第 5 章　GaussDB 数据库编程高级

本章以第 3、4 章为基础，介绍 Java 程序访问 GaussDB 以及 Python 程序访问 GaussDB，非常详细地介绍从开发环境的搭建到程序的运行的步骤。

5.1　安装及配置 JDK

安装及配置 JDK 分为下载并安装 JDK 和配置 JDK 两个步骤。

(1) 进入 Oracle 官网下载 JDK，如图 5.1 所示。

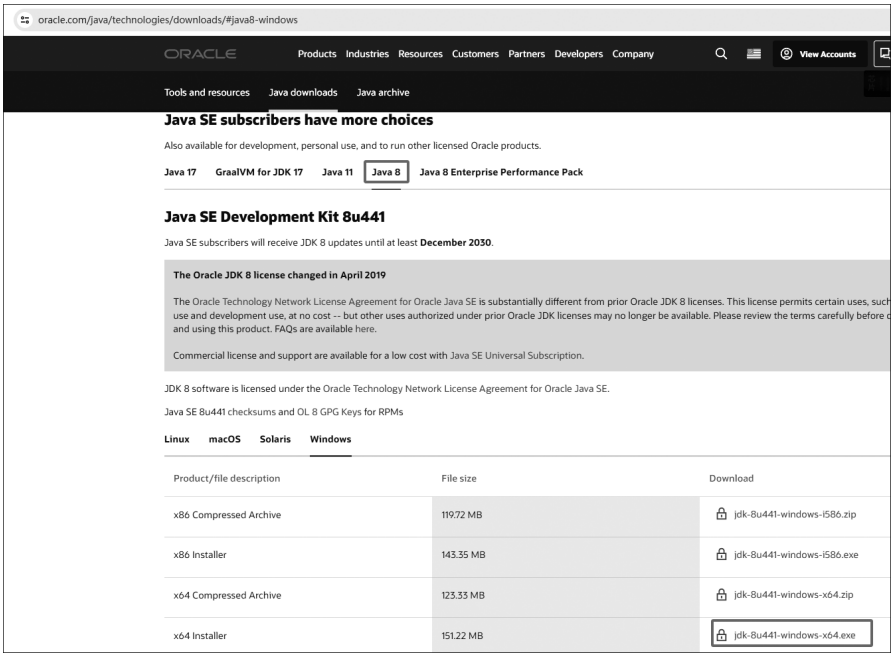

图 5.1　下载 JDK

双击 jdk-8u441-windows-x64.exe 进行安装，如图 5.2 所示，设置 JDK 的安装路径。

图 5.2　安装 JDK

接下来一直单击“下一步”按钮，直到安装完毕。

（2）配置 JDK。

右击“我的电脑”，在弹出的快捷菜单中选择“属性”命令，在打开的界面中选择“高级系统设置”选项，如图 5.3 所示。

图 5.3　高级系统设置

单击“系统属性”对话框“高级”选项卡中的“环境变量”按钮，如图 5.4 所示。

图 5.4　修改环境变量

新建 JAVA_HOME“环境变量”，如图 5.5 所示，通过该环境变量找到 Java 安装路径。

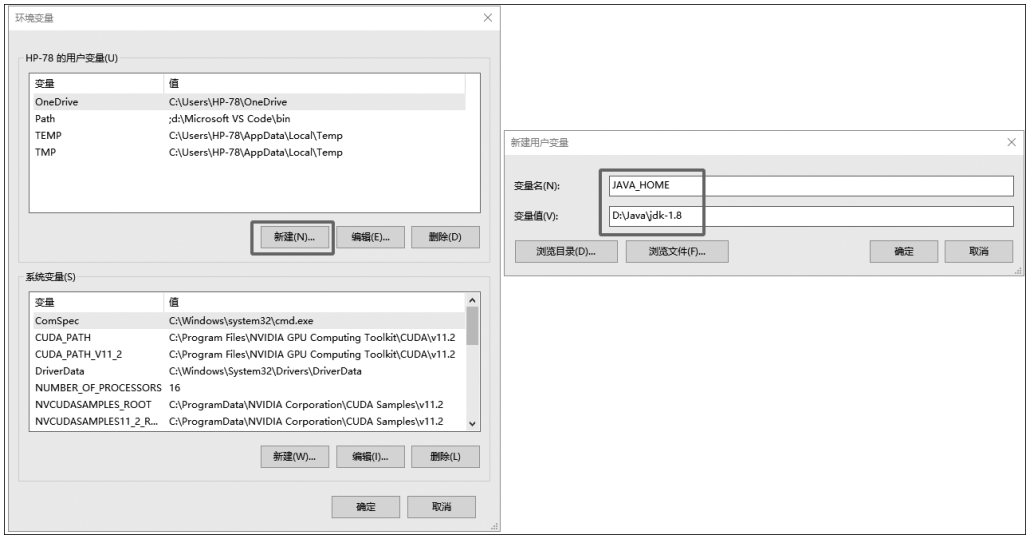

图 5.5　新建 JAVA_HOME 变量

如图 5.6 所示，编辑 Path 系统环境变量，单击“编辑”按钮，再单击“确定”按钮。

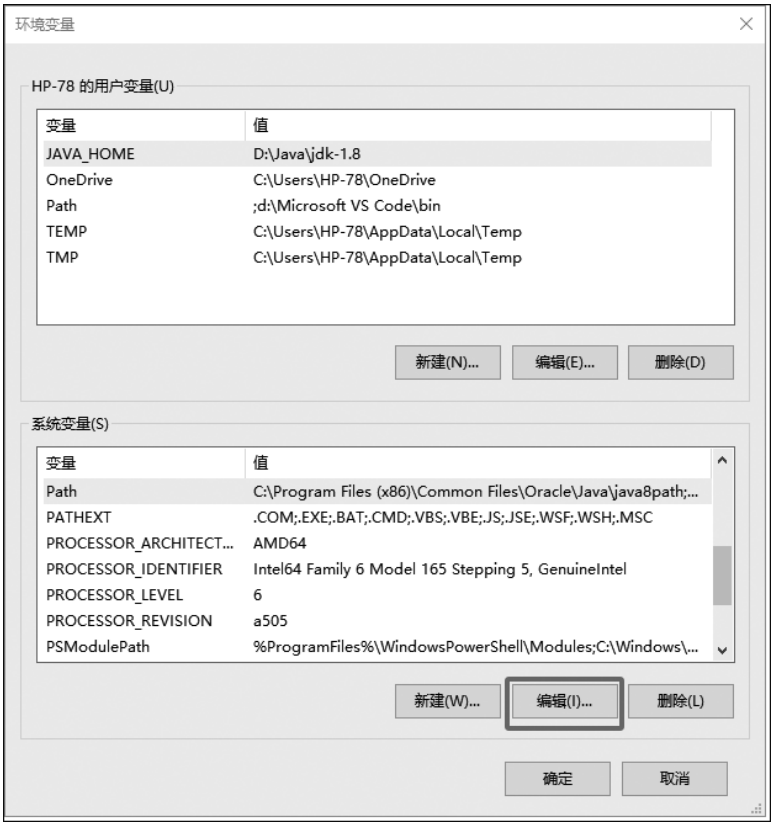

图 5.6　编辑系统变量 Path

如图 5.7 所示，新增 Java 的两条 bin 路径，再单击“确定”按钮。

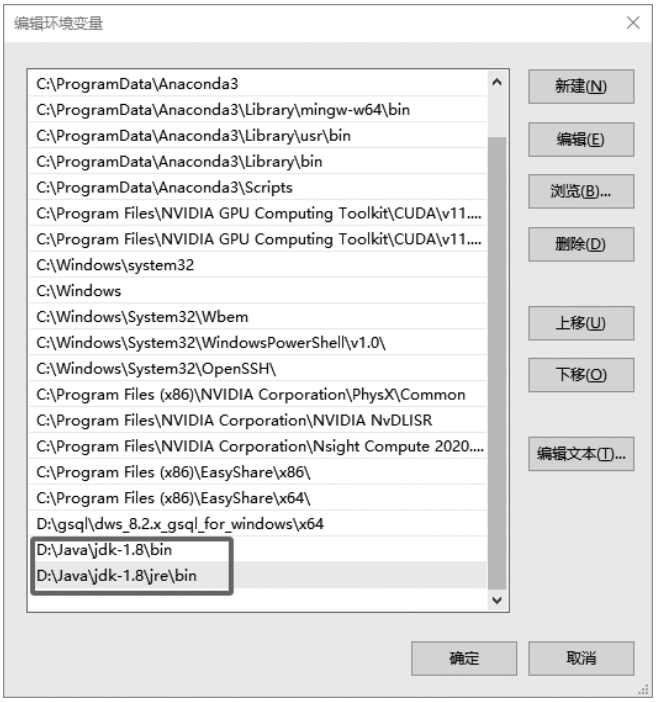

图 5.7　增加 Java 路径

(3) 测试 Java 的环境是否配置成功。

在 cmd 命令窗口中，输入 java -version，如图 5.8 所示，出现了安装的 JDK 的版本，表示 JDK 安装成功。

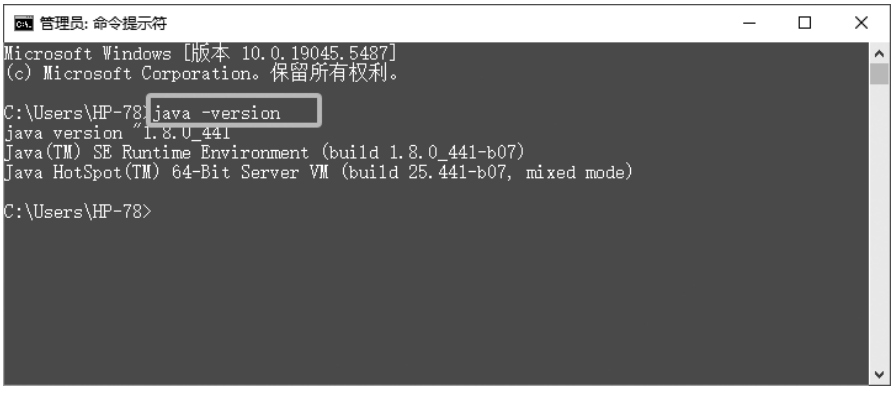

图 5.8　测试 Java 安装环境

5.2　JDBC 概述

JDBC(Java Database Connectivity，Java 数据库连接)是 Java 开发语言提供的连接关系数据库的接口标准，各驱动提供者可以使用自己的 JDBC 驱动程序来实现和扩展标准。

如图 5.9 所示，JDBC 体系结构分为三层，分别为 JDBC API、JDBC Driver Manager 以

及 JDBC Driver。JDBC API 提供应用程序到 JDBC 驱动管理器的连接，提供程序调用的接口和类。JDBC Driver Manager 对 JDBC 各类驱动进行管理载入。JDBC Driver 负责连接不同的数据库。

JDBC API 包含两个包：java.sql 和 javax.sql。java.sql 实现基本的数据库编程服务，如生成连接、执行 SQL 语句、准备语句、运行批处理查询等。javax.sql 实现扩展功能，为数据库方面的高级操作实现接口，如分布式事务、连接池、行集等。

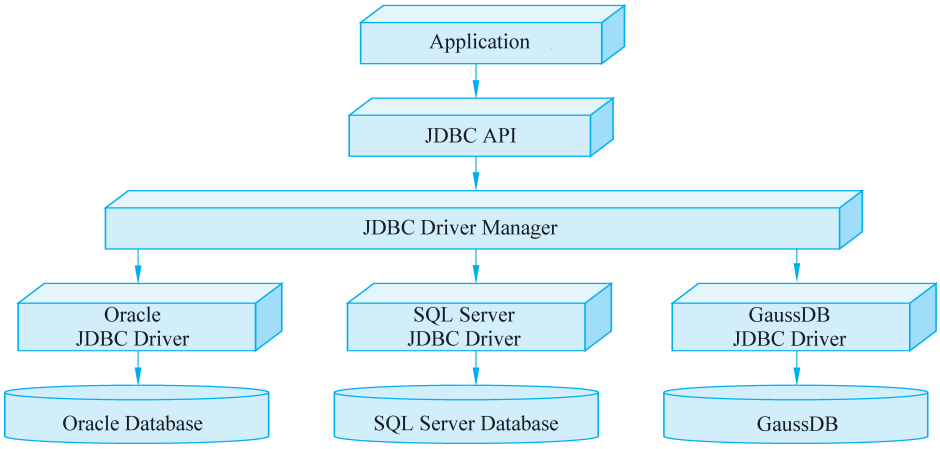

图 5.9 JDBC 体系结构

5.2.1 GaussDB JDBC 概述

GaussDB 支持 JDK8 及以上版本，支持 JDBC 4.0 的功能。GaussDB JDBC 主要增加了以下特性：支持 SHA256 加密方式登录，支持容灾切换等。

GaussDB JDBC API 如表 5.1 所示。

表 5.1 GaussDB JDBC API

JDBC 接口类名称	说　明
java.sql.Connection	数据库连接接口
java.sql.CallableStatement	存储过程执行接口
java.sql.DatabaseMetaData	数据库对象定义接口
java.sql.Driver	数据库驱动接口
java.sql.PreparedStatement	预处理语句接口，不支持方法 executeLargeUpdate
java.sql.ResultSet	执行结果集接口
java.sql.ResultsetMetaData	对 ResultSet 对象相关信息的具体描述
java.sql.Statement	SQL 语句接口
javax.sql.ConnectionPoolDataSource	数据源连接池接口
javax.sql.DataSource	数据源接口
javax.sql.PooledConnection	由连接池创建的连接接口

续表

JDBC 接口类名称	说　　明
javax.naming.Context	连接配置的上下文接口
javax.naming.spi.intialContextFactory	初始连接上下文工厂接口
CopyManager	GaussDB JDBC 中的一个 API 接口类，用于批量向集群中导入数据
PGReplicationConnection	GaussDB JDBC 中的一个 API 接口类，用于执行逻辑复制相关功能

JDBC 处理数据库的过程如图 5.10 所示，包含加载驱动、连接数据库、执行 SQL 语句、处理结果集、关闭连接 5 个步骤。

图 5.10　JDBC 处理数据库的过程

（1）在 GaussDB JDBC 中加载驱动一般采用如下语句：

```
Class.forName(com.huawei.opengauss.jdbc.Driver);
```

（2）DriverManager 是 java.sql 包中用来管理数据库驱动的类，最主要的功能就是获得数据库的连接。连接数据库有如下三个方法：

DriverManager.getConnection(String url)

DriverManager.getConnection(String url,Properties info)

DriverManager.getConnection(String url,String user,String password)

在后面的示例中，采用最常用的第三种方法直接给出连接的用户名和密码。在 getConnection 方法中，参数 url 的格式为如下几种。

jdbc:gaussdb:database 给出数据库名。

jdbc:gaussdb://host/database 给出主机地址（一般为公网 IP）和数据库名。

jdbc:gaussdb://host:port/database 给出主机地址、端口、数据库名。

jdbc:gaussdb://host:port/database?para1＝value1&para2＝value2 支持建立数据库连接时设置多个参数，如 ssl、连接超时设置等。

Jdbc:gaussdb://host1:port1,host2:port2/database? para1＝value1&para2＝value2 支持多个数据库主机，主要用于负载均衡、高可用性。

（3）在与数据库建立了 connection 连接之后，就可以应用各种类型 Statement 对象执行各种类型的 SQL 语句，如表 5.2 所示。

表 5.2　各种类型 Statement 对象方法

方　法　名	返回值类型	描 述 说 明
createStatement()	Statement	创建 Statement 对象
preparedStatement()	PreparedStatement	创建预处理 PreparedStatement 对象

续表

方 法 名	返回值类型	描 述 说 明
prepareCall(String sql)	CallableStatement	创建 CallableStatement 对象用于调用存储过程
commit()	void	使得所有更改或者回滚提交，成为持久，并释放当前 Connection 对象所持有的数据库锁
rollback()	void	取消当前事务中所进行的所有更改，并释放当前 Connection 对象所持有的数据库锁
close()	void	立即释放 Connection 对象的数据库和 JDBC 资源

例如，通过连接对象 conn 的 createStatement()方法创建一个 Statement 对象 stmt：

```
Statement stmt=conn.createStatement();
```

然后就可以使用该 Statement 对象执行 SQL 语句。但是 Statement 对象只能执行静态 SQL 语句，不接收参数。如果需要执行动态 SQL 则需要使用 PrepareStatement 对象。

例如，使用 Statement 对象的方法 excuteQuery 执行 SQL 查询，并返回一个 ResultSet 对象：

```
ResultSet rset=stmt.excuteQuery("select sname from student");
```

如果需要返回影响行数的 DDL 语句或者 DML 语句，应该使用 excuteUpdate()方法，如果 SQL 语句的类型未知，则应该使用 excute 方法。

(4) 在执行了 SQL 查询语句之后，获得一个结果集对象，处理结果集。这里可以通过 ResultSet 对象中的 next()方法遍历结果集合：

```
While(rset.next())
System.out.println(rset.getString(1));
```

在结果集对象中提取数据时，应使用 ResultSet 对象的 getXXX()方法，其中 XXX 对应 Java 数据类型。该示例查询结果集的 String 类型的第一个字段的值。

在 ResultSet 对象中，具有指向当前行的光标。在结果集 ResultSet 对象中定位的方法如表 5.3 所示。

表 5.3 ResultSet 对象的定位方法

方 法	描 述
next()	把 ResultSet 对象中的光标向后移动一行
previous()	把 ResultSet 对象中的光标向前移动一行
beforeFirst()	把 ResultSet 对象中的光标定位到第一行之前
afterLast()	把 ResultSet 对象中的光标定位到最后一行之后
first()	把 ResultSet 对象中的光标定位到第一行
last()	把 ResultSet 对象中的光标定位到最后一行

续表

方　　法	描　　述
absolute(int)	把 ResultSet 对象中的光标定位到参数指定的行数
relative(int)	把 ResultSet 对象中的光标移动参数指定的行数

获取结果集对象中的光标的方法如表 5.4 所示。

表 5.4　获取结果集对象的光标方法

方　　法	描　　述
isFirst()	是否在第一行
isLast()	是否在最后一行
isBeforeFirst()	是否在第一行之前
isAfterLast()	是否在最后一行之后
getRow()	获取当前在第几行

(5) 关闭结果集和 Statement 对象,用来释放内存,游标等,提高效率。

```
stmt.close();
rset.close();
```

5.2.2　GaussDB JDBC 的下载

在 JDK 1.8 环境中使用 gaussdbjdbc.jar,进入华为官网网址下载 GaussDB 的 JDBC 驱动,下载 GaussDB_opengauss_client_tools.zip。

下载好驱动之后,解压该压缩文件,有两个文件夹,由于所购买的数据库实例是主备版,如图 5.11 所示,打开 Centralized 文件夹。

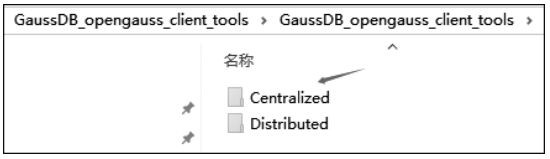

图 5.11　解压压缩文件

如图 5.12 所示选择 X86,打开该文件夹。

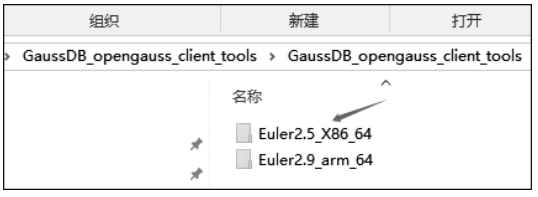

图 5.12　选择 X86 文件夹

如图 5.13 所示选择并解压该文件。

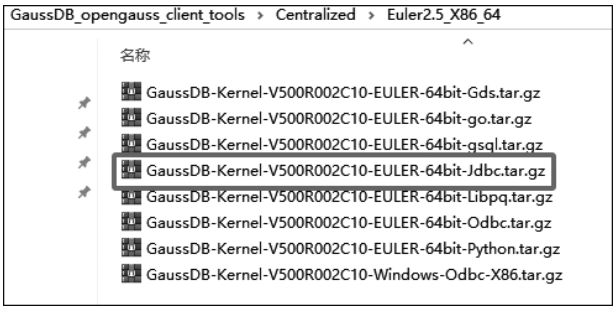

图 5.13 解压 Jdbc 压缩文件

如图 5.14 所示，该压缩文件夹下有三个 jar 包，将该压缩文件 opengaussjdbc.jar 复制至 Java 开发路径中，如 D:\Java。

图 5.14 复制 opengaussjdbc.jar 文件

5.3 Java 程序连接 GaussDB

将连接数据库的用户名和密码配置在环境变量中，比直接明文写在程序中更加安全，如图 5.15 所示。注意配置好环境变量后，需要重新启动操作系统。当然也可以采用其他的方法，如写在配置文件中，这里不详述配置文件的方法。

【例 5.1】 在 Java 项目路径中，编写 Java 程序，该程序连接 GaussDB 上的 student2024 数据库，其公网地址为 110.41.115.66，用户名的环境变量为 usernames，密码的环境变量名为 password。通过 System 的方法 getenv()获取环境变量值，数据库驱动使用 com.huawei.opengauss.jdbc.Driver 驱动。该程序首先通过 Class.forName(driver)加载 GaussDB 驱动，通过 getConnection()方法建立连接。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.PreparedStatement;
public class JavaConGaussDB{
    public static void main(String[] args) throws SQLException {
        String driver ="com.huawei.opengauss.jdbc.Driver";
```

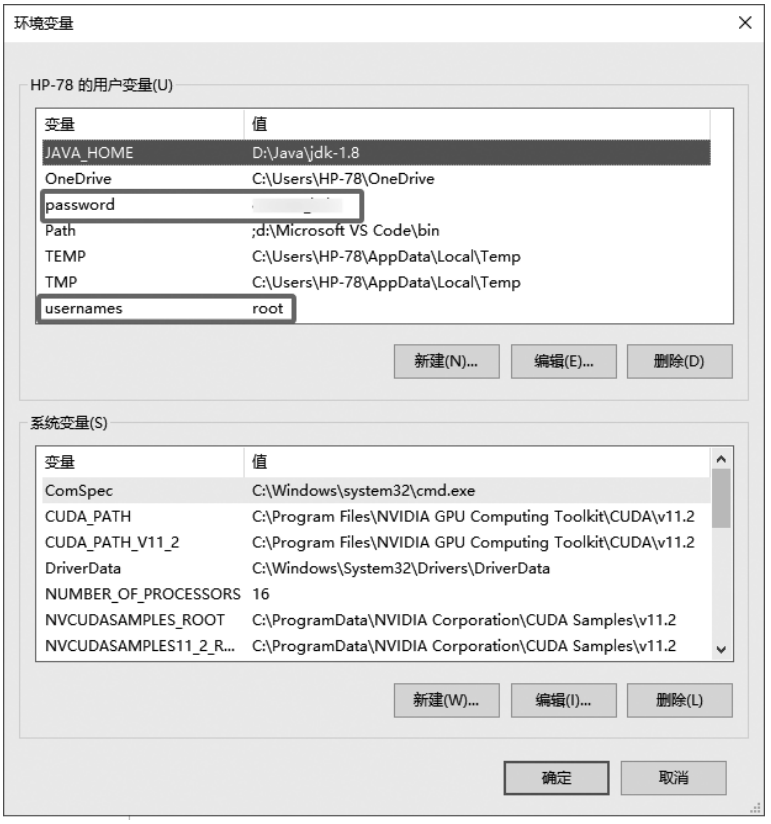

图 5.15　配置用户名和密码环境变量

```
String usernames =System.getenv("usernames");
String password =System.getenv("password");
String sourceURL ="jdbc:opengauss://110.41.115.66:8000/student2024";
Connection conn =null;
Statement stmt;
ResultSet rs;
System.out.println(usernames);
System.out.println(password);
try {
    // 加载数据库驱动
    Class.forName(driver);
} catch (Exception e) {
    e.printStackTrace();
}
try {
    // 以非加密方式创建数据库连接
    conn =DriverManager.getConnection(sourceURL, usernames, password);
    stmt=conn.createStatement();
    System.out.println("Connection succeed!");
    conn.close();
    stmt.close();
```