第 2 章

◂ MySQL的安装与配置 ▸


学习目标 Objective

MySQL 支持多种平台，不同平台下的安装与配置过程也不相同。在 Windows 平台下可以使用二进制的安装软件包或免安装版的软件包进行安装，二进制的安装包提供了图形化的安装向导过程，而免安装版直接解压缩即可使用。本章将主要讲述在 Windows 平台下 MySQL 的安装和配置过程。


内容导航 Navigation

- 掌握如何在 Windows 平台下安装和配置 MySQL 8.0
- 掌握启动服务并登录 MySQL 数据库
- 掌握 MySQL 的两种配置方法
- 熟悉 MySQL 常用图形管理工具
- 掌握常见的 MySQL 工具

2.1 在 Windows 平台下安装与配置 MySQL 8.0

在 Windows 平台下安装 MySQL 可以使用图行化的安装包，图形化的安装包提供了详细的安装向导，通过向导，读者可以一步一步地完成对 MySQL 的安装。本节将介绍使用图形化安装包安装 MySQL 的步骤。

2.1.1 安装 MySQL 8.0

要想在 Windows 中运行 MySQL，需要 32 位或 64 位 Windows 操作系统，例如 Windows 7、Windows 8、Windows 10、Windows Server 2012 等。Windows 可以将 MySQL 服务器作为服务来运行，通常在安装时需要具有系统的管理员权限。

在 Windows 平台下提供两种安装方式：MySQL 二进制分发版（.msi 安装文件）和免安

装版（.zip 压缩文件）。一般来讲，应当使用二进制分发版，因为该版本比其他的分发版使用起来要简单，不再需要其他工具来启动就可以运行 MySQL。

1. 下载 MySQL 安装文件

下载 MySQL 安装文件的具体操作步骤如下。

步骤 01　打开 IE 浏览器，在地址栏中输入网址：https://dev.mysql.com/downloads/installer/，单击【转到】按钮，打开 MySQL Community Server 8.0.13 下载页面，选择 Microsoft Windows 平台，然后根据读者的操作系统选择 32 位或者 64 位安装包，在这里选择 32 位，单击右侧的【Download】按钮开始下载，如图 2.1 所示。

步骤 02　在弹出的页面中提示开始下载，这里单击【Login】按钮，如图 2.2 所示。

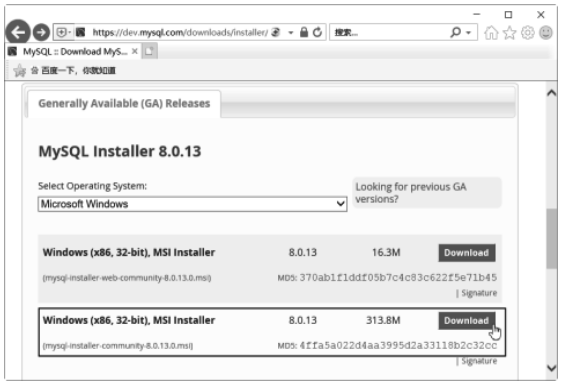

图 2.1　MySQL 下载页面

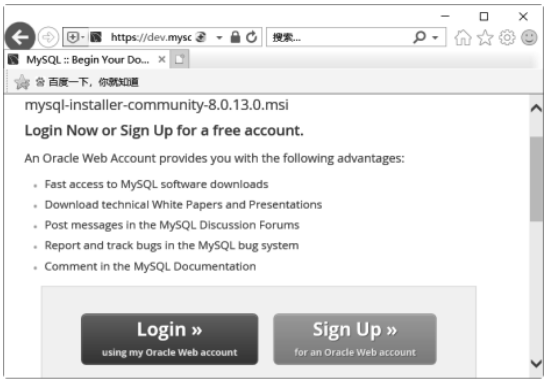

图 2.2　开始下载页面

提　示
这里有 32 位的安装程序有两个版本，分别为 mysql-installer-web-community 和 mysql-installer-community，其中 mysql-installer-web-community 为在线安装版本，mysql-installer-community 为离线安装版本。

步骤 03　弹出用户登录页面，输入用户名和密码后，单击【登录】按钮，如图 2.3 所示。

步骤 04　弹出开始下载页面，单击【Download Now】按钮，即可开始下载，如图 2.4 所示。

图 2.3　用户登录页面

图 2.4　开始下载页面

提　示
如果用户没有用户名和密码，那么可以单击【创建账户】链接进行注册。

2. 安装 MySQL 8.0

MySQL 下载完成后，找到下载文件，双击进行安装，具体操作步骤如下。

步骤 01　双击下载的 mysql-installer-community-8.0.13.0.msi 文件，如图 2.5 所示。

图 2.5　MySQL 安装文件名称

步骤 02　打开【License Agreement】（用户许可证协议）窗口，选中【I accept the license terms】（我接受许可协议）复选框，单击【Next】（下一步）按钮，如图 2.6 所示。

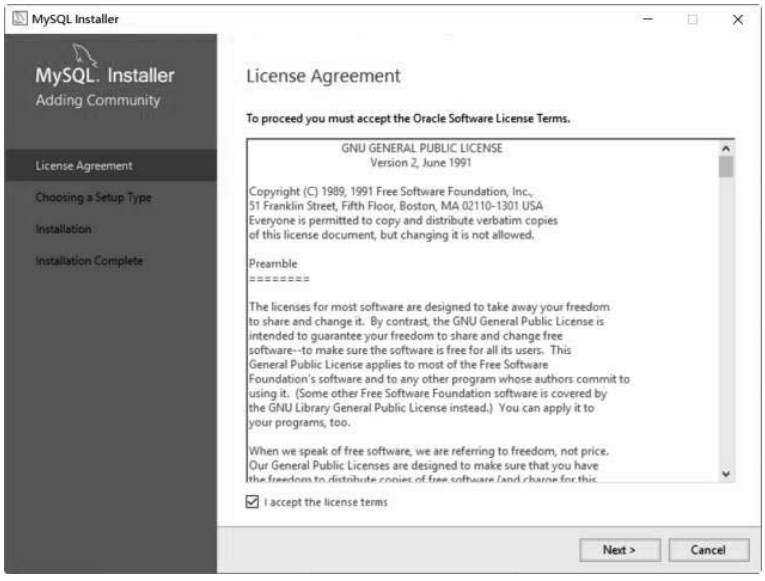

图 2.6　用户许可证协议窗口

步骤 03　打开【Choosing a Setup Type】（安装类型选择）窗口，在其中列出了 5 种安装类型，分别是：Developer Default（默认安装类型）、Server only（仅作为服务器）、Client only（仅作为客户端）、Full（完全安装）和 Custom（自定义安装类型）。这里选择【Custom】（自定义安装类型）单选按钮，单击【Next】（下一步）按钮，如图 2.7 所示。

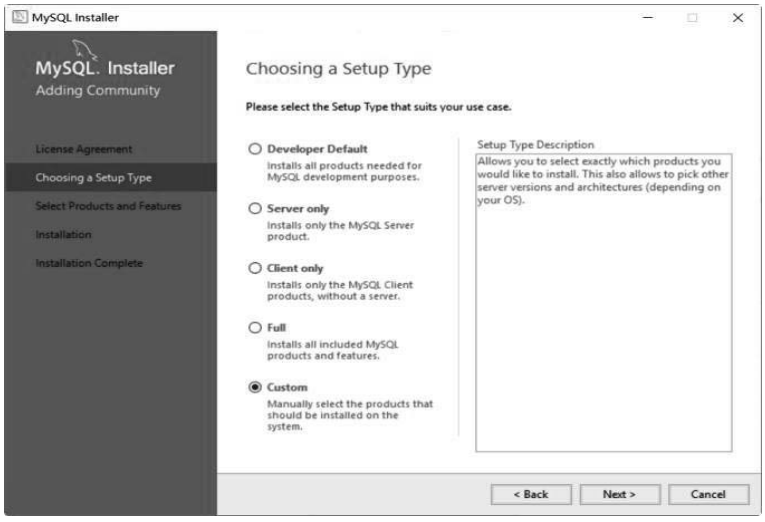

图 2.7　安装类型窗口

步骤 04　打开【Select Products and Features】（产品定制选择）窗口，选择【MySQL Server 8.0.13-x86】后，单击按钮，即可选择安装 MySQL 服务器。采用同样的方法，添加【Samples and Examples 8.0.13-x86】和【MySQL Documentation 8.0.13-x86】选项，如图 2.8 所示。

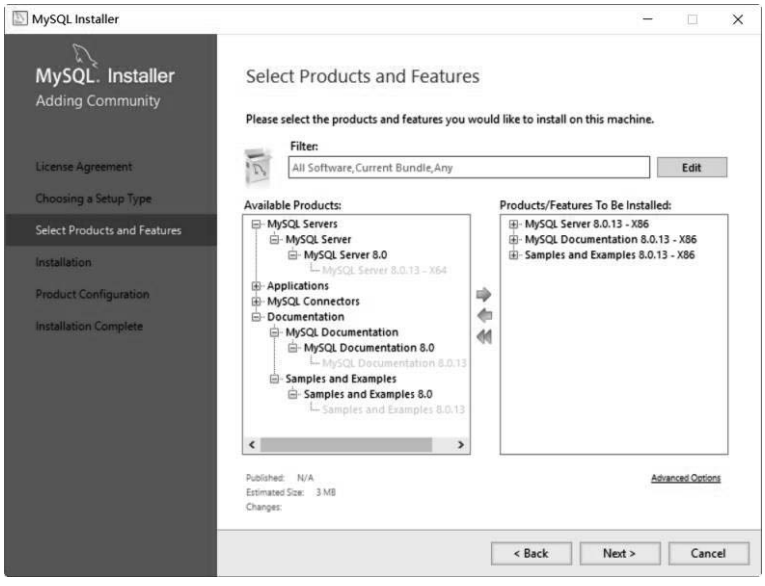

图 2.8　自定义安装组件窗口

步骤 05　单击【Next】（下一步）按钮，进入安装确认对话窗口，单击【Execute】（执行）按钮，如图 2.9 所示。

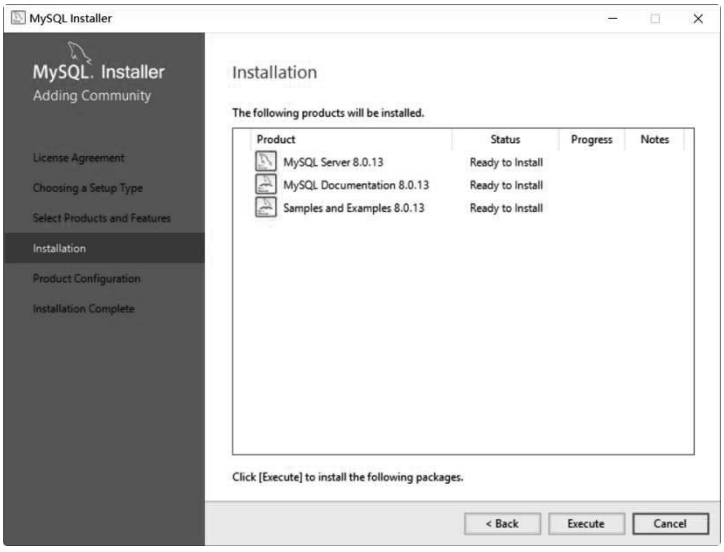

图 2.9　准备安装对话窗口

步骤 06　开始安装 MySQL 文件，安装完成后，在【Status】（状态）列表下将显示 Complete（安装完成），如图 2.10 所示。

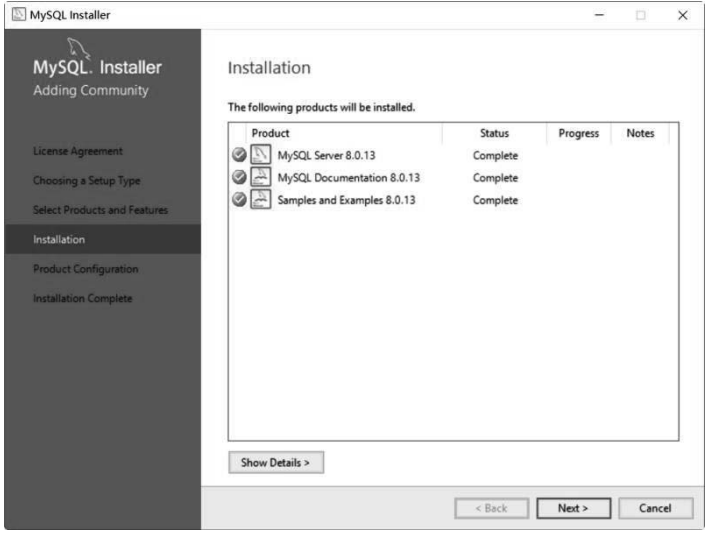

图 2.10　安装完成窗口

2.1.2　配置 MySQL 8.0

MySQL 安装完毕之后，需要对服务器进行配置，具体的配置步骤如下。

步骤 01　在 2.1.1 小节的最后一步中，单击【Next】（下一步）按钮，进入产品信息窗口，如图 2.11 所示。

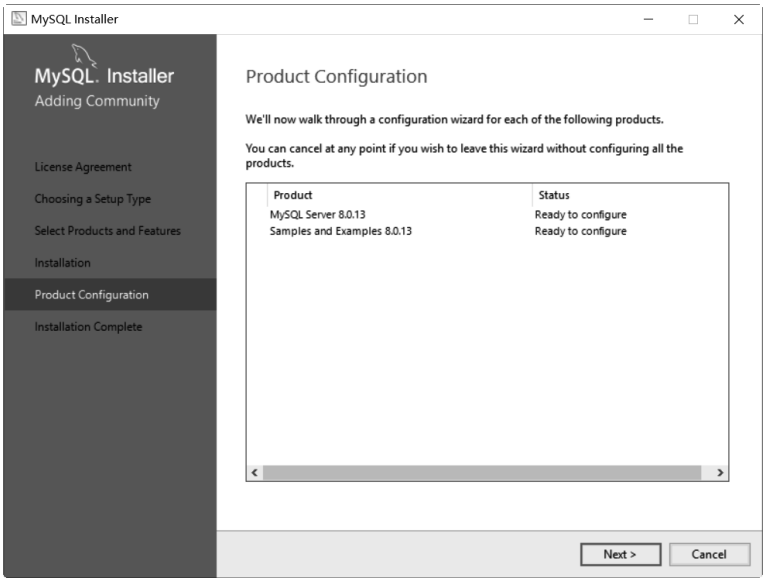

图 2.11　产品信息窗口

步骤 02　单击【Next】（下一步）按钮，进入服务器配置窗口，如图 2.12 所示。

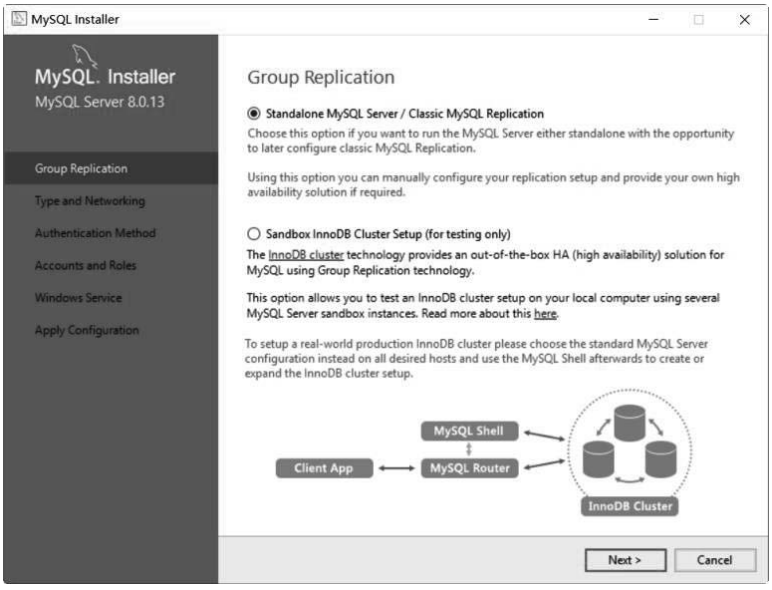

图 2.12　服务器配置窗口

步骤 03　单击【Next】（下一步）按钮，进入 MySQL 服务器配置窗口，采用默认设置，如图 2.13 所示。

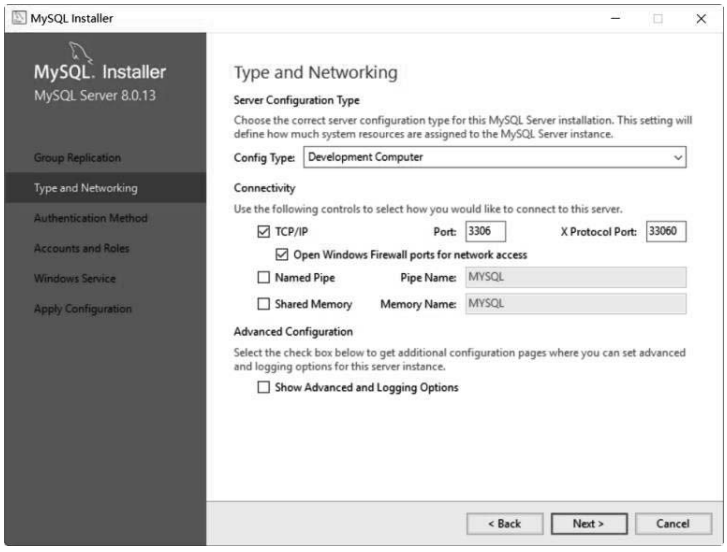

图 2.13　MySQL 服务器配置窗口

MySQL 服务器配置窗口中各个参数的含义如下。

【Server Configuration Type】：该选项用于设置服务器的类型。单击该选项右侧的向下按钮，即可看到包括 3 个选项，如图 2.14 所示。

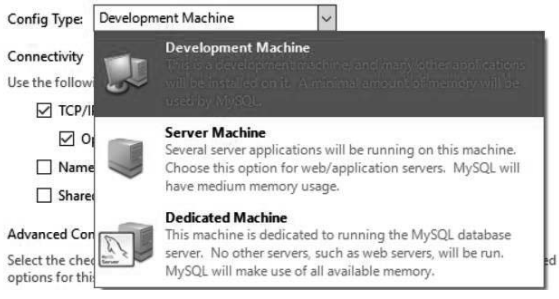

图 2.14　MySQL 服务器的类型

上图中 3 个选项的具体含义如下。

（1）Development Machine（开发机器）：该选项代表典型个人用桌面工作站。假定机器上运行着多个桌面应用程序，将 MySQL 服务器配置成使用最少的系统资源。

（2）Server Machine（服务器）：该选项代表服务器，MySQL 服务器可以同其他应用程序一起运行，例如 FTP、Email 和 Web 服务器。将 MySQL 服务器配置成使用适当比例的系统资源。

（3）Dedicated Machine（专用服务器）：该选项代表只运行 MySQL 服务的服务器。假定没有运行其他服务程序，将 MySQL 服务器配置成使用所有可用系统资源。

提　示
作为初学者，建议选择【Development Machine】（开发者机器）选项，这样占用的系统资源比较少。

步骤 04　单击【Next】（下一步）按钮，打开设置授权方式窗口。其中第一个单选项的含义：MySQL 8.0 提供的新的授权方式，采用 SHA256 基础的密码加密方法；第二个单选项的含义：传统授权方法（保留 5.x 版本兼容性）。这里选择第二个单选项，如图 2.15 所示。

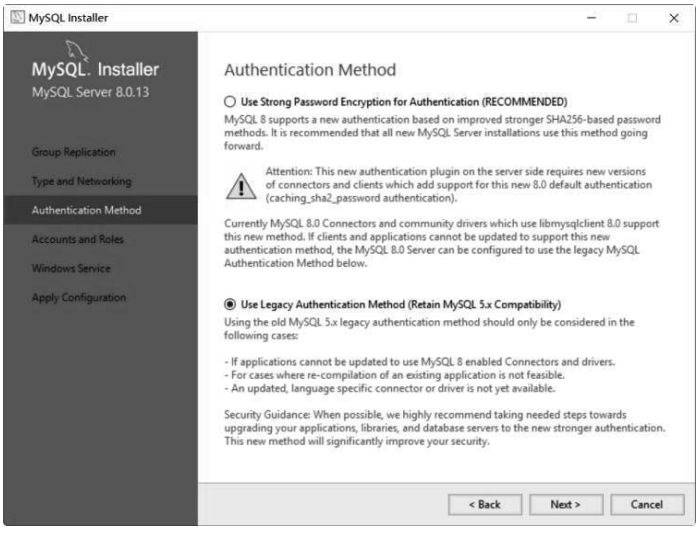

图 2.15　MySQL 服务器的类型

步骤 05　单击【Next】（下一步）按钮，打开设置服务器的密码窗口，重复输入两次同样的登录密码，如图 2.16 所示。

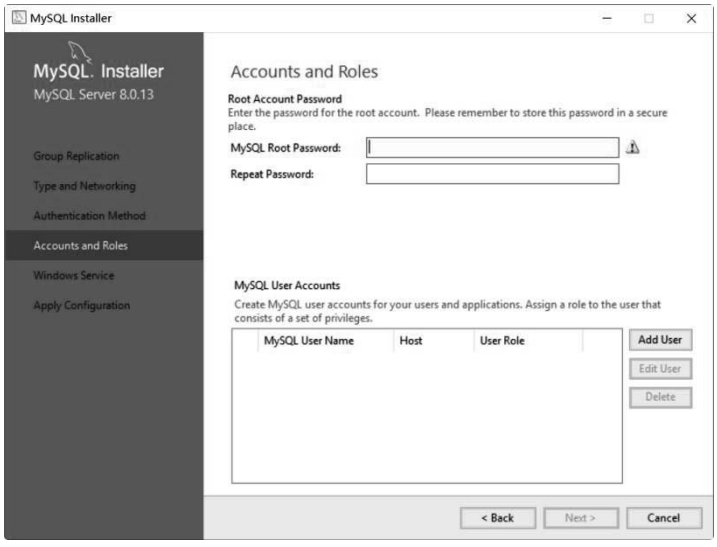

图 2.16　设置服务器的登录密码

提　示
系统默认的用户名称为 root，如果想添加新用户，可以单击【Add User】（添加用户）按钮进行添加。

步骤 06　单击【Next】（下一步）按钮，打开设置服务器名称窗口，本案例设置服务器名称为 MySQL，如图 2.17 所示。

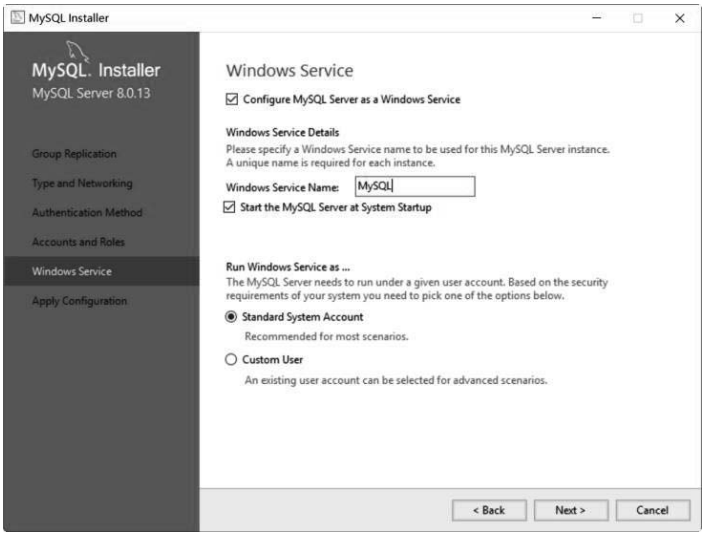

图 2.17　设置服务器的名称

步骤 07　单击【Next】（下一步）按钮，打开确认设置服务器窗口，单击【Execute】（执行）按钮，如图 2.18 所示。

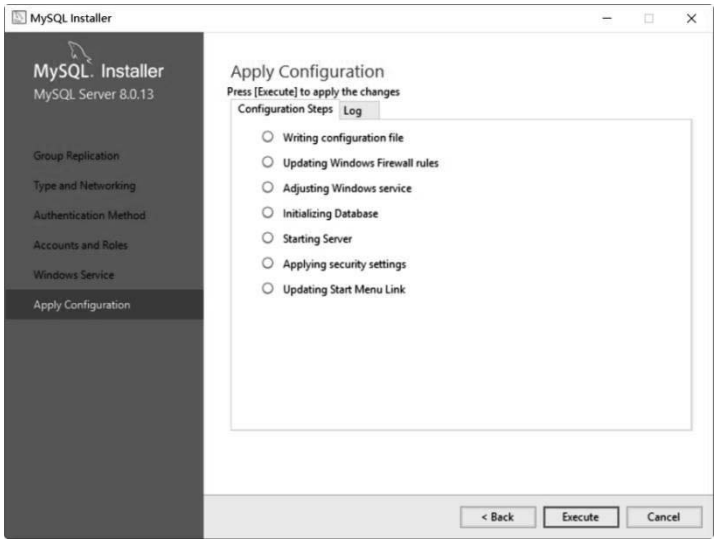

图 2.18　确认设置服务器

系统自动配置 MySQL 服务器。配置完成后，单击【Finish】（完成）按钮，即可完成服务器的配置，如图 2.19 所示。

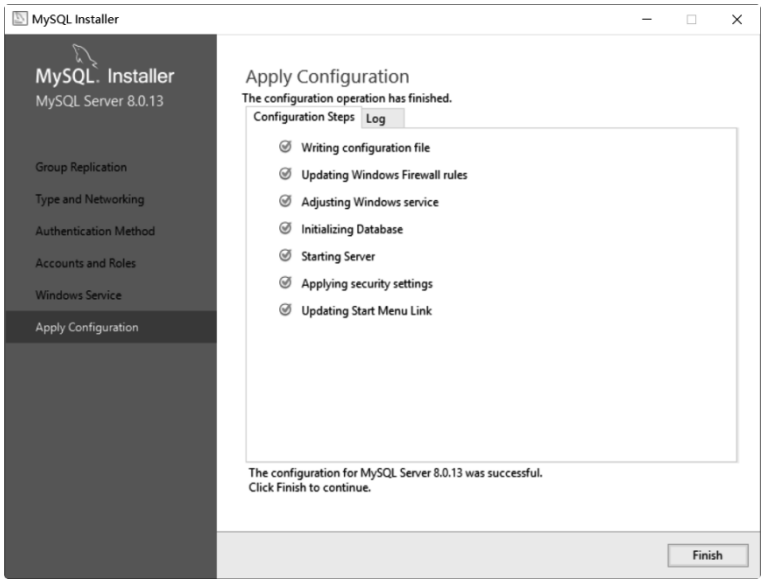

图 2.19　完成设置服务器

步骤 07　按键盘上的【Ctrl+Alt+Del】组合键，打开【任务管理器】对话框，可以看到 MySQL 服务进程 mysqld.exe 已经启动了，如图 2.20 所示。

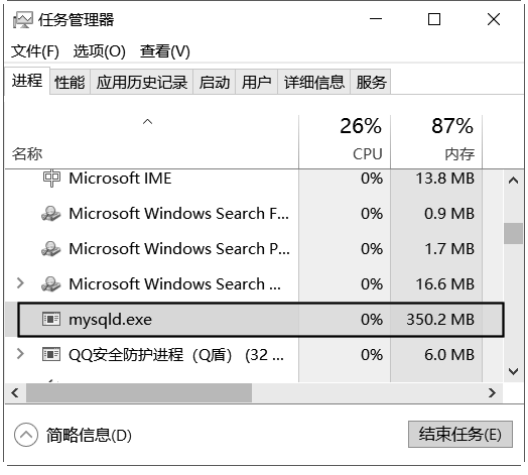

图 2.20　任务管理器窗口

至此，就完成了在 Windows 10 操作系统环境下安装 MySQL 的操作。

2.2 启动服务并登录 MySQL 数据库

MySQL 安装完毕之后，需要启动服务器进程，不然客户端无法连接数据库，客户端通过命令行工具登录数据库。本节将介绍如何启动 MySQL 服务器和登录 MySQL 的方法。

2.2.1 启动 MySQL 服务

在前面的配置过程中，已经将 MySQL 安装为 Windows 服务，当 Windows 启动、停止时，MySQL 也将自动启动、停止。不过，用户还可以使用图形服务工具来控制 MySQL 服务器或从命令行使用 NET 命令。

可以通过 Windows 的服务管理器查看，具体的操作步骤如下。

步骤 01 单击【开始】菜单，在搜索框中输入“services.msc”，按【Enter】键确认，如图 2.21 所示。

步骤 02 打开 Windows 的【服务管理器】，在其中可以看到服务名为“MySQL”的服务项，其右边状态为“已启动”，表明该服务已经启动，如图 2.22 所示。

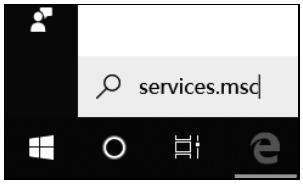

图 2.21 搜索框

名称	描述	状态	启动类型	登录为
MySQL		已启动	自动	网络服务
Net.Msmq Liste...	Rece...		禁用	网络服务
Net.Pipe Listene...	Rece...		自动	本地服务
Net.Tcp Listener...	Rece...		自动	本地服务
Net.Tcp Port Sh...	Prov...		手动	本地服务
Netlogon	为用...		手动	本地系统
Network Access ...	网络...		手动	网络服务
Network Connec...	管理...	已启动	手动	本地系统

图 2.22 服务管理器窗口

由于设置了 MySQL 为自动启动，因此在这里可以看到服务已经启动，而且启动类型为自动。如果没有“已启动”字样，就说明 MySQL 服务未启动。启动方法为：单击【开始】菜单，在搜索框中输入“cmd”，按【Enter】键确认；弹出命令提示符界面。然后输入“net start MySQL”，按回车键，就能启动 MySQL 服务了。停止 MySQL 服务的命令为：“net stop MySQL”，如图 2.23 所示。

也可以直接双击 MySQL 服务，打开【MySQL 的属性】对话框，在其中通过单击【启动】或【停止】按钮来更改服务状态，如图 2.24 所示。

提　示
输入的 MySQL 是服务的名字。如果读者的 MySQL 服务的名字是 DB 或其他名字，那么应该输入“net start DB”或其他名称。

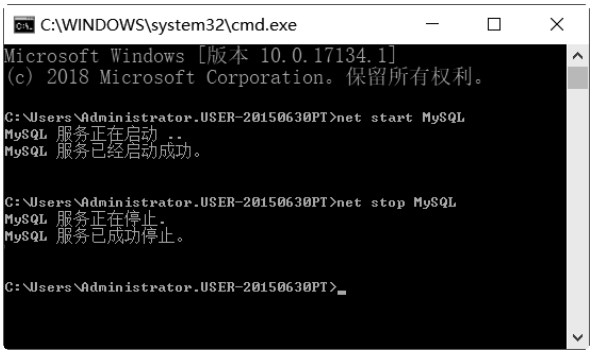

图 2.23　在命令行中启动和停止 MySQL

图 2.24　【MySQL 的属性】对话框

2.2.2　登录 MySQL 数据库

当 MySQL 服务启动后，便可以通过客户端来登录 MySQL 数据库。在 Windows 操作系统下，可以通过两种方式登录 MySQL 数据库。

1. 以 Windows 命令行方式登录

具体的操作步骤如下。

步骤 01　单击【开始】菜单，在搜索框中输入“cmd”，按【Enter】键确认，如图 2.25 所示。

步骤 02　打开 DOS 窗口，输入以下命令并按【Enter】键确认，如图 2.26 所示。

```
cd C:\Program Files\MySQL\MySQL Server 8.0\bin\
```

图 2.25　搜索框

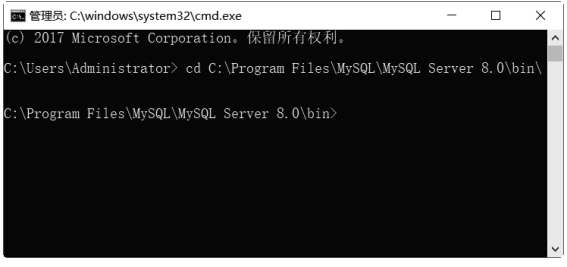

图 2.26　DOS 窗口

步骤 03 在 DOS 窗口中可以通过登录命令连接到 MySQL 数据库，连接 MySQL 的命令格式为：

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

其中，mysql 为登录命令，–h 后面的参数是服务器的主机地址，这里客户端和服务器在同一台机器上，所以输入“localhost”或者 IP 地址“127.0.0.1”；-u 后面跟登录数据库的用户名称，这里为“root”；-p 后面是用户登录密码。

接下来，输入如下命令：

```
mysql  -h localhost -u root -p
```

按【Enter】键，系统会提示输入密码“Enter password”，这里输入在前面配置向导中自己设置的密码，验证正确后，即可登录 MySQL 数据库，如图 2.27 所示。

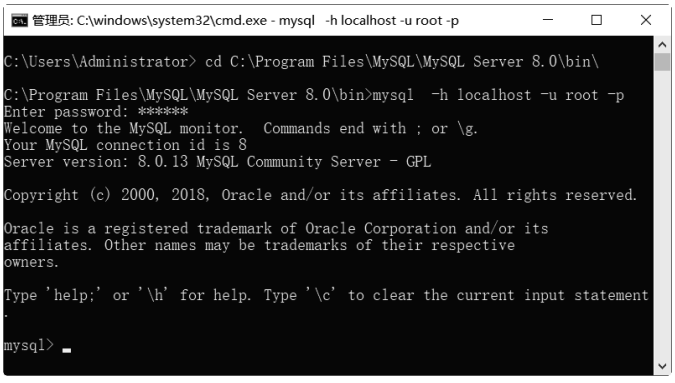

图 2.27　在 Windows 命令行登录窗口

提　示
当窗口中出现如图 2.27 所示的说明信息，命令提示符变为“mysql>”时，表明已经成功登录 MySQL 服务器了，可以开始对数据库进行操作。

2. 使用 MySQL Command Line Client 登录

依次选择【开始】|【所有程序】|【MySQL】|【MySQL 8.0 Command Line Client】菜单命令，进入密码输入窗口，如图 2.28 所示。

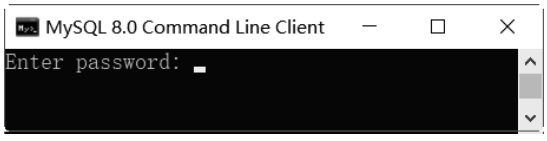

图 2.28　MySQL 命令行登录窗口

输入正确的密码之后，就可以登录 MySQL 数据库了。

2.2.3　配置 Path 变量

在前面登录 MySQL 服务器的时候，不能直接输入 MySQL 登录命令，是因为没有把 MySQL 的 bin 目录添加到系统的环境变量里面，所以不能直接使用 MySQL 命令。如果每次登录都输入“cd C:\Program Files\MySQL\MySQL Server 8.0\bin”，才能使用 MySQL 等其他命

令工具，这样比较麻烦。

下面介绍怎样手动配置 PATH 变量，具体的操作步骤如下。

步骤 01　在桌面上右击【此电脑】图标，在弹出的快捷菜单中选择【属性】菜单命令，如图 2.29 所示。

步骤 02　打开【系统】窗口，单击【高级系统设置】链接，如图 2.30 所示。

图 2.29　此电脑属性菜单

图 2.30　【系统属性】窗口

步骤 03　打开【系统属性】对话框，选择【高级】选项卡，然后单击【环境变量】按钮，如图 2.31 所示。

步骤 04　打开【环境变量】对话框，在系统变量列表中选择【Path】变量，如图 2.32 所示。

图 2.31　【系统属性】对话框

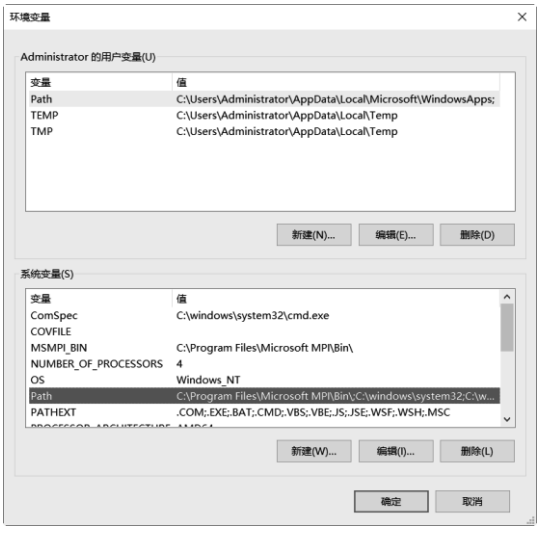

图 2.32　【环境变量】对话框

步骤 05　单击【编辑】按钮，在【编辑环境变量】对话框中，将 MySQL 应用程序的 bin 目录（C:\Program Files\MySQL\MySQL Server 8.0\bin）添加到变量值中，用分号将其与其他路径分隔开，如图 2.33 所示。

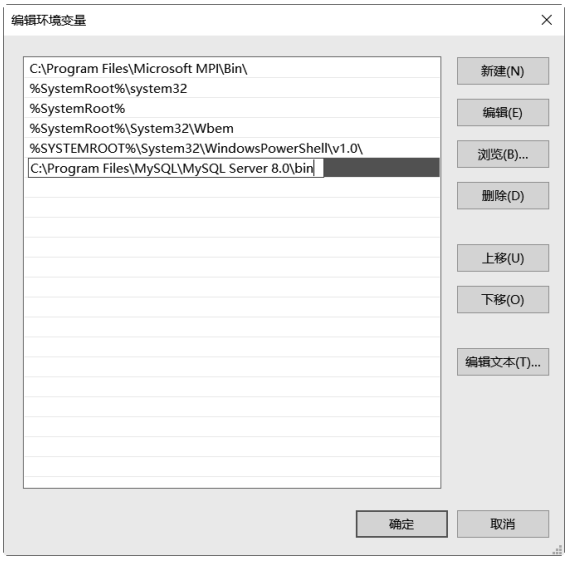

图 2.33 【编辑系统变量】对话框

步骤 06 添加完成之后，单击【确定】按钮，这样就完成了配置 PATH 变量的操作，然后就可以直接输入 MySQL 命令来登录数据库了。

2.3 小白疑难解惑

计算机技术具有很强的操作性，MySQL 的安装和配置是一件非常简单的事，但是在操作过程中也可能出现问题，读者需要多实践、多总结。

疑问 1：无法打开 MySQL 8.0 软件安装包，如何解决？

在安装 MySQL 8.0 软件安装包之前，用户需要确保系统中已经安装了.Net Framework 3.5 和.Net Framework 4.0，如果缺少这两个软件，就不能正常地安装 MySQL 8.0 软件。另外，还要确保 Windows Installer 正常安装。

疑问 2：MySQL 安装失败，如何解决？

安装过程失败多是由于重新安装 MySQL 的缘故，因为 MySQL 在删除的时候不能自动删除相关的信息。解决方法是，把以前安装的目录删除掉。删除在 C 盘的 Program File 文件夹里面 MySQL 的安装目录文件夹；同时删除 MySQL 的 DATA 目录，该目录一般为隐藏文件，其位置一般在“C:\Documents and Settings\All Users\Application Data\ MySQL”目录下，删除后重新安装即可。

2.4 习题演练

（1）下载并安装 MySQL。

（2）使用配置向导配置 MySQL 为系统服务，在【系统服务】对话框中，手动启动或者关闭 MySQL 服务。

（3）使用 net 命令启动或者关闭 MySQL 服务。

（4）使用免安装的软件包安装 MySQL。

第 3 章

◄ 操作数据库和数据表 ►

学习目标 Objective

MySQL 安装好以后，首先需要创建数据库，这是使用 MySQL 各种功能的前提。在数据库中，数据表是数据库中重要且基本的操作对象，是数据存储的基本单位。数据表被定义为列的集合，数据在表中是按照行和列的格式来存储的。每一行代表一条唯一的记录，每一列代表记录中的一个域。

内容导航 Navigation

- 掌握如何创建数据库
- 熟悉数据库的删除操作
- 掌握如何创建数据表
- 掌握查看数据表结构的方法
- 掌握如何修改数据表
- 熟悉删除数据表的方法

3.1 创建数据库

MySQL 安装完成之后，将会在其 data 目录下自动创建几个必需的数据库，可以使用 SHOW DATABASES 语句来查看当前所有存在的数据库，输入语句如下：

```
mysql> SHOW DATABASES;
+------------------------+
| Database               |
+------------------------+
| information_schema     |
| mysql                  |
| performance_schema     |
| sakila                 |
```

```
| sys                    |
| world                  |
+------------------------+
```

可以看到，数据库列表中包含 6 个数据库，其中 MySQL 是必需的，它描述用户访问权限。

创建数据库是在系统磁盘上划分一块区域用于数据的存储和管理，如果管理员在设置权限的时候为用户创建了数据库，就可以直接使用，否则需要自己创建数据库。在 MySQL 中创建数据库的基本 SQL 语法格式如下：

```
CREATE DATABASE database_name;
```

database_name 为要创建的数据库的名称，该名称不能与已经存在的数据库重名。

【例 3.1】创建测试数据库 test_db，输入语句如下：

```
CREATE DATABASE test_db;
```

数据库创建好之后，可以使用 SHOW CREATE DATABASE 语句查看数据库的定义。

【例 3.2】查看创建好的数据库 test_db 的定义，输入语句如下：

```
mysql> SHOW CREATE DATABASE test_db\G
*** 1. row ***
      Database: test_db
Create Database: CREATE DATABASE `test_db` /*!40100 DEFAULT CHARACTER SET
utf8 */
```

可以看到，如果数据库创建成功，就会显示数据库的创建信息。

再次使用 SHOW DATABASES 语句来查看当前所有存在的数据库，输入语句如下：

```
mysql> SHOW databases;
+------------------------+
| Database               |
+------------------------+
| information_schema     |
| mysql                  |
| performance_schema     |
| sakila                 |
| sys                    |
| test_db                |
| world                  |
+------------------------+
```

可以看到，数据库列表中包含刚刚创建的数据库 test_db 和其他已经存在的数据库的名称。

3.2 删除数据库

删除数据库是将已经存在的数据库从磁盘空间上清除，清除之后，数据库中的所有数据也将一同被删除。删除数据库语句和创建数据库的命令相似，MySQL 中删除数据库的基本语法格式如下：

```
DROP DATABASE database_name;
```

database_name 为要删除的数据库的名称，如果指定的数据库不存在，删除就会出错。

【例 3.3】删除测试数据库 test_db，输入语句如下：

```
DROP DATABASE test_db;
```

语句执行完毕之后，数据库 test_db 将被删除，再次使用 SHOW CREATE DATABASE 声明查看数据库的定义，结果如下：

```
mysql> SHOW CREATE DATABASE test_db\G
ERROR 1049 (42000): Unknown database 'test_db'
```

执行结果给出一条错误信息：ERROR 1049 (42000)：Unknown database 'test_db'，即数据库 test_db 已不存在，表明删除成功。

3.3 创建数据表

在创建完数据库之后，接下来的工作就是创建数据表。所谓创建数据表，指的是在已经创建好的数据库中建立新表。创建数据表的过程是规定数据列的属性的过程，同时也是实施数据完整性（包括实体完整性、引用完整性和域完整性等）约束的过程。本节将介绍创建数据表的语法形式，如何添加主键约束、外键约束、非空约束等。

3.3.1 创建表的语法形式

数据表属于数据库，在创建数据表之前，应该使用语句“USE <数据库名>”指定操作是在哪个数据库中进行的，如果没有选择数据库，就会抛出 No database selected 的错误。

创建数据表的语句为 CREATE TABLE，语法规则如下：

```
CREATE  TABLE <表名>
(
  字段名1，数据类型 [列级别约束条件] [默认值]，
  字段名2，数据类型 [列级别约束条件] [默认值]，
  ……
```

```
    [表级别约束条件]
);
```

使用 CREATE TABLE 创建表时，必须指定以下信息：

（1）要创建的表的名称，不区分大小写，不能使用 SQL 语言中的关键字，如 DROP、ALTER、INSERT 等。

（2）数据表中每一个列（字段）的名称和数据类型，如果创建多个列，就要用逗号隔开。

【例 3.4】创建员工表 tb_emp1，结构如表 3.1 所示。

表 3.1　tb_emp1 表结构

字段名称	数据类型	备　注
id	INT(11)	员工编号
name	VARCHAR(25)	员工名称
deptId	INT(11)	所在部门编号
salary	FLOAT	工资

首先创建数据库，SQL 语句如下：

```
CREATE  DATABASE test_db;
```

选择创建表的数据库，SQL 语句如下：

```
USE test_db;
```

创建 tb_emp1 表，SQL 语句如下：

```
CREATE TABLE tb_emp1
(
  id      INT(11),
  name    VARCHAR(25),
  deptId  INT(11),
  salary  FLOAT
);
```

语句执行后，便创建了一个名称为 tb_emp1 的数据表，使用 SHOW TABLES 语句查看数据表是否创建成功，SQL 语句如下：

```
mysql> SHOW TABLES;
+----------------------+
| Tables_in_ test_db    |
+----------------------+
| tb_emp1               |
+----------------------+
```

可以看到，test_db 数据库中已经有了数据表 tb_emp1，数据表创建成功。

3.3.2 使用主键约束

主键又称主码，是表中一列或多列的组合。主键约束（Primary Key Constraint）要求主键列的数据唯一，并且不允许为空。主键能够唯一地标识表中的一条记录，可以结合外键来定义不同数据表之间的关系，并且可以加快数据库查询的速度。主键和记录之间的关系如同身份证和人之间的关系，它们之间是一一对应的。主键分为两种类型：单字段主键和多字段联合主键。

1. 单字段主键

主键由一个字段组成，SQL 语句格式分为以下两种情况。

（1）在定义列的同时指定主键，语法规则如下：

```
字段名 数据类型 PRIMARY KEY [默认值]
```

【例 3.5】定义数据表 tb_emp 2，其主键为 id，SQL 语句如下：

```
CREATE TABLE tb_emp2
(
  id          INT(11) PRIMARY KEY,
  name        VARCHAR(25),
  deptId      INT(11),
  salary      FLOAT
);
```

（2）在定义完所有列之后指定主键，语法规则如下：

```
[CONSTRAINT <约束名>] PRIMARY KEY [字段名]
```

【例 3.6】定义数据表 tb_emp 3，其主键为 id，SQL 语句如下：

```
CREATE TABLE tb_emp3
(
  id INT(11),
  name VARCHAR(25),
  deptId INT(11),
  salary FLOAT,
  PRIMARY KEY(id)
);
```

上述两个例子执行后的结果是一样的，都会在 id 字段上设置主键约束。

2. 多字段联合主键

主键由多个字段联合组成，语法规则如下：

```
PRIMARY KEY [字段1, 字段2,...,字段 n]
```

【例 3.7】定义数据表 tb_emp4，假设表中没有主键 id，为了唯一确定一个员工，可以把 name、deptId 联合起来作为主键，SQL 语句如下：

```
CREATE TABLE tb_emp4
(
  name VARCHAR(25),
  deptId INT(11),
  salary FLOAT,
  PRIMARY KEY(name,deptId)
);
```

语句执行后，便创建了一个名称为 tb_emp4 的数据表，name 字段和 deptId 字段组合在一起成为 tb_emp4 的多字段联合主键。

3.3.3　使用外键约束

外键用来在两个表的数据之间建立连接，它可以是一列或者多列。一个表可以有一个或多个外键。外键对应的是参照完整性，一个表的外键可以为空值，若不为空值，则每一个外键值必须等于另一个表中主键的某个值。

对于外键来说，首先它是表中的一个字段，可以不是本表的主键，但对应另一个表的主键。外键的主要作用是保证数据引用的完整性，定义外键后，不允许删除其在另一个表中具有关联关系的行。外键的作用是保持数据的一致性、完整性。例如，部门表 tb_dept 的主键是 id，在员工表 tb_emp5 中有一个键 deptId 与这个 id 关联。

- 主表（父表）：对于两个具有关联关系的表而言，相关联字段中主键所在的那个表就是主表。
- 从表（子表）：对于两个具有关联关系的表而言，相关联字段中外键所在的那个表就是从表。

创建外键的语法规则如下：

```
[CONSTRAINT <外键名>] FOREIGN KEY 字段名1 [ ,字段名2,…]
REFERENCES <主表名> 主键列1 [ ,主键列2,…]
```

“外键名”为定义的外键约束的名称，一个表中不能有相同名称的外键；“字段名”表示子表需要添加外键约束的字段列；“主表名”即被子表外键所依赖的表的名称；“主键列”表示主表中定义的主键列，或者列组合。

【例 3.8】定义数据表 tb_emp5，并在 tb_emp5 表上创建外键约束。

创建一个部门表 tb_dept1，表结构如表 3.2 所示，SQL 语句如下：

```
CREATE TABLE tb_dept1
(
  id      INT(11) PRIMARY KEY,
```

```
  name    VARCHAR(22)  NOT NULL,
  location  VARCHAR(50)
);
```

表 3.2　tb_dept1 表结构

字段名称	数据类型	备　注
id	INT(11)	部门编号
name	VARCHAR(22)	部门名称
location	VARCHAR(50)	部门位置

定义数据表 tb_emp5，让它的键 deptId 作为外键关联到 tb_dept1 的主键 id，SQL 语句如下：

```
CREATE TABLE tb_emp5
(
  id     INT(11) PRIMARY KEY,
  name   VARCHAR(25),
  deptId  INT(11),
  salary   FLOAT,
  CONSTRAINT fk_emp_dept1 FOREIGN KEY(deptId) REFERENCES tb_dept1(id)
);
```

以上语句执行成功之后，在表 tb_emp5 上添加了名称为 fk_emp_dept1 的外键约束，外键名称为 deptId，其依赖于表 tb_dept1 的主键 id。

提　示
关联指的是在关系型数据库中，相关表之间的联系。它是通过相容或相同的属性或属性组来表示的。子表的外键必须关联父表的主键，且关联字段的数据类型必须匹配，如果类型不一样，在创建子表时，就会出现错误：ERROR 1005 (HY000): Can't create table 'database.tablename'(errno: 150)。

3.3.4　使用非空约束

非空约束（Not Null Constraint）指字段的值不能为空。对于使用了非空约束的字段，如果用户在添加数据时没有指定值，数据库系统就会报错。

非空约束的语法规则如下：

```
字段名 数据类型 not null
```

【例 3.9】定义数据表 tb_emp6，指定员工的名称不能为空，SQL 语句如下：

```
CREATE TABLE tb_emp6
(
  id     INT(11) PRIMARY KEY,
  name   VARCHAR(25) NOT NULL,
```

```
    deptId  INT(11),
    salary  FLOAT
);
```

语句执行后，在 tb_emp6 中创建了一个 name 字段，其插入值不能为空（NOT NULL）。

3.3.5　使用唯一性约束

唯一性约束（Unique Constraint）要求该列唯一，允许为空，但只能出现一个空值。唯一性约束可以确保一列或者几列不出现重复值。

唯一性约束的语法规则如下：

（1）在定义完列之后直接指定唯一性约束，语法规则如下：

```
字段名 数据类型 UNIQUE
```

【例 3.10】定义数据表 tb_dept2，指定部门的名称唯一，SQL 语句如下：

```
CREATE TABLE tb_dept2
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(22) UNIQUE,
  location  VARCHAR(50)
);
```

（2）在定义完所有列之后指定唯一性约束，语法规则如下：

```
[CONSTRAINT <约束名>] UNIQUE(<字段名>)
```

【例 3.11】定义数据表 tb_dept3，指定部门的名称唯一，SQL 语句如下：

```
CREATE TABLE tb_dept3
(
  id      INT(11),
  name    VARCHAR(22),
  location  VARCHAR(50),
  CONSTRAINT STH UNIQUE(name)
);
```

UNIQUE 和 PRIMARY KEY 的区别：一个表中可以有多个字段声明为 UNIQUE，但只能有一个 PRIMARY KEY 声明；声明为 PRIMAY KEY 的列不允许有空值，但是声明为 UNIQUE 的字段允许空值的存在。

3.3.6　使用默认约束

默认约束（Default Constraint）指定某列的默认值。比如男性同学较多，性别就可以默认为“男”。如果插入一条新的记录时没有为这个字段赋值，那么系统会自动为这个字段赋值为“男”。

默认约束的语法规则如下：

```
字段名 数据类型 DEFAULT 默认值
```

【例 3.12】定义数据表 tb_emp7，指定员工的部门编号默认为 1111，SQL 语句如下：

```
CREATE TABLE tb_emp7
(
  id     INT(11) PRIMARY KEY,
  name   VARCHAR(25) NOT NULL,
  deptId  INT(11) DEFAULT 1111,
  salary  FLOAT
);
```

以上语句执行成功之后，表 tb_emp7 上的字段 deptId 拥有了一个默认的值 1111，新插入的记录如果没有指定部门编号，那么都默认为 1111。

3.3.7 设置表的属性值自动增加

在数据库应用中，经常希望在每次插入新记录时，系统自动生成字段的主键值。可以通过为表主键添加 AUTO_INCREMENT 关键字来实现。默认地，在 MySQL 中，AUTO_INCREMENT 的初始值是 1，每新增一条记录，字段值自动加 1。一个表只能有一个字段使用 AUTO_INCREMENT 约束，且该字段必须为主键的一部分。AUTO_INCREMENT 约束的字段可以是任何整数类型（TINYINT、SMALLINT、INT、BIGINT 等）的。

设置表的属性值自动增加的语法规则如下：

```
字段名 数据类型 AUTO_INCREMENT
```

【例 3.13】定义数据表 tb_emp8，指定员工的编号自动递增，SQL 语句如下：

```
CREATE TABLE tb_emp8
(
  id     INT(11) PRIMARY KEY AUTO_INCREMENT,
  name   VARCHAR(25) NOT NULL,
  deptId  INT(11),
  salary  FLOAT
);
```

上述例子执行后，会创建名称为 tb_emp8 的数据表。表 tb_emp8 中的 id 字段的值在添加记录的时候会自动增加，在插入记录的时候，默认的自增字段 id 的值从 1 开始，每次添加一条新记录，该值自动加 1。

例如，执行如下插入语句：

```
mysql> INSERT INTO tb_emp8 (name,salary)
-> VALUES('Lucy',1000), ('Lura',1200),('Kevin',1500);
```

语句执行完后，tb_emp8 表中增加了 3 条记录，在这里并没有输入 id 的值，但系统已经

自动添加该值，使用 SELECT 命令查看记录，结果如下：

```
mysql> SELECT * FROM tb_emp8;
+----+--------+----------+-------------+
| id | name   | deptId   | salary      |
+----+--------+----------+-------------+
|  1 | Lucy   |  NULL    |   1000      |
|  2 | Lura   |  NULL    |   1200      |
|  3 | Kevin  |  NULL    |   1500      |
+----+--------+----------+-------------+
```

提　示

这里使用 INSERT 声明向表中插入记录的方法，并不是 SQL 的标准语法，这种语法不一定被其他的数据库支持，只能在 MySQL 中使用。

3.4 查看数据表结构

使用 SQL 语句创建数据表之后，可以查看表结构的定义，以确认表的定义是否正确。在 MySQL 中，查看表结构可以使用 DESCRIBE 和 SHOW CREATE TABLE 语句。本节将针对这两个语句分别进行详细讲解。

3.4.1 查看表基本结构语句 DESCRIBE

DESCRIBE/DESC 语句可以查看表的字段信息，其中包括：字段名、字段数据类型、是否为主键、是否有默认值等。语法规则如下：

```
DESCRIBE 表名;
```

或者简写为：

```
DESC 表名;
```

【例 3.14】分别使用 DESCRIBE 和 DESC 查看表 tb_dept1 和表 tb_emp1 的表结构。

查看表 tb_dept1 的表结构，SQL 语句如下：

```
mysql> DESCRIBE tb_dept1;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| name     | varchar(22)  | NO   |     | NULL    |       |
| location | varchar(50)  | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

查看表 tb_emp1 的表结构，SQL 语句如下：

```
mysql> DESC tb_emp1;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int (11)    | YES  |     | NULL    |       |
| name   | varchar(25) | YES  |     | NULL    |       |
| deptId | int (11)    | YES  |     | NULL    |       |
| salary | float       | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

其中，各个字段的含义分别解释如下：

- NULL：表示该列是否可以存储 NULL 值。
- Key：表示该列是否已编制索引。PRI 表示该列是表主键的一部分；UNI 表示该列是 UNIQUE 索引的一部分；MUL 表示在列中某个给定值允许出现多次。
- Default：表示该列是否有默认值，如果有的话，那么值是多少。
- Extra：表示可以获取的、与给定列有关的附加信息，例如 AUTO_INCREMENT 等。

3.4.2 查看表详细结构语句 SHOW CREATE TABLE

SHOW CREATE TABLE 语句可以用来显示创建表时的 CREATE TABLE 语句，语法格式如下：

```
SHOW CREATE TABLE <表名\G>;
```

提　示
使用 SHOW CREATE TABLE 语句不仅可以查看表创建时的详细语句，还可以查看存储引擎和字符编码。

如果不加“\G”参数，显示的结果就可能非常混乱，加上参数“\G”之后，可使显示结果更加直观，易于查看。

【例 3.15】使用 SHOW CREATE TABLE 查看表 tb_emp1 的详细信息，SQL 语句如下：

```
mysql> SHOW CREATE TABLE tb_emp1;
+--------+-------------------------------------------------------------+
| Table  | Create Table                                                |
+--------+-------------------------------------------------------------+
| fruits | CREATE TABLE `fruits` (
  `f_id` char(10) NOT NULL,
  `s_id` int(11) NOT NULL,
  `f_name` char(255) NOT NULL,
  `f_price` decimal(8,2) NOT NULL,
  PRIMARY KEY (`f_id`),
```

```
  KEY `index_name` (`f_name`),
  KEY `index_id_price` (`f_id`,`f_price`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+--------+-------------------------------------------------------------+
```

使用参数“\G”之后的结果如下：

```
mysql> SHOW CREATE TABLE tb_emp1\G
*** 1. row ***
       Table: tb_emp1
Create Table: CREATE TABLE `tb_emp1` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)
```

3.5 修改数据表

修改表指的是修改数据库中已经存在的数据表的结构。MySQL 使用 ALTER TABLE 语句修改表。常用的修改表的操作有：修改表名、修改字段数据类型或字段名、增加和删除字段、修改字段的排列位置、更改表的存储引擎、删除表的外键约束等。本节将对修改表及其相关的操作进行讲解。

3.5.1 修改表名

MySQL 通过 ALTER TABLE 语句来实现表名的修改，具体的语法规则如下：

```
ALTER TABLE <旧表名> RENAME [TO] <新表名>;
```

其中，TO 为可选参数，使用与否均不影响结果。

【例 3.16】将数据表 tb_dept3 改名为 tb_deptment3。

执行修改表名操作之前，使用 SHOW TABLES 查看数据库中所有的表，SQL 语句如下：

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_test_db  |
+--------------------+
| tb_dept            |
| tb_dept2           |
| tb_dept3           |
```

```
……省略部分内容
```

使用 ALTER TABLE 将表 tb_dept3 改名为 tb_deptment3，SQL 语句如下：

```
ALTER TABLE tb_dept3 RENAME tb_deptment3;
```

语句执行之后，检验表 tb_dept3 是否改名成功。使用 SHOW TABLES 查看数据库中的表，结果如下：

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_test_db   |
+---------------------+
| tb_dept             |
| tb_dept2            |
| tb_deptment3        |
……省略部分内容
```

经过比较可以看到，数据表列表中已经有了名称为 tb_deptment3 的表。

提　示
读者可以在修改表名称时使用 DESC 命令查看修改前后两个表的结构，修改表名并不修改表的结构，因此修改名称后的表和修改名称前的表的结构必然是相同的。

3.5.2　修改字段的数据类型

修改字段的数据类型就是把字段的数据类型转换成另一种数据类型。在 MySQL 中修改字段数据类型的语法规则如下：

```
ALTER TABLE <表名> MODIFY <字段名>  <数据类型>
```

其中，“表名”指要修改数据类型的字段所在表的名称，“字段名”指需要修改的字段，“数据类型”指修改后字段的新数据类型。

【例 3.17】将数据表 tb_dept1 中，name 字段的数据类型由 VARCHAR(22)修改成 VARCHAR(30)。

执行修改表名操作之前，使用 DESC 查看 tb_dept 表结构，结果如下：

```
mysql> DESC tb_dept1;
+----------+---------------+---------+--------+-------------+--------+
| Field    | Type          | Null    | Key    |Default      | Extra  |
+----------+---------------+---------+--------+-------------+--------+
| id       | int(11)       | NO      | PRI    | NULL        |        |
| name     | varchar(22)   | YES     |        | NULL        |        |
| location | varchar(50)   | YES     |        | NULL        |        |
+----------+---------------+--------+---------+-------------+--------+
```

可以看到现在 name 字段的数据类型为 VARCHAR(22)，下面修改其类型，输入如下 SQL 语句并执行：

```
ALTER TABLE tb_dept1 MODIFY name VARCHAR(30);
```

再次使用 DESC 查看表，结果如下：

```
mysql> DESC tb_dept1;
+----------+---------------+---------+--------+-------------+--------+
| Field    | Type          | Null    |  Key   |Default      | Extra |
+----------+---------------+---------+--------+-------------+--------+
| id       | int(11)       | NO      | PRI    | NULL        |       |
| name     | varchar(30)   | YES     |        | NULL        |       |
| location | varchar(50)   | YES     |        | NULL        |       |
+----------+---------------+--------+---------+-------------+--------+
```

语句执行之后，检验会发现表 tb_dept 中 name 字段的数据类型已经修改成了 VARCHAR(30)，修改成功。

3.5.3　修改字段名

在 MySQL 中，修改表字段名的语法规则如下：

```
ALTER TABLE <表名> CHANGE <旧字段名> <新字段名> <新数据类型>;
```

其中，“旧字段名”指修改前的字段名；“新字段名”指修改后的字段名；“新数据类型”指修改后的数据类型，如果不需要修改字段的数据类型，那么将新数据类型设置成与原来一样即可，但数据类型不能为空。

【例 3.18】将数据表 tb_dept1 中的 location 字段名称改为 loc，数据类型保持不变，SQL 语句如下：

```
ALTER TABLE tb_dept1 CHANGE location loc VARCHAR(50);
```

使用 DESC 查看表 tb_dept1，会发现字段的名称已经修改成功，结果如下：

```
mysql> DESC tb_dept1;
+----------+---------------+---------+--------+---------+--------+
| Field    | Type          | Null    | Key    |Default  | Extra |
+----------+---------------+---------+--------+---------+--------+
| id       | int(11)       | NO      | PRI    | NULL    |       |
| name     | varchar(30)   | YES     |        | NULL    |       |
| loc      | varchar(50)   | YES     |        | NULL    |       |
+----------+---------------+---------+--------+---------+--------+
```

【例 3.19】将数据表 tb_dept1 中的 loc 字段名称改为 location，同时将数据类型变为 VARCHAR(60)，SQL 语句如下：

```
ALTER TABLE tb_dept1 CHANGE loc location VARCHAR(60);
```

使用 DESC 查看表 tb_dept1，会发现字段的名称和数据类型均已经修改成功，结果如下：

```
mysql> DESC tb_dept1;
+----------+---------------+---------+--------+-------------+-------+
| Field    | Type          | Null    | Key    |Default      | Extra |
+----------+---------------+---------+--------+-------------+-------+
| id       | int(11)       | NO      | PRI    | NULL        |       |
| name     | varchar(30)   | YES     |        | NULL        |       |
| location | varchar(60)   | YES     |        | NULL        |       |
+----------+---------------+---------+--------+-------------+-------+
```

提　示

CHANGE 也可以只修改数据类型，实现和 MODIFY 同样的效果，方法是将 SQL 语句中的“新字段名”和“旧字段名”设置为相同的名称，只改变“数据类型”。

由于不同类型的数据在机器中存储的方式及长度并不相同，修改数据类型可能会影响数据表中已有的数据记录，因此，当数据库表中已经有数据时，不要轻易修改数据类型。

3.5.4　添加字段

随着业务需求的变化，可能需要在已经存在的表中添加新的字段。一个完整字段包括字段名、数据类型、完整性约束。添加字段的语法格式如下：

```
ALTER TABLE <表名> ADD <新字段名> <数据类型>
[约束条件] [FIRST | AFTER 已存在字段名];
```

新字段名为需要添加的字段的名称；FIRST 为可选参数，其作用是将新添加的字段设置为表的第一个字段；AFTER 为可选参数，其作用是将新添加的字段添加到指定的“已存在字段名”的后面。

提　示

“FIRST | AFTER 已存在字段名”用于指定新增字段在表中的位置，如果 SQL 语句中没有这两个参数，就默认将新添加的字段设置为数据表的最后列。

1. 添加无完整性约束条件的字段

【例 3.20】在数据表 tb_dept1 中添加一个没有完整性约束的 INT 类型的字段 managerId（部门经理编号），SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD managerId INT(10);
```

使用 DESC 查看表 tb_dept1，会发现在表的最后添加了一个名为 managerId 的 INT 类型的字段，结果如下：

```
mysql> DESC tb_dept1;
+-------------+-----------------+----------+--------+------------+--------+
| Field       | Type            | Null     | Key    | Default    | Extra  |
+-------------+-----------------+----------+--------+------------+--------+
| id          | int(11)         | NO       | PRI    | NULL       |        |
| name        | varchar(30)     | YES      |        | NULL       |        |
| location    | varchar(60)     | YES      |        | NULL       |        |
| managerId   | int(10)         | YES      |        | NULL       |        |
+-------------+-----------------+----------+--------+------------+--------+
```

2. 添加有完整性约束条件的字段

【例 3.21】在数据表 tb_dept1 中添加一个不能为空的 VARCHAR(12)类型的字段 column1，SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD column1 VARCHAR(12) not null;
```

使用 DESC 查看表 tb_dept1，会发现在表的最后添加了一个名为 column1 的 VARCHAR(12)类型且不为空的字段，结果如下：

```
mysql> DESC tb_dept1;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| id        | int(11)     | NO   | PRI | NULL    |       |
| name      | varchar(30) | YES  |     | NULL    |       |
| location  | varchar(60) | YES  |     | NULL    |       |
| managerId | int(10)     | YES  |     | NULL    |       |
| column1   | varchar(12) | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

3. 在表的第一列添加一个字段

【例 3.22】在数据表 tb_dept1 中添加一个 INT 类型的字段 column2，SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD column2 INT(11) FIRST;
```

使用 DESC 查看表 tb_dept1，会发现在表第一列添加了一个名为 column2 的 INT(11)类型的字段，结果如下：

```
mysql> DESC tb_dept1;
+-------------+--------------+----------+--------+-------------+-------+
| Field       | Type         | Null     | Key    | Default     | Extra |
+-------------+--------------+----------+--------+-------------+-------+
| column2     | int(11)      | YES      |        | NULL        |       |
| id          | int(11)      | NO       | PRI    | NULL        |       |
| name        | varchar(30)  | YES      |        | NULL        |       |
| location    | varchar(60)  | YES      |        | NULL        |       |
```

```
| managerId    | int(10)        | YES       |         | NULL          |        |
| column1      | varchar(12)    | NO        |         | NULL          |        |
+--------------+----------------+-----------+---------+---------------+--------+
```

4. 在表的指定列之后添加一个字段

【例 3.23】在数据表 tb_dept1 的 name 列后添加一个 INT 类型的字段 column3，SQL 语句如下：

```
ALTER TABLE tb_dept1 ADD column3 INT(11) AFTER name;
```

使用 DESC 查看表 tb_dept1，结果如下：

```
mysql> DESC tb_dept1;
+------------+-----------------+----------+---------+-------------+--------+
| Field      | Type            | Null     | Key     | Default     | Extra  |
+------------+-----------------+----------+---------+-------------+--------+
| column2    | int(11)         | YES      |         | NULL        |        |
| id         | int(11)         | NO       | PRI     | NULL        |        |
| name       | varchar(30)     | YES      |         | NULL        |        |
| column3    | int(11)         | YES      |         | NULL        |        |
| location   | varchar(60)     | YES      |         | NULL        |        |
| managerId  | int(10)         | YES      |         | NULL        |        |
| column1    | varchar(12)     | NO       |         | NULL        |        |
+------------+-----------------+----------+---------+-------------+--------+
```

可以看到，tb_dept1 表中增加了一个名称为 column3 的字段，其位置在指定的 name 字段后面，表明添加字段成功。

3.5.5 删除字段

删除字段是将数据表中的某个字段从表中移除，语法格式如下：

```
ALTER TABLE <表名> DROP <字段名>;
```

“字段名”指需要从表中删除的字段的名称。

【例 3.24】删除数据表 tb_dept1 中的 column2 字段。

首先，执行删除字段之前，使用 DESC 查看表 tb_dept1 的表结构，结果如下：

```
mysql> DESC tb_dept1;
+-------------+---------------+----------+---------+-------------+--------+
| Field       | Type          | Null     | Key     | Default     | Extr   |
+-------------+---------------+----------+---------+-------------+--------+
| column2     | int(11)       | YES      |         | NULL        |        |
| id          | int(11)       | NO       | PRI     | NULL        |        |
| name        | varchar(30)   | YES      |         | NULL        |        |
| column3     | int(11)       | YES      |         | NULL        |        |
```

```
| location    | varchar(60)    | YES       |          | NULL          |       |
| managerId   | int(10)        | YES       |          | NULL          |       |
| column1     | varchar(12)    | NO        |          | NULL          |       |
+-------------+----------------+-----------+----------+---------------+-------+
```

删除 column2 字段，SQL 语句如下：

```
ALTER TABLE tb_dept1 DROP column2;
```

再次使用 DESC 查看表 tb_dept1 的表结构，结果如下：

```
mysql> DESC tb_dept1;
+-------------+----------------+----------+--------+-------------+--------+
| Field       | Type           | Null     | Key    | Default     | Extr   |
+-------------+----------------+----------+--------+-------------+--------+
| id          | int(11)        | NO       | PRI    | NULL        |        |
| name        | varchar(30)    | YES      |        | NULL        |        |
| column3     | int(11)        | YES      |        | NULL        |        |
| location    | varchar(60)    | YES      |        | NULL        |        |
| managerId   | int(10)        | YES      |        | NULL        |        |
| column1     | varchar(12)    | NO       |        | NULL        |        |
+-------------+----------------+----------+--------+-------------+--------+
```

可以看到，表 tb_dept1 中已经不存在名称为 column2 的字段，删除字段成功。

3.5.6　修改字段的排列位置

对于一个数据表来说，在创建的时候，字段在表中的排列顺序就已经确定了。但表的结构并不是完全不可以改变的，我们可以通过 ALTER TABLE 命令来改变表中字段的相对位置，语法格式如下：

```
ALTER TABLE <表名> MODIFY <字段1> <数据类型> FIRST|AFTER <字段2>;
```

“字段 1”指要修改位置的字段，“数据类型”指“字段 1”的数据类型；FIRST 为可选参数，指将“字段 1”修改为表的第一个字段；“AFTER 字段 2”指将“字段 1”插入“字段 2”后面。

1. 修改字段为表的第一个字段

【例 3.25】将数据表 tb_deptl 中的 column1 字段修改为表的第一个字段，SQL 语句如下：

```
ALTER TABLE tb_dept1 MODIFY column1 VARCHAR(12) FIRST;
```

使用 DESC 查看表 tb_dept1，发现字段 column1 已经被移至表的第一列，结果如下：

```
mysql> DESC tb_dept1;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
```

```
| column1   | varchar(12)  | YES  |     | NULL    |       |
| id        | int(11)      | NO   | PRI | NULL    |       |
| name      | varchar(30)  | YES  |     | NULL    |       |
| column3   | int(11)      | YES  |     | NULL    |       |
| location  | varchar(60)  | YES  |     | NULL    |       |
| managerId | int(10)      | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

2. 修改字段到表的指定列之后

【例 3.26】将数据表 tb_dept1 中的 column1 字段插入 location 字段后面，SQL 语句如下：

```
ALTER TABLE tb_dept1 MODIFY column1 VARCHAR(12) AFTER location;
```

使用 DESC 查看表 tb_dept1，结果如下：

```
mysql> DESC tb_dept1;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int(11)      | NO   | PRI | NULL    |       |
| name      | varchar(30)  | YES  |     | NULL    |       |
| column3   | int(11)      | YES  |     | NULL    |       |
| location  | varchar(60)  | YES  |     | NULL    |       |
| column1   | varchar(12)  | YES  |     | NULL    |       |
| managerId | int(10)      | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

可以看到，tb_dept1 表中的字段 column1 已经被移至 location 字段之后。

3.5.7 删除表的外键约束

对于数据库中定义的外键，如果不再需要，可以将其删除。外键一旦删除，就会解除主表和从表之间的关联关系。MySQL 中删除外键的语法格式如下：

```
ALTER TABLE <表名> DROP FOREIGN KEY <外键约束名>
```

“外键约束名”指在定义表时 CONSTRAINT 关键字后面的参数，详细内容可参考 3.3.3 小节的“使用外键约束”。

【例 3.27】删除数据表 tb_emp9 中的外键约束。

首先创建表 tb_emp9，创建外键 deptId 关联 tb_dept1 表的主键 id，SQL 语句如下：

```
CREATE TABLE tb_emp9
(
  id     INT(11) PRIMARY KEY,
  name   VARCHAR(25),
  deptId  INT(11),
```

```
    salary   FLOAT,
    CONSTRAINT fk_emp_dept  FOREIGN KEY (deptId) REFERENCES tb_dept1(id)
  );
```

使用 SHOW CREATE TABLE 查看表 tb_emp9 的结构，结果如下：

```
  mysql> SHOW CREATE TABLE tb_emp9 \G
  *** 1. row ***
        Table: tb_emp9
  Create Table: CREATE TABLE `tb_emp9` (
    `id` int(11) NOT NULL,
    `name` varchar(25) DEFAULT NULL,
    `deptId` int(11) DEFAULT NULL,
    `salary` float DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `fk_emp_dept` (`deptId`),
    CONSTRAINT  `fk_emp_dept`  FOREIGN  KEY  (`deptId`)  REFERENCES  `tb_dept1`
(`id`)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
  1 row in set (0.00 sec)
```

可以看到，已经成功添加了表的外键。下面删除外键约束，SQL 语句如下：

```
  ALTER TABLE tb_emp9 DROP FOREIGN KEY fk_emp_dept;
```

执行完毕之后，将删除表 tb_emp9 的外键约束，使用 SHOW CREATE TABLE 再次查看表 tb_emp9 的结构，结果如下：

```
  mysql> SHOW CREATE TABLE tb_emp9 \G
  *** 1. row ***
        Table: tb_emp9
  Create Table: CREATE TABLE `tb_emp9` (
    `id` int(11) NOT NULL,
    `name` varchar(25) DEFAULT NULL,
    `deptId` int(11) DEFAULT NULL,
    `salary` float DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `fk_emp_dept` (`deptId`)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
  1 row in set (0.00 sec)
```

可以看到，表 tb_emp9 中已经不存在 FOREIGN KEY，原有的名称为 fk_emp_dept 的外键约束删除成功。