第4章

商城数据库的创建管理

4.1 项目导引:商城数据库

下面我们一起思考几个问题,图书商城中的图书信息存放在什么地方?用户的注册账户信息和提交订单时填写的邮寄地址等内容又到哪里去了?这些内容又以什么形式存放呢?

以上内容按照一定的格式存放在数据库(Data Base)中,数据库就是存放数据的仓库,其可以快速、安全的存储和处理大量的数据。

现在的网站几乎都是基于数据库的,使用 PHP 开发网站,同样也离不开数据库,PHP 可以与 MySQL、ACCESS、SQL Server、ORACAL 等多种数据库组合使用。在这些数据库中,MySQL 是世界上最为流行、开放源码、完全网络化、跨平台的数据库,能够满足多数中小型企业的需求,绝大多数 PHP 网站采用 MySQL 作为网站的数据库。本章将使用 MySQL 完成图书商城数据库的创建,为开发图书商城网站提供数据保证。

4.2 项目分析

商城中的数据存放在数据库中,那么数据在数据库中又是以什么形式存储呢?

下面,我们先看一看 Ecshop 系统的数据库,安装好 Ecshop 之后(WampServer 环境下),在浏览器中输入 http://localhost/phpmyadmin/,打开 MySQL 的管理工具 phpMyAdmin,从中选择 Ecshop 数据库,即出现如图 4-1 所示页面。在页面的左侧有许多文件,我们称它为“表”,此数据库中共有 87 张表,MySQL 数据库就是通过表来组织管理数据的。

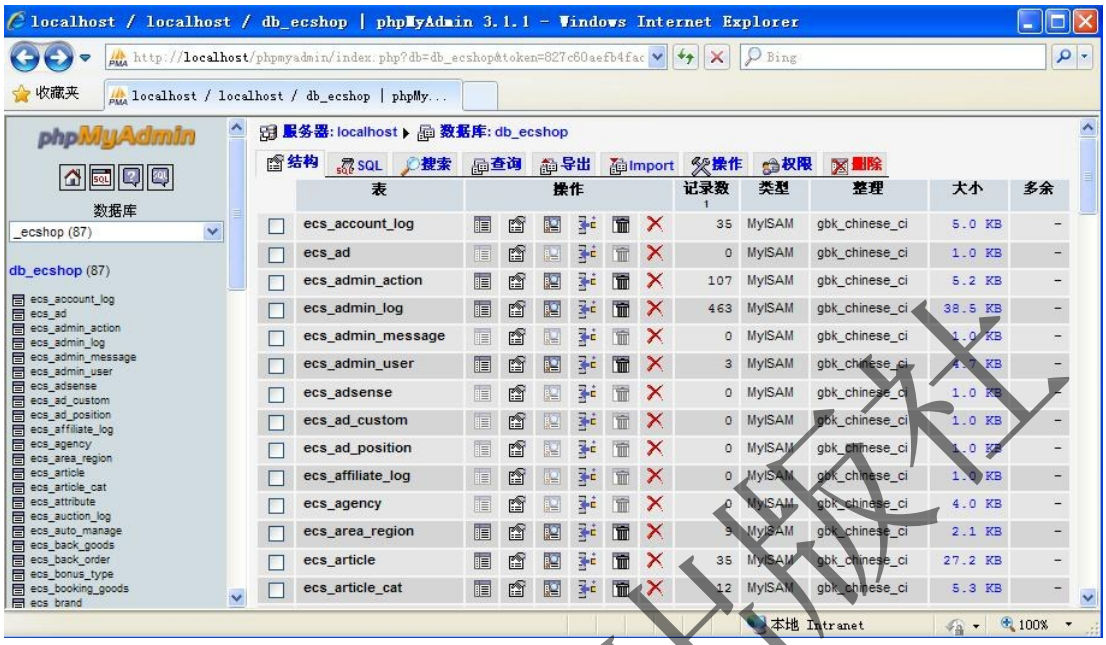

图 4-1　Ecshop 数据库中的表

任意单击一张表(如商品表 ecs_goods),就会在此页面的右侧显示出这张表的内容:表的顶部是每项内容的标题(如 goods_id、goods_name),我们称之为表的字段名;在标题栏的下面有很多行,每一行代表一个具体产品,我们称之为表的记录,如图 4-2 所示。数据就是以这种形式存储在数据库中。

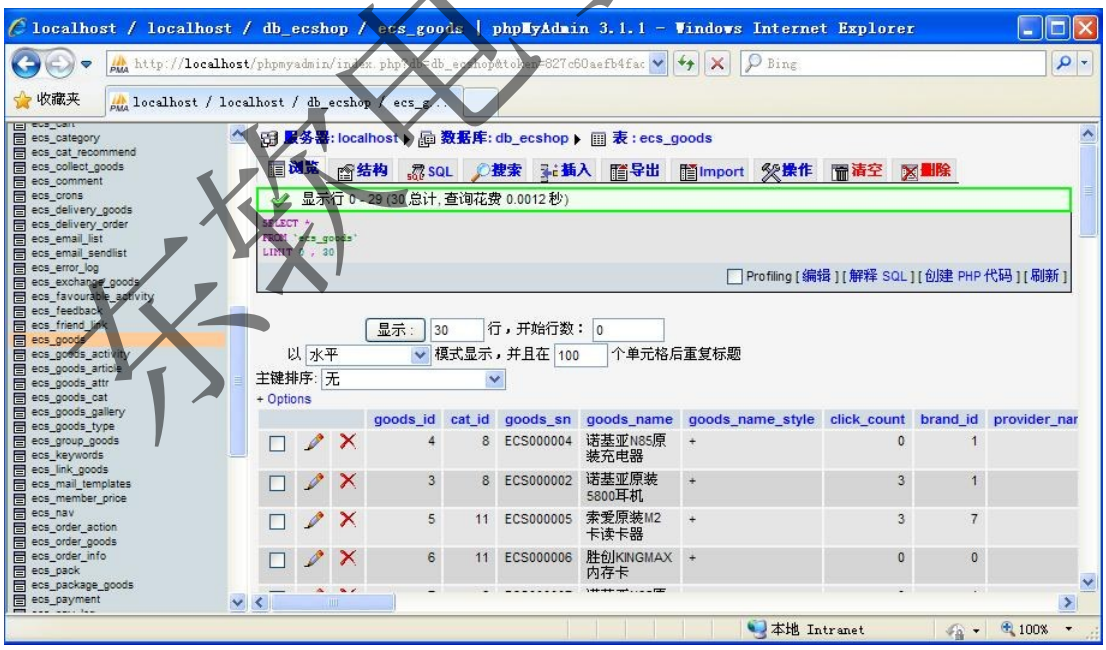

图 4-2　Ecshop 表的内容

在如图 4-2 所示页面顶部单击“结构”按钮,查看商品表的具体结构,如图 4-3 所示。表的结构主要包括:字段名称、字段的数据类型、字段的整理编码方式、字段的属性等。

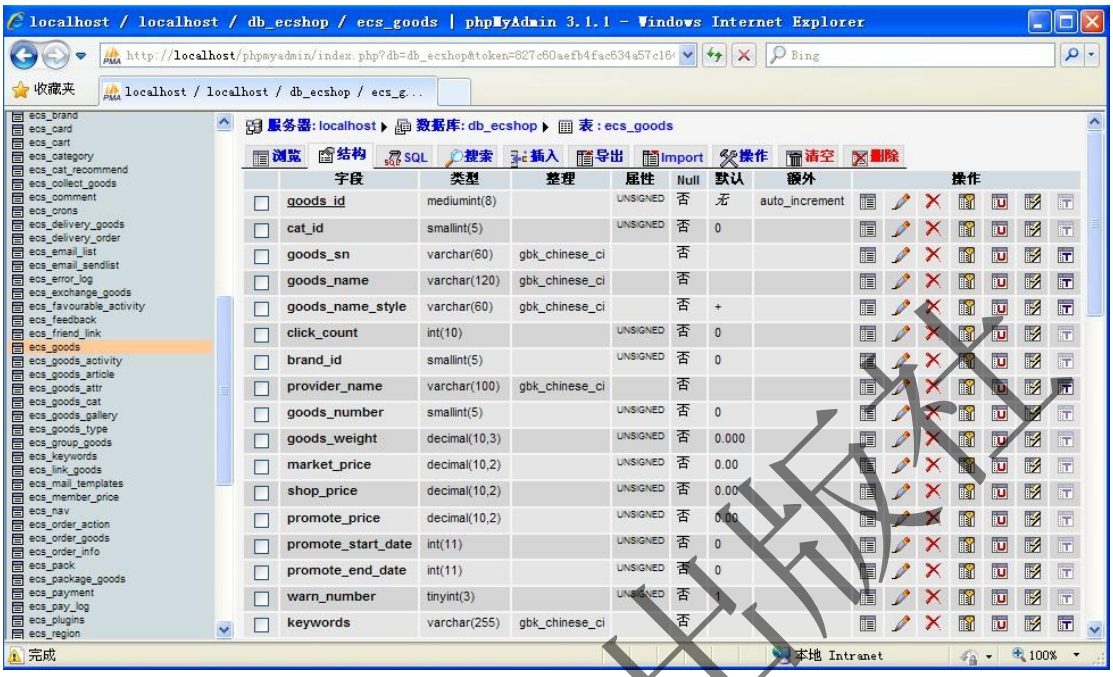

图 4-3　Ecshop 表的具体结构

我们的图书商城也应该创建一个类似的数据库，那么图书商城数据库中应该有多少个表？每个表的结构又如何呢？数据库和表又如何创建呢？下面我们就去解决这些问题。

4.3　技术准备

1. 数据表

ACCESS、MySQL、SQLServer、ORACAL 等数据库都属于关系数据库，关系数据库以二维表的形式组织、管理数据，二维表由记录(行)和字段(列)组成。例如，表 4-1 所示的图书表，包含 bookid(图书编号)、typeid(图书类别编号)、bookname(图书名称)、author(作者)、bookprice(价格)、pubdate(出版时间)6 个字段和 3 条图书记录。

表 4-1　　图书表(tb_book)

bookid	typeid	bookname	author	bookprice	pubdate
1	1	PHP	王某	46	2013 年 1 月
2	1	ASP	刘某	38	2011 年 8 月
3	1	JSP	李某	52	2012 年 3 月

我们再来看一个图书类别表，如表 4-2 所示，包含 typeid(类别编号)、typename(类别名称)、typedes(类别描述)3 个字段和 2 条图书类别记录。

表 4-2　　图书类别表(tb_type)

typeid	typename	typedes
1	计算机类	存放计算机类的图书
2	经济管理类	存放经济管类的图书

一个数据库中可以有多个数据表,每个数据表的名称必须是唯一的,表中每个字段的名称也必须是唯一的,每个字段都有对应的数据类型和取值范围。

二维表中能唯一区分、确定不同记录的属性或属性组合,称为该表的主键。主键具有唯一性和非空性。例如,图书编号为图书表的主键,用户编号为用户表的主键。

在图书类别表中,typeid(类别编号)字段为主键,在图书表中也有 typeid(类别编号)字段,并且与图书类别表中的 typeid(类别编号)字段是对应关系。这里我们把 typeid(类别编号)字段称为图书类别表的主键、图书表的外键。

2. 数据类型

在创建数据表时,必须为表中的列指定数据类型。数据类型是用来表现数据特征的,它决定了数据在计算机中的存储格式、存储长度,以及数据精度和小数位数等属性。MySQL 中的数据类型可分为 3 大类:数值类型、日期和时间类型以及字符串(字符)类型,每大类又可细分为若干小类,具体如下:

(1)数值类型。

- TINYINT:占 1 个字节,有符号数字的范围是-128 到 127,无符号的数字范围是 0 到 255。
- SMALLINT:占 2 个字节,有符号数字的范围是-32768 到 32767,无符号数字的范围是 0 到 65535。
- MEDIUMINT:占 3 个字节,有符号数字的范围是-8388608 到 8388607,无符号数字的范围是 0 到 16777215。
- INT:占 4 个字节,有符号数字的范围是-2147483648 到 2147483647,无符号数字的范围是 0 到 4294967295。
- BIGINT:占 8 个字节,有符号数字的范围是-9223372036854775808 到 9223372036854775807,无符号数字的范围是 0 到 18446744073709551615。
- FLOAT[(M,D)]:占 4 个字节,不能无符号,允许的值是-3.402823466E+38 到 -1.175494351E-38,0 和 1.175494351E-38 到 3.402823466E+38。M 是显示宽度,D 是小数的位数。
- DOUBLE[(M,D)]:占 8 个字节,不能无符号,允许的值是-1.7976931348623157E+308 到 -2.2250738585072014E-308、0 和 2.2250738585072014E-308 到 1.7976931348623157E+308。M 是显示宽度,D 是小数位数。
- DECIMAL[(M[,D])]:一个未压缩(unpack)的浮点数字,“未压缩”意味着数字作为一个字符串被存储,值的每一位使用一个字符。小数点,并且对于负数,“-”符号不在 M 中计算。如果 D 是 0,值将没有小数点或小数部分。DECIMAL 值的最大范围与 DOUBLE 相同,但是对一个给定的 DECIMAL 列,实际的范围可以通过 M 和 D 的选择被限制。如果 D 被省略,它被设置为 0。如果 M 被省掉,它被设置为 10。

(2)日期和时间类型。

- DATE:日期型,占 3 个字节,支持的范围是'1000-01-01'到'9999-12-31'。MySQL 以

'YYYY-MM-DD'格式来显示 DATE 值，但是允许使用字符串或数字把值赋给 DATE 列。

• DATETIME：日期和时间组合型，占 8 个字节，支持的范围是'1000-01-01 00：00：00'到'9999-12-31 23：59：59'。MySQL 以'YYYY-MM-DD HH：MM：SS'格式来显示 DATETIME 值，但是允许使用字符串或数字把值赋给 DATETIME 的列。

• TIMESTAMP：时间戳，占 4 个字节，范围是'1970-01-01 00：00：00'到 2037 年的某时。

• TIME：时间型，占 3 个字节，范围是'-838：59：59'到'838：59：59'。MySQL 以'HH：MM：SS'格式来显示 TIME 值，但是允许使用字符串或数字把值赋给 TIME 列。

• YEAR[(2|4)]：2 或 4 位数字格式的年(缺省是 4 位)，占 1 个字节，允许的值是 1901 到 2155，如果使用 2 位，1970-2069(70-69)。MySQL 以 YYYY 格式来显示 YEAR 值，但是允许把使用字符串或数字值赋给 YEAR 列。

(3)字符串(字符)类型。

• CHAR(M)：定长字符串，当存储时，总是用空格填满右边到指定的长度。M 的范围是 0～255个字符。

• VARCHAR(M)：变长字符串，当值被存储时，尾部的空格被删除。M 的范围是 0～255 个字符。

• BLOB：二进制对象，可以保存图片、声音等二进制数据。BLOB 类型根据其容纳值的长度不同，分为 TINYBLOB(最大长度 255 字节)、BLOB(最大长度 65535 字节)、MEDIUMBLOB(最大长度 16777215 字节)和 LONGBLOB(最大长度 4GB 字节)类型。

• TEXT：文本，用来保存字符数据。TEXT 类型根据其容纳值的长度不同，分为 TINYTEXT(最大长度 255 字节)、TEXT(最大长度 65535 字节)、MEDIUMTEXT(最大长度 16777215 字节)和 LONGTEXT(最大长度 4GB 字节)类型。

• ENUM('value1','value2',...)：枚举，一个仅有一个值的字符串对象，这个值选自值列表'value1'、'value2',...或 NULL。值列表最多能有 65535 不同的值。

• SET('value1','value2',...)：集合，有零个或多个值的一个字符串对象，其中每一个必须从值列表'value1'、'value2'、...选出。一个 SET 最多能有 64 个成员。

3. 商城数据库设计

依据“天天书屋”商城的用户需求分析和功能结构分析，商城主要包含 tb_user(用户表)、tb_type(图书类别表)、tb_book(图书表)、tb_order(订单表)和 tb_admin(系统管理员表)。每个表的具体结构如表 4-3 至表 4-7 所示。

表 4-3　　tb_user（用户表）

字段名称	数据类型	是否为空	是否主键	说明
userid	Int(11)	否	是	用户编号
username	Varchar(30)	否	否	用户姓名
password	Varchar(50)	否	否	用户密码
email	Varchar(30)	否	否	用户邮箱
address	Varchar (100)	是	否	用户地址
telephone	Varchar(15)	是	否	联系方式
regdate	Varchar(12)	是	否	注册时间

表 4-4　　tb_type(图书类别表)

字段名称	数据类型	是否为空	是否主键	说明
typeid	Int(11)	否	是	类别编号
typename	Varchar(30)	是	否	类别名称
typedes	Text	是	否	类别描述

表 4-5　　tb_book(图书表)

字段名称	数据类型	是否为空	是否主键	说明
bookid	Int(11)	否	是	图书编号
typeid	Int(11)	是	外键	类型编号
isbn	Int(11)	是	否	书号
bookname	Varchar(50)	是	否	图书名称
author	Varchar(30)	是	否	图书作者
pubhouse	Varchar(30)	是	否	出版社
pubdate	Varchar(12)	是	否	出版时间
bookprice	Float	是	否	图书价格
vipprice	Float	是	否	会员价
photo	Varchar(100)	是	否	图书图片
introduction	mediumtext	是	否	图书简介
recommend	TinyInt(1)	是	否	是否推荐
newbook	TinyInt(1)	是	否	是否新书

表 4-6　　tb_order(订单表)

字段名称	数据类型	是否为空	是否主键	说明
orderid	Int(11)	否	是	订单编号
userid	Int(11)	是	外键	客户编号
spc	Varchar(100)	是	否	商品串
slc	Varchar(100)	是	否	数量串
shouhuoren	Varchar(30)	是	否	收货人姓名
sex	Varchar(2)	是	否	收货人性别
address	Varchar(100)	是	否	收货人地址
youbian	Varchar(10)	是	否	邮编
tel	Varchar(30)	是	否	联系电话
email	Varchar(30)	是	否	邮箱地址
shff	Varchar(30)	是	否	收获方式
orderdate	Varchar(12)	是	否	下单时间
xiadanren	Varchar(30)	是	否	下单人姓名
zt	Varchar(20)	是	否	订单状态
total	Float	是	否	价格总计

表 4-7　　tb_admin（系统管理员表）

字段名称	数据类型	是否为空	是否主键	说明
id	Int(11)	否	是	用户编号
name	Varchar(30)	否	否	用户姓名
password	Varchar(50)	否	否	用户密码

4.4　项目实施

通过前面的学习我们知道商城数据库都有哪些表组成了，那么我们该如何创建数据库呢？对 MySQL 数据库的创建管理主要包括两种方式，一种是通过图形管理工具创建管理数据库，另一种是通过 MySQL 的客户端程序创建管理数据库，客户端程序的管理是通过 SQL 语句来实现。

4.4.1　phpMyAdmin 之商城数据库创建管理

MySQL 常用的图形化管理工具有 phpMyAdmin、MySQLDumper、Navicat、MySQL GUI Tools、MySQL ODBC Connector 等，这些工具需要安装之后才能使用。下面以 phpMyAdmin 为例，来创建和管理图书商城数据库。

phpMyAdmin 是一个用 PHP 开发的基于 Web 方式的 MySQL 管理工具，它的官方网站是 http://www.phpmyadmin.net。PHP 集成开发环境 WampServer 中包含有 phpMyAdmin 组件，安装好 WampServer 后，单击 WampServer 图标，执行 phpMyAdmin，即可启动该工具，其主界面如图 4-4 所示。

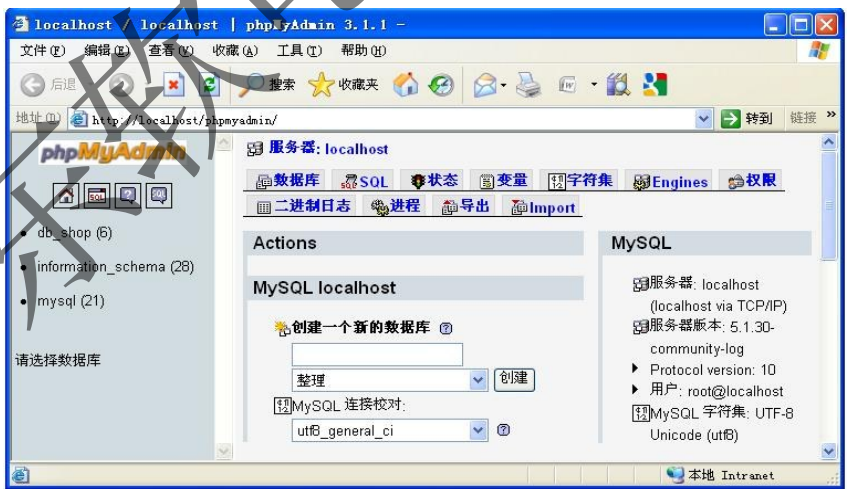

图 4-4　phpMyAdmin 主界面

1. 创建数据库

在主界面中可以创建数据库，创建数据库需要指定数据库的名字和编码方式，“天天书屋”商城系统数据库名称为 db_shop，采用 GB2312 的编码方式，如图 4-5 所示。单击“创建”按钮完

成数据库的创建。

图 4-5　创建数据库

2. 创建数据表

数据库创建后，进入创建数据表的页面，如图 4-6 所示。在此，输入数据表的名称和字段数量，例如，输入表名为“tb_type”，字段数量为“3”。

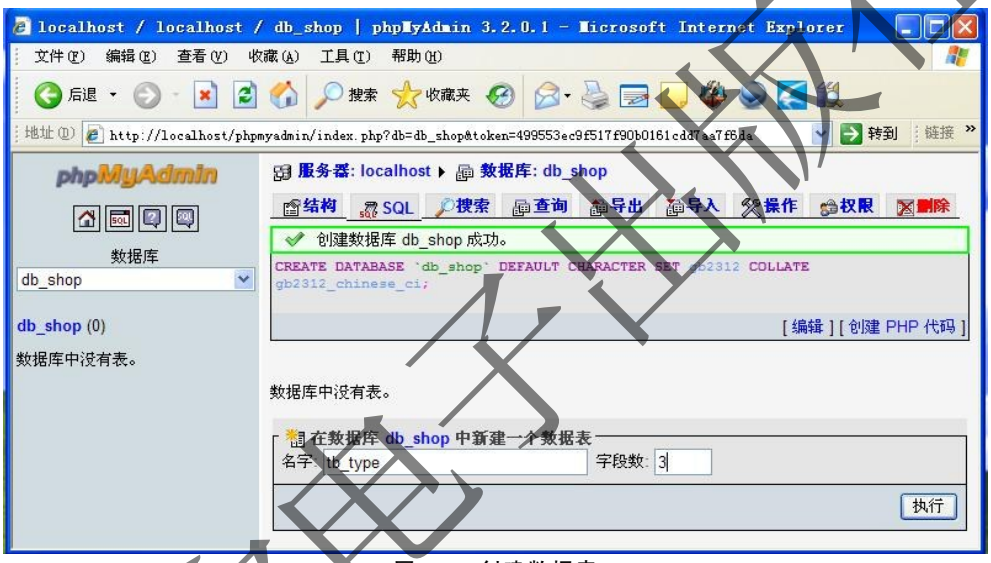

图 4-6　创建数据表

单击“执行”按钮，进入创建数据表字段的界面，如图 4-7 所示。在此可以设置字段的字段名称、数据类型、长度值、默认值、整理编码方式、属性、是否为空(Null)、索引和自动增长(A_I)等。设置完毕后，单击“保存”按钮，完成数据表结构的创建。

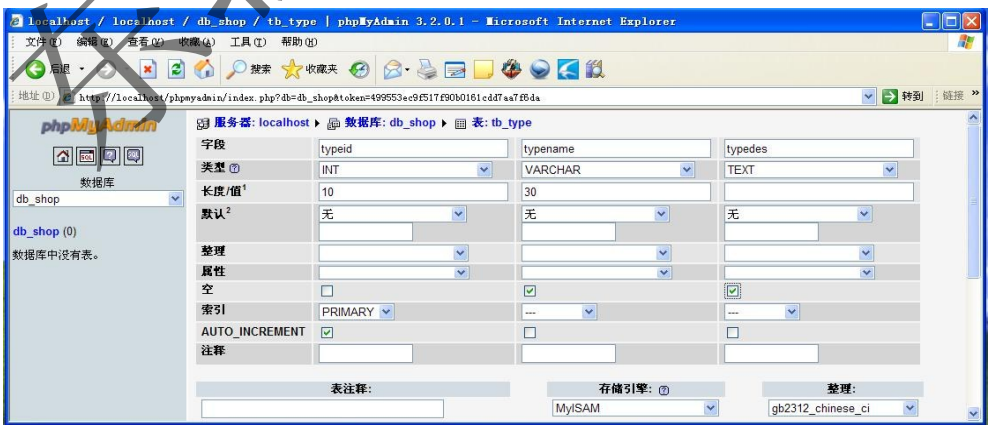

图 4-7　设置表字段

数据表创建成功后，进入数据表“结构”页面，如图 4-8 所示，在这里可以修改表结构，如添加字段、删除字段、设置主键、索引、修改字段名称等。单击导航中“删除”按钮，可以将数据表删除。

图 4-8　管理数据表

3. 添加数据

在数据表页面，单击“插入”按钮，进入添加数据页面，如图 4-9 所示，输入“值”后，单击“执行”按钮，即可将数据添加到数据表中。

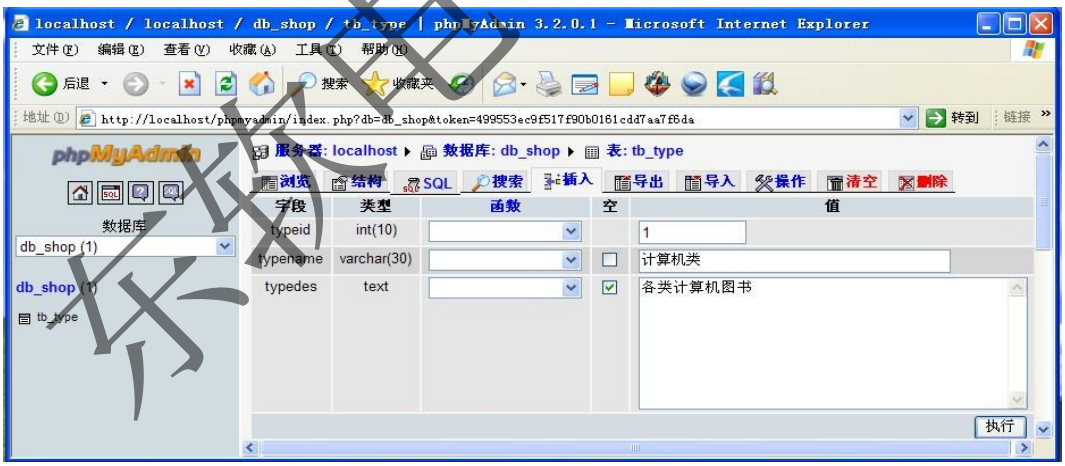

图 4-9　添加数据

4. 运行 SQL 语句

在数据库或数据表页面顶部都有“SQL”按钮，单击进入相应 SQL 语句执行页面，如图 4-10 所示。在此，可以编写运行 SQL 语句，关于 SQL 语句后面会有详细介绍。

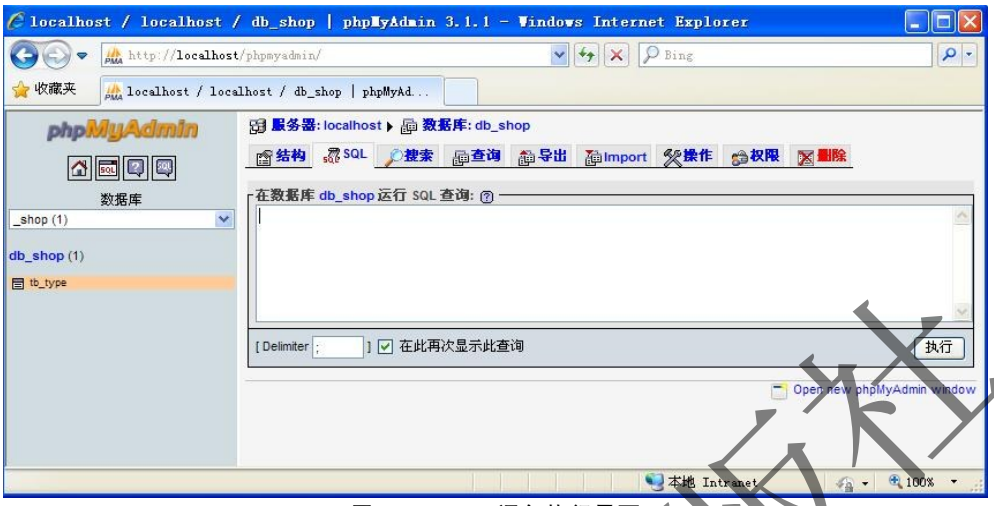

图 4-10　SQL 语句执行界面

5. 数据库的备份

对于数据库的管理经常需要备份和还原，在 phpMyAdmin 中我们通过导出、导入方式备份还原数据库。选择 db_shop 数据，单击导航栏中的“导出”按钮，进入导出界面，如图 4-11 所示。在此，我们可以选择导出的数据表，选择导出数据表的具体内容和导出的保存形式，在页面下方指定保存的文件名，最后，将 SQL 文件保存到磁盘上，完成数据备份。

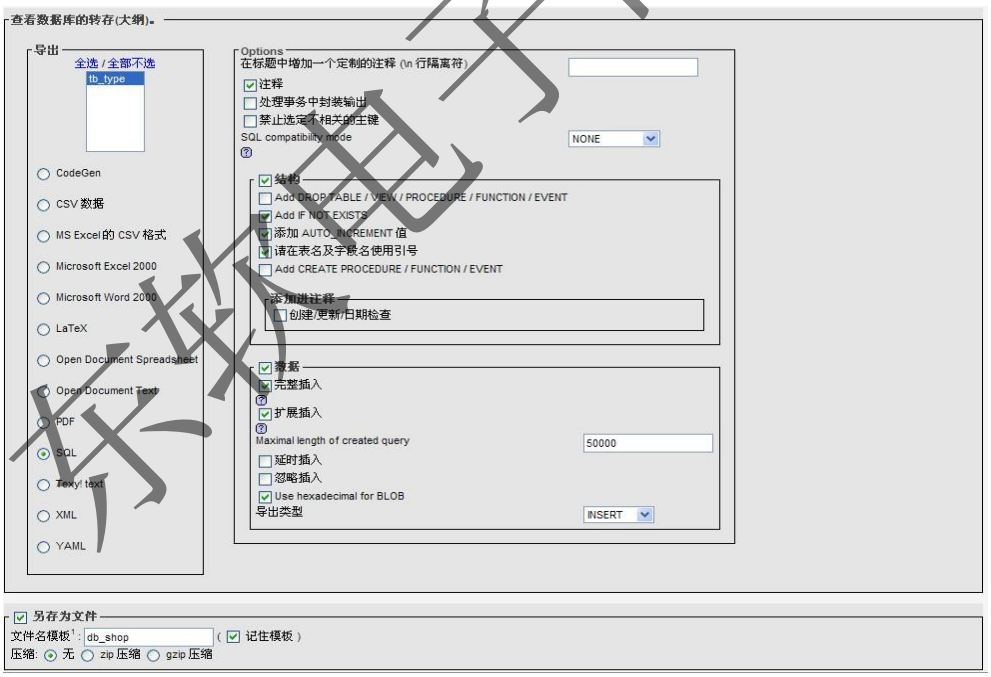

图 4-11　数据导出界面

6. 数据库的还原

单击导航栏中的“Import”按钮，进入 SQL 文件导入界面，如图 4-12 所示。单击“浏览”按钮，选择需要导入的 SQL 脚本文件，然后单击“执行”按钮，系统将执行 SQL 文件中的 SQL 命令，完成数据还原。

图 4-12　数据导入部分界面

数据库创建好之后，会保存在 MySQL 安装路径下的 data 文件中，每一个数据库对应一个文件夹，通过文件夹操作也可以实现数据库的备份还原。

掌握 phpMyAdmin 的基本操作后，使用 phpMyAdmin 完成商城数据库及数据表的创建。

4.4.2　SQL 之数据库的创建与管理

MySQL 是基于客户机/服务器结构的数据库管理系统，通过客户机连接服务器成功后，再通过必要的操作指令对其进行操作，这种数据库操作指令被称为 SQL（Structured Query Language）语言，即结构化查询语言。SQL 语言结构简洁，功能强大，自 IBM 公司 1981 年推出以来，SQL 语言得到了广泛的应用，目前 MySQL、Oracle、SQLServer、Sybase、DB2 等数据库都采用 SQL 作为查询语言。SQL 语言包含以下四部分：

- 数据定义语言（DDL）：用于定义和管理数据库对象，包括数据库、数据表、索引、视图等。例如，create、drop、alter 等语句。
- 数据操作语言（DML）：用于操作数据库对象中所包含的具体数据。例如，insert、update 和 delete 等语句。
- 数据查询语言（DQL）：用于查询数据库对象中的所包含的数据。例如，select 语句。
- 数据控制语言（DCL）：用来管理数据库的语言，包含管理权限及数据更改。例如，grant、revoke、commit 和 rollback 等语句。

下面我们在 MySQL 客户机（控制台）环境下，应用 SQL 语句完成商城数据的创建与管理。

1. 登录 MySQL 服务器

左键单击 WampServer 图标，选择“MySQL 控制台”，进入 MySQL 客户端，如图 4-13 所示。由于 WampServer 安装时没有设置用户密码，直接回车即可，当提示符变为“MySQL＞”时，就代表以 root 用户身份登录到 MySQL 服务器。

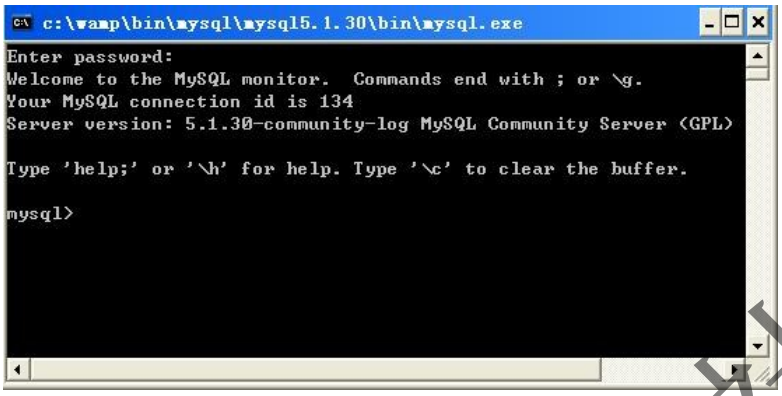

图 4-13　MySQL 控制台

2. 创建数据库

创建数据库是创建其他数据库对象的基础，其语法格式如下：

```
CREATE DATABASE db_name;
```

其中，db_name 是要创建的数据库名称；“;”是 SQL 语句的默认结束标志。

【例 4-1】创建名称为 db_shop 的数据库，命令如下所示：

```
mysql> CREATE DATABASE db_shop;
Query OK, 1 row affected (0.03 sec)
```

3. 查看数据库

查看当前服务器中数据库的语法格式如下：

```
SHOW DATABASES  [LIKE wild];
```

其中，LIKE wild 是可选项，用来指定显示模式匹配的数据库；wild 是一个字符串，它可以包含 SQL 通配符“_”（匹配单个字符）和“%”（匹配任意数目字符）。

【例 4-2】显示当前服务器中名称前两个字符是 db 的数据库，命令如下所示：

```
mysql> SHOW DATABASES LIKE 'db%';
+----------------+
| Database (db%) |
+----------------+
| db_shop        |
+----------------+
1 row in set (0.02 sec)
```

4. 指定当前数据库

服务器中可能会存有多个数据库，如果使用某个数据库，需要将其指定为当前数据库，其语法格式如下：

```
USE db_name;
```

【例 4-3】将 db_shop 数据库设置成为当前数据库，命令如下所示：

```
mysql> USE db_shop;
Database changed
```

5. 删除数据库

删除数据库的语法格式如下：

```
DROP DATABASE db_name;
```

删除命令会删除数据库中的所有表和数据，需要小心地使用该命令！

4.4.3 SQL 之数据表的创建与管理

1. 创建数据表

创建数据库后，需要为数据库创建表来保存数据。创建表的基本语法格式如下：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name [(col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT][PRIMARY KEY])]
```

其中：

- TEMPORARY：用来创建临时表；
- IF NOT EXISTS：用来判断准备新建的表是否存在，如果不存在才创建该表，从而避免出现表已经存在无法新建的错误；
- col_name type：列名及其数据类型；
- NOT NULL | NULL：指定列是否为空，默认为空(NULL)；
- DEFAULT default_value：为列指定默认值，默认值必须是常量；
- AUTO_INCREMENT：指定列为自动编号，该列必须是整数类型；
- PRIMARY KEY：指定列为主键，主键列不能为空。

【例 4-4】在 db_shop 数据库中创建 tb_type(图书分类)表，其结构见表 4-4。

在命令提示符下输入以下语句：

```
CREATE TABLE tb_type (
  typeid int(11) NOT NULL auto_increment,
  typename varchar(30) default NULL,
  typedes text,
  PRIMARY KEY  (typeid)
) ENGINE=MyISAM  DEFAULT CHARSET=gb2312 ;
```

运行结果如图 4-14 所示。

```
mysql> USE db_shop;
Database changed
mysql> CREATE TABLE tb_type (
    ->   typeid int(11) NOT NULL auto_increment,
    ->   typename varchar(30) default NULL,
    ->   typedes text,
    ->   PRIMARY KEY  (typeid)
    -> ) ENGINE=MyISAM  DEFAULT CHARSET=gb2312 ;
Query OK, 0 rows affected (0.09 sec)
```

图 4-14　创建 tb_type 表

【例 4-5】在 db_shop 数据库中我们再创建一个图书表(tb_book)，其表结构见表 4-5，创建代码如下所示：

```
CREATE TABLE tb_book (
  bookid int(11) NOT NULL auto_increment,
  typeid int(11) NOT NULL,
  isbn int(11) NOT NULL,
```

```
    bookname varchar(50) NOT NULL,
    author varchar(30) NOT NULL,
    pubhouse varchar(30) NOT NULL,
    pubdate varchar(12) NOT NULL,
    bookpricefloat NOT NULL,
    vippricefloat NOT NULL,
    photo varchar(100) NOT NULL,
    introduction varchar(1000) NOT NULL,
    recommendtinyint(1) NOT NULL,
    newbooktinyint(1) NOT NULL,
    PRIMARY KEY  (bookid)
) ENGINE=MyISAM  DEFAULT CHARSET=gb2312 ;
```

2. 查看数据库表及列信息

查看当前服务器中数据库表清单的语法格式如下：

```
SHOWTABLES  [LIKE wild];
```

查看数据表的列信息语法格式如下：

```
SHOW COLUMNS FROM tbl_name [FROM db_name] [LIKE wild];
```

【例 4-6】查看 db_shop 数据库中 tb_type 表的列信息。

在命令提示符下输入以下语句：

```
SHOW COLUMNS FROM tb_type;
```

运行结果如图 4-15 所示。

```
mysql> SHOW COLUMNS FROM tb_type;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| typeid   | int(11)     | NO   | PRI | NULL    | auto_increment |
| typename | varchar(30) | YES  |     | NULL    |                |
| typedes  | text        | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

图 4-15　tb_type 表列信息

3. 修改表

数据表创建后，可以使用 ALTER TABLE 语句对表结构进行修改，下面列举几个修改命令：

- 增加列

```
ALTER TABLE tbl_name ADD col_name TYPE;
```

- 删除列

```
ALTER TABLE tbl_name DROP col_name;
```

- 改变列

```
ALTER TABLE tbl_name MODIFY col_name TYPE;
```

- 更名列

```
ALTER TABLE tbl_name CHANGE old_col_name col_name;
```

● 更改表名

```
ALTER TABLEtbl_name RENAME new_ tbl_name;
```

4. 删除表

当数据库中不需要某些表时，可以将其删除。删除数据表的语法格式如下：

```
DROP TABLE [IF EXISTS] tbl_name [,tbl_name,...]
```

删除数据表时，表中的数据和表定义同时被删除，应该小心使用删除命令。

5. 添加数据

数据库用表存储和管理数据。新表建好后，表中并不包含任何数据记录，要想实现数据的存储必须向表中添加数据。向表中添加数据，可以使用 INSERT 语句实现。添加的数据中如果有汉字，一定要注意创建数据表时的字符编码，如字符编码为 gb2312，则在添加数据时，需要先设置编码。

(1)设置编码。

MySQL 客户端默认的字符集是 utf8，如果修改该字符集为 gb2312，可以使用“SET NAMES gb2312”命令，其运行结果如图 4-16 所示。

```
mysql> SET NAMES gb2312;
Query OK, 0 rows affected (0.00 sec)
```

图 4-16　设置 gb2312 编码

(2)添加记录。

向表中添加一条或多条数据记录，使用 INSERT...VALUES 语句实现，其语法格式如下：

```
INSERT [INTO] tbl_name SET col_name＝expression, col_name＝expression, ...
INSERT [INTO] tbl_name[(col_name,...)] VALUES (expression,...),(...),...
```

【例 4-7】为 tb_type 和 tb_book 表添加记录。

```
INSERT INTO tb_type SET typeid＝1, typename ='计算机类';
INSERT INTO tb_type (typeid, typename) VALUES(2,'经济管理类'),(3,'文艺类') ,(4,'教育类');
INSERT INTO tb_book (bookid, typeid, isbn, bookname, author, pubhouse, pubdate,bookprice, photo,
introduction, recommend) VALUES(1, 1, '9787561828052', 'VB.NET2005 程序设计实例教程', '李英杰', '天津大
学出版社', '2008-11-01', 44, '9787561828052.jpg', '介绍 VB.NET2005 程序设计基础知识', 1) ,
(2, 1, '9787121049248', 'SQL SERVER 实例教程', '杨学全', '电子工业出版社', '2007-10-01', 30,
'9787121049248.jpg', 'SQL SERVER 2005 基础知识介绍。', 1) ,(3, 3, '9787108032911', '目送', '龙应台', '生活.
读书.新知三联书店', '2009-09-01', 39, '9787108032911.jpg', '目送共由七十四篇散文组成，是为一本极具亲
情、感人至深的文集。', 0);
```

注：图书表的 photo 字段保存的是图片的完整路径。

6. 修改数据

修改数据表中的数据可以使用 UPDATE 语句，其语法格式如下：

```
UPDATE tbl_name SET col_name1＝expr1,col_name2＝expr2,...
[WHERE where_definition]
```

需要注意的是，当没有 WHERE 子句指定修改条件时，则表中所有记录的指定列都被修改。

【例 4-8】将 tb_type 表中类别名称为“教育类”的名字修改为“高职教育类”。

```
UPDATE tb_type SET typename＝'高职教育类' WHERE typename＝'教育类';
```

7. 删除数据

将数据表中数据删除的语句是 DELETE，其语法格式为：

```
DELETE FROM tbl_name [WHERE where_definition]
```

需要注意的是，当没有 WHERE 子句指定删除条件时，则删除表中所有记录。

【例 4-9】删除 tb_type 表中类别名称为“高职教育类”的记录。

```
DELETE FROM tb_type WHERE typename='高职教育类';
```

4.4.4　SQL 之数据库的查询管理

数据查询是数据库的核心操作，用户可以通过查询获得所需要的数据。查询操作可以通过 SELECT 语句实现，该语句在执行时会根据要求从一个或多个数据表中选取特定的行和列，形成一个临时表传送给用户。SELECT 语句功能强大，完整语法比较复杂，其基本的语法格式如下：

```
SELECT  select_expression,...
[FROM table_list]
[WHERE where_definition]
[GROUP BY col_name,...]
[HAVING where_definition]
[ORDER BYsorting_columns [ASC|DESC]]
[LIMIT [offset,] rows]
```

其中：

- select_expression：描述结果集的列，列与列之间用逗号分隔。
- FROM table_list：用于指定产生查询结果集的数据来源的表或视图的名称。
- WHERE where_definition：用于指定所检索的数据应该满足的条件。
- GROUP BY col_name：用于分组统计时指定分组的条件。
- HAVING where_definition：与 GROUP BY 子句一起使用，用于对分组统计的结果设置条件，进行组数据选择。
- ORDER BYsorting_columns [ASC|DESC]：指定在 SELECT 语句返回的列中所使用的排序顺序；ASC 和 DESC 用于指定行是按升序还是按降序排列。
- LIMIT[offset,] rows：用来限制 SELECT 语句返回的行数。

需要注意，SELECT 语句中的子句必须按以上语法格式的顺序给出，例如，HAVING 子句必须在 GROUP BY 子句之后，ORDER BY 子句之前。

1. 基本查询

以下查询均在 db_shop 数据库中完成。

(1)选择指定的列。

数据表中通常存有多列信息，如果只需要查询部分列信息时，可以把需要的列名按照用户要求依次列在 SELECT 语句后面即可，字段与字段之间用英文半角逗号隔开，如果需要查询所有信息，可以使用“*”号代替表中所有的字段。

【例 4-10】查询 tb_book 表中图书的名称(bookname)及其作者名称(author)。

```
SELECT bookname,author FROM tb_book;
```

(2)指定列别名。

在缺省情况下,查询结果中的列标题是表中的列名或者无列标题。如果希望对列标题进行修改,或者为没有标题的列加上标题时,可以使用 AS 子句改变查询结果中的列标题。其语法格式为:

```
SELECTcolumn_name AS column_alias
```

【例 4-11】查询 tb_book 表中图书的名称(bookname)及其作者名称(author),结果中的标题显示为图书名称和作者。

```
SELECT bookname AS 图书名称,author AS 作者 FROM tb_book;
```

2. 条件查询

数据表中可以存储成千上万条记录,一般情况下,我们查询时并不需要将数据表中所有数据都显示出来,而是有目的的查询数据。为了在数据表中查找满足条件的记录,需要使用 WHERE 子句,通过使用它指定一系列查询条件来保证查询结果集中只包含所需的记录。条件查询的语法格式如下:

```
SELECT  select_expression,...FROM table_list WHERE where_definition
```

在使用 WHERE 子句时,它必须紧跟在 FROM 子句后面,其中 where_definition 条件是一个表达式,其常用运算符如下:

- 关系比较:=、>、<、>=、<=。
- 范围比较:BETWEEN AND(在某个范围内)、NOT BETWEEN AND(不在某个范围内)。
- 列表比较:IN(在某个列表内)、NOT IN(不在某个列表中)。
- 字符串模式匹配:LIKE(标准的 SQL 模式匹配)、REGEXP(扩展的正则表达式模式匹配)。
- 多重条件:AND(&&)、OR(||)、NOT(!)。

(1)关系比较查询。

【例 4-12】查询 tb_book 表中图书价格高于 36 元的图书名称(bookname)及其作者名称(author)。

```
SELECT bookname,author FROM tb_book WHERE bookprice>36;
```

(2)范围比较查询。

当要查询的条件是某个范围时,使用 BETWEEN 关键字指出查询范围,其语法格式如下:

```
expression [NOT] BETWEEN expression1 AND expression2
```

其中,expression1 是范围的下限值,expression2 是范围的上限值;使用 BETWEEN 时,数据范围包含边界值,使用 NOT BETWEEN 时,数据范围不包含边界值。

【例 4-13】查询 tb_book 表中图书价格在 36 元至 48 元的图书名称(bookname)及其作者名称(author)。

```
SELECT bookname, author FROM tb_book WHERE bookprice BETWEEN 36 AND 48;
```

(3)列表比较查询。

当要查询的条件属于某一列表值之一时,使用 IN 关键字指出查询列表,其语法格式如下:

```
expression [NOT] IN(expression1,expression2,expression3,…)
```

【例 4-14】查询 tb_book 表中作者为“杨学全”、“李英杰”、“龙应台”的图书名称(bookname)

及其作者名称(author)。

```
SELECT bookname, author FROM tb_book WHERE author IN ('杨学全','李英杰','龙应台');
```

(4)字符串模式匹配查询。

字符串模式匹配查询,又称为模糊查询,是在查询条件中使用字符串匹配符号 LIKE 或 REGEXP 把表达式与含有通配符的字符串进行比较,实现对字符串的模糊查询。

①使用 LIKE 运算符的标准的 SQL 模式匹配查询。

LIKE 是 MySQL 提供的标准的 SQL 模式匹配中用到的运算符,它通常和"_"(匹配单个字符)、"%"(匹配任意数目字符)通配符搭配使用进行模糊查询,其语法格式如下:

```
expression [NOT] LIKE match_ expression
```

【例 4-15】查询 tb_book 表中图书名称中含"VB"内容的图书名称(bookname)及其作者(author)。

```
SELECT bookname, author FROM tb_book WHERE bookname LIKE '%VB%';
```

②使用 REGEXP 运算符的扩展的正则表达式模式匹配查询。

MySQL 除了使用标准的 SQL 模式进行简单的模糊查询外,还可以使用正则表达式完成复杂的模糊查询。当使用这类模式进行匹配查询时,使用 REGEXP 和 NOT REGEXP 操作符(或 RLIKE 和 NOT RLIKE,它们作用相同),其语法格式如下:

```
expression [NOT] REGEXP|RLIKE match_ expression
```

扩展正则表达式常用的一些匹配字符有:

- ^:匹配字符串的开始部分。
- $:匹配字符串的结尾部分。
- ?.:匹配任何单个的字符。
- ? *:匹配零个或多个星号前面的字符。
- +:匹配 1 个或多个加号前面的字符。
- |:匹配符号左边或右边出现的字符串。
- {n}:匹配括号前的内容出现 n 次。
- ():匹配括号里面的内容。
- [...]:匹配在方括号内的任何字符。例如,"[abc]"匹配"a"、"b"或"c"。为了命名字符的一个范围,使用一个"-"。"[a-z]"匹配任何小写字母,而"[0-9]"匹配任何数字。

【例 4-16】查询 tb_book 表中图书名称中含"VB"或"SQL"内容的图书名称(bookname)及其作者(author)。

```
SELECT bookname, author FROM tb_book WHERE bookname REGEXP 'VB|SQL';
```

(5)多重条件查询。

多重条件查询是指查询条件有多个,需要使用逻辑运算符进行条件连接的查询,常用的逻辑运算符有 NOT、AND、OR。

【例 4-17】查询 tb_book 表中图书名称中含"VB"内容并且作者姓"李"的图书名称(bookname)及其作者(author)。

```
SELECT bookname, author FROM tb_book WHERE bookname LIKE '%VB%' AND author LIKE '李%';
```

3. 排序查询

在执行查询操作时,默认情况下查询结果以数据存储在表中的顺序显示。为了有助于数据

的浏览和查阅，在实际应用中经常需要对数据进行排序。对查询结果排序的子句是 ORDER BY，其语法格式如下：

```
ORDER BY sorting_columns [ASC | DESC],...
```

【例 4-18】按价格降序显示 tb_book 表中图书名称(bookname)和价格(bookprice)。

```
SELECT bookname, bookprice  FROM tb_book ORDER BY bookprice DESC;
```

4. 统计查询

在实际应用中，经常会对查询结果进行分类、统计和汇总等操作，这些操作可以通过聚合函数、GROUP BY 子句等来实现。常用的聚合函数有 AVG()、COUNT()、MAX()、MIN()、SUM()等。

(1)使用聚合函数。

【例 4-19】统计 tb_book 表中图书名称中含“VB”内容的图书的种数、平均价格、最高价格和最低价格。

```
SELECT COUNT( * ) AS 书的种数, AVG(bookprice) AS 平均价格 ,MAX(bookprice) AS 最高价格,MIN(bookprice)AS 最低价格 FROM tb_book WHERE bookname LIKE '%VB%';
```

(2)使用 GROUP BY 子句。

单独使用聚合函数只能返回单一的汇总结果，如果需要显示分组统计数据，就需要使用 GROUP BY 子句的简单形式。该子句和聚合函数一起使用可以实现对数据的分组统计，在查询结果中，每一组将统计出一个结果。

【例 4-20】统计显示 tb_book 表中每个出版社图书的图书种数。

```
SELECT pubhouse,count( * ) AS 图书种数 FROM tb_book  GROUP BY pubhouse;
```

(3)使用 HAVING 子句限制分组结果。

HAVING 与 WHERE 子句都是搜索条件语句，其区别在于 WHERE 搜索条件在进行分组操作之前应用，跟在 FROM 子句之后；而 HAVING 搜索条件在进行分组操作之后应用，跟在 GROUP BY 子句之后。

【例 4-21】统计显示 tb_book 表中图书种数大于 3 的出版社的名称及图书种数。

```
SELECT pubhouse,count( * ) AS 图书种数 FROM tb_book  GROUP BY pubhouse HAVING count( * )>3;
```

5. 连接查询

在数据库应用中，经常需要从两个或多个表中查询数据，这时，就需要使用连接查询。连接查询主要包含内连接和外连接。

(1)内连接。

使用内连接时，如果连接表中的相关字段满足连接条件，则从连接表中提取数据并组合成新的记录作为结果集，其语法格式如下：

```
SELECTselect_expression FROM  table_reference1 [INNER] JOIN table_reference2 ON conditional_expr
```

【例 4-22】连接 tb_type 表和 tb_book 表，查询图书的分类情况，查询结果包含图书的分类号(typeid)、分类名称(typename)及图书名称(bookname)。

```
SELECT tb_type.typeid, tb_type.typename, tb_book.bookname  FROM tb_type  JOIN tb_book ON tb_book.typeid=tb_type.typeid;
```

(2)外连接。

在内连接中，只有满足条件的记录才能在结果集里输出。而外连接扩展了内连接的结果，除返回所有满足条件的记录外，还会返回一部分或全部不满足条件的记录。外连接分为左外连

接、右外连接和全外连接，其语法格式如下：

```
SELECTselect_expression FROM table_reference1 LEFT| RIGHT |FULL [OUTER] JOIN table_reference2 ON conditional_expr
```

其中：

• LEFT：指左外连接，结果集中除了满足连接条件的记录外，还有左表中不满足连接条件的记录，该记录对应的右表列上自动填充 NULL 值。

• RIGHT：指右外连接，结果集中除了满足连接条件的记录外，还有右表中不满足连接条件的记录，该记录对应的左表列上自动填充 NULL 值。

• FULL：指全外连接，结果集中除了满足连接条件的记录外，还有左、右表中不满足连接条件的记录，在左、右表的相应列上填充 NULL 值。

【例 4-23】用 tb_type 表左外连接 tb_book 表，查询图书的分类情况，查询结果包含图书的分类号(typeid)、分类名称(typename)和图书名称(bookname)。

```
SELECT tb_type.typeid, tb_type.typename, tb_book.bookname  FROM tb_type  LEFT JOIN tb_book ON tb_book.typeid=tb_type.typeid;
```

6. 子查询

在查询语句中，可以将一条查询语句嵌套在另外一条查询语句中作为条件的一部分使用，被嵌套的查询语句称为子查询，嵌套子查询的查询语句称为父查询。嵌套查询的求解方法是由里向外，即每个子查询在上一级查询处理之前求解，子查询的结果用于建立父查询的查找条件。子查询通常与 IN 或者比较运算符结合使用，当子查询的结果为单值时，使用比较运算符，结果为多值时，使用 IN 运算符。

【例 4-24】使用子查询显示“计算机类”图书信息。

```
SELECT * FROM tb_book WHERE typeid =(
SELECT typeid FROM tb_type WHERE typename='计算机类') ;
```

7. 限制返回行数

使用 LIMIT 子句可以限制查询语句返回的记录数，其语法格式如下：

```
LIMIT [offset,] rows
```

其中，offset 用来指定要返回的第一行的偏移量，其初始行的偏移量是 0，rows 用来指定返回行的行数。例如，“LIMIT 2”表示返回结果集中前 2 条记录；“LIMIT 1,5”表示从第 2 行开始返回 5 条记录，即返回第 2～6 条记录。

【例 4-25】查询显示 tb_book 表中第 2 和第 3 条记录。

```
SELECT * FROM tb_book LIMIT1,2;
```

4.5　技术拓展

4.5.1　数据完整性

数据完整性是指存储在数据库中的数据正确无误并且相关数据具有一致性。数据完整性可以由约束来实现，约束是数据库系统提供的自动强制数据完整性的机制，它通过定义表中列

的取值规则来维护数据的完整性。约束可以在创建数据表时创建，也可以在已存在的表上添加。在 MySQL 数据库中，可以创建主键约束、唯一约束、非空约束、默认约束、外键约束等。

1. 主键约束

主键约束是最常用的约束，一般的数据表中都应该有一个主键约束。主键约束是将表中的一列或多列定义成一个主键来唯一标识表中的每行记录，其有如下特点：每个表中只能有一个主键，主键可以由一列组成，也可以由多列的组合而成；主键值必须唯一并且不能为空，对于多列组合的主键，某列值可以重复，但列的组合值必须唯一。例如，在图书表中可以将图书编号设置为主键，用来保证表中的图书记录具有唯一性。在项目实施中，每个表都创建了主键，创建数据表的同时创建主键约束的代码参考项目实施代码即可，下面介绍修改主键的方法。

添加主键约束的语法格式如下：

```
ALTER TABLE tbl_name ADD PRIMARY KEY (index_col_name,...)
```

删除主键约束的法格式如下：

```
ALTER TABLE tbl_name DROP PRIMARY KEY
```

【例 4-26】将 tb_type 表中 typeid 设置为主键。

```
ALTER TABLE tb_type ADD PRIMARY KEY (typeid );
```

2. 唯一约束

唯一约束用来限制表中非主键列中不允许输入重复值。唯一约束有如下特点：一个表中可以定义多个唯一约束；每个唯一约束可以定义到一列上，也可以定义到多列上；空值可以出现在某列中一次。例如在图书类别表中可以将类别编码作为主键，用来保证记录的唯一性。如果不允许有同名类别存在，应该为类别名称列定义唯一约束，保证非主键列中不出现重复值。

添加唯一约束语法格式如下：

```
ALTER TABLE tbl_name ADD UNIQUE (index_col_name,...)
```

删除唯一约束语法格式如下：

```
ALTER TABLE tbl_name DROP INDEX index_col_name
```

【例 4-27】为 tb_type 表中 TypeName 字段设置唯一约束。

```
ALTER TABLE tb_type ADD UNIQUE (TypeName);
```

3. 非空约束

非空约束用于设定某列值不能为空来实现数据完整性。如果指定某列不能为空，则在进行添加记录时，此列必须添加数据。例如，对于管理员表，有一个管理员，就必须有相应的密码，这时，就应该设置密码列不能空。非空约束使用 NOT NULL 选项实现，如下面创建管理员表(tb_admin)的代码中，用户名(tb_admin)和密码(password)字段都设置了非空约束。

```
CREATE TABLE IF NOT EXISTS tb_admin (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(30) NOT NULL,
  password varchar(50) NOT NULL,
  PRIMARY KEY (id)
) ENGINE=MyISAM  DEFAULT CHARSET=gb2312;
```

在此代码中，id int(11) NOT NULL AUTO_INCREMENT 的作用是为表创建自增量字段 id，自增量字段必须是数值类型，且其值不能重复，其作用是每增加一条记录，该字段的值就自动加 1。

4. 默认约束

默认约束用来为表中某列建立一个默认值，当用户添加记录时，如果没有为该列提供输入值，则系统会自动将默认值赋给该列。使用默认约束可以提高输入记录的速度。添加默认约束的语法格式为：

```
ALTER TABLE tbl_name ALTER COLUMN columnname SET DEFAULT default values;
```

【例 4-28】为图书信息表(tb_book)中出版社字段(pubhouse)添加默认值“东软电子出版社”。

```
ALTER TABLE  tb_book  ALTER COLUMN  pubhouse SET DEFAULT  '东软电子出版社';
```

5. 外键约束

外键是指一个表中的一列或列组合，这些列可以引用同一个表或其他表的主键约束列或唯一约束列。通过外键约束可以为相关联的两个表建立联系，实现数据的参照完整性，维护两表之间数据的一致性关系。外键的取值要依赖于它引用的主键值或唯一约束字段的值，也就是引用表中主键或唯一约束列存在什么值，外键才可以使用什么值。例如，如果要求 tb_book 中“TypeID”列的取值，必须是 tb_type 表中“TypeID”列的列值之一，这就应该在 tb_book 表的“TypeID”上创建外键约束，使 tb_book 表和 tb_type 表中的图书类别号具有一致性，从而防止 tb_book 表中出现错误的图书类别号。添加外键约束的语法格式如下：

```
ALTER TABLE tbl_name ADD [CONSTRAINT symbol] FOREIGN KEY (col_name,...) REFERENCES tbl_name(col_name,...)
```

【例 4-29】为 tb_book 表中 TypeID 字段添加外键约束，从而保证在 tb_book 表中输入 TypeID 值有效，即与 tb_type 表中的图书类型号(TypeIID)具有一致性。

```
ALTER TABLE tb_book ADD CONSTRAINT fk_id FOREIGN KEY(TypeID) REFERENCES tb_type(typeId);
```

4.5.2　索引

索引是以数据表中的列为基础创建的数据库对象，它包含由数据表中的一列或多列生成的排序索引列，并且记录了索引列在数据表中的物理存储位置，实现了表中数据的逻辑排序。索引对大数据量的表有较大意义，可以提高数据的查询效率。如果没为表创建索引，在进行数据条件查询时，MySQL 将从第一条记录开始，然后依次扫描表中的所有记录，实现整表扫描后，才能找出符合条件的记录。这样，表中的数据越多，查询花费的时间越多。如果对查询的条件列建立索引，MySQL 就可以不用整表扫描，而通过索引定位，快速查询到所有符合条件的记录。

1. 创建索引

创建索引的语法格式如下：

```
CREATE [UNIQUE] INDEX index_name ON tbl_name (col_name[(length)],... )
```

【例 4-30】为 tb_book 表中 BookName 字段创建索引 I_bookname。

```
CREATE INDEXI_bookname ON tb_book (BookName(50));
```

2. 删除索引

删除索引的语法格式如下：

```
DROP INDEX index_name ON tbl_name
```

3. 重建索引

重建索引的语法格式如下：

```
REPAIR TABLE index_name QUICK;
```

4. 查看索引

查看索引的语法格式如下：

```
SHOW INDEX FROMindex_name
```

4.5.3 视图

视图是在数据表的基础上，通过查询语句生成的虚拟表。视图与数据表相似，也是由一些数据列和数据行组成，可以作为查询语句的数据源使用。视图与数据表的区别在于：视图只存储构成视图的结构，不保存实际数据，其所对应的数据存储在视图所引用的数据表中，在使用视图时动态生成。使用视图可以简化复杂查询的结构，从而方便用户对数据的操作。例如，当需要查询的信息涉及多张表时，编写查询语句就相当繁琐，这时，就可以把复杂的查询语句定义为一个视图，以后用户再进行查询时就不必编写复杂的查询语句，而只需要一条简单的查询视图语句即可。

1. 创建视图

视图的创建者必须拥有创建视图的权限，并且对视图中引用的数据表有许可权。创建视图的语法格式如下：

```
CREATE VIEW view_name[(column_list)]
AS
select_statement
```

其中：

- view_name：是新建视图的名称，视图名称要唯一；
- column_list：是视图中的列名，如果没有指定列名，其列名由 SELECT 语句指派；
- select_statement：是创建视图的 SELECT 语句，该语句可以使用多个表或其他视图。

【例 4-31】创建分类统计图书种数的视图 V_countbook，其包含图书类别(typename)及图书种数。

```
CREATE VIEW V_countbook
AS
SELECT typename AS 图书类别,COUNT( * ) AS 图书种数 FROM tb_type INNER JOIN tb_book ON tb_type.
typeid=tb_book.typeid
GROUP BY typename;
```

2. 使用视图

视图创建之后，可以像使用基本表一样对视图进行查询。

【例 4-32】使用视图 V_countbook 查看图书种数。

```
SELECT * FROM V_countbook;
```

3. 修改视图

修改视图的语法格式如下：

```
ALTER VIEW view_name
  AS
  select_statement
```

4. 删除视图

如果不再需要一个视图，可以将其定义从数据库中删除，其语法格式如下：

```
DROP VIEW view_name;
```

4.5.4　存储过程

存储过程是为了实现某个特定功能，由一组经过预编译的 SQL 语句组成的数据库对象；存储过程存储在服务器端，由用户在客户端通过指定存储过程的名称调用执行。使用存储过程也可以保证数据的完整性，提高应用的性能，因此，任何设计良好的数据库应用程序都会用到存储过程。

1. 创建与执行存储过程

在 MySQL 中，创建存储过程的语句是 CREATE PROCEDURE，其语法格式如下：

```
CREATE PROCEDURE procedure_name ([[IN|OUT|INOUT]parameter type[,...]])
routine_body
```

其中：

- procedure_name：将要创建的存储过程的名字。
- parameter type：存储过程的参数及参数数据类型。
- IN|OUT|INOUT：IN 表示参数为输入参数；OUT 表示参数为输出参数；INOUT 表示参数为输入/输出参数。
- routine_body：存储过程中的 SQL 语句块，一般以 begin 开始，end 结束。

执行存储过程的语法格式如下：

```
CALL procedure_name(parameter[,...])
```

(1)创建与执行不带参数的存储过程。

【例 4-33】创建并执行存储过程 proc_Book，使用该存储过程显示图书的编号和名称。其代码如下：

```
USE db_shop;
--把语句结束符改为"//"
DELIMITER //
--创建存储过程
CREATE PROCEDUREproc_Book()
BEGIN
SELECTbookid,bookname FROM tb_book;
END//
--把语句结束符重新改为";"
DELIMITER ;
--执行存储过程
CALL proc_Book ;
```

(2)创建与执行带参数的存储过程。

【例 4-34】创建并执行存储过程 proc_Type，使用该存储过程为 tb_type 添加数据。其代码如下：

```
USEdb_shop;
```

```
--把语句结束符改为"//"
DELIMITER //
--创建存储过程
CREATE PROCEDUREproc_Type (IN iid INT ,IN itypename VARCHAR(30))
BEGIN
INSERT INTO tb_type(typeid,typename)VALUES(iid,itypename);
END //
--把语句结束符重新改为";"
DELIMITER ;
--执行存储过程
CALL proc_Type(4,'农林') ;
```

2. 查看、修改和删除存储过程

(1)查看存储过程的定义信息。

查看存储过程定义的语法格式如为:SHOW CREATE PROCEDURE procedure_name;

(2)查看当前数据库中存储过程列表。

查看存储过程列表,包含存储过程的名称、拥有者、类型和创建时间等信息,其语法格式为:

```
SHOW  PROCEDURE STATUS;
```

(3)修改存储过程。

修改存储过程的语法格式如下:

```
ALTER PROCEDURE procedure_name ([[IN|OUT|INOUT]parameter type [,...]])
routine_body;
```

(4)删除存储过程。

删除存储过程的语法格式如下:

```
DROP PROCEDURE procedure_name;
```

4.5.5 触发器

触发器是一种特殊类型的存储过程,与存储过程类似,由 SQL 语句组成,可以实现一定的功能;不同的是,触发器的执行不能通过名称调用来完成,而是当用户对数据库发生事件时,如 INSERT、DELETE、UPDATE 数据时,将会自动触发与该事件相关的触发器,使其自动执行。

1. 创建及使用触发器

触发器不允许带参数,它的定义与表紧密相连,可以作为表的一部分,其创建语法格式如下:

```
CREATE TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW
BEGIN
trigger_stmt
END
```

其中:

- trigger_name:触发器名称。
- trigger_time:触发程序的动作时间,它可以是 BEFORE 或 AFTER,以指明触发程序是

在激活它的语句之前或之后触发。

• trigger_event：触发事件，用来指明激活触发程序的语句的类型，其值可以是 INSERT、UPDATE 或 DELETE 语句。

• tbl_name：被定义触发器的表。

• trigger_stmt：触发程序执行的 SQL 语句。

在触发器中，可以使用特殊名称的虚拟表 OLD 和 NEW 引用与触发程序相关表中的列。在 INSERT 触发程序中，仅能使用 NEW. col_name 来引用将要添加的新行的一列；在 DELETE 触发程序中，仅能使用 OLD. col_name 引用已有行中的一列；在 UPDATE 触发程序中，可以使用 OLD. col_name 来引用更新前的某一行的列，也能使用 NEW. col_name 来引用更新后的行中的列。

【例 4-35】创建 tb_type 表的备份表 tb_type_bak，在 tb_type 表上创建触发器 t_bak，其作用是将 tb_type 中新添加的记录备份到 tb_type_bak 中。

```
USE db_shop;
CREATE TABLE tb_type_bak(typeid INT,typename varchar(30));
--把语句结束符改为"//"
DELIMITER //
--创建触发器
CREATE TRIGGER t_bak AFTER INSERT
ON tb_type FOR EACH ROW
BEGIN
INSERT INTO tb_type_bak(typeid,typename)VALUES(NEW.TYPEID,NEW.TYPENAME);
END //
--把语句结束符重新改为";"
DELIMITER ;
--为 tb_type 添加数据
INSERT INTO tb_type (typeid,typename)VALUES('15','电子类');
--检查触发器执行情况
SELECT * FROM tb_type;
SELECT * FROM tb_type_bak;
```

2. 删除触发器

删除触发器的语法格式如下：

```
DROP TRIGGER trigger_name;
```

4.6　本章小结

本章是开发图书商城网站的数据库设计与实现部分，主要介绍了 MySQL 数据库的创建与管理、数据类型及数据表的创建与管理、数据添加、修改、删除和查询等 SQL 操作，并实现了图书商城网站数据库。

数据库用来存储数据，数据以表的形式存储在数据库中，数据库设计要根据系统需求，确定

需要哪些表，每个表中都有哪些列及每一列的数据类型，哪些列允许空值，哪里需要索引，哪些列是主键，哪些列是外键等。

创建数据库，可以在 MySQL 控制台上完成，但是操作比较繁琐，也可以使用 MySQL 图形化管理工具，使用图形化管理工具是创建数据库的常用方式。

4.7 强化练习

一、填空题

1. SQL 的含义是(　　)。

A. 结构化查询语言　　B. 过程化语言

C. 非过程化语言　　D. 非标准化语言

2. 创建数据表时，不允许某列为空可以使用(　　)命令。

A. NO NULL　　B. NOT NULL

C. NULL　　D. NOT BLANK

3. 一个表中的主键(　　)。

A. 可以有多个　　B. 值可以为空

C. 值必须唯一　　D. 值可以重复

4. 如果一条查询语句涉及以下子句，哪一子句在最后(　　)。

A. WHERE　　B. ORDER BY

C. HAVING　　D. LIMIT

5. 从图书表(tb_book)中查询名称(bookname)以“PHP”开头的图书信息的语句为(　　)。

A. SELECT * FROM tb_book WHERE bookname LIKE 'PHP%'

B. SELECT * FROM tb_book WHERE bookname='PHP%'

C. SELECT * FROM tb_book WHERE bookname='PHP_'

D. SELECT * FROM tb_book WHERE bookname LIKE 'PHP_'

6. 如果图书表(tb_book)中有 100 条记录，查询语句 SELECT * FROM tb_book LIMIT 5,3 将显示多少条记录(　　)。

A. 5　　B. 3

C. 8　　D. 15

7. 在查询语句中，统计记录数量的函数是(　　)。

A. AVG()　　B. COUNT()

C. MAX()　　D. SUM()

8. 在子查询中，如果查询结果为多值时，子查询和父查询连接的运算符为(　　)。

A. IN　　B. =

C. LIKE　　D. OR

9. 视图保存的内容是(　　)。

A. 数据　　B. 查询语句

C. 表结构　　D. 添加语句

10. DELIMITER 语句的作用是(　　)。

A. 设置存储过程环境　　B. 改变语句结束符号

C. 执行存储过程的命令　　D. 设置存储过程参数

二、简答题

1. MySQL 提供了哪些数据类型?

2. 修改表数据和删除表数据的 SQL 命令是什么?

3. MySQL 数据库系统提供的约束主要有什么?

4. 主键约束的特点是什么?

5. 创建触发器时,用到的系统虚拟表有哪些?