

## 第 28 章 数据库编程

数据必须以某种方式来存储才可以有用，数据库实际上是一组相关数据的集合。例如，某个医疗机构中所有信息的集合可以被称为一个“医疗机构数据库”，这个数据库中的所有数据都与医疗机构的相关。

数据库编程相关的技术很多，涉及具体的数据库安装、配置和管理，还要掌握SQL语句，最后才能编写程序访问数据库。本章重点介绍MySQL数据库的安装和配置，以及JDBC数据库编程。

## 28.1 数据持久技术概述

把数据保存到数据库中只是一种数据持久化方式。凡是将数据保存到存储介质中，需要的时候能够找到它们，并能够对数据进行修改，这些就属于数据持久化。

Java中数据持久化技术有很多：

### 01. 文本文件

通过Java I/O流技术将数据保存到文本文件中，然后进行读写操作，这些文件一般是结构化的文档，如XML、JSON和CSV等文件。结构化文档就是文件内部采取某种方式将数据组织起来。

### 02. 对象序列化

序列化用于将某个对象以及它的状态写到文件中，它保证了被写入的对象之间的关系，当需要这个对象时，可以完整地重新构造出来，并保持原来的状态。在Java中实现java.io.Serializable接口的对象才能被序列化和反序列化。Java还提供了两个流：ObjectInputStream和ObjectOutputStream。但序列化不支持事务处理、查询或者向不同的用户共享数据。序列化只适用于最简单的应用，或者在某些无法有效地支持数据库的嵌入式系统中。

### 03. 数据库

将数据保存到数据库中是不错的选择，数据库的后面是一个数据库管理系统，它支持事务处理、并发访问、高级查询和SQL语言。Java对象保存到数据库中主要的技术有：JDBC<sup>1</sup>、EJB<sup>2</sup>和ORM<sup>3</sup>框架等。JDBC是本书重点介绍的技术。

<sup>1</sup>Java数据库连接（Java Database Connectivity，简称JDBC）。

<sup>2</sup>企业级JavaBean（Enterprise JavaBean，简称EJB）是一个用来构筑企业级应用的服务器端组件。

<sup>3</sup>对象关系映射（Object-Relational mapping，简称ORM），它能将对象保存到数据库表中，对象与数据库表结构之间是有某种对应关系的。

## 28.2 MySQL数据库管理系统

介绍JDBC技术一定会依托某个数据库管理系统（Database Management System，缩写DBMS），还会使用SQL语句，所以本节先介绍一下数据库管理系统。

数据库管理系统负责对数据进行管理、维护和使用。现在主流数据库管理系统有Oracle、SQL Server、DB 2、Sysbase和MySQL等，本节介绍MySQL数据库管理系统使用和管理。

MySQL（<https://www.mysql.com>）是流行的开放源码SQL数据库管理系统，它是由MySQL AB公司开发，先被Sun公司收购，后来又被Oracle公司收购，现在MySQL数据库是Oracle旗下的数据库产品，Oracle负责提供技术支持和维护。

### 28.2.1 数据库安装与配置

目前Oracle提供了多个MySQL版本，其中社区版MySQL Community Edition是免费的，社区版本比较适合中小企业数据库，本书也采用这个版本介绍。

社区版下载地址是<https://dev.mysql.com/downloads/windows/installer/5.7.html>，如图28-1所示，可以选择不同的平台版本，MySQL可运行在Windows、Linux和UNIX等操作系统上安装和运行。本书选择的是Windows 版中的mysql-installer-community-5.7.18.1.msi安装文件。

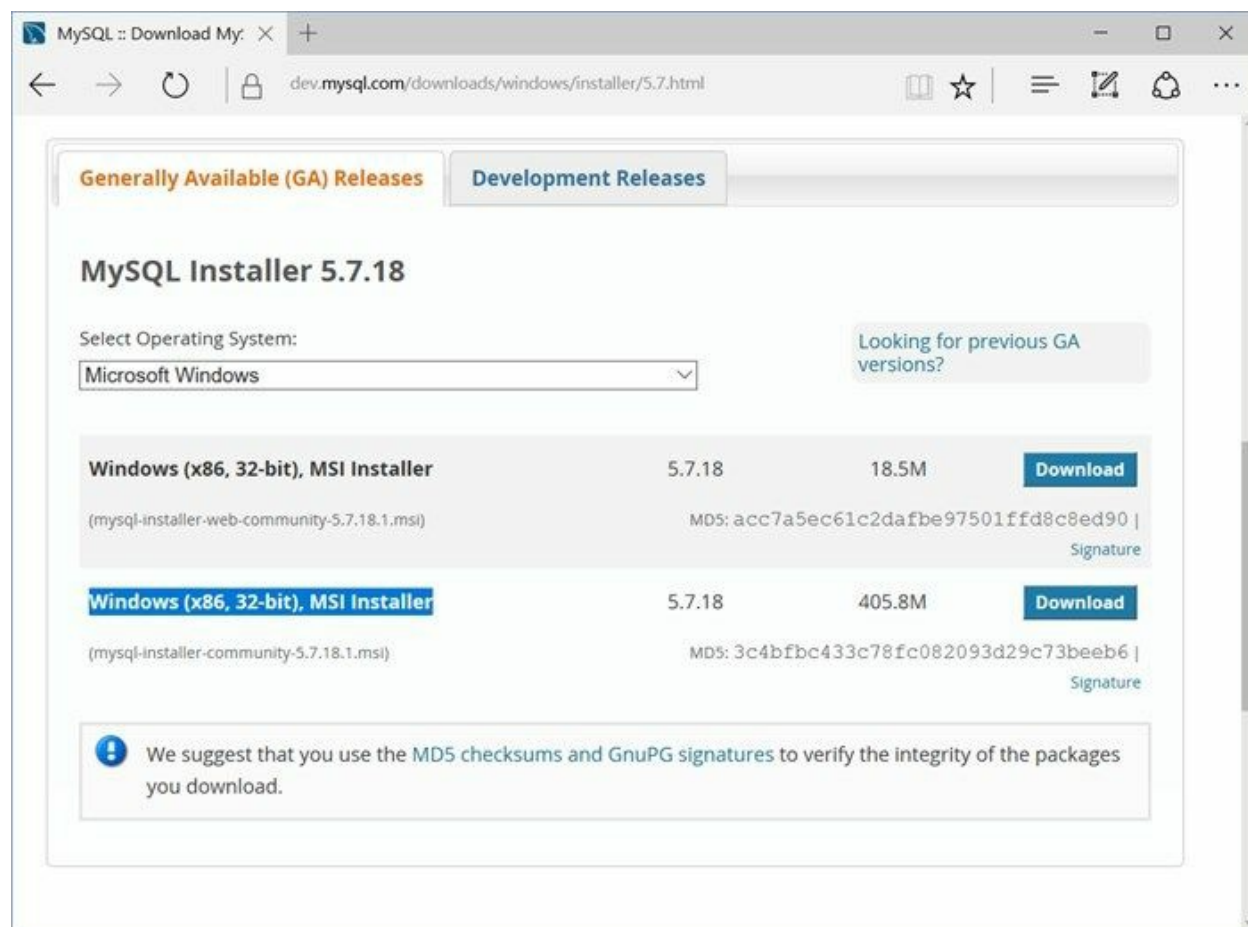


图28-1 MySQL数据库社区版下载

下载成功后，可以双击.msi文件启动安装过程，安装过程比较简单，这里介绍一个关键步骤。

## 01. 安装类型选择

如图28-2所示是安装类型选择对话框。在这个页面中可以选择安装类型，有5种安装类型：Developer Default（开发者安装）、Server only（只安装服务器）、Client only（只安装客户端）、Full（全部安装）和Custom（自定义安装）。对于学习和开发可以选择Developer Default安装。

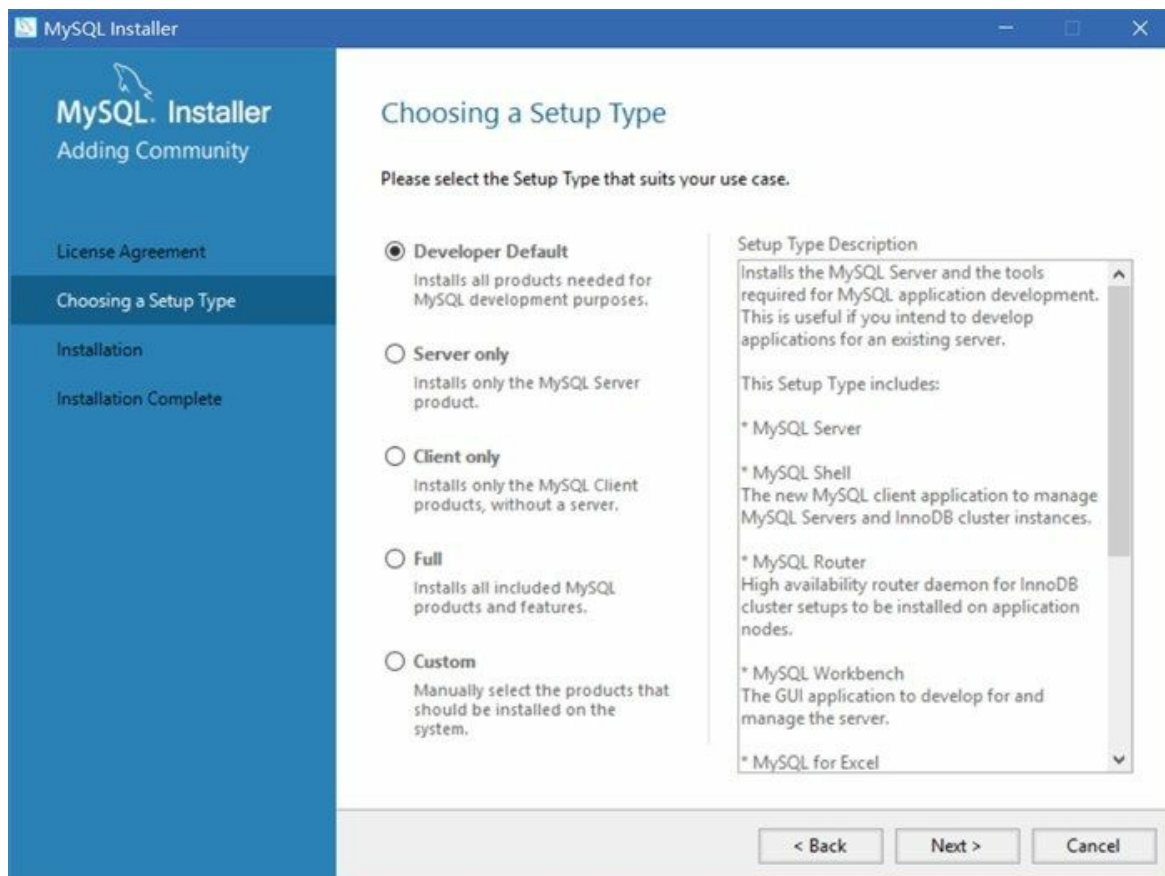


图28-2 安装类型选择

## 02. 安装环境检查

在Windows下安装时，由于Windows版本多样性。安装过程会检查你的需要，缺少哪些Windows安装包，安装过程会给出提示，如图28-3所示，安装MySQL Server需要Microsoft Visual C++ 2013 Runtime，则需要到微软网站下载Microsoft Visual C++ 2013 Runtime安装包，安装好Microsoft Visual C++ 2013 Runtime后，再重新安装MySQL。

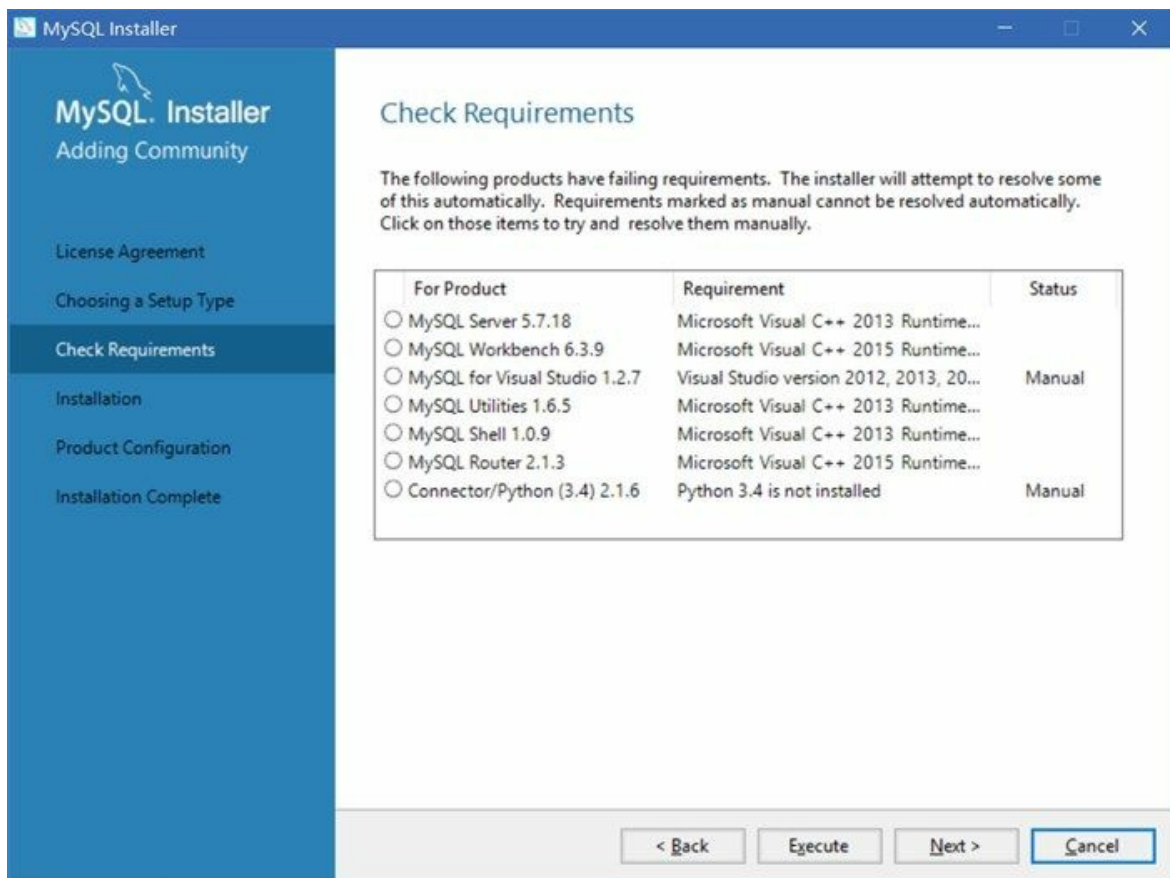


图28-3 安装环境检查

### 03. 配置过程

所需要的文件安装完成后，就会进入MySQL的配置过程。首先如图28-4所示是数据库类型选择对话框，Standalone是单个服务器，InnoDB Cluster是数据库集群。

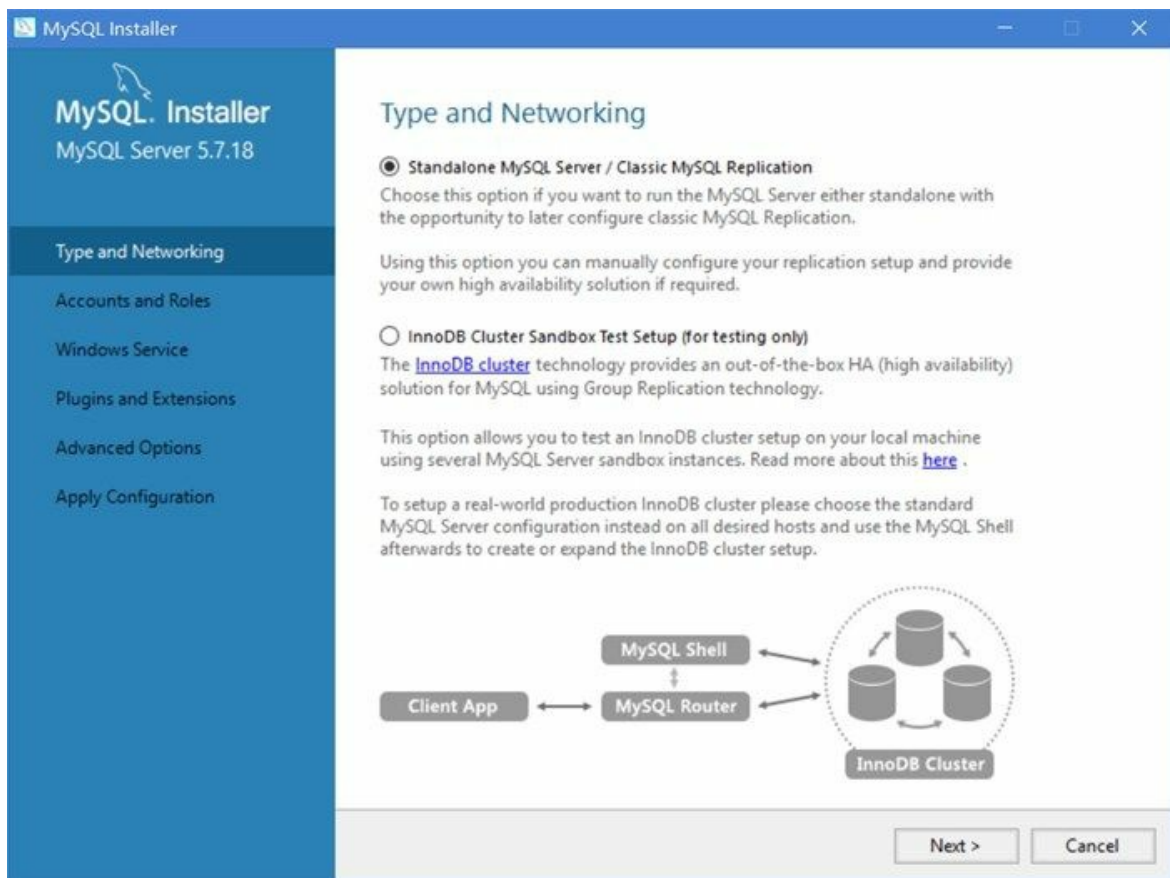


图28-4 数据库类型选择对话框

在图28-4所示的对话框中选择Standalone，单击Next按钮进入如图28-5所示的服务器配置类型选择对话框。在这里可以选择配置类型、通信协议和端口等，单击Config Type下拉列表可以选择如下的配置类型：

- **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。
- **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。MySQL服务器配置成使用适当比例的系统资源。
- **Dedicated Machine（专用MySQL服务器）**：该选项代表只运行MySQL服务的服务器。假定没有运行其它应用程序。MySQL服务器配置成使用所有可用系统资源。

根据自己的需要选择配置类型，其他的配置项目保持默认值，单击Next按钮进入如图28-6所示的账号和用户角色设置对话框。

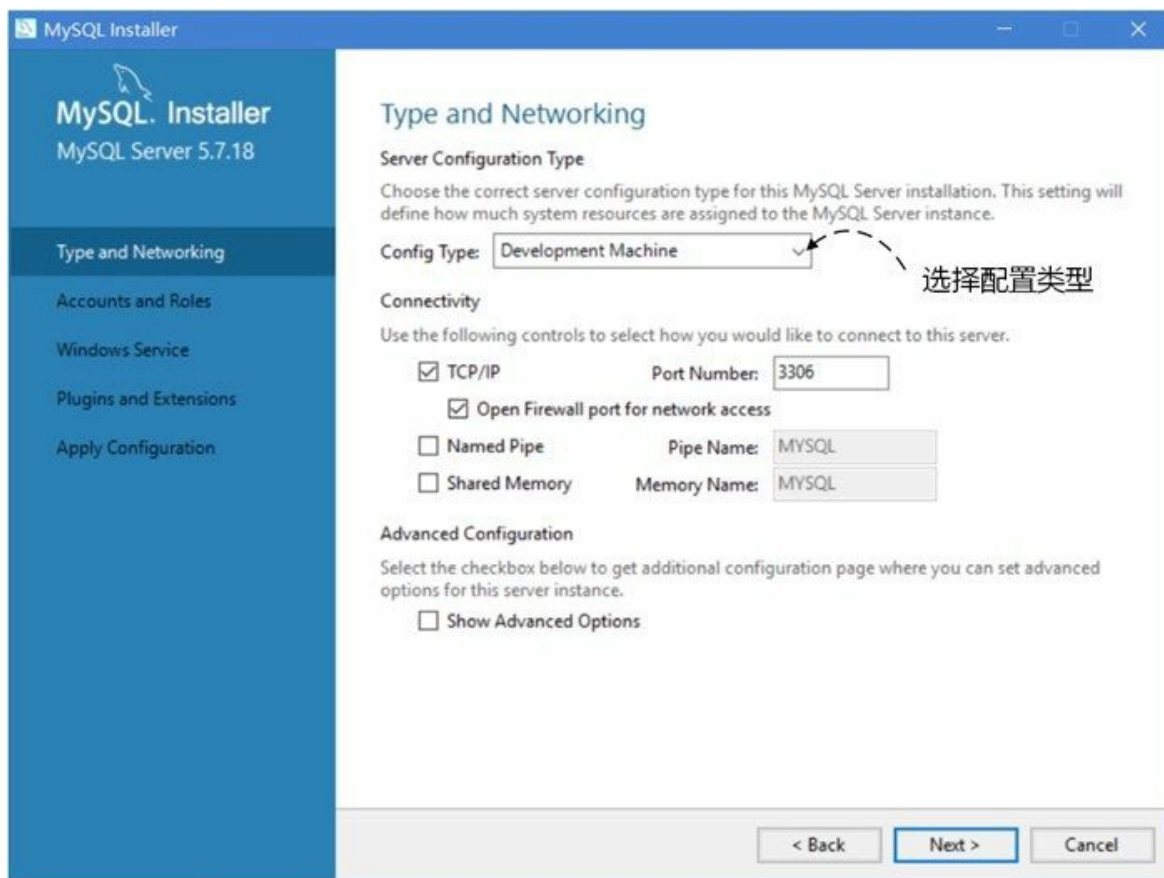


图28-5 服务器配置类型对话框

在图28-6所示的对话框中可以进行设置root密码，以及添加其他账号等操作。root密码必须是4位以上，根据需要设置root密码。此外，还可以单击Add User按钮添加其他的账号。

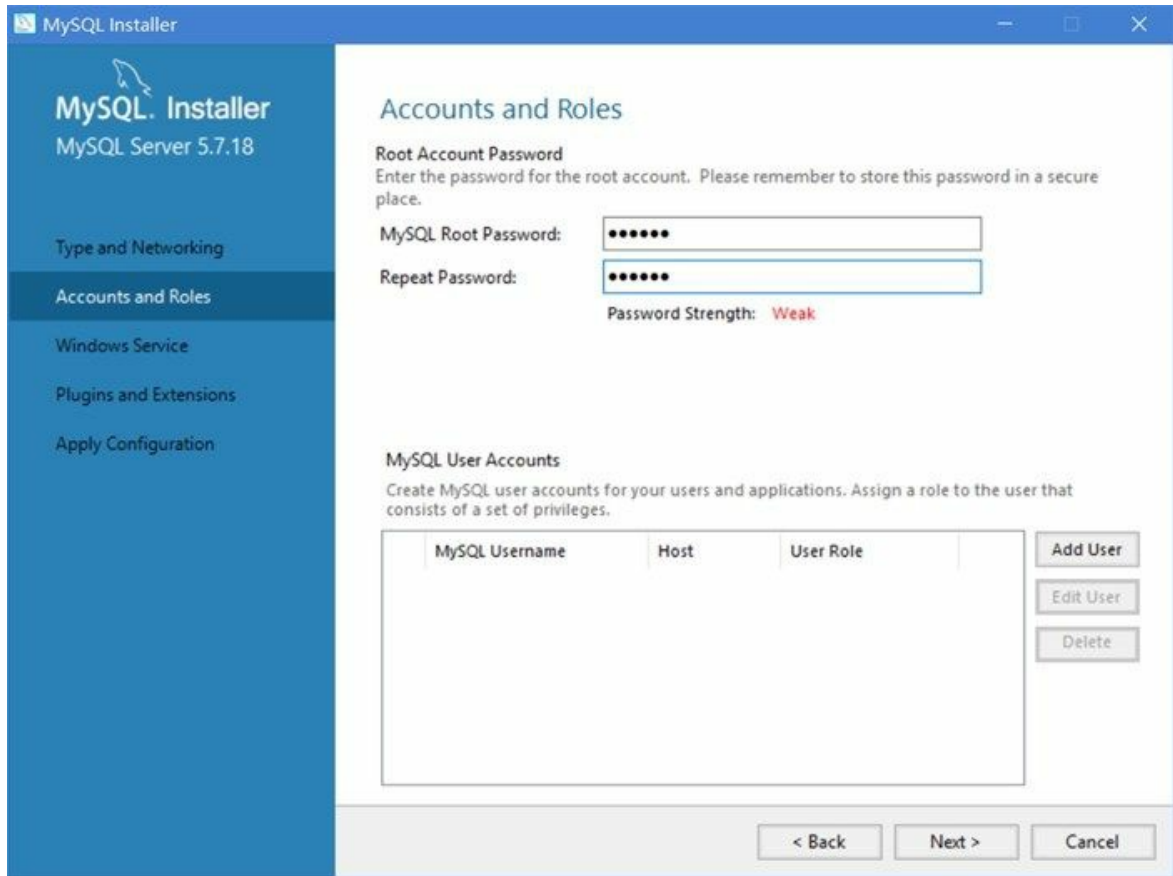


图28-6 账号和用户角色设置对话框

在图28-6对话框设置完成后，单击Next按钮进入图28-7所示的配置Windows服务对话框，在这里可以将MySQL数据库配置成为一个Windows服务，Windows服务可以在后台随着Windows已启动而启动，不需要人为干预。其实默认的服务名是MySQL57。



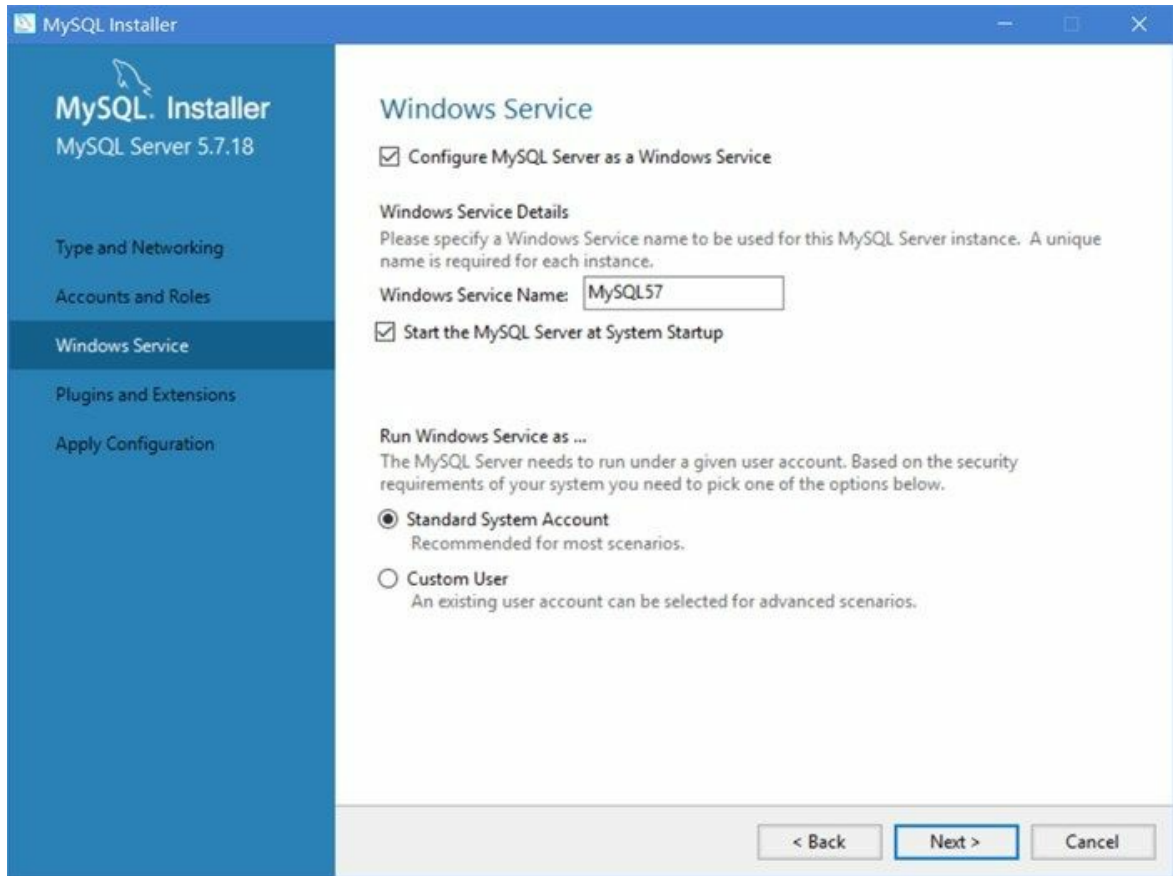



图28-7 配置Windows服务对话框

在图28-7所示配置界面之后，不需要再进行配置了，只需要单击Next按钮，这里不再赘述。

### 28.2.2 连接MySQL服务器

由于MySQL是C/S（客户端/服务器）结构的，所以应用程序包括它的客户端必须连接到服务器才能使用其服务功能。下面主要介绍MySQL本身客户端如何连接到服务器。

#### 01. 快速连接服务器方式

MySQL for Windows版本提供一个菜单项目可以快速连接服务器，打开过程右击屏幕左下角的Windows图标，在“最近添加”中找到MySQL 5.7 Command Line Client，则会在打开一个终端窗口如图28-8所示对话框。

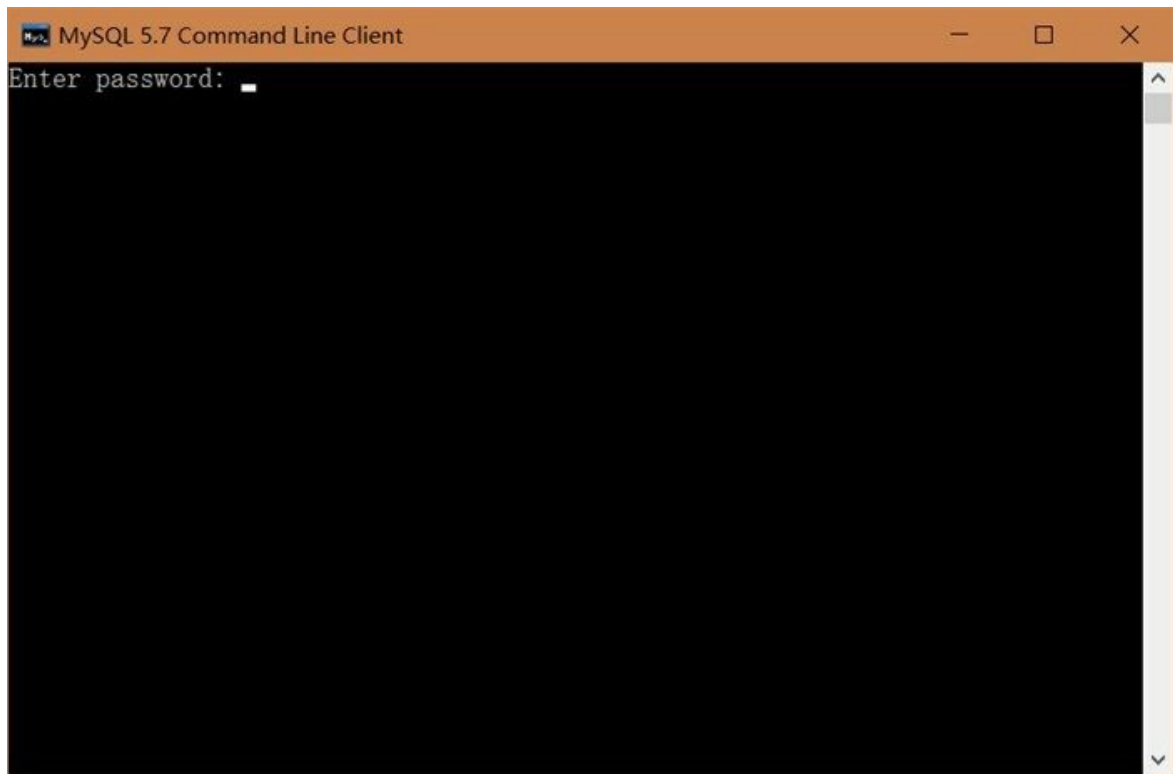
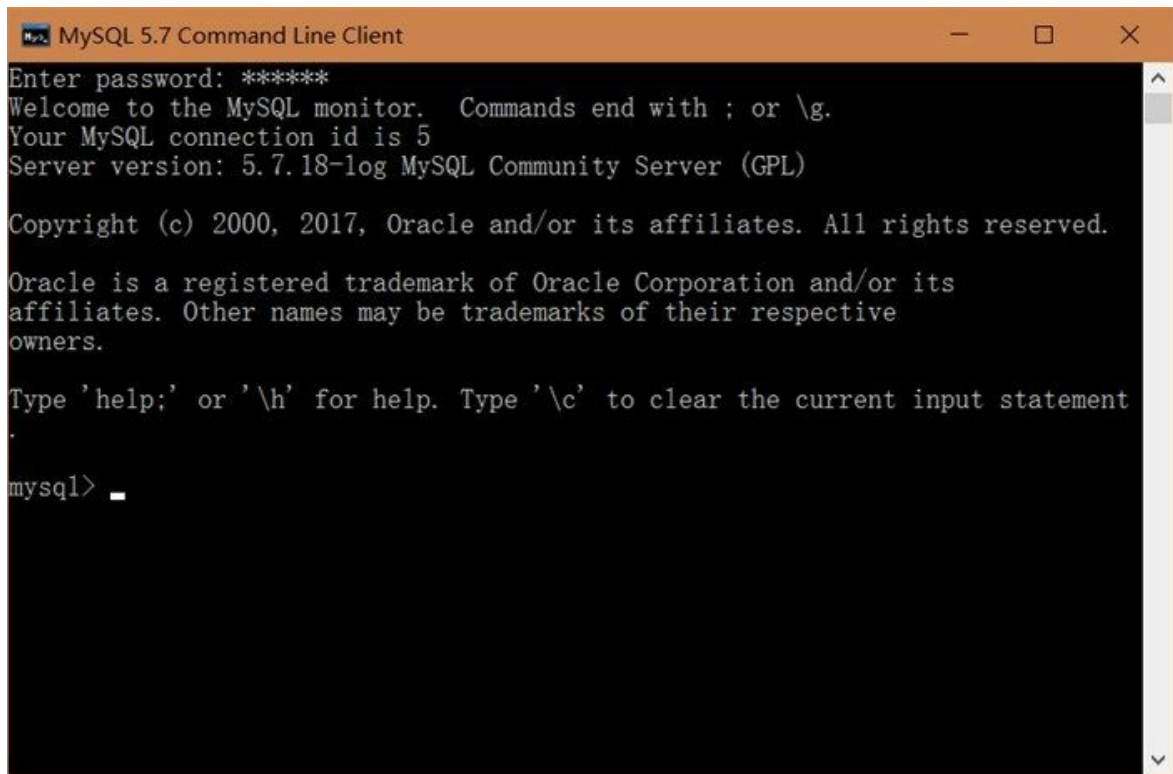


图28-8 MySQL命令行客户端

这个工具就是MySQL命令行客户端工具，可以使用MySQL命令行客户端工具连接到MySQL服务器，要求输入root密码。输入root密码按Enter键，如果密码正确则连接到MySQL服务器，如图28-9所示。



```
MySQL 5.7 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

mysql> _
```

图28-9 使用命令行客户端连接到服务器

## 02. 通用的连接方式

快速连接服务器方式连接的是本地数据库，如果服务器不在本地，而是在一个远程主机上，那么需要可以使用通用的连接方式。

首先在操作系统下打开一个终端窗口，Windows下是命令行工具，在次输入mysql -h localhost -u root -p命令。如图28-10所示，如果出现“'MySQL' 不是内部或外部命令，也不是可运行的程序或批处理文件。”的错误，则说明在环境变量的Path没有配置MySQL的Path。参考2.1.2追加C:\Program Files\MySQL\MySQL Server 5.7\bin到环境变量Path之后。

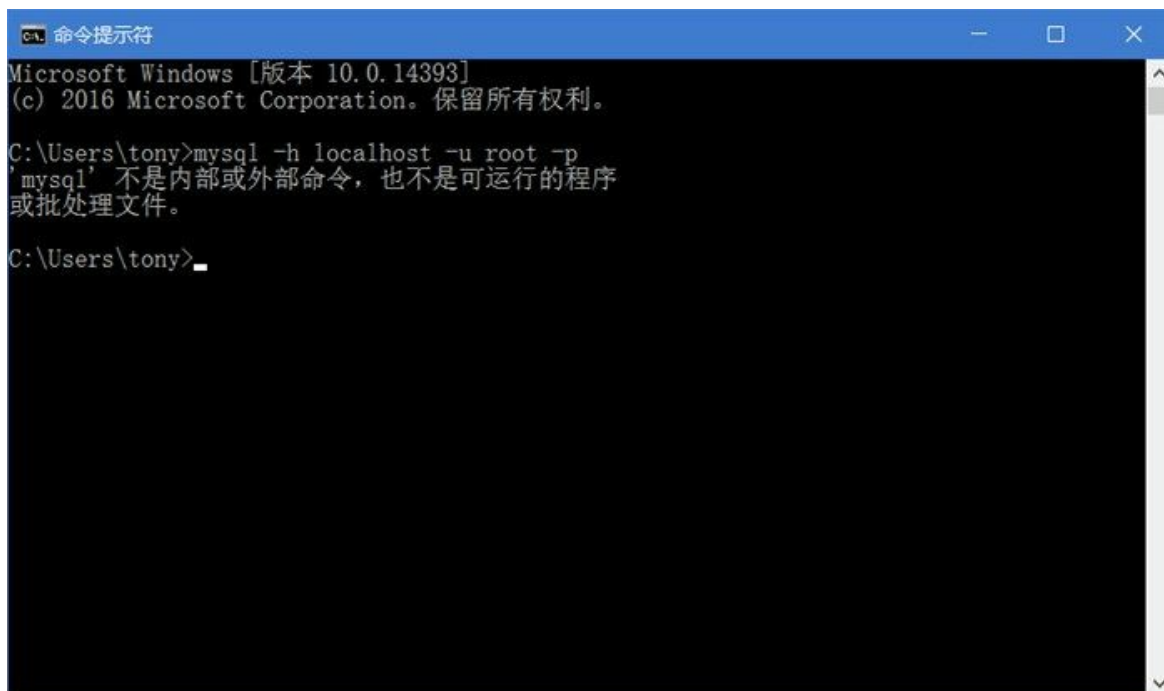


图28-10 环境变量中没有MySQL的Path

如果Path环境变量添加成功，重新打开命令行，再次输入mysql -h localhost -u root -p命令，然后系统会提示你输入root密码，输入密码按下Enter键，如果密码正确成功连接到服务器，会看到如图28-9所示的界面。

提示： mysql -h localhost -u root -p命令,参数说明：

-h: 要连接的服务器主机名或IP地址，可以是远程的一个服务器主机，也可以是-hlocalhost方式没有空格。

-u: 是服务器要验证的用户名，这个用户一定是数据库中存在的，并且具有连接服务器的权限，也可以是-uroot方式没有空格。

-p: 是与上面用户对应的密码，也可以直接输入密码-p123456，123456是root密码。

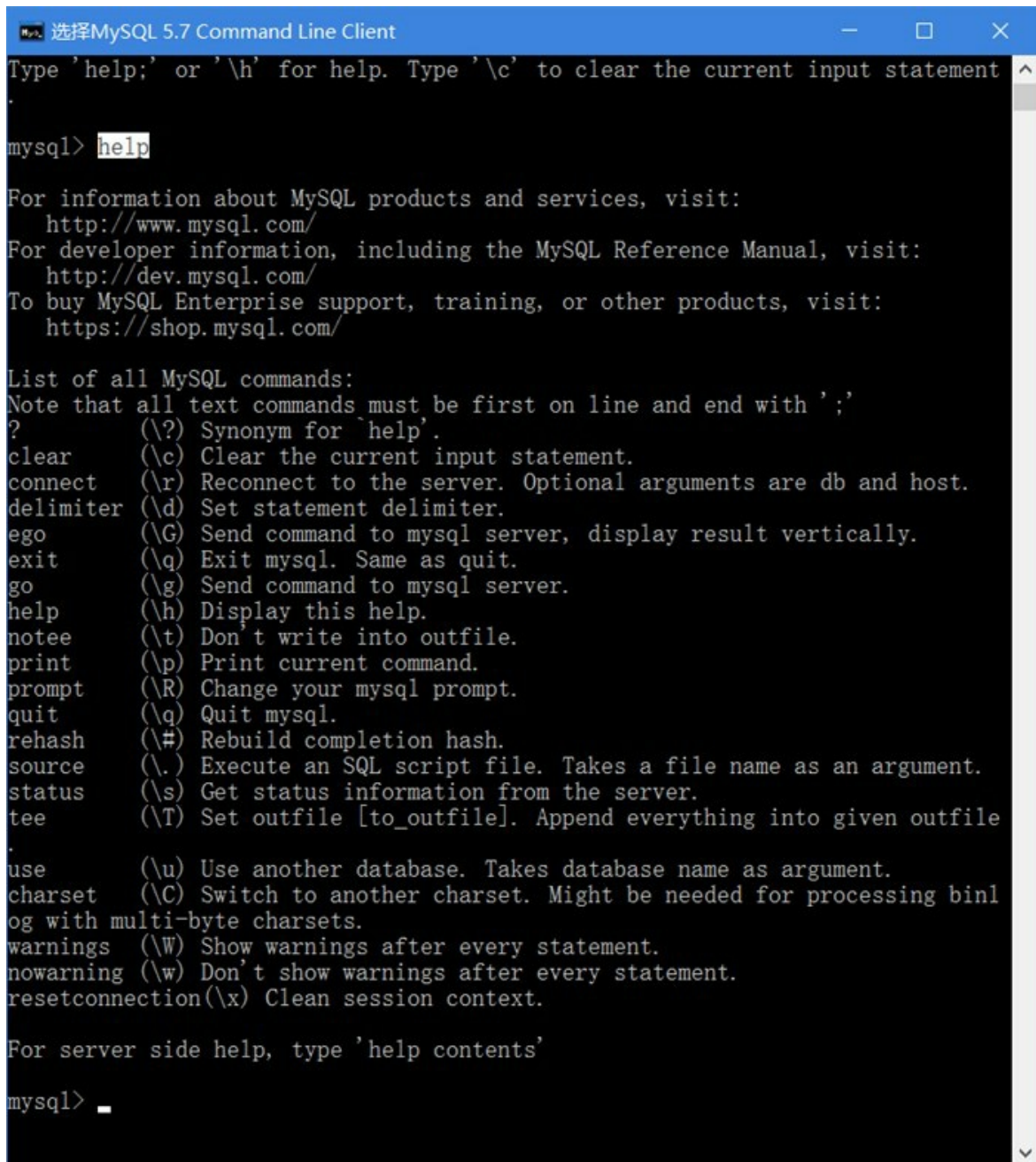
所以mysql -h localhost -u root -p命令也可以替换为mysql -hlocalhost -uroot -p123456。

### 28.2.3 常见的管理命令

通过命令行客户端管理MySQL数据库，需要了解一些常用的命令。

#### 01. help

第一个应该熟悉的的就是help命令，help命令能够列出MySQL其他命令的帮助，在命令行客户端中输入help，不需要分号结尾，直接按下Enter键，如图28-11所示，这里都是MySQL的管理命令，这些命令大部分不需要分号结尾。



```
选择MySQL 5.7 Command Line Client
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.
mysql> help

For information about MySQL products and services, visit:
  http://www.mysql.com/
For developer information, including the MySQL Reference Manual, visit:
  http://dev.mysql.com/
To buy MySQL Enterprise support, training, or other products, visit:
  https://shop.mysql.com/

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?      (\?) Synonym for 'help'.
clear  (\c) Clear the current input statement.
connect (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
ego    (\G) Send command to mysql server, display result vertically.
exit   (\q) Exit mysql. Same as quit.
go     (\g) Send command to mysql server.
help   (\h) Display this help.
notee  (\t) Don't write into outfile.
print  (\p) Print current command.
prompt (\R) Change your mysql prompt.
quit   (\q) Quit mysql.
rehash (\#) Rebuild completion hash.
source (\.) Execute an SQL script file. Takes a file name as an argument.
status (\s) Get status information from the server.
tee     (\T) Set outfile [to_outfile]. Append everything into given outfile
.
use     (\u) Use another database. Takes database name as argument.
charset (\C) Switch to another charset. Might be needed for processing binl
og with multi-byte charsets.
warnings (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
resetconnection (\x) Clean session context.

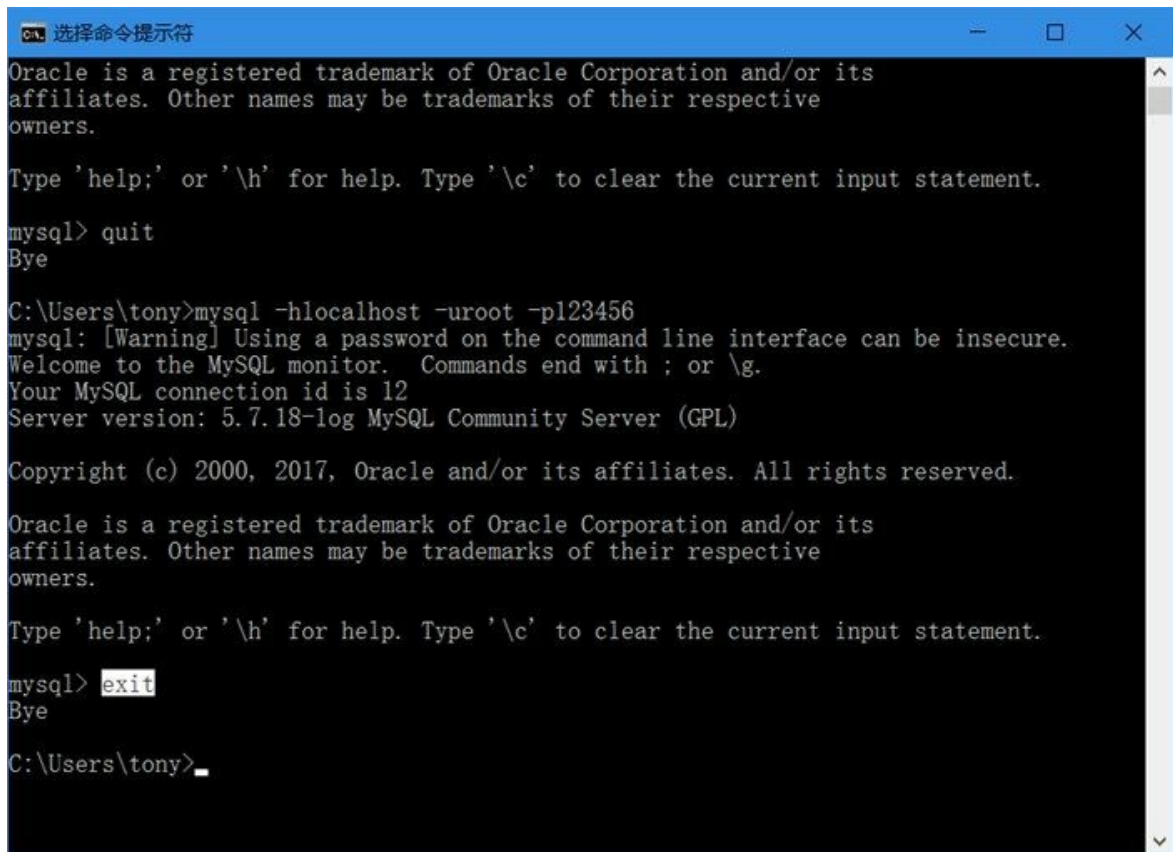
For server side help, type 'help contents'

mysql> _
```

图28-11 使用help命令

## 02. 退出命令

如果命令行客户端中退出，可以在命令行客户端中使用quit或exit命令，如图28-12所示。注意这两个命令也不需要分号结尾。



```
C:\Users\tony>mysql -hlocalhost -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> quit
Bye

C:\Users\tony>mysql -hlocalhost -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye

C:\Users\tony>
```

图28-12 使用退出命令

### 03. 数据库管理

在使用数据库的过程中，有时需要知道服务器中哪些数据库或自己创建和删除数据库。查看数据库的命令是show databases;，如图28-13所示，注意该命令后面是有分号结尾的。

```
命令提示符 - mysql -hlocalhost -uroot -p123456
C:\Users\tony>mysql -hlocalhost -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql       |
| performance_schema |
| sakila      |
| sys         |
| world       |
+-----+
6 rows in set (0.00 sec)

mysql> _
```

图28-13 查看数据库信息

创建数据库可以使用`create database testdb;`命令，如图28-14所示，`testdb`是自定义数据库名，注意该命令后面是有分号结尾的。



```
选择命令提示符 - mysql -hlocalhost -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database testdb;
Query OK, 1 row affected (0.00 sec)

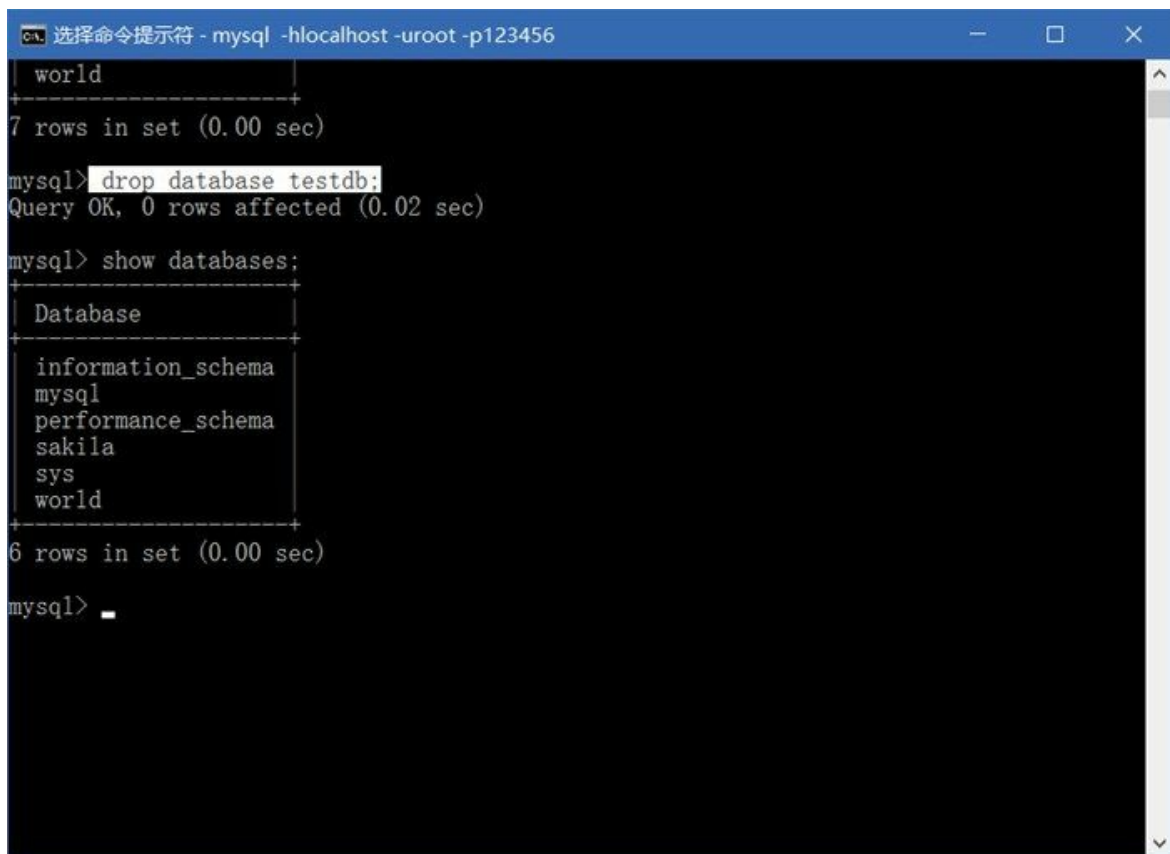
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sakila         |
| sys           |
| testdb        |
| world         |
+-----+
7 rows in set (0.00 sec)

mysql> _
```

图28-14 创建数据库

想要删除数据库可以使用`database testdb;`命令，如图28-15所示，`testdb`是自定义数据库名，注意该命令后面是有分号结尾的。





```
选择命令提示符 - mysql -hlocalhost -uroot -p123456
+-----+
| world |
+-----+
7 rows in set (0.00 sec)

mysql> drop database testdb;
Query OK, 0 rows affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
6 rows in set (0.00 sec)

mysql> _
```

图28-15 删除数据库

#### 04. 数据表管理

在使用数据库的过程中，有时需要知道某个数据库下有多少个数据表，并想查看表结构等信息。

查看有多少个数据表的命令是show tables;，如图28-16所示，注意该命令后面是有分号结尾的。因为一个服务器中有很多数据库，应该先使用use 选择数据库，如图28-16所示，use world命令结尾没有分号。如果没有选择数据库，会发生错误，如图28-16所示。

```
选择命令提示符 - mysql -hlocalhost -uroot -p123456

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> use world
Database changed
mysql> show tables;
+-----+
| Tables_in_world |
+-----+
| city             |
| country          |
| countrylanguage  |
+-----+
3 rows in set (0.00 sec)

mysql> _
```

图28-16 查看数据库中表信息

知道了有哪些表后，还需要知道表结构，可以使用desc命令，例如像知道city表结构可以使用命令desc city;命令，如图28-17所示，注意该命令后面是有分号结尾的。

```
选择命令提示符 - mysql -hlocalhost -uroot -p123456

+-----+
| Tables_in_world |
+-----+
| city             |
| country          |
| countrylanguage  |
+-----+
3 rows in set (0.00 sec)

mysql> desc city;
+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+
| ID         | int(11)   | NO   | PRI | NULL    | auto_increment |
| Name       | char(35)  | NO   |     |          |                |
| CountryCode | char(3)   | NO   | MUL |          |                |
| District   | char(20)  | NO   |     |          |                |
| Population | int(11)   | NO   |     | 0        |                |
+-----+
5 rows in set (0.00 sec)

mysql> _
```

图28-17 查看表结构

## 28.3 JDBC技术

Java中数据库编程是通过JDBC（Java Database Connectivity）实现的。使用JDBC技术涉及到三种不同的角色：Java官方、开发人员和数据库厂商。

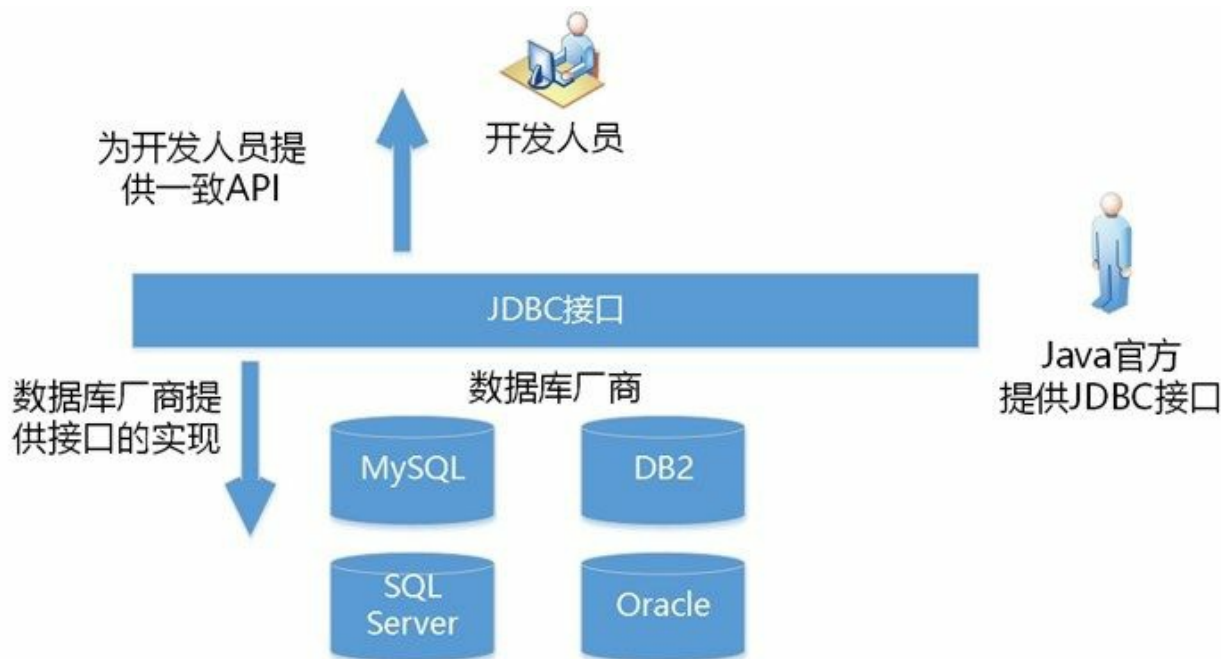


图28-18 JDBC技术涉及到三种不同的角色

- Java官方提供JDBC接口，如Connection、Statement和ResultSet等。
- 数据库厂商为了支持Java语言使用自己的数据库，他们根据这些接口提供了具体的实现类，这些具体实现类称为JDBC Driver（JDBC驱动程序），例如Connection是数据库连接接口，如何能够高效地连接数据库或许只有数据库厂商自己清楚，因此他们提供的JDBC驱动程序当然是最高效的，当然针对某种数据库也可能有其他第三方JDBC驱动程序。
- 对于开发人员而言，JDBC提供了一致的API，开发人员不用关心实现接口的细节。

### 28.3.1 JDBC API

JDBC API为Java开发者使用数据库提供了统一的编程接口，它由一组Java类和接口组成。这种类和接口来自于java.sql和javax.sql两个包。

- **java.sql:** 这个包中的类和接口主要针对基本的数据库编程服务，如创建连接、执行语句、语句预编译和批处理查询等。同时也有一些高级的处理，如批处理更新、事务隔离和可滚动结果集等。
- **javax.sql:** 它主要为数据库方面的高级操作提供了接口和类，提供分布式事务、连接池和行集等。

### 28.3.2 加载驱动程序

在编程实现数据库连接时，JVM必须先加载特定厂商提供的数据库驱动程序。使用Class.forName()方

法实现驱动程序加载过程，该方法在26章介绍过。

不同驱动程序的装载方法如下：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");    //JDBC-ODBC桥接，Java自带  
Class.forName("特定的JDBC驱动程序类名");        //数据库厂商提供
```

例如加载MySQL驱动程序代码如下：

```
Class.forName("com.mysql.jdbc.Driver");
```

如果直接这样运行程序会抛出如下的ClassNotFoundException异常。

```
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
```

这是因为程序无法找到MySQL驱动程序com.mysql.jdbc.Driver类，这需要配置当前项目的类路径（Classpath），在类路径中包含MySQL驱动程序。MySQL驱动程序是在MySQL安装目录中Connector.J 5.1中的mysql-connector-java-xxx-bin.jar文件。笔者默认安装驱动程序文件路径如下：

```
"C:\Program Files (x86)\MySQL\Connector.J 5.1\mysql-connector-java-5.1.41-bin.jar"
```

数据库厂商提供驱动程序一般都是.jar文件。

提示 一般在发布java文件时，会把字节码文件（class文件）打包成.jar文件，.jar文件是一种基于.zip结构的压缩文件。

在Eclipse中将.jar文件添加到项目步骤，首先在Eclipse中选择项目，右击菜单中选择“构建路径”→“配置构建路径”，在弹出对话框中选择“Java构建路径”→“库”如图28-19所示，其中“添加JAR”是在当前项目中找.jar文件，“添加外部JAR”是在项目之外找.jar文件。

提示 笔者推荐前一种方式（添加JAR），它的好处在于当你把Eclipse项目复制给别人时，这些.jar文件也一起复制，项目重新编译和运行时，不会发生找不到.jar文件情况。而添加外部JAR方法，只是添加了一个基于本机的绝对路径，当你的项目复制给别人时，会发生找不到.jar文件情况。



图28-19 构建路径对话框

在图28-19对话框中如果单击“添加JAR”按钮，则弹出如图28-20所示的对话框。该对话框只能选择该项目内.jar文件，所以要保证所添加的.jar文件应该在Eclipse项目中,如果文件存在，选择.jar文件单击“确定”按钮添加。



图28-20 添加jar文件到项目类路径

如果文件在Eclipse中没有，需要从操作系统的资源管理器中复制到Eclipse中，参考图28-21，将文件复制到Eclipse中。

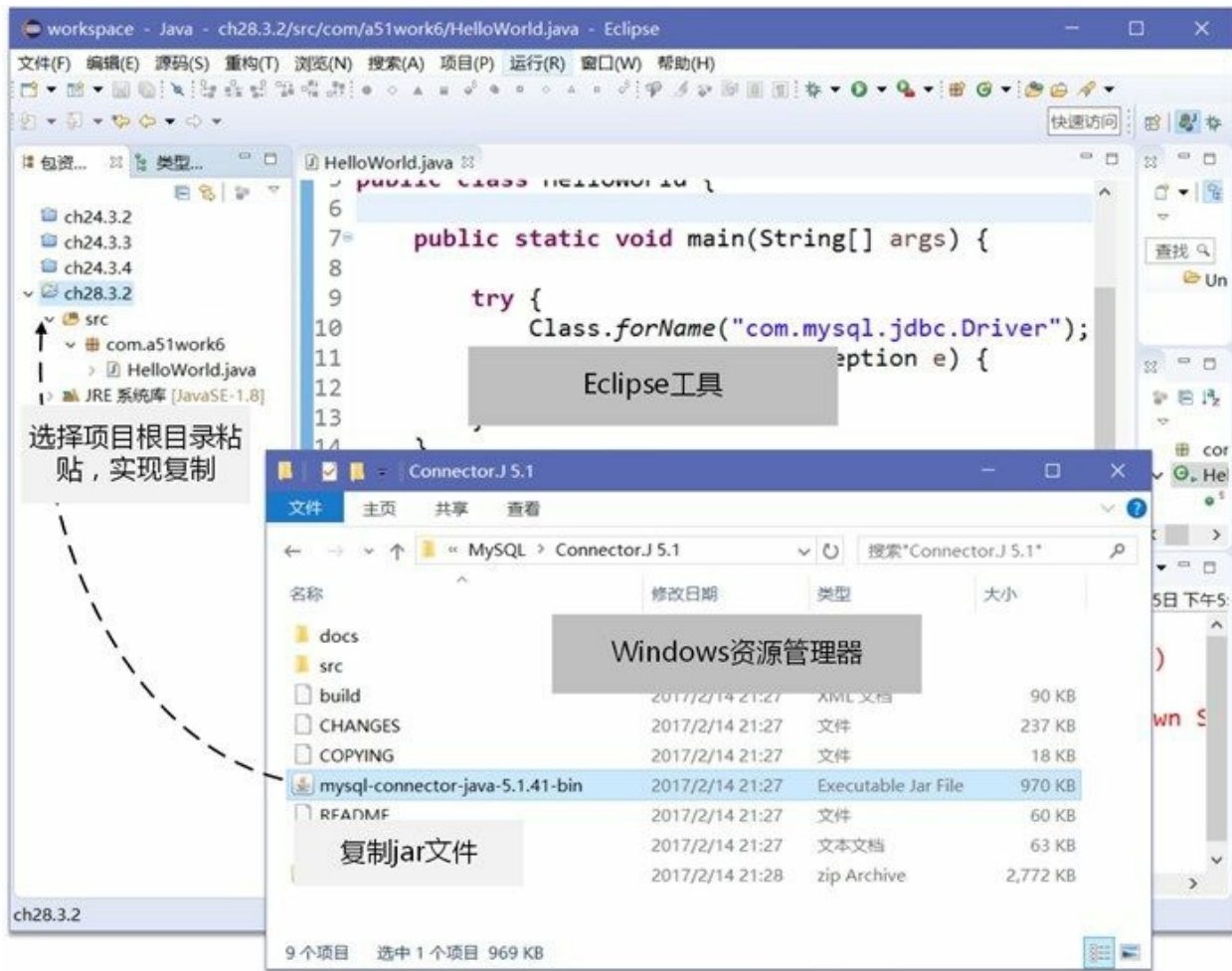


图28-21 从资源管理器复制文件到Eclipse

将驱动程序.jar文件添加类路径中后，再运行上面的程序看看是否还有ClassNotFoundException异常。

### 28.3.3 建立数据连接

驱动程序加载成功就可以进行数据库连接了。建立数据库连接可以通过调用DriverManager类的getConnection()方法实现，该方法有几个重载版本，如下所示。

- static Connection getConnection(String url): 尝试通过一个URL建立数据库连接，调用此方法时，DriverManager会试图从已注册的驱动中选择恰当的驱动来建立连接。
- static Connection getConnection(String url, Properties info): 尝试通过一个URL建立数据库连接，一些连接参数（如user和password）可以按照键值对的形式放置到info中，Properties是Hashtable的子类，它是一种Map结构。
- static Connection getConnection(String url, String user, String password): 尝试通过一个URL建立数据库连接，指定数据库用户名和密码。

上面的几个getConnection()方法都会抛出受检查的SQLException异常，注意处理这个异常。

JDBC的URL类似于其他场合的URL，它的语法如下：



```
jdbc:<subprotocol>:<subname>
```

这里有三个部分，它们用冒号隔离。

01. 协议：jdbc表示协议，它是唯一的，JDBC只有这一种协议。
02. 子协议：主要用于识别数据库驱动程序，也就是说，不同的数据库驱动程序的子协议不同。
03. 子名：它属于专门的驱动程序，不同的专有驱动程序可以采用不同的实现。

对于不同的数据库，厂商提供的驱动程序和连接的URL都不同，在这里总结后如表28-1所示。

表 28-1 数据库厂商提供的驱动程序和连接的URL

数据库名	驱动程序	URL
MS SQLServer	com.microsoft.jdbc.sqlserver.SQLServerDriver	jdbc:microsoft:sqlserver://[ip]:[port];user=[user];password=[password]
JDBC-ODBC	sun.jdbc.odbc.JdbcOdbcDriver	jdbc:odbc:[odbcsource]
Oracle thin Driver	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@[ip]:[port]:[sid]
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://ip/database

建立数据连接示例代码如下：

```
//HelloWorld.java文件
package com.a51work6;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class HelloWorld {

    public static void main(String[] args) {

        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("驱动程序加载成功...");
        } catch (ClassNotFoundException e) {
            System.out.println("驱动程序加载失败...");
            // 退出
            return;
        }

        String url = "jdbc:mysql://localhost:3306/MyDB";
        String user = "root";
        String password = "123456";

        try (Connection conn = DriverManager.getConnection(url, user, password)) { ①

            System.out.println("数据库连接成功: " + conn);

        } catch (SQLException e) {
            e.printStackTrace();
        }

    }

}
```

```
}
```

上述代码第①行使用DriverManager的getConnection(String url, String user, String password)方法建立数据库连接，在url中3306是数据库端口号，MyDB是MySQL服务器中的数据库。

另外Connection对象是通过自动资源管理技术释放资源的。

注意 Connection对象代表的数据库连接不能被JVM的垃圾收集器回收，在使用完连接后必须关闭（调用close()方法），否则连接会保持一段比较长的时间，直到超时。Java 7之前都在finally模块中关闭数据库连接。Java 7之后可以Connection接口继承了AutoCloseable接口，可以通过自动资源管理技术释放资源。

事实上上面代码虽然可以成功建立连接，但是控制台会有警告，警告如下：

```
WARN: Establishing SSL connection without server's identity verification is not recommended. Acco
```

这是由于现在的网络通信为了提高网络完全，都要求使用SSL<sup>4</sup>(Secure Sockets Layer 安全套接层) 安全协议。但是由于各种原因，很多服务器并未使用SSL安全协议，特别是对于学习和测试阶段可以不使用SSL安全协议。为此，需要修改url连接字符串：

<sup>4</sup>SSL为网络通信提供安全及数据完整性的一种安全协议，SSL在传输层对网络连接进行加密。

```
"jdbc:mysql://localhost:3306/MyDB?verifyServerCertificate=false&useSSL=false"
```

其中verifyServerCertificate设置为false表示不进行安全认证，useSSL设置为false表示不使用SSL进行网络通信。

数据库连接的url字符串可以有很多参数对，包括数据库用户名和密码也都可以参数对形式放到url字符串中，有的url字符串会很长维护起来不方便，可以把这些参数对放置到Properties对象中，示例代码如下：

```
//HelloWorldWithProp.java文件
package com.a51work6;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class HelloWorldWithProp {

    public static void main(String[] args) {

        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("驱动程序加载成功...");

        } catch (ClassNotFoundException e) {
            System.out.println("驱动程序加载失败...");
            // 退出
            return;
        }

        String url = "jdbc:mysql://localhost:3306/MyDB";
        //创建Properties对象
```

```

        Properties info = new Properties();           ①
        info.setProperty("user", "root");           ②
        info.setProperty("password", "123456");      ③
        info.setProperty("verifyServerCertificate", "false"); ④
        info.setProperty("useSSL", "false");         ⑤

        try (Connection conn = DriverManager.getConnection(url, info)) {    ⑥

            System.out.println("数据库连接成功: " + conn);

        } catch (SQLException e) {
            e.printStackTrace();
        }

    }
}

```

上述代码第①行是创建Properties对象，代码第②行~第⑤行是设置参数，setProperty()方法键和值都是字符串类型。代码第⑥行是DriverManager的getConnection(String url, Properties info)方法建立数据连接。

但是上述代码还是有不尽人意的地方，就是这些参数都是“硬编码<sup>5</sup>”在程序代码中的，程序编译之后不能修改。但是数据库用户名、密码、服务器主机名、端口等等这一切，在开发阶段和部署阶段可能完全不同，这些参数信息应该可以配置的，可以放到一个属性文件中，借助于输入流，可以在运行时读取属性文件内容到Properties对象中。具体示例代码如下：

<sup>5</sup>硬编码俗称“写死”，指将可变变量用一个固定值来代替的方法，用这种方法编译后，如果以后需要更改此变量就非常困难了。

```

//HelloWorldWithPropFile.java文件
package com.a51work6;

...

public class HelloWorldWithPropFile {

    public static void main(String[] args) {

        //加载驱动程序
        ...

        Properties info = new Properties();           ①
        try {
            InputStream input = HelloWorldWithPropFile.class.getClassLoader()
                .getResourceAsStream("config.properties"); ②

            info.load(input);                           ③

        } catch (IOException e) {
            // 退出
            return;
        }

        String url = "jdbc:mysql://localhost:3306/MyDB";

        try (Connection conn = DriverManager.getConnection(url, info)) {

            System.out.println("数据库连接成功: " + conn);

        } catch (SQLException e) {
            e.printStackTrace();
        }

    }
}

```

```
}  
}
```

上述代码第①行创建一个Properties对象。代码第②行获得config.properties属性文件输入流对象，属性文件一般放到src目录与源代码文件放置在一起，但是编译时，这些文件会复制到字节码文件所在的目录中，这种目录称为资源目录，获得资源目录要通过Java反射机制，HelloWorldWithPropFile.class.getClassLoader().getResourceAsStream("config.properties")语句能够获得运行时config.properties的文件输入流对象。

代码第③行是从流中加载信息到Properties对象中。

config.properties文件内容如下：

```
#Tony  
user=root  
password=123456  
useSSL=false  
verifyServerCertificate=false
```

在开发和部署阶段使用文本编辑器修改该文件，不需要修改程序代码。

## 28.3.4 三个重要接口

下面重点介绍一下JDBC API中最重要的三个接口：Connection、Statement和ResultSet。

### 01. Connection接口

java.sql.Connection接口的实现对象代表与数据库的连接，也就是在Java程序和数据库之间建立连接。Connection接口中常用的方法：

- `Statement createStatement()`: 创建一个语句对象，语句对象用来将SQL语句发送到数据库。
- `PreparedStatement prepareStatement(String sql)`: 创建一个预编译的语句对象，用来将参数化的SQL语句发送到数据库，参数包含一个或者多个问号“?”占位符。
- `CallableStatement prepareCall(String sql)`: 创建一个调用存储过程的语句对象，参数是调用的存储过程，参数包含一个或者多个问号“?”为占位符。
- `close()`: 关闭到数据库的连接，在使用完连接后必须关闭，否则连接会保持一段比较长的时间，直到超时。
- `isClosed()`: 判断连接是否已经关闭。

### 02. Statement接口

java.sql.Statement称为语句对象，它提供用于向数据库发出SQL语句，并且访问结果。Connection接口提供了生成Statement的方法，一般情况下可以通过`connection.createStatement()`方法就可以得到Statement对象。

有三种Statement接口：java.sql.Statement、java.sql.PreparedStatement和java.sql.CallableStatement，其中PreparedStatement继承Statement接口，CallableStatement继承PreparedStatement接口。Statement实现对象用于执行基本的SQL语句，PreparedStatement实现对象用于执行预编译的SQL语句，CallableStatement实现对象用于用来调用数据库中的存储过程。

注意 预编译SQL语句是在程序编译的时一起进行编译，这样的语句在数据库中执行时候，不需要编译过程，直接执行SQL语句，所以速度很快。在预编译SQL语句会有一些程序执行时才能确定的参数，这些参数采用“?”占位符，直到运行时再用实际参数替换。

Statement提供了许多方法，最常用的方法如下：

- `executeQuery()`: 运行查询语句，返回ResultSet对象。
- `executeUpdate()`: 运行更新操作，返回更新的行数。
- `close()`: 关闭语句对象。
- `isClosed()`: 判断语句对象是否已经关闭。

Statement对象用于执行不带参数的简单SQL语句，它的典型使用如下。

```
Connection conn = DriverManager.getConnection("jdbc:odbc:accessdb", "admin", "admin");
Statement stmt = conn.createStatement();
ResultSet rst = stmt.executeQuery("select userid, name from user");
```

PreparedStatement对象用于执行带参数的预编译SQL语句，它的典型使用如下。

```
Connection conn = DriverManager.getConnection("jdbc:odbc:accessdb", "admin", "admin");
PreparedStatement pstmt = conn.prepareStatement("insert into user values(?, ?)");
pstmt.setInt(1,10);           //绑定第一个参数
pstmt.setString(2,"guan");    //绑定第二个参数
pstmt.executeUpdate();        //执行SQL语句
```

上述SQL语句"insert into user values(?, ?)"在Java源程序编译时一起编译，两个问号占位符所代表的参数，在运行时绑定。

注意 绑定参数时需要注意两个问题：绑定参数顺序和绑定参数的类型，绑定参数索引是从1开始的，而不是从0开始的。根据绑定参数的类型不同选择对应的set方法。

CallableStatement对象用于执行对数据库已存储过程的调用，它的典型使用如下。

```
Connection conn = DriverManager.getConnection("jdbc:odbc:accessdb", "admin", "admin");
strSQL = "{call proc_userinfo(?, ?)}";
java.sql.CallableStatement sqlStmt = conn.prepareCall(strSQL);
sqlStmt.setString(1, "tony");
sqlStmt.setString(2, "tom");
//执行存储过程
int i = sqlStmt.executeUpdate();
```

### 03. ResultSet接口

在Statement执行SQL语句时，如果是SELET语句会返回结果集，结果集通过接口java.sql.ResultSet描述的，它提供了逐行访问结果集的方法，通过该方法能够访问结果集中不同字段的内容。

ResultSet提供了检索不同类型字段的方法，最常用的方法介绍如下：

- close(): 关闭结果集对象。
- isClosed(): 判断结果集对象是否已经关闭。
- next(): 将结果集的光标从当前位置向后移一行。
- getString(): 获得在数据库里是CHAR 或 VARCHAR等字符串类型的数据，返回值类型是String。
- getFloat(): 获得在数据库里是浮点类型的数据，返回值类型是float。
- getDouble(): 获得在数据库里是浮点类型的数据，返回值类型是double。
- getDate(): 获得在数据库里是日期类型的数据，返回值类型是java.sql.Date。
- getBoolean(): 获得在数据库里是布尔数据的类型，返回值类型是boolean。
- getBlob(): 获得在数据库里是Blob(二进制大型对象)类型的数据，返回值类型是Blob类型。
- getClob(): 获得在数据库里是Clob(字符串大型对象)类型的数据，返回值类型是Clob。

这些方法要求有列名或者列索引，如getString()方法的两种情况：

`public String getString(int columnIndex) throws SQLException`

`public String getString(String columnName) throws SQLException`

方法`getXXX`提供了获取当前行中某列值的途径，在每一行内，可按任何次序获取列值。使用列索引有时会比较麻烦，这个顺序是`select`语句中的顺序：

```
select * from user
select userid, name from user
select name,userid from user
```

注意 `columnIndex`列索引是从1开始的，而不是从0开始的。这个顺序与`select`语句有关，如果`select`使用`*`返回所有字段，如`select * from user`语句，那么列索引是数据表中字段的顺序；如果`select`指定具体字段，如`select userid, name from user`或`select name,userid from user`，那么列索引是`select`指定字段的顺序。

ResultSet示例代码如下：

```
//HelloWorldWithPropFile.java文件
...
String url = "jdbc:mysql://localhost:3306/MyDB";

try ( // 自动资源管理技术释放资源
    Connection conn = DriverManager.getConnection(url, info);
    Statement stmt = conn.createStatement();
    ResultSet rst = stmt.executeQuery("select name,userid from user")) {

    while (rst.next()) {
        System.out.printf("name:%s    id:%d\n", rst.getString("name"), rst.getInt(2));
    }

} catch (SQLException e) {
    e.printStackTrace();
}
```

上述代码可见`Connection`对象、`Statement`对象和`ResultSet`对象的释放采用自动资源管理。

在遍历结果集时使用了`rst.next()`方法，`next()`是将结果集光标从当前位置向后移一行，结果集光标最初位于第一行之前；第一次调用 `next` 方法使第一行成为当前行；第二次调用使第二行成为当前行，依此类推。如果新的当前行有效，则返回 `true`；如果不存在下一行，则返回`false`。

注意 `Connection`对象、`Statement`对象和`ResultSet`对象都不能被JVM的垃圾收集器回收，在使用完后都必须关闭（调用它们的`close()`方法）。Java 7之前都在`finally`模块中关闭释放资源。Java 7之后它们都继承了`AutoCloseable`接口，可以通过自动资源管理技术释放资源。

## 28.4 案例：数据CRUD操作

对数据库表中数据可以进行4类操作：数据插入（Create）、数据查询（Read）、数据更新（Update）和数据删除（Delete），也是俗称的“增、删、改、查”。

本节通过一个案例介绍如何通过JDBC技术实现Java对数据的CRUD操作。

### 28.4.1 数据库编程一般过程

在讲解案例之前，有必要先介绍一下通过JDBC进行数据库编程的一般过程。

如图28-22所示是数据库编程的一般过程，其中查询（Read）过程最多需要7个步骤，修改（C插入、U更新、D删除）过程最多需要5个步骤。这个过程采用了预编译语句对象进行数据操作，所以有可能进行绑定参数，见第4步骤。

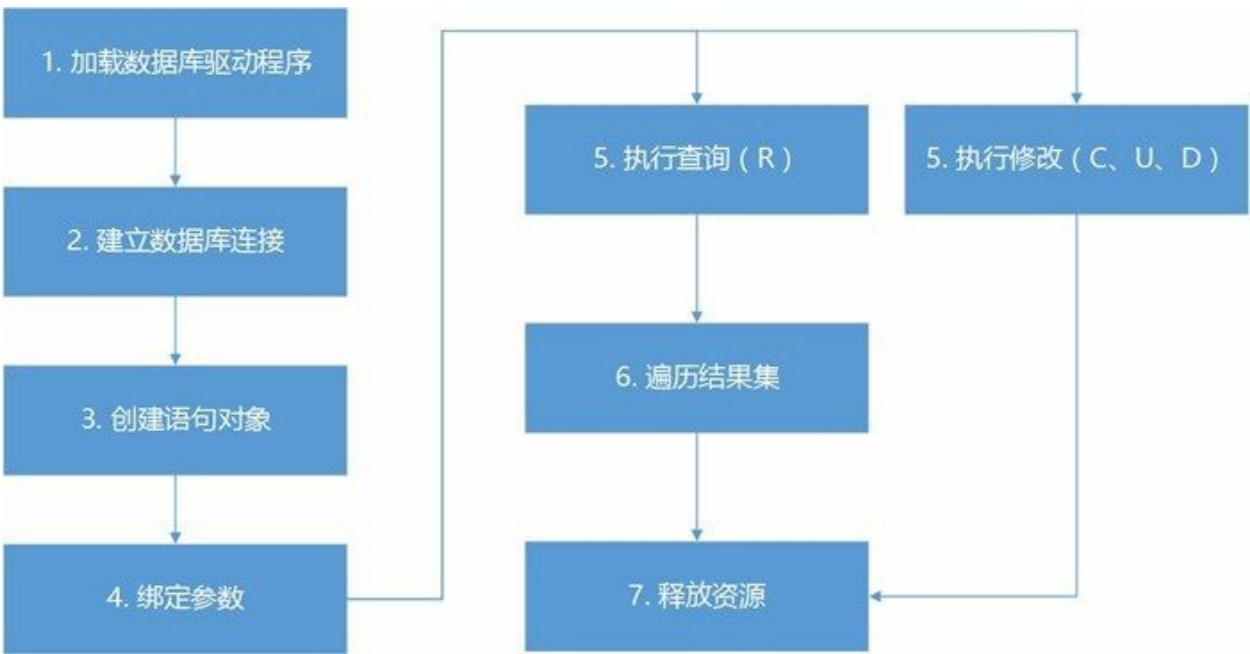


图28-22 数据库编程的一般过程

上述步骤是基本的一般步骤，实际情况会有所变化，例如没有参数需要绑定，则第4步骤就省略了。还有，如果Connection对象、Statement对象和ResultSet对象都采用自动资源管理技术释放资源，那么第7步骤也可以省略。

### 28.4.2 数据查询操作

为了介绍数据查询操作案例，这里准备了一个User表，它有两个字段name和userid，如表28-2所示。

表 28-2 User表结构



字段名	类型	是否可以为 Null	主键
name	varchar(20)	是	否
userid	int	否	是

下面介绍实现如下两条SQL语句查询功能：

```
select name,userid from user where userid > ? order by userid    //有条件查询
select max(userid) from user                                     //使用max等函数，无条件查询
```

01. 有条件查询实现代码如下：

```
//CRUDSample.java文件
package com.a51work6;

import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

public class CRUDSample {

    // 连接数据库url
    static String url;
    // 创建Properties对象
    static Properties info = new Properties();

    // 1.驱动程序加载
    static {
        // 获得属性文件输入流
        InputStream input
            = CRUDSample.class.getClassLoader().getResourceAsStream("config.properties");

        try {
            // 加载属性文件内容到Properties对象
            info.load(input);
            // 从属性文件中取出url
            url = info.getProperty("url");
            // Class.forName("com.mysql.jdbc.Driver");
            // 从属性文件中取出driver
            String driverClassName = info.getProperty("driver");
            Class.forName(driverClassName);
            System.out.println("驱动程序加载成功...");
        } catch (ClassNotFoundException e) {
            System.out.println("驱动程序加载失败...");
        } catch (IOException e) {
            System.out.println("加载属性文件失败...");
        }
    }

    public static void main(String[] args) {

        // 查询数据
        read();
    }
}
```

```

}

// 数据查询操作
public static void read() {

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        // 2. 创建数据库连接
        conn = DriverManager.getConnection(url, info);
        // 3. 创建语句对象
        pstmt = conn.prepareStatement("select name,userid from "
            + "user where userid > ? order by userid");
        // 4. 绑定参数
        pstmt.setInt(1, 0);
        // 5. 执行查询 (R)
        rs = pstmt.executeQuery();
        // 6. 遍历结果集
        while (rs.next()) {
            System.out.printf("id: %d      name: %s\n", rs.getInt(2), rs.getString("name"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // 7. 释放资源
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
            }
        }
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
            }
        }
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
            }
        }
    }
}
}

```

上述代码第①行~第②行是静态代码块，在静态代码块中读取属性文件内容到Properties对象和加载驱动程序，这两个操作只需执行一次，所以它们最好放到静态代码块中。另外，需要注意本例中将驱动程序类名和数据库连接的url字符串都放到属性文件中config.properties，这样一来更加方便配置，config.properties内容如下：

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/MyDB
user=root
password=123456
useSSL=false
verifyServerCertificate=false
```

上述代码第③行read()方法是数据查询方法，查询完成之后采用finally代码块释放资源，见代码第④行~第⑤行。本例也可以使用自动资源管理技术，但会引起try语句发生嵌套，反而会有些麻烦。

## 02. 无条件查询实现代码如下：

```
// 1.驱动程序加载
static {
    ...
}
...

// 查询最大的用户Id
public static int readMaxUserId() {

    int maxId = 0;
    try (
        // 2.创建数据库连接
        Connection conn = DriverManager.getConnection(url, info);
        // 3. 创建语句对象
        PreparedStatement pstmt = conn.prepareStatement("select max(userid) from user");
        // 4. 绑定参数
        // pstmt.setInt(1, 0);
        // 5. 执行查询 (R)
        ResultSet rs = pstmt.executeQuery()) {
        // 6. 遍历结果集
        if (rs.next()) {
            maxId = rs.getInt(1);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return maxId;
}
```

上述代码使用了自动资源管理技术，由于没有参数需要绑定，所以ResultSet对象可以与Connection对象和PreparedStatement对象放在一个try代码块中进行管理。而前面的有条件查询read()方法则不行。

### 28.4.3 数据修改操作

数据修改操作包括了：数据插入、数据更新和数据删除。

#### 01. 数据插入

数据插入代码如下：

```
// 数据插入操作
public static void create() {

    try ( // 2.创建数据库连接
        Connection conn = DriverManager.getConnection(url, info);
        // 3. 创建语句对象
        PreparedStatement pstmt
            = conn.prepareStatement("insert into user (userid, name) values (?,?)") { ①

        // 查询最大值
        int maxId = readMaxUserId();
```

```

        // 4. 绑定参数
        pstmt.setInt(1, ++maxId);           ②
        pstmt.setString(2, "Tony" + maxId); ③
        // 5. 执行修改 (C、U、D)
        int affectedRows = pstmt.executeUpdate(); ④

        System.out.printf("成功插入%d条数据。\\n", affectedRows);

    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

代码第①行是创建插入语句对象，其中有两个占位符。因此需要绑定参数，代码第②行是绑定第一个参数，代码第③行是绑定第二个参数。代码第④行executeUpdate()方法执行SQL语句，该方法与查询方法executeQuery()不同。executeUpdate()方法返回的是整数，成功影响的记录数，即成功插入记录数。

## 02. 数据更新

数据更新代码如下：

```

// 数据更新操作
public static void update() {

    try ( // 2.创建数据库连接
        Connection conn = DriverManager.getConnection(url, info);
        // 3. 创建语句对象
        PreparedStatement pstmt
            = conn.prepareStatement("update user set name = ? where userid > ?")) {

        // 4. 绑定参数
        pstmt.setString(1, "Tom");
        pstmt.setInt(2, 30);
        // 5. 执行修改 (C、U、D)
        int affectedRows = pstmt.executeUpdate();

        System.out.printf("成功更新%d条数据。\\n", affectedRows);

    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

## 03. 数据删除

数据删除代码如下：

```

// 数据删除操作
public static void delete() {

    try ( // 2.创建数据库连接
        Connection conn = DriverManager.getConnection(url, info);
        // 3. 创建语句对象
        PreparedStatement pstmt = conn.prepareStatement("delete from user where userid = ?"))

        // 查询最大值
        int maxId = readMaxUserId();

        // 4. 绑定参数
        pstmt.setInt(1, maxId);

```

```
        // 5. 执行修改 (C、U、D)
        int affectedRows = pstmt.executeUpdate();

        System.out.printf("成功删除%d条数据。\\n", affectedRows);

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

数据更新、数据删除与数据插入程序结构上非常类似，差别主要在于SQL语句的不同，绑定参数的不同。具体代码不再解释。

## 本章小结

本章首先介绍MySQL数据库的安装、配置和日常的管理命令，然后重点讲解了JDBC数据库编程技术。读者需要重点掌握三个主要接口，熟悉一般数据库编程过程。