第 9 章　创建Web数据库

在本章中，我们将介绍如何建立一个能够在Web站点上使用的MySQL数据库。

在本章中，我们将主要介绍以下内容：

- 创建一个数据库
- 设置用户权限
- 权限系统的介绍
- 创建数据库表
- 创建索引
- 选择MySQL中的列类型

在本章中，我们仍将以第8章所介绍的Book-O-Rama在线书店应用程序为例。以下给出了Book-O-Rama应用程序的数据库模式：

```
Customers(CustomerID, Name, Address, City)

Orders(OrderID, CustomerID, Amount, Date)

Books(ISBN, Author, Title, Price)

Order_Items(OrderID, ISBN, Quantity)

Book_Reviews(ISBN, Reviews)
```

请记住，带下画线的是主键而斜体的是外键。

为了使用本节中这些内容，必须具有访问MySQL的权限。通常，这就意味着已经在Web服务器上完成了MySQL的基本安装。这些安装包括：

- 安装文件。
- 为MySQL创建一个用户，并且以该用户身份运行所创建的数据库。
- 设置路径。
- 如果需要，运行`mysql_install_db`。
- 设置`root`密码。
- 删除匿名用户和仅供测试使用的数据库。
- 启动MySQL服务器并将其设置为自动运行。

如果已经完成了以上这些操作，就可以继续本章的学习了。如果尚未完成这些操作，可以在附录A中找到完成这些安装的说明。

如果在学习本章内容时遇到任何问题，可能是由于MySQL系统安装不正确。如果发生这种情况，可以回到上一步，查看附录A以确认安装是否正确。

拥有访问安装在某台机器上的MySQL数据库的权限，不需要具有管理该机器的权限。该机器可以是Web主机服务器，也可以是工作室中的一台机器等。

如果你具有这样权限，而且要使用本例或者创建你自己的数据库，就应该让管理员为你创建一个用户和将要使用的数据库，并告诉你用户名、密码、以及他们分配给你的数据库名。

读者可以跳过本章关于如何创建用户和数据库的介绍，但阅读这些内容可以更好地向系统管理员解释需求。一个普通用户不能够执行这些命令来创建用户和数据库。

本章给出的例子都是在MySQL的最新版本5.1下创建并测试的。MySQL早些版本支持的功能较少。应该安装或者升级到目前最新的稳定版本。可以从MySQL站点下载MySQL的最新版本：http://mysql.com。

在本书中，我们使用命令行客户端工具与MySQL进行交互，该工具叫做MySQL监视器，它会在MySQL的每一个安装中出现。但是，也可以使用其他客户端工具。例如，如果在主机托管的Web环境使用MySQL，系统管理员通常会提供基于浏览器的phpMyAdmin工具。不同的GUI客户端工具与我们这里结果的操作存在差异，但是我们可以很快掌握那些工具所提供的功能。

9.1 使用MySQL监视程序

在本章和下一章的MySQL例子中，每个命令之间都用分号（;）分开，分号将告诉MySQL执行这个命令。如果漏掉了这些分号，MySQL将不会执行这些命令。对于新用户来说，这是一个非常常见的问题。

漏掉分号的结果是：可能在一个命令中间添加新行。我们使用这样的模式是为了使得本例更容易阅读。因为MySQL提供了一个持续符号，你将看到我们在什么地方使用了这个方法。持续符号是一个箭头，如下所示：

```
mysql> grant select
    ->
```

这个符号表示MySQL期待着更多的输入。每次按Enter键时都会出现这些提示符，直到输入分号才没有提示符。

另外还需要注意的一点是SQL语句不区分大小写，但数据库和表的名称则区分大小写（我们将在后面的内容详细介绍）。

9.2 登录到MySQL

要完成登录操作，首先要进入机器的命令行界面并输入如下所示的命令：

```
mysql -h hostname -u username -p
```

mysql命令将调用MySQL监视程序。这是一个可以将我们连接到MySQL服务器的客户端命令行工具。

-h命令选项用于指定所希望连接的主机，即运行MySQL服务器的机器。如果正在该MySQL服务器所运行的机器上运行该命令，可以忽略该选项和hostname参数。如果不是，必

须用运行MySQL服务器的主机名称来代替主机名称参数。

-u命令选项用于指定连接数据库时使用的用户名称。如果不指定，默认值是登录该操作系统时使用的用户名。

如果你在自己的机器或服务器上安装了MySQL，必须以root身份进行登录并且创建本节中将使用到的数据库。假设已经安装了MySQL数据库，而且root用户是进行各项操作的唯一用户。如果在其他人管理的机器上使用MySQL，必须使用他们提供的用户名。

-p命令选项用来告诉服务器要使用一个密码来连接它。如果登录时使用的用户名没有设置密码，可以忽略此选项。

如果以root用户的身份登录并且没有设置root密码，我们强烈建设参阅附录A，马上参阅！没有root密码，系统是不安全的。

我们不必在本行命令中包含密码，MySQL服务器会向你询问密码的。实际上，没有这样做更好。如果在命令行输入密码，它将以普通文本方式显示在屏幕上，很容易被其他用户发现。

在输入前述命令之后，会得到如下响应：

```
Enter password:
```

(如果这行命令没有出现，请确认MySQL服务器是否正在运行，并且上述mysql命令应该包含在路径中。)

必须输入密码。如果一切顺利，将得到类似如下所示的响应：

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.1.25-rc-community MySQL
Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c to clear the buffer.

mysql>
```

在你自己的机器上，如果没有得到类似的响应，请确认mysql_install_db是否已经运行（如果需要的话），是否设置了root用户密码，并确认输入的密码是否正确。

我们现在应该位于MySQL命令提示符下，可以开始创建数据库了。如果使用的是我们自己的机器，需要遵循下一节给出的说明。如果使用的是别人的机器，这些操作已经设置完成了。可以直接进入到9.7节。你可能希望阅读相关章节以了解更多背景知识，但是不能够运行那些命令（或者至少不应该能够那样！）。

9.3　创建数据库和用户

MySQL数据库系统可以支持许多不同的数据库。通常，每个应用程序需要一个数据库。在Book-O-Rama例子中，数据库名为books。

创建数据库是最容易的部分。在MySQL命令提示符下，输入如下所示命令：

```
mysql> create database dbname;
```

应该用所希望的数据库名称来代替"dbname"字符串。在Book-O-Rama例子中，我们要创

建一个名为books的数据库。

就这样，你应该能够看到如下所示的响应（执行时间会因为机器不同而不同）：

```
Query OK, 1 row affected (0.0 sec)
```

这意味着一切正常。如果没有得到该响应，请确认在上面的命令行后面输入了分号。分号将告诉MySQL已经完成了命令输入，它应该执行该命令了。

9.4 设置用户与权限

一个MySQL系统可能有许多用户。为了安全起见，root用户通常只用作管理目的。对于每个需要使用该系统的用户，应该为他们创建一个账号和密码。这些用户名和密码不必与MySQL之外的用户名称和密码（例如，UNIX或NT用户名和密码）相同。同样的原则也适合于root用户。对于系统用户和MySQL用户最好使用不同的密码，这一点对root用户尤其应该这样。

为用户设置密码不是必需的，但是我们强烈建议为所有创建的用户设定密码。要建立一个Web数据库，最好为每个网站应用程序建立一个用户。我们可能会问，“为什么要这么做呢？”——答案在于权限。

9.5 MySQL权限系统的介绍

MySQL的最好特性之一是支持复杂的权限系统。权限是对特定对象执行特定操作的权力，它与特定用户相关。其概念非常类似于文件的权限。当在MySQL中创建一个用户时，就赋予了该用户一定的权限，这些权限指定了该用户在本系统中可以做什么和不可以做什么。

9.5.1 最少权限原则

最少权限原则可以用来提高任何计算机系统的安全性。它是一个基本的、但又是非常重要的而且容易为我们忽略的原则。该原则包含如下内容：

一个用户（或者一个进程）应该拥有能够执行分配给他的任务的最低级别的权限。

该原则同样适用于MySQL，就像它应用于其他地方一样。例如，要在网站上运行查询，用户并不需要root用户所拥有的所有权限。因此，我们应该创建另一个用户，他只有访问我们刚刚建立的数据库的必要权限。

9.5.2 创建用户：GRANT命令

GRANT和REVOKE命令分别用来授予和取消MySQL用户的权限，这些权限分4个级别。它们分别是：

- 全局
- 数据库
- 表
- 列

稍后，我们将详细介绍如何应用每个权限。

GRANT命令用来创建用户并赋予他们权限。GRANT命令的常见形式是：

```
GRANT privileges [columns]
ON item
TO user_name [IDENTIFIED BY 'password']
[REQUIRE ssl_options]
[WITH [GRANT OPTION | limit_options] ]
```

方括号内的子句是可选的。在本语法中，出现了许多占位符。第一个占位符是privileges，应该是由逗号分开的一组权限。MySQL已经有一组已定义的权限。它们在下一节详细介绍。

占位符columns是可选的。可以用它对每一个列指定权限。也可以使用单列的名称或者用逗号分开的一组列的名称。

占位符item是新权限所应用于的数据库或表。可以将项目指定为*.*，而将权限应用于所有数据库。这叫做赋予全局权限。如果没有使用在何特定的数据库，也可以通过只指定*完成赋予全局权限。更常见的是，以dbname.*的形式指定数据库中所有的表，以dbname.tablename的形式指定单个表，或者通过指定tablename来指定特定的列。这些分别表示其他3个可以利用的权限：数据库、表、列。如果在输入命令的时候正在使用一个数据库，tablename本身将被解释成当前数据库中的一个表。

user_name应该是用户登录MySQL所使用的用户名。请注意，它不必与登录系统时所使用的用户名相同。MySQL中的user_name也可以包含一个主机名。可以用它来区分如laura（解释成laura@localhost）和laura@somewhere.com。这是非常有用的一项能力，因为来自不同域的用户经常可能使用同一个名字。这也提高了安全性能，因为可以指定用户从什么地方连接到本机，甚至可以指定他们在特定的地方可以访问哪些表和数据库。

password应该是用户登录时使用的密码。常见的密码选择规则在这里都适用。我们后面将更详细地讲述安全问题，但是密码应该不容易被猜出来。这意味着，密码不应该是一个字典单词或与用户名相同。理想的密码应该是大、小写字母和非字母的组合。

REQUIRE子句允许指定用户是否必须通过加密套接字连接，或者指定其他的SSL选项。关于SSL到MySQL连接的更多信息，请参阅MySQL手册。

WITH GRANT OPTION选项，如果指定，表示允许指定的用户向别人授予自己所拥有的权限。

我们也可以指定如下所示的WITH子句：

```
MAX_QUERIES_PER_HOUR n
```

或者

```
MAX_UPDATES_PER_HOUR n
```

或者

```
MAX_CONNECTIONS_PER_HOUR n
```

这些子句可以指定每一个用户每小时执行的查询、更新和连接的数量。在共享的系统上限制单个用户的负载时，这些子句是非常有用的。

权限存储在名为mysql的数据库中的5个系统表中。这些表分别是mysql.user、mysql.db、

mysql.host、mysql.tables_priv和mysql.columns_priv。作为GRANT命令的替代，可以直接修改这些表。我们将在第12章中更详细讨论它们。

9.5.3 权限的类型和级别

MySQL中存在3个基本类型的权限：适用于赋予一般用户的权限、适用于赋予管理员的权限和几个特定的权限。任何用户都可以被赋予这3类权限，但是根据最少权限原则，最好严格限定只将管理员类型的权限赋予管理员。

我们应该只赋予用户访问他们必须使用的数据库和表的权限。而不应该将访问mysql的权限赋予不是管理员的人。mysql数据库是所有用户名、密码等信息存储的地方。（我们将在第12章详细介绍该数据库）。

常规用户的权限直接与特定的SQL命令类型以及用户是否被允许运行它们相关。我们将在下一章中详细讨论这些SQL命令。这里，我们将对这些权限所能实现的操作的概念性描述。表9-1所示的是基本用户权限。“应用于”列下面的对象给出了该类型权限可以授予的对象。

表9-1　用户的权限

权　限	应 用 于	描　述
SELECT	表，列	允许用户从表中选择行（记录）
INSERT	表，列	允许用户在表中插入新行
UPDATE	表，列	允许用户修改现存表里行中的值
DELETE	表	允许用户删除现存表的行
INDEX	表	允许用户创建和拖动特定表索引
ALTER	表	允许用户改变现存表的结构，例如，可添加列、重命名列或表、修改列的数据类型
CREATE	数据库，表	允许用户创建新数据库或表。如果在GRANT中指定了一个特定的数据库或表，他们只能够创建该数据库或表，即他们必须首先删除（drop）它
DROP	数据库，表	允许用户拖动（删除）数据库或表

从系统的安全性方面考虑，适于常规用户的权限大多数是相对无害的。ALTER权限通过重命名表可能会影响权限系统，但是大多数用户需要它。安全性常常是可用性与保险性的折中。遇到ALTER的时候，应当作出自己的选择，但是通常还是会将这个权限授予用户。

除了表9-1给出的权限，还有两种权限：REFERENCES权限和EXECUTE权限，但是如今已经不再使用了，而GRANT权限是以WITH GRANT OPTION选项给出，而不是在权限列表里列出的。

表9-2给出了适用于管理员用户使用的权限。
可以将这些权限授予非管理员用户，这样做的时候要非常小心。

FILE权限有些不同，它对普通用户非常有用，因为它可以将数据从文件载入数据库，从而可以节省许多时间，否则，每次将数据输入数据库都需要重新输入，这很浪费时间。

然而，文件载入可以用来载入MySQL可识别的任何文件，包括属于其他用户的数据库和潜在的密码文件。授予该权限的时候需要小心，或者自己为用户载入数据。

此外，还存在两个特别的权限，如表9-3所示。

表9-2　管理员权限

权　限	描　述
CREATE TEMPORARY TABLES	允许管理员在CREATE TABLE语句中使用TEMPORARY关键字
FILE	允许将数据从文件读入表，或从表读入文件
LOCK TABLES	允许使用LOCK TABLES语句
PROCESS	允许管理员查看属于所有用户的服务器进程
RELOAD	允许管理员重新载入授权表、清空授权、主机、日志和表
REPLICATION CLIENT	允许在复制主机（Master）和从机（Slave）上使用SHOW STATUS。复制将在第12章详细介绍
REPLICATION SLAVE	允许复制从服务器连接到主服务器。复制将在第12章详细介绍
SHOW DATABASES	允许使用SHOW DATABASES语句查看所有数据库列表。没有这个权限，用户只能看到他们能够看到的数据库
SHUTDOWN	允许管理员关闭MySQL服务器
SUPER	允许管理员关闭属于任何用户的线程

表9-3　特别的权限

权　限	描　述
ALL	授予表9-1和表9-2列出的所有权限。也可以将ALL写成ALL PRIVILEGES
USAGE	不授予权限。这创建一个用户并允许他登录，但是不允许进行任何操作。通常在以后会授予该用户更多的权限

9.5.4　REVOKE命令

与GRANT相反的命令是REVOKE。它用来从一个用户收回权限。在语法上它与GRANT非常相似：

```
REVOKE privileges [( columns )]
ON item
FROM user_name
```

如果已经给出了WITH GRANT OPTION子句，可以按如下方式撤销它（以及所有其他权限）：

```
REVOKE All PRIVILEGES, GRANT
FROM user_name
```

9.5.5　使用GRANT和REVOKE的例子

要创建一个管理员，可以输入如下所示命令：

```
mysql> grant all
    -> on *
    -> to fred identified by 'mnb123'
    -> with grant option;
```

以上命令授予了用户名为Fred、密码为mnb123的用户使用所有数据库的所有权限，并允许他向其他人授予这些权限。

如果不希望用户在系统中存在，可以按如下方式撤销：

```
mysql> revoke all privileges, grant
    -> from fred;
```

现在，我们可以按如下方式创建一个没有任何权限的常规用户：

```
mysql> grant usage
    -> on books.*
    -> to sally identified by 'magic123';
```

在与Sally交谈之后，我们对她需要进行的操作有了进一步了解，因此按如下方式可以授予她适当的权限：

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on books.*
    -> to sally;
```

请注意，要完成这些，并不需要指定Sally的密码。

如果我们认为Sally权限过高，可能会决定按如下方式减少一些权限：

```
mysql> revoke alter, create, drop
    -> on books.*
    -> from sally;
```

后来，当她不再需要使用数据库时，可以按如下方式撤销所有的权限：

```
mysql> revoke all
    -> on books.*
    -> from sally;
```

9.6　创建一个Web用户

要通过PHP连接到MySQL，需要为PHP脚本创建一个用户。这里，我们同样应用最少权限原则：脚本能够进行哪些操作呢？

在大多数情况下，它们只需要从表中选择（SELECT）、插入（INSERT）、删除（DELETE）和更新（UPDATE）查询。因此，可以按如下方式设定这些权限：

```
mysql> grant select, insert, delete, update
    -> on books.*
    -> to bookorama identified by 'bookorama123';
```

很明显，为了安全起见，应该选择一个更好的密码。

如果使用了Web主机服务，通常可以使用主机服务商在一个数据库上创建的其他用户类型权限。典型地，他们将提供相同的用户名和密码以用于命令行（建立表等）和用于Web脚本连接（查询数据库）。这是不够安全的。我们可以建立其他具有相同权限级别的用户，如下所示：

```
mysql> grant select, insert, update, delete, index, alter, create, drop
```

```
-> on books.*
-> to bookorama identified by 'bookorama123';
```

继续上面的工作，可以再创建一个用户，因为我们将在下一节中使用这个用户。

可以输入quit命令退出MySQL监视程序。最好应该再以Web用户的身份登录，测试每件事情是否工作正常。如果所运行的GRANT语句已经执行了，但是尝试登录时，又被拒绝了，这通常是因为安装过程中还没有删除匿名账户。以root重新登录并且查阅附录A给出的关于如何删除匿名账户的介绍。删除匿名账户后，应该能够以Web用户身份重新登录了。

9.7　使用正确的数据库

如果已经开始使用数据库了，因为刚刚创建这个数据库，或者因为Web服务器管理员刚刚创建它，应该以普通用户级别的MySQL账号登录以测试这些样本代码。

登录进入后，要做的第一件事是指定要使用的数据库。可以输入如下命令来完成：

```
mysql> use dbname;
```

这里dbname是数据库名称。

或者，也可以通过在登录的时候指定数据库而避免使用命令。如下所示：

```
mysql -D dbname -h hostname -u username -p
```

在这个例子中，我们将使用books数据库：

```
mysql> use books;
```

当输入该命令后，MySQL应该给出如下所示的响应：

```
Database changed
```

如果开始工作之前并没有选择数据库，MySQL将给出如下所示的错误信息：

```
ERROR 1046 (3D000): No Database Selected
```

9.8　创建数据库表

创建数据库的下一步是创建实际的表。可以使用SQL命令CREATE TABLE来完成它。CREATE TABLE语句的常见形式如下所示：

```
CREATE TABLE tablename (columns)
```

提示　你可能会注意到，MySQL提供了多个表类型和存储引擎，其中包括一些事务安全的类型。

我们将在第13章介绍这些表类型。目前，books数据库中所有表都使用了默认存储引擎，MyISAM。

应该用要创建的表名代替tablename占位符，用逗号分开的列名称列表代替columns占位符。每一列应该有一个名字，该名字后面紧跟其数据类型。

这里再次给出了Book-O-Rama的模式：

```
Customers(CustomerID, Name, Address, City)
```

```
Orders(OrderID, CustomerID, Amount, Date)

Books(ISBN, Author, Title, Price)

Order_Items(OrderID, ISBN, Quantity)

Book_Reviews(ISBN, Reviews)
```

程序清单9-1显示了如何使用SQL来创建这些表。可以在文件中找到这个SQL脚本，该脚本保存于文件chapter9/bookorama.sql中。

可以运行现有的SQL文件，例如，从文件中载入的一个文件，输入以下语句：

```
> mysql -h host -u bookorama -D books -p < bookorama.sql
```

(请记住，用你的主机名称替换host并且指定bookorama.sql文件的完整路径)

在这里，使用文件重定向是相当方便的，因为它意味着在执行之前，可以在所选择的文本编辑器中编辑SQL。

程序清单9-1 bookorama.sql——创建Book-O-Rama数据库表的SQL脚本

```
create table customers
( customerid int unsigned not null auto_increment primary key,
  name char(50) not null,
  address char(100) not null,
  city char(30) not null
);

create table orders
( orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null,
  amount float(6,2),
  date date not null
);

create table books
( isbn char(13) not null primary key,
  author char(50),
  title char(100),
  price float(4,2)
);

create table order_items
( orderid int unsigned not null,
  isbn char(13) not null,
  quantity tinyint unsigned,

  primary key (orderid, isbn)

);
create table book_reviews
```

```
(
  isbn char(13) not null primary key,
  review text
);
```

每个表由一个独立的CREATE TABLE语句所创建。可以看到我们已经创建了模式中的每个表，以及我们在上一章中为每个表所设计的列。每一列的名字后面都有一个数据类型。一些列还有其他特别项。

9.8.1 理解其他关键字的意思

NOT NULL的意思是表中所有行的此属性必须有一个值。如果没有指定，该列可以为空(NULL)。

AUTO_INCREMENT是一个特殊的MySQL特性，可以在整数列中使用它。它的意思是在表中插入行的时候，如果将该字段设置为空，那么MySQL将自动产生一个唯一的标识符值。该值比本列中现存的最大值更大。在每个表中只能有一个这样的值。指定AUTO_INCREMENT的列必须是索引列。

列名称后面的PRIMARY KEY表示该列是表的主键。本列中的输入必须唯一。MySQL将自动索引该列。在程序清单9-1的customers表中使用customerid时，我们将customerid列指定为AUTO_INCREMENT。主键的自动索引功能将管理AUTO_INCREMENT所要求的索引列。

在列的名称后面指定PRIMARY KEY，这只用于单列主键。Order_items语句结尾的分句PRIMARY KEY是一个可选格式，在这里，我们用到它是因为这个表的主键由两列组成（也将根据两列来创建索引）。

整数类型后面的UNSIGNED意思是它只能是0或者一个正数。

9.8.2 理解列的类型

我们首先看看第一个表的例子：

```
create table customers
( customerid int unsigned not null auto_increment primary key,
  name char(50) not null,
  address char(100) not null,
  city char(30) not null
);
```

在创建一个表的时候，需要确定列的数据类型。

对于customers表，我们在模式里指定它有4个列。第一列customerid，是主键，我们已经直接将它指定为主键。确定该列的数据类型是一个整数（数据类型int），而且这些ID应该是无符号的（unsigned）。我们还使用了auto_increment工具，这样MySQL就可以为我们管理这些，我们就不需要担心它。

其他列都是字符串类型数据。我们为这些列选择了char类型。同时，还将它们定义为固定长度的字段，该长度是在括号里指定的，例如姓名最多可以有50个字符宽度。

该数据类型将为姓名分配50个字符的存储空间，尽管姓名的长度通常不会长达30个字符。MySQL将用空格填充空余的部分。或者，我们还可以选择使用varchar类型，该数据类型可以根据需要分配存储空间（加一个字节）。这可能会有一些不足——因为，虽然varchar类型数据占用空间较小，但是char类型数据速度更快。

请注意，我们所声明的所有列都是NOT NULL（不为空），这是一个小小的优化措施，可以使得那些需要的地方速度更快。我们将在第12章中详细介绍优化。

其他一些CREATE语句在语法上有些不同。让我们来看看orders表：

```
create table orders
( orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null,
  amount float(6,2) ,
  date date not null
);
```

amount列被指定为浮点类型数据（float）。对于大多数据浮点数据类型来说，可以指定显示宽度和小数点后的位数。在这个例子中，订单总量将以美元计算，因此我们将允许合理大小的订单总金额（宽6位），小数位数到美分（2位）。

Date（日期）列数据类型为date。

在这个表中，我们将所有列指定为NOT NULL，这是为什么呢？因为当一个订单输入数据库的时候，将在orders表中创建一个记录，将所购物品添加到order_items表中，然后计算出总金额。在创建订单之前，我们可能还无法知道订单的总金额，所以必须允许amount列为NULL。

books表具有一些类似的特性：

```
create table books
( isbn char(13) not null primary key,
  author char(50),
  title char(100),
  price float(4,2)
);
```

在这个例子中，我们不必生成主键，因为可以将ISBN作为主键，而ISBN在其他地方可以生成。我们将其他字段设置为NULL，因为书店可能在知道书本标题（title）、作者（author）或价格（price）之前就已经知道此书的ISBN了。

order_items表显示了如何创建多列主键：

```
create table order_items
( orderid int unsigned not null,
  isbn char(13) not null,
  quantity tinyint unsigned,

  primary key (orderid, isbn)
);
```

该表将图书的数量指定为TINYINT UNSIGNED数据类型，其取值范围为0～255之间的一个整数。

正如前面已经提到的，多列主键需要通过一个特定的主键子句指定。在这里，我们就要使用它。

最后，考虑book_reviews表：

```
create table book_reviews
(
  isbn char(13) not null primary key,
  review text
);
```

它使用了本书尚未讨论过的新数据类型，text。该数据类型用于更长的文本，例如一篇文章。基于这个数据类型，还有一些变量，我们将在本章后续内容中详细讨论。

要更详细地理解创建表，我们应该先简单地介绍一下列名称和标识符，然后再介绍可以为列指定的数据类型。但是首先，让我们先了解已经创建的数据库。

9.8.3 用SHOW和DESCRIBE来查看数据库

登录到MySQL监视程序并使用books数据库。输入如下命令，可以查看数据库中的所有表：

```
mysql> show tables;
```

MySQL将显示该数据库中所有表的清单：

```
+----------------+
| Tables in books |
+----------------+
| book_reviews    |
| books           |
| customers       |
| order_items     |
| orders          |
+----------------+
5 rows in set (0.06 sec)
```

也可以使用show命令来查看数据库列表，输入如下命令：

```
mysql> show databases;
```

如果没有SHOW DATABASES权限，你将只看到权限范围内的数据库。

要查看某个特定表（例如，books表）的详细信息，可以使用DESCRIBE命令：

```
mysql> describe books;
```

MySQL将显示你在创建数据库的时提供的信息，如下所示：

```
+--------+-----------+------+-----+---------+-------+
| Field  | Type      | Null | Key | Default | Extra |
+--------+-----------+------+-----+---------+-------+
| isbn   | char(13)  | NO   | PRI | NULL    |       |
```

```
| author | char(50)    | YES  |     | NULL    |       |
| title  | char(100)   | YES  |     | NULL    |       |
| price  | float(4,2)  | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

这些命令是非常有用的，可以通过这些命令了解列的数据类型，或者浏览不是由你创建的数据库。

9.8.4 创建索引

由于设计主键将在这些列上创建索引，所以我们已经简单介绍过索引。

MySQL的新用户可能面临的一个常见问题是他们抱怨数据库的性能非常低下，因为他们曾经听说数据库速度很快。这个性能问题通常会在数据库上没有创建任何索引的情况下发生（创建没有主键或索引的表是可能的）。

要开始创建索引，可以使用自动创建的索引。如果发现需要对一个不是主键的列运行许多查询，我们可能希望在该列上添加索引来改善性能。可以使用CREATE INDEX语句来实现。该语句的常见形式如下所示：

```
CREATE [UNIQUE|FULLTEXT] INDEX index_name
ON table_name (index_column_name [(length)] [ASC|DESC], ...])
```

（FULLTEXT索引用来索引文本字段；我们将在第13章详细介绍它们的使用。）

可选的length字段允许指定只有该字段前length个字符将被索引。也可以指定一个索引的排序为升序或降序；默认值是升序。

9.9 理解MySQL的标识符

在MySQL中，提供了5种类型的标识符——Database（数据库）、Table（表）、Column（列）、index（索引）和Alias（别名）。前4类标识符我们已经熟悉，对于别名标识符，我们在下一章详细介绍。

MySQL中的数据库将被映射到具有某种文件结构的目录，而表则映射到文件。这可能对赋予它们的名字有直接影响。它也可以影响这些名字的大小写——如果操作系统区分目录与文件的大小写，那么数据库名称和表名称也会区分大小写（例如，在UNIX中），否则不区分（例如在Windows中）。列的名称和别名的名称不区分大小写，但是不能在同一个SQL语句中使用不同的大小写。

值得注意的是，目录和包含数据的文件的位置需要在配置中设置。可以使用mysqladmin命令来检查它们在系统中的位址，如下所示：

```
> mysqladmin -h host -u root -p variables
```

然后再查询datadir变量。

表9-4给出了所有标识符的总结。唯一的例外是在标识符中不能使用ASCII（0）、ASCII（255）或引号字符（实际上，这3个字符不会用到）。

表9-4　MySQL的标识符

类　型	最大长度	是否区分大小写	允许的字符
Database	64	与操作系统（O/S）相同	允许在操作系统目录名中出现的任何字符，不包括“/”、“\”和“.”字符
Table	64	与操作系统（O/S）相同	允许在操作系统目录名中出现的任何字符，不包括“/”和“.”字符
Column	64	否	任何字符
Index	64	否	任何字符
Alias	255	否	任何字符

这些规则是非常开放的。你甚至可以使用所有类型的单词和特殊字符作为标识符，唯一的限制是如果使用这样奇怪的标识符，必须用后引号将其括起来（位于大多数键盘上角的波浪号之下）。例如：

```
create database `create database`;
```

该规则在MySQL版本中（3.23.6版本前）有更严格的限制，它不允许这么做。

当然，对这些自由需要运用常识。只因为可以调用数据库“create database”，并不意味着应该这么做。同样的原则也适用于其他编程语言——使用有意义的标识符。

9.10　选择列数据类型

MySQL中3种基本的列数据类型：数字、日期和时间、字符串。而每个类型又包含了许多种类型。在这里，我们将总结这些类型，在第12章中，我们详细讨论每一类型的优点和弱点。

这3种类型需要不同的存储空间。一般说来，选择列数据类型的时候，基本原则是选择可以满足数据的最小类型。

对许多数据类型来说，当创建该类型列的时候，可以指定最大的显示长度。在如下数据类型总结表中，显示的就是M。如果该类型是可选的，它就显示在方括号内。M的最大值可为255。

这些描述中可选值显示在方括号中。

9.10.1　数字类型

数字类型分为整型和浮点型两类。对于浮点数字来说，可以指定小数点后数字的位数。本书中即为D。可以指定D的最大值为30或M-2（也就是，最大显示长度减去2——一个小数点和一个此数字的整数部分）。

对于整型数据，也可以将它们指定为无符号型，如程序清单9-1所示。

对所有数字类型，也可以指定ZEROFILL属性。当显示ZEROFILL字段中的值时，空余部分用前导0来补充。如果将一个字段指定为ZEROFILL，它将自动成为UNSIGNED数据类型。

整数类型如表9-5所示。请注意，本表第一行显示的范围是有符号整数的取值范围，而第二行显示的则是无符号整数的范围。

浮点类型如表9-6所示。

表9-5　整数数据类型

类　型	取值范围	存储空间（单位为字节）	描　述
TINYINT[(M)]	-127..128或0..255	1	非常小的整数
BIT			TINYINT的同义词
BOOL			TINYINT的同义词
SMALLINT[(M)]	-32768..32767或0..65535	2	小型整数
MEDIUMINT[(M)]	-8388608..8388607或0..16777215	3	中型整数
INT[(M)]	$-2^{31}..2^{31}-1$或$0..2^{32}-1$	4	一般整数
INTEGER[(M)]			INT的同义词
BIGINT[(M)]	$-2^{63}..2^{63}-1$或$0..2^{64}-1$	8	大型整数

表9-6　浮点数据类型

类　型	取值范围	存储空间（单位为字节）	描　述
FLOAT（精度）	取决于精度	可变	可用于指定单精度和双精度浮点数
FLOAT[(M, D)]	±1.175494351E-38 ±3.402823466E+38	4	单精度浮点数。等同于FLOAT(4)，但是指定显示宽度和小数位数
DOUBLE[(M, D)]	±1.7976931348623157E+308 ±2.2250738585072014E-308	8	双精度浮点数，等同FLOAT(8)，但是指定显示宽度和小数位数
DOUBLE PRECISION[(M, D)]	同上		DOUBLE[(M, D)]的同义词
REAL[(M, D)]	同上		DOUBLE[(M,D)]的同义词
DECIMAL[(M[,D])]	可变	M+2	浮点数，以char存储。范围取决于显示宽度M
NUMERIC[(M, D)]	同上		DECIMAL的同义词
DEC[(M, D)]	同上		DECIMAL的同义词
FIXED[(M, D)]	同上		DECIMAL的同义词

9.10.2　日期和时间类型

MySQL支持多种日期和时间类型。如表 9-7所示。使用这些类型，可以以字符串或数字格式输入数据。值得注意的是，如果不手动设置，特定行中的TIMESTAMP列将被设置为最近修改该行的日期和时间。这对于事务记录是很有意义的。

表9-7　日期和时间数据类型

类　型	取值范围	描　述
DATE	1000-01-01 9999-12-31	一个日期，以YYYY-MM-DD格式显示
TIME	-838:59:59 838:59:59	一个时间，以HH：MM：SS形式显示。注意其范围比想象的宽得多
DATETIME	1000-01-0100:00:00 9999-12-31 23:59:59	日期和时间。以YYYY-MM-DD HH：MM：SS格式显示

（续）

类　型	取值范围	描　述
TIMESTAMP[(M)]	1970-01-01 00:00:00	时间标签，在处理报告中有意义。显示格式取决于M的值（参阅表9-8）
	2037年的某个时间	范围的最高值取决于UNIX的限制
YEAR[(2|4)]	70-69（1970-2069）	年份。可以指定2位数字或4位数字的格式。各有不同的范围，如左边所示
	1901-2155	

表9-8显示了TIMESTAMP所有不同的可显示类型。

表9-8　TIMESTAMP显示类型

指定的类型	显　示	指定的类型	显　示
TIMESTAMP	YYYYMMDDHHMMSS	TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(14)	YYYYMMDDHHMMSS	TIMESTAMP(6)	YYMMDD
TIMESTAMP(12)	YYMMDDHHMMSS	TIMESTAMP(4)	YYMM
TIMESTAMP(10)	YYMMDDHHMM	TIMESTAMP(2)	YY

9.10.3　字符串类型

字符串类型分为3类。第一类为普通字符串，即小段文本，包括CHAR（固定长度字符）类型和VARCHAR（可变长度字符）类型。可以指定每种类型的宽度。无论数据大小是多少，CHAR类型的列都会用空格填补空白，但是VARCHAR列宽随数据大小变化。（请注意，获取CHAR类型数据的时候与存储VARCHAR数据的时候，MySQL将过滤多余的空格。）这两种类型都有速度与存储空间的问题，我们将在第12章中详细讨论。

第2类为TEXT和BLOB类型。这些类型大小可变，它们分别适用于长文本或二进制数据。

BLOB全称为大二进制对象（binary large objects）。它支持任何数据，例如，图像或声音数据。

在实际应用中，除了TEXT区分大小写而BLOB不区分之外，TEXT和BLOB列是相同的。

因为这些列类型可以容纳大量的数据，所以在使用它们时需要特别考虑。我们将在第12章中详细讨论。

第3类包括两种特殊类型，SET和ENUM。SET类型用来指定列中的值必须来自一个特定集合中的指定值。列值可以包含来自该集合的多个值。在指定的集合中，最大可以有64个元素。

ENUM就是枚举。与SET类型非常类似，但是该类型的列可以只有一个指定集合中的值或者NULL，在枚举中最大还可以有65 535个元素。

表9-9、表9-10、表9-11分别给出了这3类字符串类型数据的总结。表9-9给出了普通字符串类型。

表9-10给出了TEXT和BLOB类型。以字符计算的TEXT字段最大长度是可以存储在该字段中文件的最大字节数。

表9-11给出了ENUN和SET类型。

表9-9　常规字符串类型

类　型	取值范围	描　述
[NATIONAL] CHAR(M) [BINARY|ASCII|UNICODE]	0～255个字符	固定长度为M的字符串，其中M的取值范围为0～255。NATIONAL关键字指定了应该使用的默认字符集。虽然这是MySQL的默认值，但包含它的原因是因为它是ANSI SQL标准的一部分。BINARY关键字指定了数据是区分大小写的（默认值就是区分大小写的）。ASCII关键字指定了在该列中使用latin1字符集。UNICODE关键字指定了使用Ucs字符集
CHAR		CHAR(1)的同义词
[NATIONAL]VARCHAR(M) [BINARY]	1～255个字符	除了可变长度，其他与上一项相同

表9-10　TEXT和BLOB类型

类　型	最大长度（字符数）	描　述
TINYBLOB	2^8-1（即255）	小二进制大对象（BLOB）字段
TINYTEXT	2^8-1（即255）	小TEXT字段
BLOB	$2^{16}-1$（即65 535）	常规大小BLOB字段
TEXT	$2^{16}-1$（即65 535）	常规大小TEXT字段
MEDIUMBLOB	$2^{24}-1$（即16 777 215）	中型大小BLOB字段
MEDIUMTEXT	$2^{24}-1$（即16 777 215）	中型大小TEXT字段
LONGBLOB	$2^{32}-1$（即4 294 967 295）	长BLOB字段
LONGTEXT	$2^{32}-1$（即4 294 967 295）	长TEXT字段

表9-11　SET和ENUM类型

类　型	集合中的最大值	描　述
ENUM('value1', 'value2', ...)	65 535	该类型的列只可以容纳所列值之一或者为NULL
SET('value1', value2', ...)	64	该类型的列可以容纳一组值或者为NULL

9.11　进一步学习

要了解更多信息，可以在MySQL在线手册上阅读创建数据库的相关内容：http://www.mysql.com/。

9.12　下一章

到目前为止，我们已经了解了如何创建用户、数据库以及表。现在，我们可以集中精力学习如何与数据库进行交互。在下一章中，我们将介绍如何向表输入数据，如何更新和删除数据，以及如何查询数据库。