


## 1.1 下载安装 MySQL

### 1.1.1 下载 MySQL

MySQL 是一款开源的数据库软件，由于其免费特性得到了全世界用户的喜爱，是目前使用人数最多的数据库。下面将详细讲解如何下载和安装 MySQL 库。

 **说明：**由于 MySQL 的版本在持续更新，所以本章选择相对稳定的 MySQL 5.7 版本下载使用，读者也可以下载当前最新的 MySQL 8.0 版本，本书内容都是通用的。

在浏览器的地址栏中输入地址 “<https://dev.mysql.com/downloads/windows/installer/5.7.html>”，并按下〈Enter〉键，将进入到当前最新版本 MySQL 5.7 的下载页面，选择离线安装包，如图 1.1 所示。



图 1.1 下载 MySQL

单击 “Download” 按钮下载，进入开始下载页面，如果有 MySQL 的账户，可以单击 Login 按钮，登录账户后下载，如果没有可以直接单击下方的 “No thanks, just take me to the download.” 超链接，跳过注册步骤，直接下载，如图 1.2 所示。



图 1.2 不注册下载

### 1.1.2 安装 MySQL

下载完成以后，开始安装 MySQL。双击安装文件，在所示界面中勾选“I accept the license terms”，点击“next”，进入选择设置类型界面。在选择设置中有 5 种类型，说明如下：

- ☒ **Developer Default:** 安装 MySQL 服务器以及开发 MySQL 应用所需的工具。工具包括开发和管理服务器的 GUI 工作台、访问操作数据的 Excel 插件、与 Visual Studio 集成开发的插件、通过 NET/Java/C/C++/ODBC 等访问数据的连接器、例子和教程、开发文档。
- ☒ **Server only:** 仅安装 MySQL 服务器，适用于部署 MySQL 服务器。
- ☒ **Client only:** 仅安装客户端，适用于基于已存在的 MySQL 服务器进行 MySQL 应用开发的情况。
- ☒ **Full:** 安装 MySQL 所有可用组件。
- ☒ **Custom:** 自定义需要安装的组件。

MySQL 会默认选择“Developer Default”类型，这里我们选择纯净的“Server only”类型，如图 1.3 所示，然后一直默认选择安装。

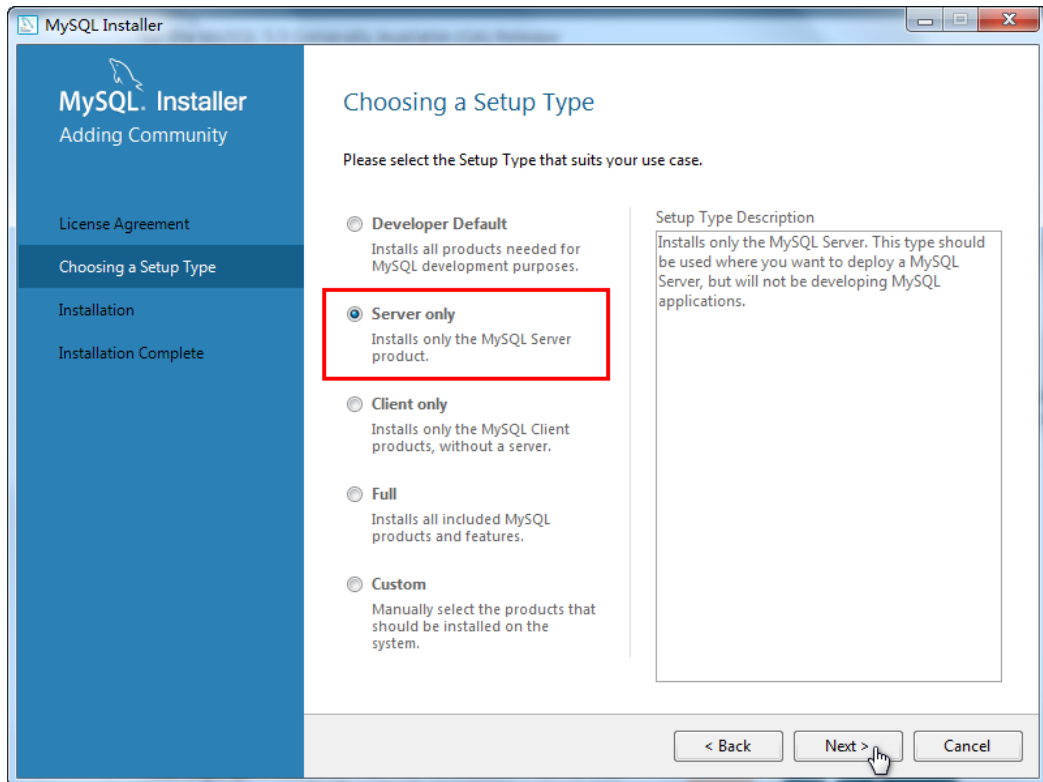


图 1.3 选择安装类型

### 1.1.3 设置环境变量

安装完成以后，默认的安装路径是“C:\Program Files\MySQL\MySQL Server 5.7\bin”。下面设置环境变量，以便在任意目录下使用 MySQL 命令。右键单击“计算机”→选择“属性”→选择“高级系统设置”→选择“环境变量”→选择“PATH”→单击“编辑”。将“C:\Program Files\MySQL\MySQL Server 5.7\bin”写在变量值中。如图 1.4 所示。

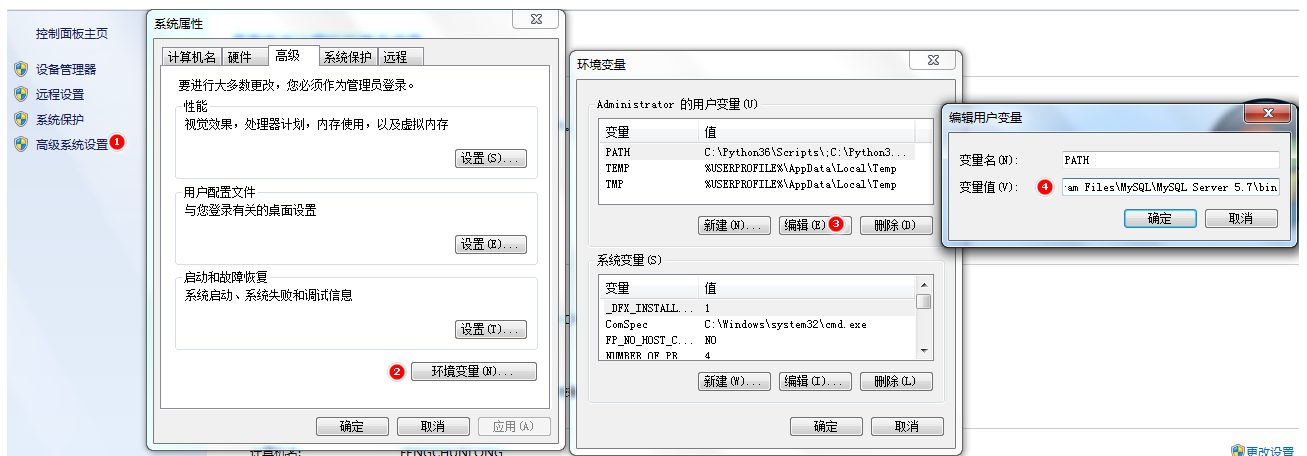


图 1.4 设置环境变量

### 1.1.4 启动和关闭 MySQL 服务

使用 MySQL 数据库前，需要先启动 MySQL。在 cmd 窗口中，输入命令行“net start mysql57”，来启动 MySQL 5.7。启动成功后，使用账户和密码进入 MySQL。输入命令“mysql -u root -p”，接着提示“Enter password:”，输入密码“root”即可进入 MySQL。如图 1.5 所示。

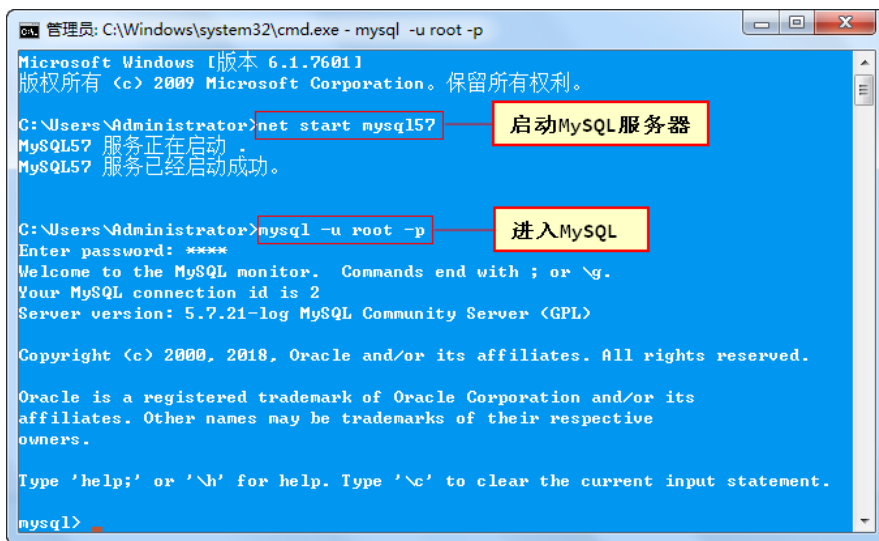


图 1.5 启动 MySQL

在 MySQL 控制台中，输入“exit()”即可退出 MySQL 控制台。然后输入“net stop mysql57”即可关闭 MySQL 服务。

## 1.2 操作 MySQL 数据库

针对 MySQL 数据库的操作可以分为创建、选择和删除三种，下面分别介绍这三种操作。

### 1.2.1 创建数据库

在 MySQL 中，应用 create database 语句创建数据库。其语法格式如下：

```
create database 数据库名;
```

在创建数据库时，数据库的命名要遵循如下规则：

- ☑ 不能与其他数据库重名。
- ☑ 名称可以是任意字母、阿拉伯数字，下划线（\_）或者“\$”组成，可以使用上述的任意字符开头，但不能使用单独的数字，那样会造成它与数值相混淆。

- ☑ 名称最长可为 64 个字符组成（还包括表、列和索引的命名），而别名最多可长达 256 个字符。
- ☑ 不能使用 MySQL 关键字作为数据库、表名。
- ☑ 默认情况下，Windows 下数据库名、表名的字母大小写是不敏感的，而在 Linux 下数据库名、表名的字母大小写是敏感的。为了便于数据库在平台间进行移植，建议读者采用小写字母来定义数据库名和表名。

下面通过 `create database` 语句创建一个名称为 `db_userss` 的数据库。在创建数据库时，首先连接 MySQL 服务器，然后编写“`create database db_users;`”SQL 语句，数据库创建成功。运行结果如图 1.6 所示。

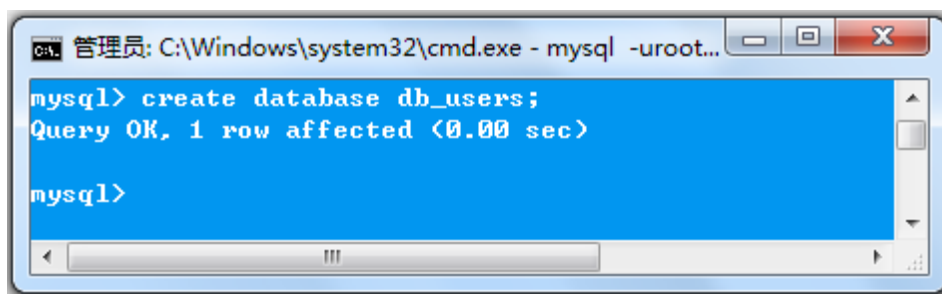


图 1.6 创建数据库

创建 `db_users` 数据库后，MySQL 管理系统会自动在 MySQL 安装目录下的“`MySQL\data`”目录下创建 `db_users` 数据库文件夹及相关文件实现对该数据库的文件管理。

## 1.2.2 选择数据库

`use` 语句用于选择一个数据库，使其成为当前默认数据库。其语法如下：

`use 数据库名;`

例如，选择名称为 `db_users` 的数据库，操作命令如图 1.7 所示。

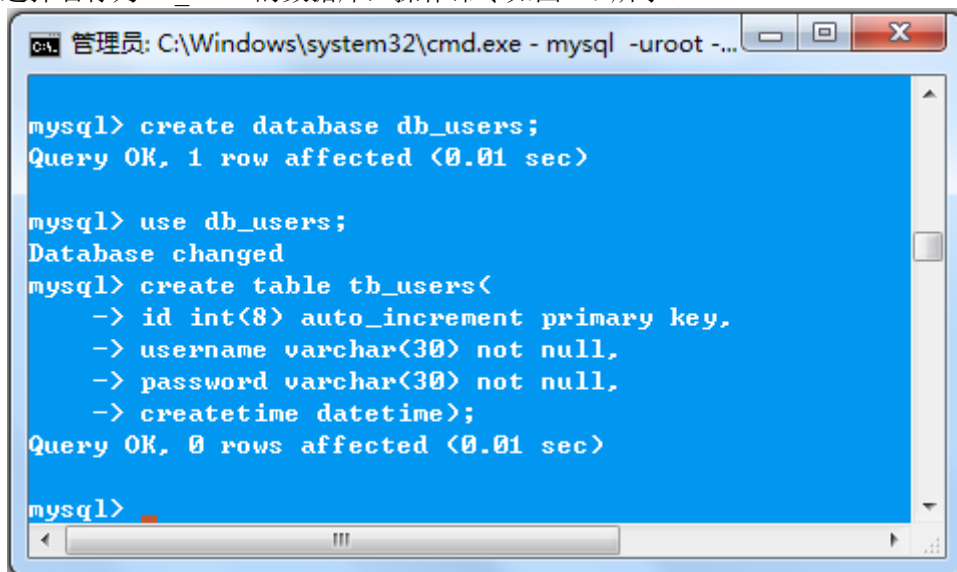


图 1.7 选择数据库

选择了 `db_users` 数据库之后，才可以操作该数据库中的所有对象。

### 1.2.3 查看数据库

数据库创建完成后，可以使用 `show databases` 命令查看 MySQL 数据库中所有已经存在的数据库。语法如下：

```
show databases
```

例如，使用“`show databases`”命令显示本地 MySQL 数据库中所有存在的数据库名，如图 1.8 所示。

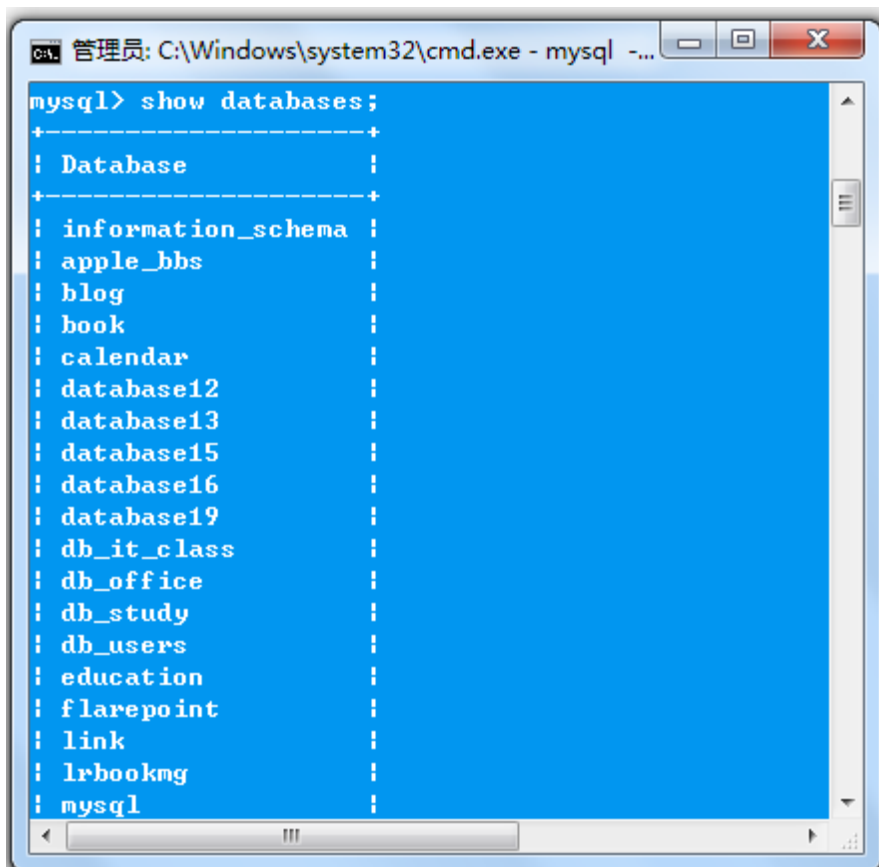


图 1.8 显示所有数据库名



注意：“`show databases`”是复数形式，并且所有命令都以英文分号“`;`”结尾。

### 1.2.4 删除数据库

删除数据库使用的是 `drop database` 语句，语法如下：

```
drop database 数据库名;
```

例如，在 MySQL 命令窗口中使用“drop database db\_userss;” SQL 语句即可删除 db\_users 数据库，如图 1.9 所示。删除数据库后，MySQL 管理系统会自动删除 MySQL 安装目录下的“MySQL\data\db\_users”目录及相关文件。

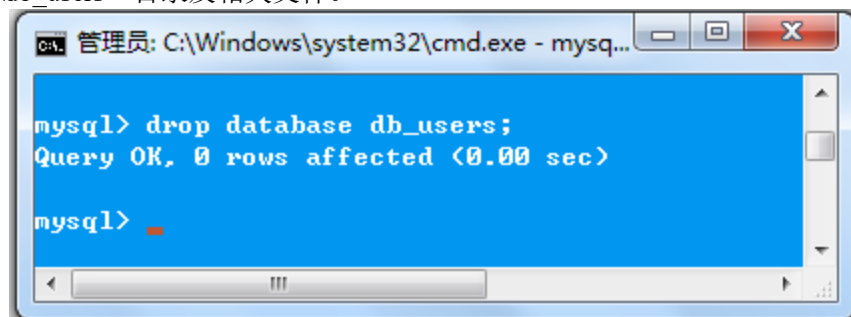


图 1.9. 删除数据库



**注意：**对于删除数据库的操作，应该谨慎使用，一旦执行这项操作，数据库的所有结构和数据都会被删除，没有恢复的可能，除非数据库有备份。

## 1.3 MySQL 数据类型

在 MySQL 数据库中，每一条数据都有其数据类型。MySQL 支持的数据类型主要分成三类：数字类型、字符串（字符）类型、日期和时间类型。

### 1.3.1 数字类型

MySQL 支持的数字类型包括准确数字的数据类型（NUMERIC、DECIMAL、INTEGER 和 SMALLINT），还包括近似数字的数据类型（FLOAT、REAL 和 DOUBLE PRECISION）。其中的关键字 INT 是 INTEGER 的简写，关键字 DEC 是 DECIMAL 的简写。

一般来说，数字类型可以分成整型和浮点型两类，详细内容如表 1.1 和表 1.2 所示。

表 1.1 整数数据类型

数据类型	取值范围	说明	单位
TINYINT	符号值：-127~127 无符号值：0~255	最小的整数	1 字节
BIT	符号值：-127~127 无符号值：0~255	最小的整数	1 字节
BOOL	符号值：-127~127 无符号值：0~255	最小的整数	1 字节
SMALLINT	符号值：-32768~32767 无符号值：0~65535	小型整数	2 字节

数据类型	取值范围	说明	单位
MEDIUMINT	符号值: -8388608~8388607 无符号值: 0~16777215	中型整数	3 字节
INT	符号值: -2147683648~2147683647 无符号值: 0~4294967295	标准整数	4 字节
BIGINT	符号值: -9223372036854775808~ 9223372036854775807 无符号值: 0~18446744073709551615	大整数	8 字节

续表

表 1.2 浮点数据类型

数据类型	取值范围	说明	单位
FLOAT	+(-) 3. 402823466E+38	单精度浮点数	8 字节或 4 字节
DOUBLE	+(-) 1. 7976931348623157E+308 +(-) 2. 2250738585072014E-308	双精度浮点数	8 字节
DECIMAL	可变	一般整数	自定义长度



**说明:** 在创建表时, 使用哪种数字类型, 应遵循以下原则:

- (1) 选择最小的可用类型, 如果值永远不超过 127, 则使用 TINYINT 要比使用 INT 好。
- (2) 对于完全都是数字的, 可以选择整数类型。
- (3) 浮点类型用于可能具有小数部分的数。例如, 货物单价、网上购物交付金额等。

### 1.3.2 字符串类型

字符串类型可以分为三类: 普通的文本字符串类型 (CHAR 和 VARCHAR)、可变类型 (TEXT 和 BLOB) 和特殊类型 (SET 和 ENUM)。它们之间都有一定的区别, 取值的范围不同, 应用的地方也不同。

(1) 普通的文本字符串类型, 即 CHAR 和 VARCHAR 类型, CHAR 列的长度在创建表时指定, 取值在 1~255 之间; VARCHAR 列的值是变长的字符串, 取值和 CHAR 一样。下面介绍普通的文本字符串类型如表 1.3 所示。

表 1.3 普通的文本字符串类型

类型	取值范围	说明
[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。Binary 关键字指定了数据是否区分大小写 (默认是区分大小写的)。ASCII 关



类型	取值范围	说明
		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 续表
char	0~255 个字符	char (M) 类似
[national] varchar (M) [binary]	0~255 个字符	长度可变, 其他和 char (M) 类似

(2) TEXT 和 BLOB 类型。它们的大小可以改变, TEXT 类型适合存储长文本, 而 BLOB 类型适合存储二进制数据, 支持任何数据, 如文本、声音和图像等。下面介绍 TEXT 和 BLOB 类型, 如表 1.4 所示。

表 1.4 TEXT 和 BLOB 类型

类型	取值范围	说明
TINYBLOB	2 <sup>8</sup> ~1 (225)	小 BLOB 字段
TINYTEXT	2 <sup>8</sup> ~1 (225)	小 TEXT 字段
BLOB	2 <sup>16</sup> ~1 (65 535)	常规 BLOB 字段
TEXT	2 <sup>16</sup> ~1 (65 535)	常规 TEXT 字段
MEDIUMBLOB	2 <sup>24</sup> ~1 (16 777 215)	中型 BLOB 字段
MEDIUMTEXT	2 <sup>24</sup> ~1 (16 777 215)	中型 TEXT 字段
LONGBLOB	2 <sup>32</sup> ~1 (4 294 967 295)	长 BLOB 字段
LONGTEXT	2 <sup>32</sup> ~1 (4 294 967 295)	长 TEXT 字段

### (3) 特殊类型 SET 和 ENUM

特殊类型 SET 和 ENUM 的介绍如表 1.5 所示。

表 1.5 TEXT 和 BLOB 类型

类型	最大值	说明
Enum ( "value1", "value2", ... )	65 535	该类型的列只可以容纳所列值之一或为 NULL
Set ( "value1", "value2", ... )	64	该类型的列可以容纳一组值或为 NULL



**说明:** 在创建表时, 使用哪种数字类型, 应遵循以下原则:

- (1) 从速度方面考虑, 要选择固定的列, 可以使用 CHAR 类型。
- (2) 要节省空间, 使用动态的列, 可以使用 VARCHAR 类型。
- (3) 要将列中的内容限制在一种选择, 可以使用 ENUM 类型。

- (4) 允许在一个列中有多于一个的条目，可以使用 SET 类型。
- (5) 如果要搜索的内容不区分大小写，可以使用 TEXT 类型。
- (6) 如果要搜索的内容区分大小写，可以使用 BLOB 类型。

1.3.3 日期和时间类型

日期和时间类型包括：DATETIME、DATE、TIMESTAMP、TIME 和 YEAR。其中的每种类型都有其取值的范围，如赋予它一个不合法的值，将会被“0”代替。下面介绍日期和时间数据类型，如表 1.6 所示。

表 1.6 日期和时间数据类型

类型	取值范围	说明
DATE	1000-01-01 9999-12-31	日期，格式 YYYY-MM-DD
TIME	-838:58:59 835:59:59	时间，格式 HH:MM:SS
DATETIME	1000-01-01 00:00:00 9999-12-31 23:59:59	日期和时间，格式 YYYY-MM-DD HH:MM:SS
TIMESTAMP	1970-01-01 00:00:00 2037 年的某个时间	时间标签，在处理报告时使用的显示格式取决于 M 的值
YEAR	1901-2155	年份可指定两位数字和四位数字的格式

在 MySQL 中，日期的顺序是按照标准的 ANSISQL 格式进行输入的。

1.4 操作数据表

数据库创建完成后，即可在命令提示符下对数据库进行操作，如创建数据表、更改数据表结构以及删除数据表等。

1.4.1 创建数据表

MySQL 数据库中，可以使用 create table 命令创建数据表。语法如下：

```
create[TEMPORARY] table [IF NOT EXISTS] 数据表名
[(create_definition,...)][table_options] [select_statement]
```

create table 语句的参数说明如表 1.7 所示。

表 1.7 create table 语句的参数说明

关键字	说明
TEMPORARY	如果使用该关键字，表示创建一个临时表
IF NOT EXISTS	该关键字用于避免表存在时 MySQL 报告的错误
create_definition	这是表的列属性部分。MySQL 要求在创建表时，表要至少包含一列
table_options	表的一些特性参数
select_statement	SELECT 语句描述部分，用它可以快速地创建表

下面介绍列属性 create\_definition 的使用方法，每一列具体的定义格式如下：

```
col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT]
[PRIMARY KEY ] [reference_definition]
```

属性 create\_definition 的参数说明如表 1.8 所示。

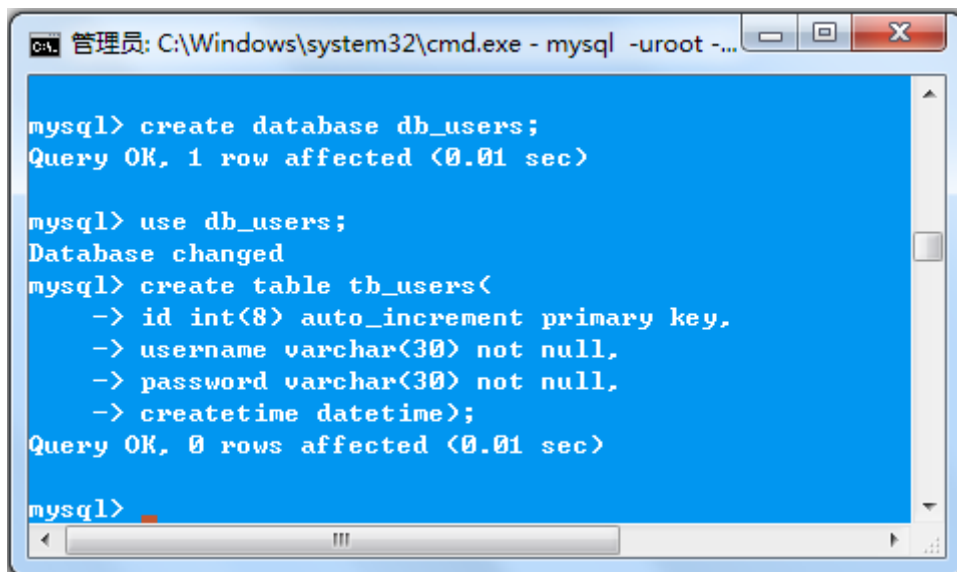
表 1.8 属性 create\_definition 的参数说明

参数	说明
col_name	字段名
type	字段类型
NOT NULL   NULL	指出该列是否允许是空值，但是数据“0”和空格都不是空值，系统一般默认允许为空值，所以当不允许为空值时，必须使用 NOT NULL
DEFAULT default_value	表示默认值
AUTO_INCREMENT	表示是否是自动编号，每个表只能有一个 AUTO_INCREMENT 列，并且必须被索引
PRIMARY KEY	表示是否为主键。一个表只能有一个 PRIMARY KEY。如表中没有一个 PRIMARY KEY，而某些应用程序要求 PRIMARY KEY，MySQL 将返回第一个没有任何 NULL 列的 UNIQUE 键，作为 PRIMARY KEY
reference_definition	为字段添加注释

在实际应用中，使用 create table 命令创建数据表的时候，只需指定最基本的属性即可，格式如下：

```
create table table_name (列名 1 属性, 列名 2 属性 ...);
```

例如，在命令提示符下应用 create table db\_users 创建 db\_users 数据库，然后使用 create table 命令，在数据库 db\_users 中创建一个名为 tb\_users 的数据表，表中包括 id、user、pwd 和 createtime 等字段，实现过程如图 1.10 所示。



```
mysql> create database db_users;
Query OK, 1 row affected (0.01 sec)

mysql> use db_users;
Database changed
mysql> create table tb_users(
    -> id int(8) auto_increment primary key,
    -> username varchar(30) not null,
    -> password varchar(30) not null,
    -> createtime datetime);
Query OK, 0 rows affected (0.01 sec)

mysql>
```

图 1.10 创建 MySQL 数据表



说明：按下<Enter>键即可换行，结尾分号“;”表示该行语句结束。

## 1.4.2 查看表结构

成功创建数据表后，可以使用 `show columns` 命令或 `describe` 命令查看指定数据表的表结构。下面分别对这两个语句进行介绍。

### 1.4.2.1 show columns 命令

`show columns` 命令的语法格式如下：

```
show [full] columns from 数据表名 [from 数据库名];
```

或写成：

```
show [full] columns FROM 数据库名.数据表名;
```

例如，应用 `show columns` 命令查看数据表 `tb_users` 表结构，如图 1.11 所示。

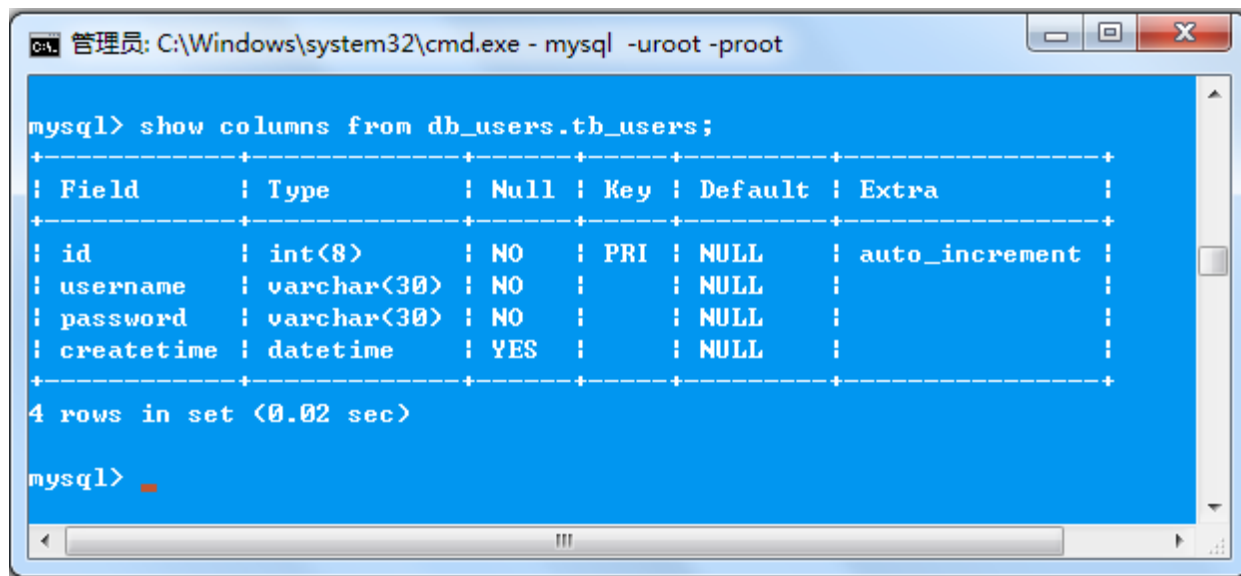


图 1.11 查看 tb\_users 表结构

#### 1.4.2.2 describe 命令

describe 命令的语法格式如下：

describe 数据表名；

其中，describe 可以简写为 desc。在查看表结构时，也可以只列出某一列的信息，语法格式如下：

describe 数据表名 列名；

例如，应用 describe 命令的简写形式查看数据表 tb\_users 的某一列信息，如图 1.12 所示。

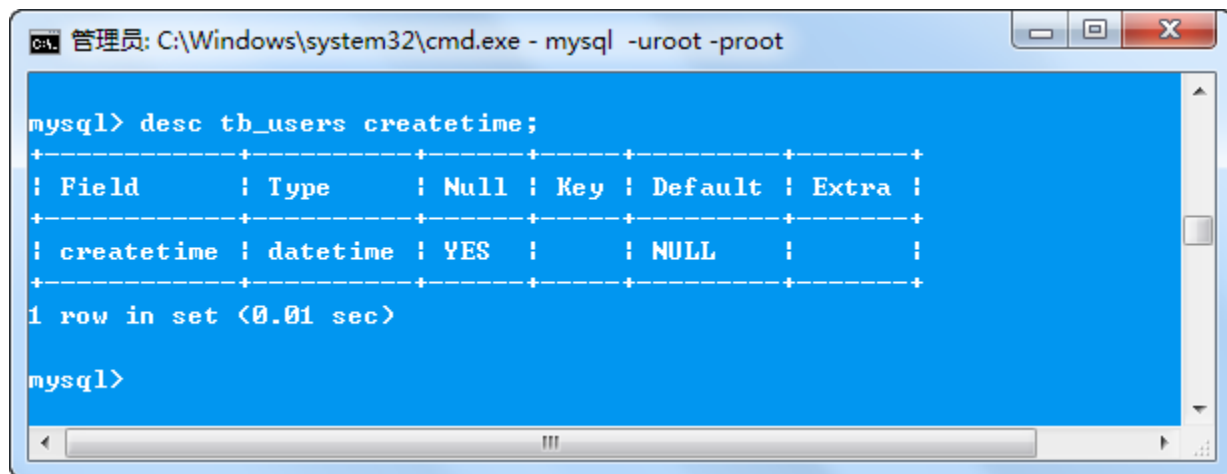


图 1.12 查看 tb\_users 表 createtime 列的信息

### 1.4.3 修改表结构

修改表结构采用 `alter table` 命令。修改表结构指增加或者删除字段、修改字段名称或者字段类型、设置取消主键外键、设置取消索引以及修改表的注释等。

语法：

```
alter [IGNORE] table 数据表名 alter_spec[,alter_spec]...
```

注意，当指定 `IGNORE` 时，如果出现重复关键的行，则只执行一行，其他重复的行被删除。其中，`alter_spec` 子句用于定义要修改的内容，语法如下：

```
alter_specification:
    ADD [COLUMN] create_definition [FIRST | AFTER column_name ]  --添加新字段
    | ADD INDEX [index_name] (index_col_name,...)                --添加索引名称
    | ADD PRIMARY KEY (index_col_name,...)                       --添加主键名称
    | ADD UNIQUE [index_name] (index_col_name,...)               --添加唯一索引
    | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT} --修改字段名称
    | CHANGE [COLUMN] old_col_name create_definition             --修改字段类型
    | MODIFY [COLUMN] create_definition                          --修改子句定义字段
    | DROP [COLUMN] col_name                                     --删除字段名称
    | DROP PRIMARY KEY                                           --删除主键名称
    | DROP INDEX index_name                                       --删除索引名称
    | RENAME [AS] new_tbl_name                                    --更改表名
    | table_options
```

`alter table` 语句允许指定多个动作，动作间使用逗号分隔，每个动作表示对表的一个修改。

例如，向 `tb_users` 表中添加一个新的字段 `address`，类型为 `varchar(60)`，并且不为空值 “not null”，将字段 `user` 的类型由 `varchar(30)` 改为 `varchar(50)`，然后再用 `show colume` 命令查看修改后的表结构，如图 1.13 所示。

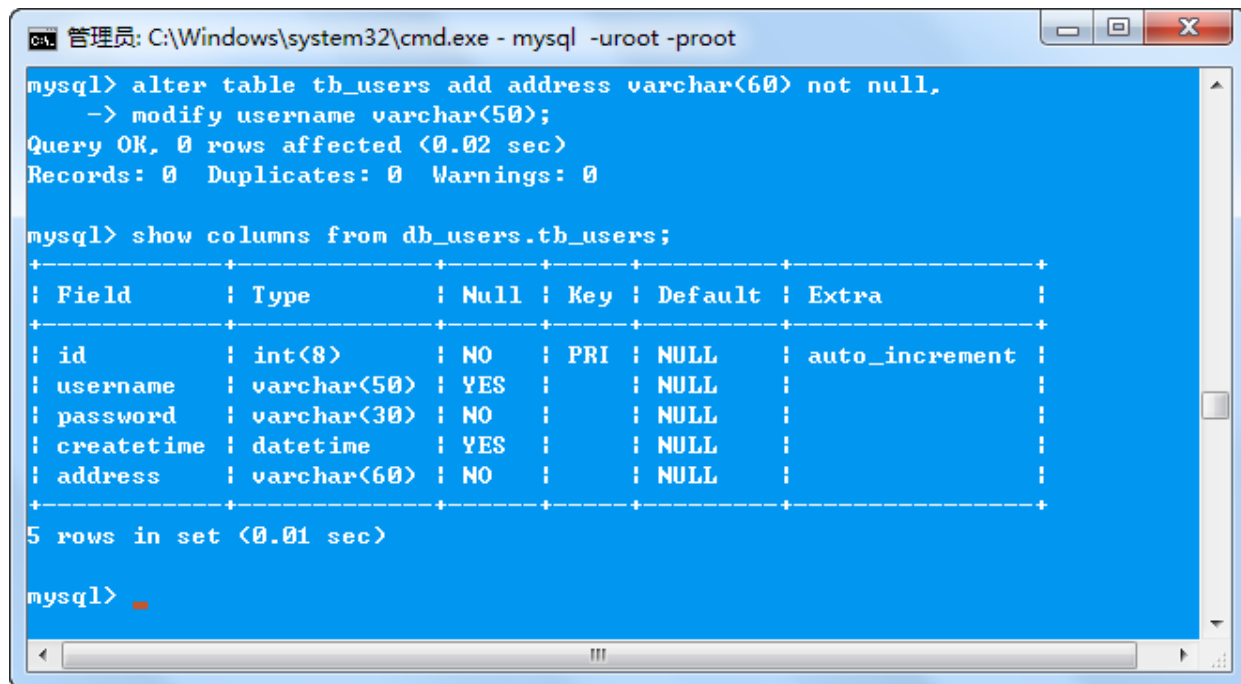


图 1.13 修改 tb\_users 表结构

#### 1.4.4 删除数据表

删除数据表的操作很简单，与删除数据库的操作类似，使用 `drop table` 命令即可实现。格式如下：

`drop table 数据表名;`

例如，在 MySQL 命令窗口中使用“`drop table tb_member;`”SQL 语句即可删除 `tb_member` 数据表。删除数据表后，MySQL 管理系统会自动删除“`D:\phpStudy\MySQL\data\db_member`”目录下的表文件。



**注意：**删除数据表的操作应该谨慎使用。一旦删除了数据表，那么表中的数据将会全部清除，没有备份则无法恢复。

在删除数据表的过程中，如果删除一个不存在的表将会产生错误，这时在删除语句中加入 `if exists` 关键字就可避免出错。格式如下：

`drop table if exists 数据表名;`



**注意：**在对数据表进行操作之前，首先必须选择数据库，否则是无法对数据表进行操作的。

例如，先使用 `drop table` 语句删除一个 `tb_user` 表，查看提示信息，然后使用 `drop table if exists` 语句删除 `tb_user` 表。运行结果如图 1.14 所示。

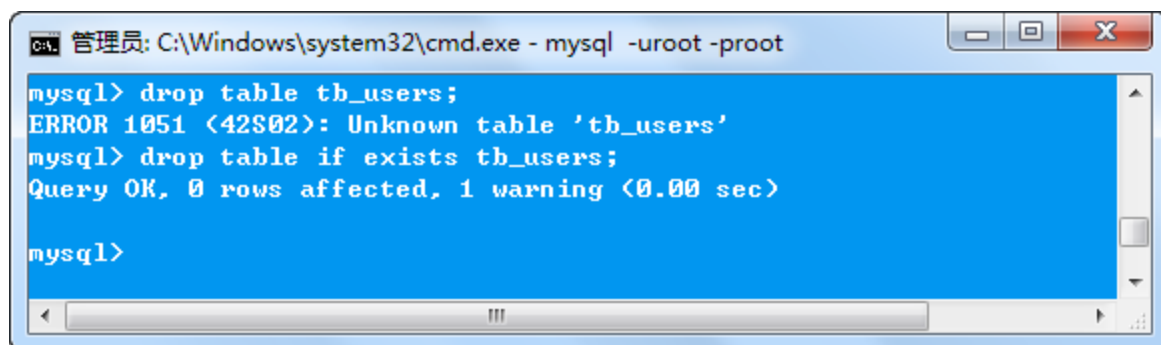


图 1.14 删除 tb\_member 数据表

## 1.5 操作数据表记录

数据库中包含数据表，而数据表中包含数据。更多时候，操作最多的是数据表中的数据，因此如何更好地操作和使用这些数据才是使用 MySQL 数据库的重点。

向数据表中添加、查询、修改和删除记录可以在 MySQL 命令行中使用 SQL 语句完成。下面介绍如何在 MySQL 命令行中执行基本的 SQL 语句。


### 1.5.1 数据表记录的添加

建立一个空的数据库和数据表时，首先要想到的就是如何向数据表中添加数据。这项操作可以通过 insert 命令来实现。

语法格式如下：

```
insert into 数据表名(column_name,column_name2, ... ) values (value1, value2, ... );
```

在 MySQL 中，一次可以同时插入多行记录，各行记录的值清单在 values 关键字后以逗号“,”分隔，而标准的 SQL 语句一次只能插入一行。

 **说明：**值列表中的值应与字段列表中字段的个数和顺序相对应，值列表中值的数据类型必须与相应字段的数据类型保持一致。

例如，向用户信息表 tb\_member 中插入一条数据信息，如图 1.15 所示。

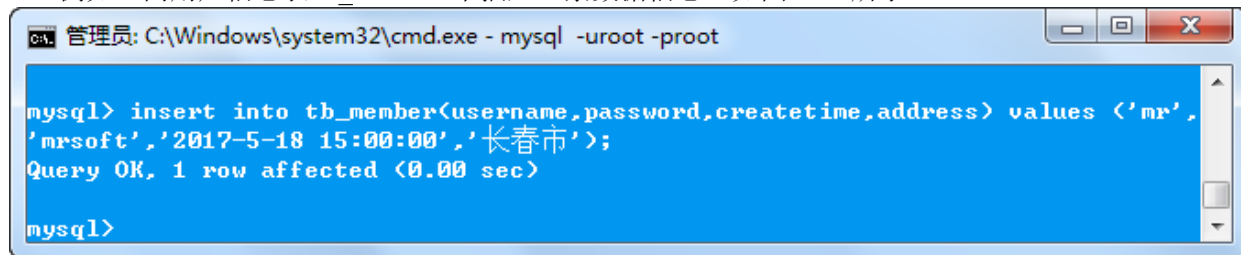


图 1.15 tb\_member 表插入新记录

当向数据表中的所有列添加数据时，insert 语句中的字段列表可以省略，例如，



```
insert into tb_member values('2', '小明', 'xiaoming', '2017-6-20 12:12:12', '长春市');
```

### 1.5.2 数据表记录的查询

数据表中插入数据后，可以使用 `select` 命令来查询数据表中的数据。该语句的格式如下：

```
select selection_list from 数据表名 where condition;
```

其中，`selection_list` 是要查找的列名，如果有要查询多个列，可以用“,” 隔开；如果查询所有列，可以用“\*” 代替。`where` 子句是可选的，如果给出该子句，将查询出指定记录。

例如，查询 `tb_member` 表中数据。运行结果如图 1.16 所示。

```
mysql> select * from tb_member;
+----+-----+-----+-----+-----+
| id | username | password | createtime | address |
+----+-----+-----+-----+-----+
| 2 | 小明 | xiaoming | 2017-06-20 12:12:12 | 长春市 |
| 1 | mr | mrsoft | 2017-05-18 15:00:00 | 长春市 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select id,username from tb_member where id = 1;
+----+-----+
| id | username |
+----+-----+
| 1 | mr |
+----+-----+
1 row in set (0.00 sec)

mysql>
```

图 1.16 select 查找数据

### 1.5.3 数据表记录的修改

要执行修改的操作可以使用 `update` 命令，该语句的格式如下：

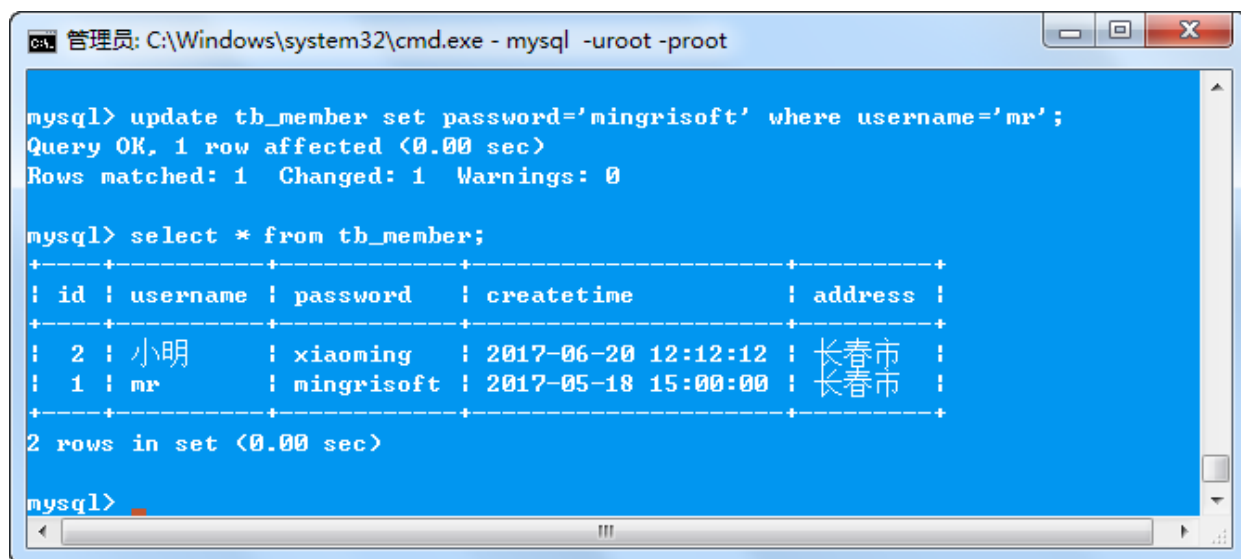
```
update 数据表名 set column_name = new_value1, column_name2 = new_value2, ...where condition;
```

其中，`set` 子句指出要修改的列及其给定的值；`where` 子句是可选的，如果给出该子句将指定记录中哪行应该被更新，否则，所有的记录行都将被更新。

例如，将用户信息表 `tb_member` 中用户名为 `mr` 的管理员密码“`mrsoft`”修改为“`mingrisoft`”，SQL 语句如下：

```
update tb_member set password='mingrisoft' where username='mr';
```

运行结果如图 1.17 所示。



```
管理员: C:\Windows\system32\cmd.exe - mysql -uroot -proot

mysql> update tb_member set password='mingrisoft' where username='mr';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from tb_member;
+----+-----+-----+-----+-----+
| id | username | password | createtime | address |
+----+-----+-----+-----+-----+
| 2 | 小明 | xiaoming | 2017-06-20 12:12:12 | 长春市 |
| 1 | mr | mingrisoft | 2017-05-18 15:00:00 | 长春市 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

图 1.17 更改数据表记录

### 1.5.4 数据表记录的删除

在数据库中有些数据已经失去意义或者是错误的,这时就需要将它们删除,此时可以使用 `delete` 命令。该命令的格式如下:

```
delete from 数据表名 where condition;
```



**注意:** 该语句在执行过程中,如果没有指定 `where` 条件,将删除所有的记录;如果指定了 `where` 条件,将按照指定的条件进行删除。

使用 `delete` 命令删除整个表的效率并不高,还可以使用 `truncate` 命令,利用它可以快速删除表中所有的内容。

例如,删除用户信息表 `tb_users` 中用户名为“mr”的记录信息,SQL 语句如下:

```
delete from tb_member where username = 'mr';
```

删除后,使用 `select` 命令查看结果。运行结果如图 1.18 所示。

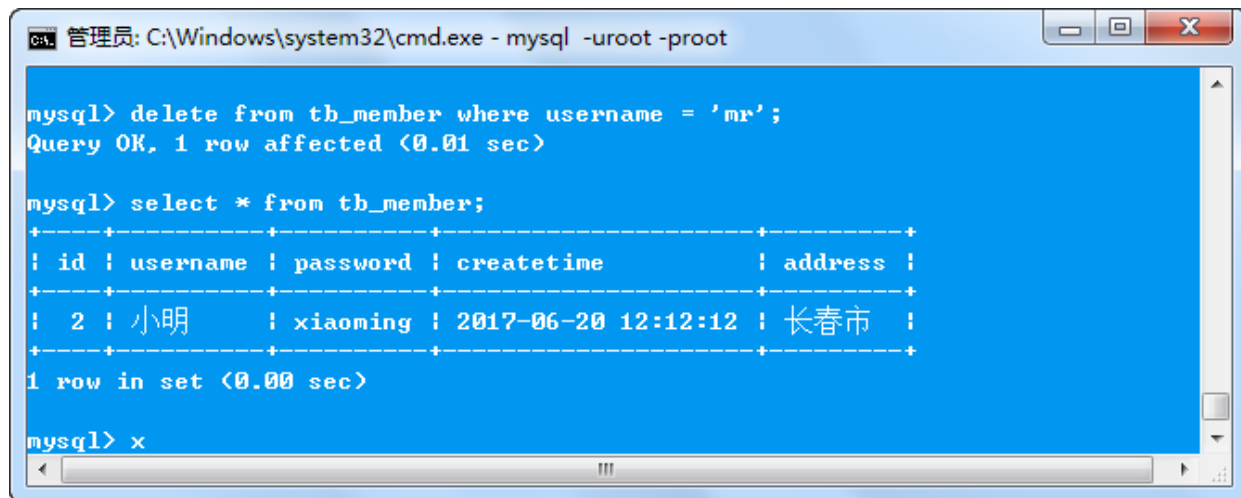


图 1.18 delete 命令删除记录

## 1.6 数据表记录的查询操作

对于数据表的“增删改查”，最常用的就是查询操作。在 1.5.2 节中，我们只是介绍了最基础的查询操作，实际应用中查询的条件要复杂的多。再来看一下复杂一点的 select 语法：

select selection_list	--要查询的内容，选择哪些列
from 数据表名	--指定数据表
where primary_constraint	--查询时需要满足的条件，行必须满足的条件
group by grouping_columns	--如何对结果进行分组
order by sorting_columns	--如何对结果进行排序
having secondary_constraint	--查询时满足的第二条件
limit count	--限定输出的查询结果

下面对它的参数进行详细的讲解。

### 1. selection\_list

设置查询内容。如果要查询表中所有列，可以将其设置为“\*”；如果要查询表中某一列或多列，则直接输入列名，并以“,”为分隔符。例如，查询 tb\_mrbook 数据表中所有列和查询 id 和 bookname 列的代码如下：

```
select * from tb_mrbook;           # 查询数据表中所有数据
select id,bookname from tb_mrbook; # 查询数据表中 id 和 bookname 列的数据
```

## 2. table\_list

指定查询的数据表。即可以从一个数据表中查询，也可以从多个数据表中进行查询，多个数据表之间用“,”进行分隔，并且通过 WHERE 子句使用连接运算来确定表之间的联系。

例如，从 tb\_mrbook 和 tb\_bookinfo 数据表中查询 bookname='PHP 自学视频教程'的 id 编号、书名、作者和价格，其代码如下：

```
select tb_mrbook.id,tb_mrbook.bookname,
       author,price from tb_mrbook,tb_bookinfo
where tb_mrbook.bookname = tb_bookinfo.bookname
and tb_bookinfo.bookname = 'python 自学视频教程';
```

在上面的 SQL 语句中，因为两个表都有 id 字段和 bookname 字段，为了告诉服务器要显示的是哪个表中的字段信息，要加上前缀。语法如下：

表名.字段名

tb\_mrbook.bookname=tb\_bookinfo.bookname 将表 tb\_mrbook 和 tb\_bookinfo 连接起来，叫作等同连接；如果不使用 tb\_mrbook.bookname=tb\_bookinfo.bookname，那么产生的结果将是两个表的笛卡儿积，叫作全连接。



**多学两招：**笛卡尔乘积是指在数学中，两个集合 X 和 Y 的笛卡尔积 (Cartesian product)，又称直积，表示为  $X \times Y$ ，第一个对象是 X 的成员而第二个对象是 Y 的所有可能有序对的其中一个成员。

## 3. where 条件语句

在使用查询语句时，如要从很多的记录中查询出想要的记录，就需要一个查询的条件。只有设定了查询的条件，查询才有实际的意义。设定查询条件应用的是 WHERE 子句。

WHERE 子句的功能非常强大，通过它可以实现很多复杂的条件查询。在使用 WHERE 子句时，需要使用一些比较运算符，常用的比较运算符如表 1.9 所示。

表 1.9 常用的 WHERE 子句比较运算符

字段名	默认值或绑定	默认值或绑定	默认值或绑定	默认值或绑定	描述
=	等于	id=10	is not null	n/a	id is not null
>	大于	id>10	between	n/a	id between 1 and 10
<	小于	id<10	in	n/a	id in (4, 5, 6)
>=	大于等于	id>=10	not in	n/a	name not in (a, b)
<=	小于等于	id<=10	like	模式匹配	name like ( 'abc%' )
!=或<>	不等于	id!=10	not like	模式匹配	name not like ( 'abc%' )

字段名	默认值或绑定	默认值或绑定	默认值或绑定	默认值或绑定	描述
is null	n/a	id is null	regexp	常规表达式	name 正则表达式

表 1.9 中列举的是 WHERE 子句常用的比较运算符，示例中的 id 是记录的编号，name 是表中的用户名。

例如，应用 WHERE 子句，查询 tb\_mrbook 表，条件是 type（类别）为 PHP 的所有图书，代码如下：

```
select * from tb_mrbook where type = 'PHP';
```

#### 4. DISTINCT 在结果中去除重复行

使用 DISTINCT 关键字，可以去除结果中重复的行。

例如，查询 tb\_mrbook 表，并在结果中去掉类型字段 type 中的重复数据，代码如下：

```
select distinct type from tb_mrbook;
```

#### 5. ORDER BY 对结果排序

使用 ORDER BY 可以对查询的结果进行升序和降序（DESC）排列，在默认情况下，ORDER BY 按升序输出结果。如果要按降序排列可以使用 DESC 来实现。

对含有 NULL 值的列进行排序时，如果是按升序排列，NULL 值将出现在最前面，如果是按降序排列，NULL 值将出现在最后。例如，查询 tb\_mrbook 表中的所有信息，按照“id”进行降序排列，并且只显示五条记录。其代码如下：

```
select * from tb_mrbook order by id desc limit 5;
```

#### 6. LIKE 模糊查询

LIKE 属于较常用的比较运算符，通过它可以实现模糊查询。它有两种通配符：“%”和下划线“\_”。“%”可以匹配一个或多个字符，而“\_”只匹配一个字符。例如，查找所有书名（bookname 字段）包含“PHP”的图书，代码如下：

```
select * from tb_mrbook where bookname like('%PHP%');
```



**说明：**无论是一个英文字符还是中文字符都算做一个字符，在这一点上英文字母和中文没有什么区别。

#### 7. CONCAT 联合多列

使用 CONCAT 函数可以联合多个字段，构成一个总的字符串。例如，把 tb\_mrbook 表中的书名（bookname）和价格（price）合并到一起，构成一个新的字符串。代码如下：

```
select id,concat(bookname,":",price) as info,type from tb_mrbook;
```

其中，合并后的字段名为 CONCAT 函数形成的表达式“bookname:price”，看上去十分复杂，通过 AS 关键字给合并字段取一个别名，这样看上去就更加清晰了。

## 8. LIMIT 限定结果行数

LIMIT 子句可以对查询结果的记录条数进行限定，控制它输出的行数。例如，查询 tb\_mrbook 表，按照图书价格升序排列，显示十条记录，代码如下：

```
select * from tb_mrbook order by price asc limit 10;
```

使用 LIMIT 还可以从查询结果的中间部分取值。首先要定义两个参数，参数 1 是开始读取的第一条记录的编号（在查询结果中，第一个结果的记录编号是 0，而不是 1）；参数 2 是要查询记录的个数。

例如，查询 tb\_mrbook 表，从第 3 条记录开始，查询 6 条记录，代码如下：

```
select * from tb_mrbook limit 2,6;
```

## 9. 使用函数和表达式

在 MySQL 中，还可以使用表达式来计算各列的值，作为输出结果。表达式还可以包含一些函数。例如，计算 tb\_mrbook 表中各类图书的总价格，代码如下：

```
select sum(price) as totalprice,type from tb_mrbook group by type;
```

在对 MySQL 数据库进行操作时，有时需要对数据库中的记录进行统计，例如求平均值、最小值、最大值等，这时可以使用 MySQL 中的统计函数，其常用的统计函数如表 1.10 所示。

表 1.10 MySQL 中常用的统计函数

名称	说明
avg（字段名）	获取指定列的平均值
count（字段名）	如指定了一个字段，则会统计出该字段中的非空记录。如在前面增加 DISTINCT，则会统计不同值的记录，相同的值当作一条记录。如使用 COUNT（*）则统计包含空值的所有记录数
min（字段名）	获取指定字段的最小值
max（字段名）	获取指定字段的最大值
std（字段名）	指定字段的标准背离值
stddev（字段名）	与 STD 相同
sum（字段名）	获取指定字段所有记录的总和

除了使用函数之外，还可以使用算术运算符、字符串运算符，以及逻辑运算符来构成表达式。例如，可以计算图书打九折之后的价格，代码如下：

```
select *, (price * 0.9) as '90%' from tb_mrbook;
```

## 10. GROUP BY 对结果分组

通过 GROUP BY 子句可以将数据划分到不同的组中，实现对记录进行分组查询。在查询时，所查询的列必须包含在分组的列中，目的是使查询到的数据没有矛盾。在与 AVG()函数或 SUM()函数一起使用时，GROUP BY 子句能发挥最大作用。例如，查询 tb\_mrbook 表，按照 type 进行分组，求每类图

书的平均价格，代码如下：

```
select avg(price),type from tb_mrbook group by type;
```

### 11. 使用 having 子句设定第二个查询条件

having 子句通常和 group by 子句一起使用。在对数据结果进行分组查询和统计之后，还可以使用 having 子句来对查询的结果进行进一步的筛选。having 子句和 where 子句都用于指定查询条件，不同的是 where 子句在分组查询之前应用，而 having 子句在分组查询之后应用，而且 having 子句中还可以包含统计函数。例如，计算 tb\_mrbook 表中各类图书的平均价格，并筛选出图书的平均价格大于 60 的记录，代码如下：

```
select avg(price),type from tb_mrbook group by type having avg(price)>60;
```