык проектировался с целью упростить описание взаимоотношений между таблицами и строками, поэтому базы данных используют его для модификации данных в таблицах.

SQL – это стандартный язык, используемый в таких базах данных, как MySQL, Oracle или Microsoft SQL Server. Он разрабатывался специально для извлечения, добавления и манипулирования данными, размещаемыми в базах данных. Подробнее с особенностями MySQL мы познакомим вас в главе 8, а пока рассмотрим лишь самые простые команды. Начнем с создания таблиц.

Каждая система управления базами данных привносит свои расширения в стандартный язык SQL. Например, команда `truncate` моментально удаляет все данные из таблицы. Она поддерживается многими базами данных, но при этом не является частью стандарта. Используйте ее очень осторожно, потому что `truncate` удаляет данные безвозвратно.

Создание таблиц

Для определения структуры новой таблицы базы данных служит команда `CREATE TABLE`. Когда создается таблица базы данных, каждый столбец может содержать дополнительные параметры, помимо имени и типа данных. Если при добавлении новой записи в таблицу поле не должно

оставаться пустым, в его определении указывается ключевое слово NOT NULL. Ключевое слово PRIMARY KEY определяет, какое поле будет использоваться в качестве первичного ключа. Автоматическое заполнение ключевого поля можно определить с помощью ключевого слова AUTO_INCREMENT.

Чтобы создать эти таблицы, выполните следующий код с помощью клиента командной строки MySQL. Если вам интересно опробовать SQL-код следующих примеров, прочитайте главу 8 – из нее вы узнаете, как обратиться к клиенту MySQL, как определить права доступа с помощью команды GRANT, как создать базу данных и выбрать нужную базу данных для работы.

Код примера 7.1 создаст таблицу books, в которой используются типы данных, описанные в табл. 8.10 ниже.

Пример 7.1. Создание таблиц books и authors

```
CREATE TABLE books (
  title_id INT NOT NULL AUTO_INCREMENT,
  title VARCHAR (150),
  pages INT,
PRIMARY KEY (title_id));

CREATE TABLE authors (
  author_id INT NOT NULL AUTO_INCREMENT,
  title_id INT NOT NULL,
  author VARCHAR (125),
PRIMARY KEY (author_id));
```

Если все в порядке, вы увидите примерно такие же результаты работы MySQL по созданию таблицы books, как показано в примере 7.2 (время исполнения запроса у вас может отличаться от 0,06 с).

Пример 7.2. Создание тестовых таблиц

```
mysql> CREATE TABLE books (
    -> title_id INT NOT NULL AUTO_INCREMENT,
    -> title VARCHAR (150),
    -> pages INT,
    -> PRIMARY KEY (title_id));
Query OK, 0 rows affected (0.06 sec)
mysql> CREATE TABLE authors (
    -> author_id INT NOT NULL AUTO_INCREMENT,
    -> title_id INT,
    -> author VARCHAR (125),
    -> PRIMARY KEY (author_id));
Query OK, 0 rows affected (0.06 sec)
```

Описание полученных результатов:

- первый столбец называется title_id и имеет целочисленный тип. Ключевое слово auto_increment обеспечивает уникальность значений этого поля, генерируемых автоматически в процессе вставки новых записей;

- столбец `title` предназначен для хранения текста длиной до 150 символов;
- столбец `pages` является целым числом;
- ключевое слово `PRIMARY KEY` сообщает MySQL, какое поле будет играть роль первичного ключа.

Первичный ключ должен быть уникальным и не может иметь значение `NULL`. Желательно, чтобы все таблицы имели первичный ключ, так как это позволит MySQL увеличить скорость доступа при извлечении данных из нескольких таблиц или поиске определенной строки с использованием ключа. В MySQL это делается с помощью специальной структуры данных, называемой индексом. *Индекс* в чем-то можно сравнить с ярлыком карточек библиотечного каталога. Проверить, насколько успешно прошло создание таблицы, можно с помощью команды `DESCRIBE`:

```
DESCRIBE books;
```

Эта команда вернет:

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| title_id | int(11)      | NO   | PRI | NULL    | auto_increment |
| title    | varchar(150) | YES  |     | NULL    |                |
| pages    | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

А команда :

```
describe authors;
```

выведет:

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| author_id | int(11)      | NO   | PRI | NULL    | auto_increment |
| title_id  | int(11)      | NO   |     |         |                |
| author    | varchar(125) | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

Все так, как мы указали в определении.

Обратите внимание: мы не указывали размеры столбцов, предназначенных для хранения целых чисел. В MySQL для этих целей по умолчанию используется 11 десятичных разрядов.

Добавление данных в таблицу

Для добавления данных предназначена команда `INSERT`. Используется она следующим образом: `INSERT INTO` *`table`* `COLUMNS` *`([столбцы])`* `VALUES (`*`[значения]`*`);`. Здесь видно, что в команде необходимо указать, в какую таблицу будут добавляться данные, и определить список значений. Если

перечень столбцов (COLUMNS) не указан, значения должны следовать в том же порядке, в каком определялись столбцы при создании таблицы (если вы не пропускаете какие-либо значения). Есть определенные правила, регламентирующие порядок заполнения базы данных с помощью команд SQL:

- числовые значения должны указываться без кавычек;
- строковые значения всегда должны быть в кавычках;
- значения даты и времени всегда должны быть в кавычках;
- функции должны указываться без кавычек;
- значение NULL никогда не должно заключаться в кавычки.

Наконец, если в строке отсутствует какое-либо значение, оно по умолчанию подразумевается равным значению NULL. Однако если поле не может иметь значение NULL (то есть когда оно было определено как NOT NULL), и вы не указали значение для этого поля, будет сгенерировано сообщение об ошибке.

Например:

```
INSERT INTO books VALUES (1,'Linux in a Nutshell',112);
INSERT INTO authors VALUES (NULL,1,'Ellen Siever');
INSERT INTO authors VALUES (NULL,1,'Aaron Weber');
```

Пока никаких ошибок не было, и вы должны получить:

```
mysql> INSERT INTO books VALUES (1,"Linux in a Nutshell",112);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO authors VALUES (NULL,1,"Ellen Siever");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO authors VALUES (NULL,1,"Aaron Weber");
Query OK, 1 row affected (0.00 sec)
```

При добавлении данных вы должны указывать все столбцы, даже если для некоторых из них значения отсутствуют. Хотя мы и не задаем значение поля author_id, позволяя MySQL сделать это за нас, следует оставить на его месте метку-заполнитель.

Добавляем другую книгу:

```
INSERT INTO books VALUES (2,'Classic Shell Scripting',256);
INSERT INTO authors VALUES (NULL,2,'Arnold Robbins');
INSERT INTO authors VALUES (NULL,2,'Nelson Beebe');
```

В результате этих действий в таблице books появятся две записи. Теперь, когда вы знаете, как создать таблицу и записать в нее данные, нужно научиться извлекать эту информацию.

Манипулирование определениями таблиц

Создав таблицу и начав заполнять ее информацией, вы можете обнаружить, что потребовалось изменить типы полей. Например, увеличить размер поля, вмещающего 30 символов, до 100 символов. Можно было

бы начать все с нуля, полностью переопределив таблицу, но при этом будут утеряны данные. К счастью, MySQL позволяет изменять типы полей без потери данных. В следующих примерах предполагается, что были созданы все таблицы, описываемые в этой главе.

Переименование таблицы

Чтобы переименовать таблицу, следует использовать команду `ALTER TABLE` *`имя_таблицы`* `RENAME` *`новое_имя_таблицы`*. Следующая команда переименует таблицу `books` в `publications`:

```
ALTER TABLE books RENAME publications;
```

Результат должен выглядеть так, как показано на рис. 7.10.

Рис. 7.10. *Переименование таблицы*

Изменение типа данных столбца

Чтобы изменить тип данных столбца, следует использовать команду `ALTER TABLE` *`имя_таблицы`* `MODIFY` *`имя_столбца тип_данных`*. Следующая команда изменит поле `author` таким образом, что оно будет вмещать до 150 символов.

```
ALTER TABLE authors MODIFY author VARCHAR(150);
```

Результат изменения типа данных столбца должен выглядеть, как показано на рис. 7.11.

Рис. 7.11. *Изменение типа данных столбца*

Кроме того, команда `MODIFY` может принимать два необязательных параметра, изменяющих порядок следования столбцов в таблице. С помощью ключевого слова `FIRST` можно сделать столбец первым в таблице, а с помощью ключевого слова `AFTER` *`имя_столбца`* – поместить столбец после указанного. Например, следующая команда разместит столбец `author` после столбца `author_id`:

```
ALTER TABLE authors MODIFY author varchar(125) AFTER author_id;
```

Необходимо указывать полное определение столбца, даже если оно не изменяется.

Добавление столбца

Добавить новый столбец позволяет команда `ALTER TABLE` *`имя_таблицы`* `ADD` *`имя_столбца тип_данных`*. Следующая команда добавит в таблицу `publications` столбец типа `TIMESTAMP`.

```
ALTER TABLE publications ADD time TIMESTAMP;
```

Результат работы команды приведен на рис. 7.12.

В этой команде, как и в конструкции `ALTER TABLE MODIFY`, можно определить позицию вставляемого столбца с помощью ключевых слов `FIRST` и `AFTER` *`имя_столбца`*.

Рис. 7.12. *Добавление столбца*

Переименование столбца

Чтобы переименовать столбец, следует использовать команду `ALTER TABLE` *`имя_таблицы`* `CHANGE` *`новое_имя_столбца старое_имя_столбца`*. Ниже приводится пример переименования столбца `author` в `author_name`. При работе с этой командой вы можете одновременно изменять определение столбца. Однако даже если определение столбца не изменяется, вам все же придется указывать его полное определение:

```
ALTER TABLE authors CHANGE author author_name varchar(125);
```

Результат работы команды приведен на рис. 7.13.

Рис. 7.13. *Переименование столбца*

Удаление столбца

Если спустя некоторое время вы решите, что какой-то столбец вам больше не нужен, его можно просто удалить. Чтобы удалить столбец, следует использовать команду `ALTER TABLE` *`имя_таблицы`* `DROP` *`имя_столбца`*. Следующая команда удалит столбец `pages`, после чего мы уже не сможем узнать, сколько страниц содержат книги, перечисленные в базе данных:

```
ALTER TABLE publications DROP COLUMN pages;
```

Результат работы команды приведен на рис. 7.14.

***Рис. 7.14.** Удаление столбца*

Удаление всей таблицы

Иногда требуется удалить и целую таблицу. Полное удаление таблицы со всеми данными выполняется с помощью команды `DROP`:

```
DROP TABLE test_table;
```

Будьте осторожны при удалении столбцов или таблиц. После выполнения операции данные будут безвозвратно утеряны, а отсутствие некоторых таблиц или столбцов может нарушить нормальную работу ваших программ.

Выполнение запросов к базе данных

Бестолку хранить данные в таблицах, если у вас нет возможности просматривать их. Данные извлекают с помощью команды `SELECT`, которой передаются имя таблицы и условия для выборки строк. Синтаксис команды `SELECT`: `SELECT` *`столбцы`* `FROM` *`таблицы`* `[WHERE условие отбора строк] [ORDER BY` *`порядок сортировки`*`];`.

Здесь *`столбцы`* – перечень имен полей, значения которых будут отбираться из *`таблиц`*. Необязательное предложение `WHERE` задает ограничение на отбор строк, другими словами, предложение `WHERE` ограничивает результаты, возвращаемые запросом. Например, строки могут быть отвергнуты запросом, если некоторое их поле не равно какому-либо значению, либо больше или меньше его. Предложение `ORDER BY` позволяет определить требуемый порядок сортировки информации, возвращаемой запросом. Если в команде `SELECT` указаны несколько таблиц и нет предложения `WHERE`, результирующий набор будет представлять *декартово* произведение, в котором каждая строка из первой таблицы будет возвращена со всеми строками из второй таблицы, то же самое – для второй строки и т. д. То есть объем результирующего набора данных будет огромным!

Самый простой запрос, позволяющий просмотреть содержимое таблицы, выглядит так:

```
SELECT * FROM books;
```

В нашем случае он выведет:

```
+----------+-------------------------+-------+
| title_id | title                   | pages |
+----------+-------------------------+-------+
|        1 | Linux in a Nutshell     |   112 |
|        2 | Classic Shell Scripting |   256 |
+----------+-------------------------+-------+
2 rows in set (0.01 sec)
```

Иногда в запросе вместо символа звездочки удобнее перечислить отбираемые столбцы:

```
SELECT author_id, title_id, author FROM authors;
```

Этот запрос выведет:

```
+-----------+----------+------------------+
| author_id | title_id | author           |
+-----------+----------+------------------+
|         1 |        1 | Ellen Siever     |
|         2 |        1 | Aaron Weber      |
|         3 |        2 | Arnold Robbins   |
|         4 |        2 | Nelson Beebe     |
+-----------+----------+------------------+
5 rows in set (0.01 sec)
```

Ограничение результатов с помощью предложения WHERE

Если вас интересует только книга `Classic Shell Scripting`, вы можете ограничить набор возвращаемых данных с помощью предложения `WHERE`:

```
SELECT * FROM books WHERE title="Classic Shell Scripting";
```

Этот запрос вернет:

```
+----------+-------------------------+-------+
| title_id | title                   | pages |
+----------+-------------------------+-------+
|        2 | Classic Shell Scripting |   256 |
+----------+-------------------------+-------+
1 row in set (0.00 sec)
```

Также можно просто перечислить столбцы таблицы, представляющие интерес:

```
SELECT books.pages FROM books WHERE title="Classic Shell Scripting";
```

Этот запрос вернет:

```
+-------+
| pages |
+-------+
|   256 |
+-------+
1 row in set (0.00 sec)
```

Условные выражения должны следовать за ключевым словом `WHERE`. С помощью логических операторов `AND` и `OR` в предложении `WHERE` можно определить сразу несколько условий. Порядок исполнения логических операторов изменяется с помощью круглых скобок `()`.

Рано или поздно вам понадобится извлечь данные из нескольких таблиц в одном запросе. Поэтому следует взять в привычку использовать полные имена столбцов в формате: `ТАБЛИЦА.СТОЛБЕЦ`. Это позволит избежать путаницы при выборке столбцов с одинаковыми именами из разных таблиц. Например, поле `description` может присутствовать в двух таблицах, и если не указать полное имя столбца, будет непонятно, поле которой таблицы имеется в виду.

Определение порядка сортировки

Как уже говорилось, изменить порядок сортировки результирующего набора данных позволяет ключевое слово `ORDER BY`. По умолчанию `ORDER BY` задает сортировку в порядке возрастания, поэтому для сортировки списка авторов книг в алфавитном порядке можно просто указать `ORDER BY author`. Чтобы назначить противоположный порядок сортировки, следует добавить ключевое слово `DESC` после имени поля `author`. Например, получить список авторов, отсортированный в алфавитном порядке, можно следующим запросом:

```
SELECT * FROM authors ORDER BY author;
```

Он выведет:

```
+-----------+----------+-------------------+
| author_id | title_id | author            |
+-----------+----------+-------------------+
|         2 |        1 | Aaron Weber       |
|         5 |        9 | Alex Martelli     |
|         3 |        2 | Arnold Robbins    |
|         1 |        1 | Ellen Siever      |
|         4 |        2 | Nelson Beebe      |
+-----------+----------+-------------------+
```

Далее мы попробуем сделать выборку данных сразу из нескольких таблиц.

Соединение таблиц

Инструкция `SELECT` позволяет выполнять запросы сразу к нескольким таблицам. В примере 7.3 создается таблица `purchases` (покупки), в которую добавляются несколько строк для примера.

Пример 7.3. SQL-запрос, который создает и заполняет таблицу покупок, связывая поле purchase_id (покупка) с полями user_id (пользователь) и title_id (название книги)

```
CREATE TABLE purchases (
    purchase_id int NOT NULL AUTO_INCREMENT,
    user_id varchar(10) NOT NULL,
    title_id int(11) NOT NULL,
    purchased timestamp NOT NULL default CURRENT_TIMESTAMP,
    PRIMARY KEY (purchase_id));
    INSERT INTO `purchases` VALUES (1, 'mdavis', 2, '2005-11-26 17:04:29');
    INSERT INTO `purchases` VALUES (2, 'mdavis', 1, '2005-11-26 17:05:58');
```

В результате исполнения примера 7.3 получим:

```
SELECT * FROM purchases;
+-------------+---------+----------+---------------------+
| purchase_id | user_id | title_id | purchased           |
+-------------+---------+----------+---------------------+
|           1 | mdavis  |        2 | 2005-11-26 17:04:29 |
|           2 | mdavis  |        1 | 2005-11-26 17:05:58 |
+-------------+---------+----------+---------------------+
2 rows in set (0.00 sec)
```

Чтобы сформировать запрос для получения списка всех купленных книг с указанием автора и числа страниц, введите следующую команду SELECT:

```
SELECT books.*, author FROM books, authors WHERE books.title_id = authors.
title_id;
```

В результате вы получите:

```
+----------+-------------------------+-------+--------------------+
| title_id | title                   | pages | author             |
+----------+-------------------------+-------+--------------------+
|        1 | Linux in a Nutshell     |   112 | Ellen Siever       |
|        1 | Linux in a Nutshell     |   112 | Aaron Weber        |
|        2 | Classic Shell Scripting |   256 | Arnold Robbins     |
|        2 | Classic Shell Scripting |   256 | Nelson Beebe       |
+----------+-------------------------+-------+--------------------+
4 rows in set (0.00 sec)
```

Часть запроса books.*, author сообщает о необходимости выбрать все поля из таблицы books и единственное поле author из таблицы authors. Часть запроса WHERE books.title_id = authors.title_id связывает таблицы по полю title_id.

Вы могли бы определить список отбираемых столбцов как (*), тогда в результирующий набор попали бы все поля обеих таблиц, но в этом случае поле title_id было бы включено дважды, поскольку оно имеется в обеих таблицах. Количество соединяемых столбцов и таблиц не ограничено.

Естественные соединения

Получить те же результаты, но меньше вводя с клавиатуры, позволяет ключевое слово NATURAL JOIN. При выполнении естественного соединения MySQL автоматически соединяет одноименные поля двух таблиц. В нашем случае это поле title_id. Естественное соединение достаточно сообразительно, чтобы в следующем запросе не выводить поле title_id дважды, и при этом вывести поле author_id таблицы authors:

```
SELECT * FROM books NATURAL JOIN authors;
```

Результат будет таким:

```
+----------+-------------------------+-------+-----------+--------------------
| title_id | title                   | pages | author_id | author
|
+----------+-------------------------+-------+-----------+--------------------
|        1 | Linux in a Nutshell     |   112 |         1 | Ellen Siever
|
|        1 | Linux in a Nutshell     |   112 |         2 | Aaron Weber
|
|        2 | Classic Shell Scripting |   256 |         3 | Arnold Robbins
|
|        2 | Classic Shell Scripting |   256 |         4 | Nelson Beebe
|
+----------+-------------------------+-------+-----------+--------------------
4 rows in set (0.00 sec)
```

Конструкция JOIN ON

Конструкция `JOIN ON` похожа на инструкцию естественного соединения, но предоставляет возможность явно определить поля, по которым следует выполнять соединение, не полагаясь на автоматический выбор по их именам. Эта конструкция имеет следующий синтаксис: `SELECT` *столбцы* `FROM` *имя_таблицы* `JOIN` *таблицы* `ON (`*условия*`)`. Например, запрос `SELECT * FROM books JOIN authors ON (books.title_id = authors.title_id);` вернет тот же набор строк, что и предыдущий запрос, выполняющий естественное соединение.

Псевдонимы

Перечисляя таблицы в запросе, используйте псевдонимы (aliases). Чтобы определить псевдоним таблицы, нужно после ее полного имени поставить ключевое слово `AS` и затем указать псевдоним. Например, присвоим в запросе таблице `books` псевдоним `b`, а таблице `purchases` псевдоним `p`:

```
SELECT * FROM books AS p,authors AS b WHERE b.title_id = p.title_id;
```

В результате получим:

```
+--------+------------------------+-----+---------+----------+-------------+
|title_id| title                  |pages|author_id| title_id | author
+--------+------------------------+-----+---------+----------+-------------+
|      1 | Linux in a Nutshell    | 112 |       1 |        1 | Ellen Siever
|      1 | Linux in a Nutshell    | 112 |       2 |        1 | Aaron Weber
|       2 | Classic Shell Scripting| 256 |         3 |          2 | Arnold
Robbins
|      2 | Classic Shell Scripting| 256 |       4 |        2 | Nelson Beebe
+--------+------------------------+-----+---------+----------+-------------+
4 rows in set (0.00 sec)
```

Определив псевдоним таблицы, можно обращаться к ней по псевдониму в любом месте запроса. Псевдонимы удобны в качестве подмены длинных имен таблиц. Кроме того, они позволяют дважды включать в запрос одну и ту же таблицу и определять, в каком случае какой экземпляр таблицы следует использовать.

Модификация данных в базе данных

Если вы допустили ошибку при вводе данных, например указали неверное число страниц в книге, ошибку можно исправить с помощью команды `UPDATE`. Для внесения изменений в таблицы есть много причин, например изменение пароля пользователя.

В команде `UPDATE` используется то же предложение `WHERE`, что и в инструкции `SELECT`, но в ней присутствует команда `SET`, с помощью которой определяется новое значение столбца.

Если вы забудете включить предложение `WHERE` в команду `UPDATE`, она изменит все записи в таблице.

Например, обновим таблицу books:

```
UPDATE books SET pages = 476 WHERE title = 'Linux in a Nutshell';
```

Этот запрос вернет:

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Данный запрос изменит значение поля pages для всех книг с названием «Linux in a Nutshell» в таблице books, установив его равным значению 476. Этот прием позволяет исправлять ошибочные данные.

```
SELECT * FROM books;
```

На этот раз запрос вернет:

```
+----------+-------------------------+-------+
| title_id | title                   | pages |
+----------+-------------------------+-------+
|        1 | Linux in a Nutshell     |   476 |
|        2 | Classic Shell Scripting |   256 |
+----------+-------------------------+-------+
2 rows in set (0.00 sec)
```

Удаление данных из базы

Команда DELETE удаляет строки или записи из таблицы. В команде DELETE используется то же предложение WHERE, что и в инструкции UPDATE: удаляются все строки, соответствующие условию. В случае отсутствия предложения WHERE будут удалены все записи в таблице.

Прежде чем воспользоваться командой DELETE, не забудьте создать резервные копии своих данных, в противном случае вы рискуете потерять все данные, нажив кучу неприятностей.

В следующем примере из базы данных будут удалены все книги, автором которых является Ellen Siever:

```
DELETE FROM books WHERE author= "Ellen Siever";
```

Функции поиска

Как вы заметили в предыдущих примерах, MySQL обладает возможностью отыскивать конкретные данные. Однако мы пока еще не рассматривали синтаксис поиска. В MySQL роль шаблонного символа исполняет символ (%), используемый совместно с ключевым словом LIKE. То есть этим символом можно буквально представить все, что угодно. Это напоминает поиск файлов в Проводнике Windows по строке *.doc – будут найдены все файлы документов, независимо от имен. По умолчанию поиск выполняется без учета регистра букв.

Например, выполнить общий поиск можно с помощью следующего синтаксиса:

```
SELECT * FROM authors WHERE author LIKE "%b%";
```

В результате будет получено:

```
+-----------+----------+------------------+
| author_id | title_id | author           |
+-----------+----------+------------------+
|         2 |        1 | Aaron Weber      |
|         3 |        2 | Arnold Robbins   |
|         4 |        2 | Nelson Beebe     |
+-----------+----------+------------------+
3 rows in set (0.00 sec)
```

Этот запрос нашел все записи, в значении поля `author` которых есть символ (`b`). Заметим, что здесь мы использовали два символа (`%`), окружив ими символ (`b`) – (`%b%`). Это означает, что до и после искомого символа может быть что угодно. Если хотите, можете использовать только один шаблонный символ – жесткого правила на этот счет нет.

Символ (`%`), помещенный в любое место строки в инструкции `LIKE`, означает, что на этом месте в строке может быть что угодно.

Еще один шаблонный символ – символ подчеркивания (_). Он соответствует любому единственному символу. С использованием этого шаблонного символа можно выполнить такой поиск:

```
SELECT * FROM authors WHERE author like 'Aaron Webe_'
```

В результате будут получены все строки, где имя автора начинается с «Aaron Webe» и кончается любым символом.

Логические операторы

В предложении `WHERE` можно использовать те же логические операторы, что и в условных логических конструкциях языка PHP.

В предложении `WHERE` запроса вы можете использовать операторы `AND`, `OR` и `NOT`:

```
SELECT * FROM authors WHERE NOT (author = "Ellen Siever" );
```

Этот запрос вернет все записи, кроме тех, где в качестве автора указана Ellen Siever. Круглые скобки просто увязывают оператор `NOT` с операцией сравнения, в запросе они необязательны.

Этот запрос возвращает информацию о книге и авторе:

```
SELECT *
  FROM books, authors
WHERE title = "Linux in a Nutshell"
  AND author = "Aaron Weber"
  AND books.title_id = authors.title_id;
```

Этот запрос вернет все записи, где указан автор Aaron Weber или Ellen Siever.

```
SELECT *
  FROM books, authors
WHERE (author = "Aaron Weber"
       OR author = "Ellen Siever")
      AND books.title_id=authors.title_id
```

В этом запросе круглые скобки очень важны, потому что с их помощью указывается, что оператор `OR` в условии выбора автора должен выполниться *раньше*, чем оператор `AND` в условии соединения таблиц по автору и названию книги.

К настоящему моменту получены все необходимые начальные сведения. В следующей главе мы рассмотрим основные принципы проектирования баз данных, способы резервного копирования и расширенные возможности языка SQL. Мы успешно продвигаемся к созданию блога (сетевого дневника), которым заканчивается книга.

Вопросы к главе 7

Вопрос 7.1

Какую команду нужно ввести в командной строке, чтобы получить доступ к MySQL? (Путь к каталогу *bin*, в который установлены исполняемые файлы MySQL, записан в системную переменную окружения `PATH`.)

Вопрос 7.2

Создайте таблицу с именем `month`, в которую будут записаны названия месяцев и количество дней в каждом из них.

Вопрос 7.3

Заполните таблицу `month` с помощью инструкций `INSERT`.

Вопрос 7.4

Напишите инструкцию `SELECT`, выводящую информацию о месяцах.

Вопрос 7.5

Напишите инструкцию `SELECT`, выводящую информацию только о месяцах, в которых 28 дней.

Вопрос 7.6

Напишите запрос, который выводил бы список месяцев, названия которых заканчиваются на «брь».

Ответы на эти вопросы приводятся в разделе «Глава 7» приложения.