

第11章 JDBC数据库操作

导读

主要内容

- MySQL数据库管理系统
- 连接MySQL数据库
- JDBC
- 连接数据库
- 查询操作
- 更新、添加与删除操作
- 使用预处理语句
- 事务
- 批处理

重点和难点

- 重点：创建数据源和掌握JDBC连接的方法；实现查询功能
- 难点：预处理，事务

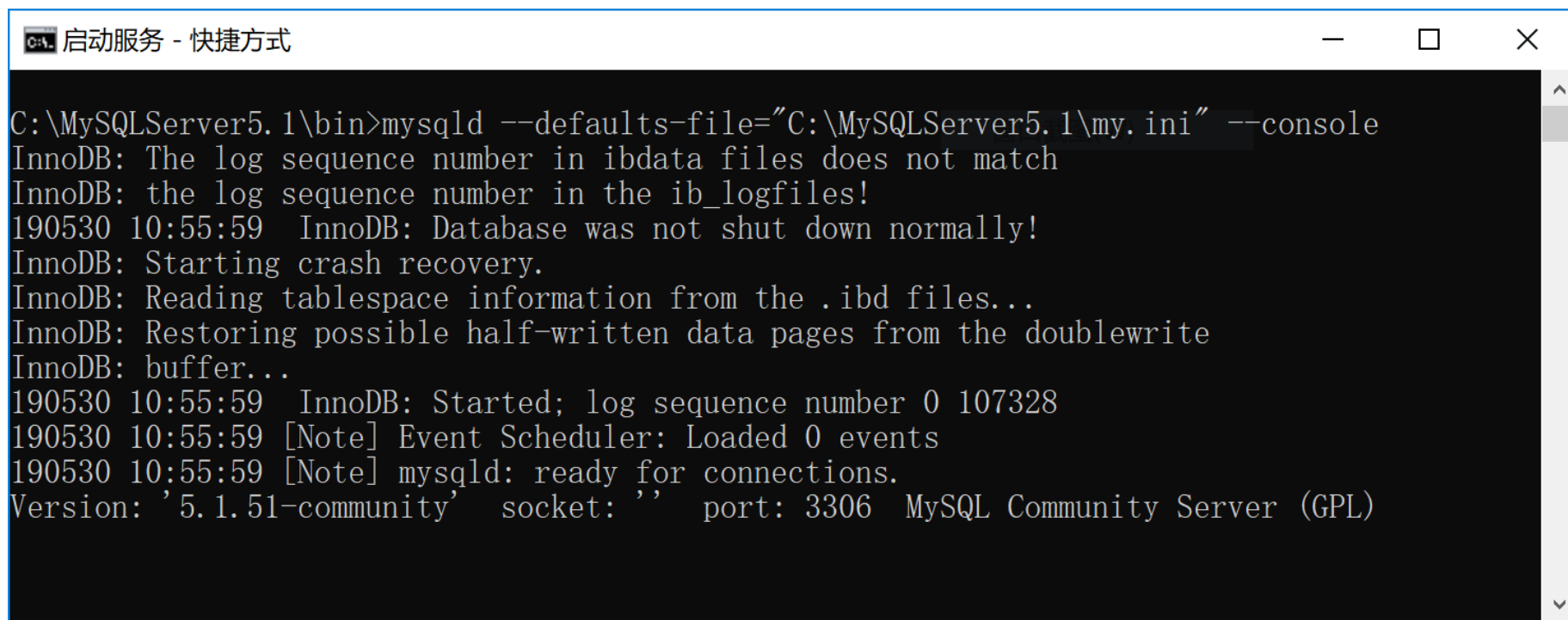
§ 11.1 MySQL数据库管理系统

- MySQL数据库管理系统，简称MySQL，是世界上最流行的开源数据库管理系统，其社区版（MySQL Community Edition）是最流行的免费下载的开源数据库管理系统。
- MySQL最初由瑞典MySQL AB公司开发，目前由Oracle公司负责源代码的维护和升级，Oracle将MySQL分为社区版和商业版，并保留MySQL开放源码这一特点。
- 目前许多应用开发项目都选用MySQL，其主要原因是MySQL的社区版性能卓越，可满足许多应用的需求，而且MySQL的社区版是开源系统、可降低软件的开发和使用成本。

11.2 启动MySQL数据库服务器

网络课堂中共享的MySQL 5.1压缩包下载后，**直接在C盘根目录下解压后**，进入bin目录，执行“启动服务.cmd”即可启动服务器（如果未放在C盘根目录下，需要修改配置文件my.ini和启动服务.cmd）。

注意：使用MySQL期间不可关闭本窗口。可使用Ctrl+C关闭MySQL服务器。



```
C:\MySQLServer5.1\bin>mysql --defaults-file="C:\MySQLServer5.1\my.ini" --console
InnoDB: The log sequence number in ibdata files does not match
InnoDB: the log sequence number in the ib_logfiles!
190530 10:55:59 InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Restoring possible half-written data pages from the doublewrite
InnoDB: buffer...
190530 10:55:59 InnoDB: Started; log sequence number 0 107328
190530 10:55:59 [Note] Event Scheduler: Loaded 0 events
190530 10:55:59 [Note] mysql: ready for connections.
Version: '5.1.51-community' socket: '' port: 3306 MySQL Community Server (GPL)
```

如果启动时提示端口已经被占用（例如开机后自动启动了MySQL 8），可修改my.ini文件中的端口配置，由3306改为其它空闲的端口号。

2. root用户

MySQL数据库服务器启动后，MySQL默认授权可以访问该服务器的用户只有一个，名字是root，密码为空。

应用程序以及MySQL客户端管理工具软件，都必须借助MySQL授权的“用户”来访问数据库服务器。

MySQL数据库服务器启动后，不仅可以用root用户访问数据库服务器，而且可以再授权能访问数据库服务器的新用户（只有root用户有权利建立新的用户）。

11.3 MySQL客户端管理工具

Navicat for MySQL: 可以在搜索引擎搜索**Navicat for MySQL**或登录:
<http://www.navicat.com.cn/download> 下载试用版或购买商业版, 例如
下载navicat112_mysql_cs_x64.exe安装即可
(温馨提示: 可以到网络课堂下载一个可用版)

启动navicat for MySQL 出现主界面。



图 11.8 启动 navicat forMySQL 客户端管理工具

§ 11.4 JDBC

- Java提供了专门用于操作数据库的API，即JDBC（Java DataBase Connection）。JDBC操作不同的数据库仅仅是连接方式上的差异而已，使用JDBC的应用程序一旦和数据库建立连接，就可以使用JDBC提供的API操作数据库。
- 程序经常使用JDBC进行如下的操作：
 - (1)与一个数据库建立连接。
 - (2)向数据库发送SQL语句。
 - (3)处理数据库返回的结果。

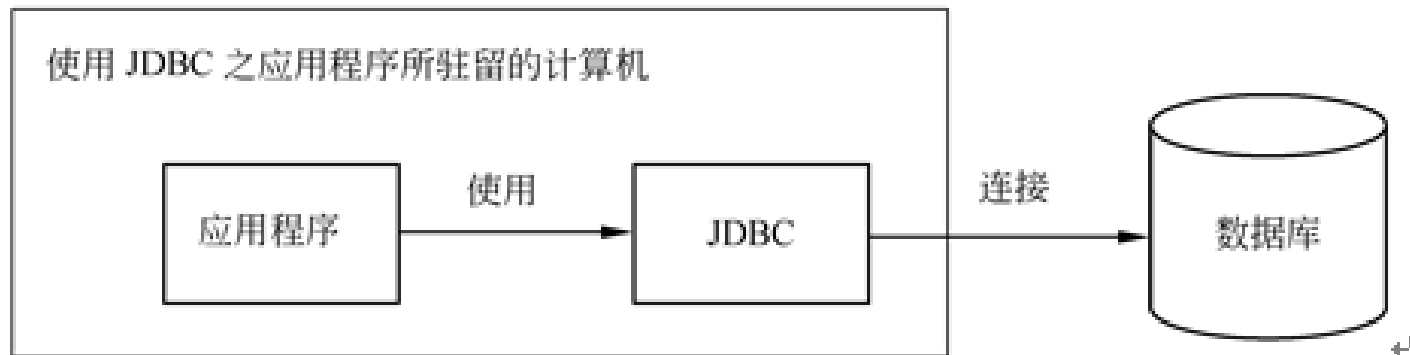


图 11.15 使用 JDBC 操作数据库

§ 11.5连接MySQL数据库

MySQL数据库服务器启动后，应用程序为了能和数据库交互信息，必须首先和MySQL数据库服务器上的数据库建立连接。目前常用的连接数据库的方式是**加载JDBC-数据库驱动**（连接器，即调用本地的JDBC-数据库驱动和相应的数据库建立连接。Java运行环境将JDBC-数据库驱动转换为DBMS所使用的专用协议来实现和特定的DBMS交互信息。

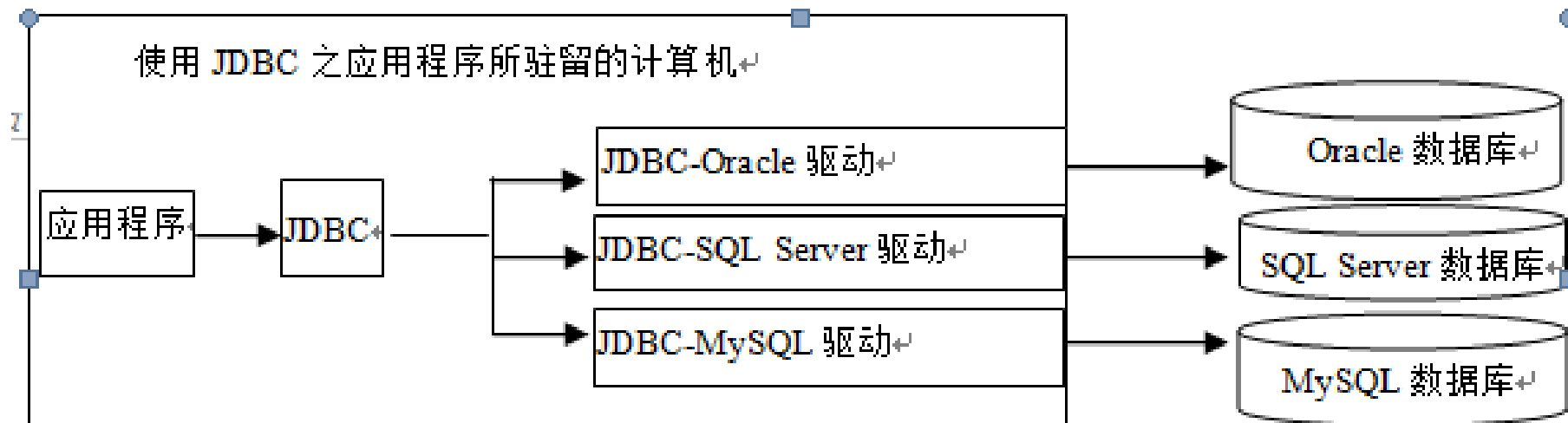


图 11.16 使用 JDBC-数据库驱动

1. 下载JDBC-MySQL数据库驱动

可登录MySQL官方网站：www.mysql.com，下载JDBC-MySQL数据库驱动（JDBC Driver for MySQL）。

教材下载的是**mysql-connector-java-5.1.40.zip**，将该zip文件解压至硬盘，在解压后的目录下的**mysql-connector-java-5.1.40-bin.jar**文件就是连接MySQL数据库的JDBC-数据库驱动。将该驱动复制到JDK的扩展目录中，比如：**E:\jdk1.8\jre\lib\ext**。

mysql-connector-java-5.1.40-bin.jar上传到了网盘，下载地址是<http://pan.baidu.com/s/1i5g87sD>

2.加载JDBC-MySQL数据库驱动

应用程序负责加载的JDBC-MySQL数据库驱动，代码如下：

```
try{  
    Class.forName("com.mysql.jdbc.Driver");  
}  
catch(Exception e){  
}
```

MySQL数据库驱动被封装在Driver类中，该类的包名是com.mysql.jdbc，该类不是Java运行环境类库中的类，所以需要放置在jre的扩展中

不要忘记将下载的mysql-connector-java-5.1.40-bin.jar文复制到JDK的扩展目录中。

3.连接数据库

应用程序要和MySQL数据库服务器管理的数据库students(在11.3节建立的数据库)建立连接，而有权访问数据库**students**的用户的id和密码分别是root和空，那么使用

Connection getConnection(java.lang.String)

方法建立连接的代码如下：

```
Connection con;
String uri =
"jdbc:mysql://192.168.100.1:3306/students?user=root&password=&useSSL=true";
try{
    con = DriverManager.getConnection(uri); //连接代码
}
catch(SQLException e){
    System.out.println(e);
}
```

如果root用户密码是99，将&password=更改为&password=99即可

使用

Connection getConnection(java.lang.String, java.lang.String, java.lang.String)
方法建立连接的代码如下：

```
Connection con;  
String uri = "jdbc:mysql:// 192.168.100.1:3306/students? useSSL=true";  
String user = "root";  
String password = "";  
try{  
    con = DriverManager.getConnection(uri,user,password); //连接代码  
}  
catch(SQLException e){  
    System.out.println(e);  
}
```

4.注意汉字问题

需要特别注意的是，如果数据库的表中的记录有汉字，那么在建立连接时需要额外多传递一个参数**characterEncoding**，并取值gb2312或utf-8

```
String uri =
```

```
"jdbc:mysql://localhost/students?useSSL=true&characterEncoding=utf-8";
```

```
con = DriverManager.getConnection(uri, "root",""); //连接代码
```

11.6 查询操作

查询操作的具体步骤如下

1. 得到SQL查询语句对象

```
try{  
    Statement sql=con.createStatement();  
}  
catch(SQLException e){}
```

2. 处理查询结果

有了SQL语句对象后，这个对象就可以调用相应的方法实现对数据库中表的查询和修改，并将查询结果存放在一个ResultSet类声明的对象中。也就是说SQL查询语句对数据库的查询操作将返回一个ResultSet对象，ResultSet对象是按“列”（字段）组织的数据行构成。

```
ResultSet rs = sql.executeQuery("SELECT * FROM students");
```

结果集rs的列数是4列，刚好和students的列数相同

对于

```
ResultSet rs = sql.executeQuery("SELECT sname, ssex FROM students");
```

内存的结果集对象rs列数只有两列，第一列是sname列，第2列是ssex列

ResultSet对象一次只能看到一个数据行，使用**next()**方法移到下一个数据行，获得一行数据后，**ResultSet**对象可以使用**getXxx**方法获得字段值（列值），将位置索引（第一列使用1，第二列使用2等）或列名传递给**getXxx**方法的参数即可。[表11.1](#)给出了**ResultSet**对象的若干方法。

无论字段是何种属性，总可以使用
getString(int columnIndex)或
getString(String columnName)
方法返回字段值的字符串表示

3. 关闭连接

ResultSet对象和数据库连接对象（Connection对象）实现了紧密的绑定，一旦连接对象被关闭，ResultSet对象中的数据立刻消失。这就意味着，应用程序在使用ResultSet对象中的数据时，就必须始终保持和数据库的连接，直到应用程序将ResultSet对象中的数据查看完毕。

如果在代码

```
ResultSet rs = sql.executeQuery("SELECT * FROM students");
```

之后立刻关闭连接

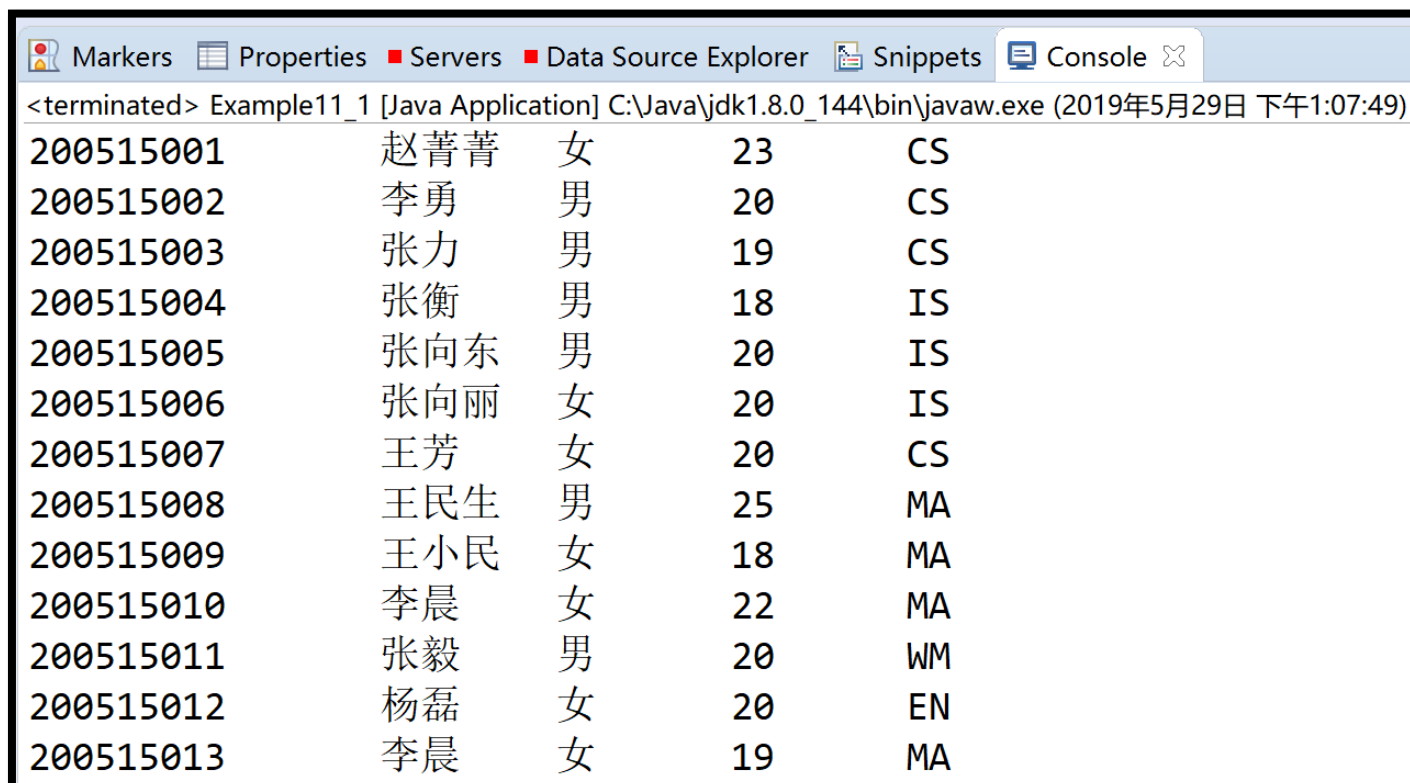
```
con.close();
```

程序将无法获取rs中的数据

11.6.1 顺序查询

所谓顺序查询，是指ResultSet对象一次只能看到一个数据行，使用next()方法移到下一个数据行，next()方法最初的查询位置，即游标位置，位于第一行的前面。next()方法向下（向后、数据行号大的方向）移动游标，移动成功返回true，否则返回false。

例子1查询学生_教学数据库中student表的全部记录。



200515001	赵菁菁	女	23	CS
200515002	李勇	男	20	CS
200515003	张力	男	19	CS
200515004	张衡	男	18	IS
200515005	张向东	男	20	IS
200515006	张向丽	女	20	IS
200515007	王芳	女	20	CS
200515008	王民生	男	25	MA
200515009	王小民	女	18	MA
200515010	李晨	女	22	MA
200515011	张毅	男	20	WM
200515012	杨磊	女	20	EN
200515013	李晨	女	19	MA

```
import java.sql.*;
public class Example11_1 {
    public static void main(String args[]) {
        Connection con=null;    Statement sql;    ResultSet rs;
        try{ Class.forName("com.mysql.jdbc.Driver"); //加载JDBC_MySQL驱动
        }
        catch(Exception e){ }
        String uri = "jdbc:mysql://localhost:3306/教学库?useSSL=true";
        String user ="root";    String password ="123";
        try{ con = DriverManager.getConnection(uri,user,password); //连接代码
        }
        catch(SQLException e){ }
        try { sql=con.createStatement();
            rs=sql.executeQuery("SELECT * FROM student"); //查询student表
            while(rs.next()) {
                String sno=rs.getString(1);    String sname=rs.getString(2);
                String ssex=rs.getString(3);    int sage=rs.getInt(4);
                String sdept=rs.getString(5);
                System.out.printf("%s\t",sno);    System.out.printf("%s\t",sname);
                System.out.printf("%s\t",ssex);    System.out.printf("%d\t",sage);
                System.out.printf("%s\n",sdept);
            } con.close();
        } catch(SQLException e) {    System.out.println(e);    }
    }
}
```

11.6.2 控制游标

为了得到一个**可滚动的结果集**，需使用下述方法获得一个Statement对象。

```
Statement stmt = con.createStatement(int type,int concurrency);
```

例子2 随机查询**student**表的2条记录，首先将游标移动到最后一行，再获取最后一行的行号，以便获得表中的记录数目。

(本例用到了第8章例子18中的GetRandomNumber类的static方法)

```
public static int [] getRandomNumber(int max,int amount)
```

返回1至max之间的amount个不同的随机数

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> Example11_2 [Java Application] C:\Java\jdk1.8.0_144\bin\javaw.exe (2019年5月29日 下午1:09:15)

表共有29条记录,随机抽取2条记录:

200515018	李贵	男	19	EN
200515002	李勇	男	20	CS

将数据库连接的代码单独封装到一个[GetDatabaseConnection](#)类中。

```
package ch11;
import java.sql.*;
public class GetDBConnection {
    public static Connection connectDB(String DBName,String id,String p) {
        Connection con = null;
        String
        uri =
        "jdbc:mysql://localhost:3306/"+DBName+"?useSSL=true&characterEncoding=utf-8";
        try{ Class.forName("com.mysql.jdbc.Driver"); //加载JDBC-MySQL驱动
        }
        catch(Exception e){ }
        try{
            con = DriverManager.getConnection(uri,id,p); //连接代码
        }
        catch(SQLException e){ }
        return con;
    }
}
```

```

import java.sql.*;
public class Example11_2 {
    public static void main(String args[]) {
        Connection con;    Statement sql;    ResultSet rs;
        con = GetDBConnection.connectDB("教学库","root","123");
        if(con == null ) return;
        try {
            sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                     ResultSet.CONCUR_READ_ONLY);
            rs = sql.executeQuery("SELECT * FROM student ");           rs.last();
            int max = rs.getRow(); System.out.println("表共有"+max+"条记录,随机抽取2条记录: ");
            int [] a =GetRandomNumber.getRandomNumber(max,2);//得到1-max之间2个不同随机数
            for(int i:a){
                rs.absolute(i);//游标移动到第i行
                String sno=rs.getString(1);        String sname=rs.getString(2);
                String ssex=rs.getString(3);        int sage=rs.getInt(4);
                String sdept=rs.getString(5);
                System.out.printf("%s\t",sno);        System.out.printf("%s\t",sname);
                System.out.printf("%s\t",ssex);        System.out.printf("%d\t",sage);
                System.out.printf("%s\n",sdept);
            }
            con.close();
        } catch(SQLException e) {        System.out.println(e);    }
    }
}

```

```
import java.util.*;
public class GetRandomNumber {
    public static int [] getRandomNumber(int max,int amount) {
        //1-max之间的amount个不同随机整数
        int [] randomNumber = new int[amount];
        int index =0;    randomNumber[0]= -1;
        Random random = new Random();
        while(index<amount){
            int number = random.nextInt(max)+1;    boolean isInArrays=false;
            for(int m:randomNumber){//m依次取数组randomNumber元素的值
                if(m == number)
                    isInArrays=true; //number在数组里了
            }
            if(isInArrays==false){//如果number不在数组randomNumber中:
                randomNumber[index] = number;
                index++;
            }
        }
        return randomNumber;
    }
}
```

11.6.3 条件与排序查询

1. where子语句

一般格式：

select 字段 from 表名 where 条件

（1）字段值和固定值比较，例如：

```
select sname, ssex from student where sname='李四'
```

（2）字段值在某个区间范围，例如：








```
select * from student where sage<23 and sage>19
```

2. 排序

用order by子语句对记录排序

```
select * from student where sname like '%林%' order by sname
```

例子3 查询student表中姓名至少包括2个汉字、sage大于22的女生，并按学号排序。

 Markers  Properties  Servers  Data Source Explorer  Snippets  Console 				
<terminated> Example11_3 [Java Application] C:\Java\jdk1.8.0_144\bin\javaw.exe (2019年5月29日)				
200515001	赵菁菁	女	23	CS
200515025	朱小鸥	女	30	WM


```

import java.sql.*;
public class Example11_3 {
    public static void main(String args[]) {
        Connection con;    Statement sql;    ResultSet rs;
        con = GetDBConnection.connectDB("教学库","root","123");
        if(con == null ) return;
        String c1=" ssex='女' and sage>22";//条件1
        String c2=" sname Like '__%"; //条件2
        String sqlStr = "select * from student where "+c1+" and "+c2+" order by sno";
        try {
            sql=con.createStatement();    rs = sql.executeQuery(sqlStr);
            while(rs.next()) {
                String sno=rs.getString(1);    String sname=rs.getString(2);
                String ssex=rs.getString(3);    int sage=rs.getInt(4);
                String sdept=rs.getString(5);
                System.out.printf("%s\t",sno);    System.out.printf("%s\t",sname);
                System.out.printf("%s\t",ssex);    System.out.printf("%d\t",sage);
                System.out.printf("%s\n",sdept);
            }
            con.close();
        }
        catch(SQLException e) {
            System.out.println(e);    }
    }
}

```

11.7 更新、添加与删除操作

1.更新

update 表 set 字段 = 新值 where <条件子句>

2.添加

insert into 表(字段列表) values (对应的具体的记录)

或

insert into 表 values (对应的具体的记录)

3.删除

delete from 表名 where <条件子句>

例子4向student插入2条记录

```
import java.sql.*;
public class Example11_4 {
    public static void main(String args[]) {
        Connection con;    Statement sql;    ResultSet rs;
        con = GetDBConnection.connectDB("教学库","root","123");
        if(con == null ) return;
        String jiLu="('201800101','王三','男',19, '软件工程系'),"+
            "('201800102','王珊','女',20, '计算机科学系')"; //2条记录
        String sqlStr ="insert into student values "+jiLu;
        try {
            sql=con.createStatement();    int ok = sql.executeUpdate(sqlStr);
            rs = sql.executeQuery("select * from student");
            while(rs.next()) {
                String sno=rs.getString(1);    String sname=rs.getString(2);
                String ssex=rs.getString(3);    int sage=rs.getInt(4);
                String sdept=rs.getString(5);
                System.out.printf("%s\t",sno);    System.out.printf("%s\t",sname);
                System.out.printf("%s\t",ssex);    System.out.printf("%d\t",sage);
                System.out.printf("%s\n",sdept);
            }
            con.close();
        }
        catch(SQLException e) { System.out.println("记录中sno值不能重复"+e);    }
    }
}
```

11.8 使用预处理语句

11.8.1 预处理语句优点

如果应用程序能针对连接的数据库，事先就将SQL语句解释为数据库底层的内部命令，然后直接让数据库去执行这个命令，显然不仅减轻了数据库的负担，而且也提高了访问数据库的速度。

Connection和某个数据库建立了连接对象con，那么con就可以调用 **prepareStatement(String sql)** 方法对参数sql指定的SQL语句进行预编译处理，生成该数据库底层的内部命令，并将该命令封装在 **PreparedStatement** 对象中，那么该对象调用下列方法都可以使得该底层内部命令被数据库执行。

```
ResultSet executeQuery()
```

```
boolean execute()
```

```
int executeUpdate()
```

11.8.2 使用通配符?

```
String str = "select * from student where height < ? and name= ? "  
PreparedStatement sql = con.prepareStatement(str);
```

在sql对象执行之前，必须调用相应的方法设置通配符?代表的具体值，如：

```
sql.setFloat(1,1.76f);  
sql.setString(2, "武泽");
```

预处理SQL语句sql中第1个通配符?代表的值是1.76，第2个通配符?代表的值是'武泽'。通配符按着它们在预处理SQL语句中从左到右依次出现的顺序分别被称为第1个、第2个、.....、第m个通配符。

例子5中使用预处理语句向student表添加记录并查询了姓刘的记录

```

import java.sql.*;
public class Example11_5 {
    public static void main(String args[]) {
        Connection con;    PreparedStatement preSql;    ResultSet rs;
        con = GetDBConnection.connectDB("教学库","root","123");
        if(con == null ) return;
        String sqlStr ="insert into student values(?,?,?,?,?)";
        try { preSql = con.prepareStatement(sqlStr);//得到预处理语句对象preSql
            preSql.setString(1,"A001");  preSql.setString(2,"刘伟1");
            preSql.setString(3,"男");    preSql.setFloat(4,20);    //设置第4个?代表的值
            preSql.setString(5,"通信系");    //设置第5个?代表的值
            int ok = preSql.executeUpdate();
            sqlStr="select * from student where sname like ? ";
            preSql = con.prepareStatement(sqlStr);//得到预处理语句对象preSql
            preSql.setString(1,"刘%"); rs = preSql.executeQuery();
            while(rs.next()) {
                String sno=rs.getString(1); String sname=rs.getString(2);  String ssex=rs.getString(3);
                int sage=rs.getInt(4);    String sdept=rs.getString(5);
                System.out.printf("%s\t",sno);    System.out.printf("%s\t",sname);
                System.out.printf("%s\t",ssex);    System.out.printf("%d\t",sage);
                System.out.printf("%s\n",sdept);
            } con.close();
        } catch(SQLException e) { System.out.println("记录中sno值不能重复"+e);
        }
    }
}

```

11.9 通用查询

编写一个类，只要用户将数据库名、SQL语句传递给该类对象，那么该对象就用一个二维数组返回查询的记录。

结果集ResultSet对象rs调用`getMetaData()`方法返回一个`ResultSetMetaData`对象（结果集的元数据对象）：

```
ResultSetMetaData metaData = rs.getMetaData();
```


metaData,调用`getColumnCount()`方法就可以返回结果集rs中的列的数目：

```
int columnCount = metaData.getColumnCount();
```

metaData调用`getColumnName(int i)`方法就可以返回结果集rs中的第i列的名字：

```
String columnName = metaData.getColumnName(i);
```

例子6将数据库名以及SQL语句传递给Query类的对象，用表格（JTable组件）显示查询到的记录。



sno	sname	ssex	sage	sdept
200515001	赵菁菁	女	23	CS
200515002	李勇	男	20	CS
200515003	张力	男	19	CS
200515004	张衡	男	18	IS
200515005	张向东	男	20	IS
200515006	张向丽	女	20	IS
200515007	王芳	女	20	CS
200515008	王民生	男	25	MA
200515009	王小民	女	18	MA
200515010	李晨	女	22	MA
200515011	张毅	男	20	WM
200515012	杨磊	女	20	EN
200515013	李晨	女	19	MA
200515014	张丰毅	男	22	CS
200515015	李蕾	女	21	EN
200515016	刘社	男	21	CM
200515017	刘星耀	男	18	CM
200515018	李贵	男	19	EN
200515019	林自许	男	20	WM
200515020	马翔	男	21	
200515021	刘峰	男	25	CS
200515022	牛站强	男	22	
200515023	李婷婷	女	18	
200515024	严丽	女	20	
200515025	朱小鸥	女	30	WM
201800101	王三	男	19	软件工程系
201800102	王珊	女	20	计算机科学系
A001	刘伟	男	19	通信系
A002	刘伟1	男	20	通信系


```
package ch11;

import javax.swing.*;

public class Example11_6 {
    public static void main(String args[]) {
        String [] tableHead;
        String [][] content;
        JTable table ;
        JFrame win= new JFrame();
        Query findRecord = new Query();
        findRecord.setDatabaseName("教学库");
        findRecord.setSQL("select * from student");
        content = findRecord.getRecord();
        tableHead=findRecord.getColumnName();
        table = new JTable(content,tableHead);
        win.add(new JScrollPane(table));
        win.setBounds(12,100,400,200);
        win.setVisible(true);
        win.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```
package ch11;
import java.sql.*;
public class Query {
    String databaseName="";    //数据库名
    String SQL;                //SQL语句
    String [] columnName;      //全部字段（列）名
    String [][] record;        //查询到的记录

    public Query() {
        try{ Class.forName("com.mysql.jdbc.Driver");//加载JDBC-MySQL驱动
        } catch(Exception e){}
    }

    public void setDatabaseName(String s) {    databaseName=s.trim(); }
    public void setSQL(String SQL) {    this.SQL=SQL.trim(); }
    public String[] getColumnName() {
        if(columnName ==null ){ System.out.println("先查询记录"); return null; }
        return columnName;
    }
    public String[][] getRecord() {
        startQuery(); return record; }

    private void startQuery() {
        ...
    }
}
```

```
package ch11;  import java.sql.*;
public class Query {
    //.....
    private void startQuery() {
        Connection con;    Statement sql;    ResultSet rs;
        String uri="jdbc:mysql://localhost:3306/" + databaseName + "?useSSL=true&characterEncoding=utf-8";
        try { con=DriverManager.getConnection(uri,"root","123");
            sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                    ResultSet.CONCUR_READ_ONLY);
            rs=sql.executeQuery(SQL);    ResultSetMetaData metaData = rs.getMetaData();
            int columnCount = metaData.getColumnCount();//字段数目
            columnName=new String[columnCount];
            for(int i=1;i<=columnCount;i++){ columnName[i-1]=metaData.getColumnName(i);    }
            rs.last();    int recordAmount =rs.getRow(); //结果集中的记录数目
            record = new String[recordAmount][columnCount];
            int i=0;    rs.beforeFirst();
            while(rs.next()) {
                for(int j=1;j<=columnCount;j++){ record[i][j-1]=rs.getString(j); //第i条记录放入数组的第i行
                }
                i++;
            }    con.close();
        }
        catch(SQLException e) {    System.out.println("请输入正确的表名"+e);    }
    }
}
```

11.10 事务

11.10.1 事务及处理

事务由一组SQL语句组成，所谓事务处理是指：应用程序保证事务中的SQL语句要么全部都执行，要么一个都不执行。

11.10.2 JDBC事务处理步骤

1. 用setAutoCommit(boolean b)方法关闭自动提交模式
2. 用commit()方法处理事务
3. 用rollback()方法处理事务失败

下面的例子7使用了事务处理，将student表中sno字段是A001的ssage的值减少2，并将减少的2增加到字段是A002的ssage上。

```

import java.sql.*;
public class Example11_7{
    public static void main(String args[]){
        Connection con = null;    Statement sql;    ResultSet rs;    String sqlStr;
        con = GetDBConnection.connectDB("教学库","root","123");
        if(con == null ) return;
        try{ int n = 2;           con.setAutoCommit(false);           //关闭自动提交模式
            sql = con.createStatement(); sqlStr = "select sname,sage from student where sno='A001'";
            rs = sql.executeQuery(sqlStr);          rs.next();          int h1 = rs.getInt(2);
            System.out.println("事务之前"+rs.getString(1)+"年龄:"+h1);
            sqlStr = "select sname,sage from student where sno='A002'";
            rs = sql.executeQuery(sqlStr);          rs.next();          int h2 = rs.getInt(2);
            System.out.println("事务之前"+rs.getString(1)+"年龄:"+h2); h1 = h1-n;    h2 = h2+n;
            sqlStr = "update student set sage =" +h1+" where sno='A001'"; sql.executeUpdate(sqlStr);
            sqlStr = "update student set sage =" +h2+" where sno='A002'"; sql.executeUpdate(sqlStr);
            con.commit(); //开始事务处理,如果发生异常直接执行catch块
            con.setAutoCommit(true); //恢复自动提交模式
            String s = "select sname,sage from student"+" where sno='A001' or sno='A002'";
            rs = sql.executeQuery(s);
            while(rs.next()){ System.out.println("事务后"+rs.getString(1)+"年龄:"+rs.getFloat(2));
                } con.close();
        }catch(SQLException e){try{ con.rollback();           //撤销事务所做的操作           }
            catch(SQLException exp){}
        }
    }
}

```

连接MySQL 8的注意事项

1. 要使用专门的jar包。下载驱动mysql-connector-java-8.0.11.jar
2. Java代码连接时设置的驱动程序名（需要注意MySQL8.0和低版本不同的地方：**DB_DRIVER**要写com.mysql.cj.jdbc.Driver）

```
try{  
    Class.forName("com.mysql.cj.jdbc.Driver");  
}  
catch(Exception e){  
}
```

3. 建立连接的URI稍有变化（需要注意MySQL8.0和低版本不同的地方：**DB_URL**后面要加上useSSL和serverTimezone）

```
String uri =
```

```
"jdbc:mysql://localhost:3306/mh?useSSL=false&serverTimezone=UTC";
```

```
con = DriverManager.getConnection(uri, "root",""); //连接代码
```

```
import java.sql.*;
public class TestMySQL8 {
    public static void main(String[] args) {
        String driver = "com.mysql.cj.jdbc.Driver"; // 驱动程序名
        String url = "jdbc:mysql://localhost:3306/mh?useSSL=false&serverTimezone=UTC";
        String user = "root"; String password = "123";
        try { Class.forName(driver); // 加载驱动程序
            Connection con = DriverManager.getConnection(url, user, password);
            Statement statement = con.createStatement();
            String sql = "select * from student";
            ResultSet rs = statement.executeQuery(sql);
            System.out.println("姓名" + "\t" + "年龄" + "\t" + "专业");
            String name = null; String position = null; int age = 0;
            while(rs.next()){
                name = rs.getString("name"); age = rs.getInt("age"); position =
rs.getString("major");
                System.out.println(name + "\t" + age + "\t" + position);
            }
            rs.close(); con.close();
        } catch (Exception e) { e.printStackTrace(); }
    }
}
```

11.11 连接SQL Server数据库

要使用专门的jar包。下载驱动`sqljdbc42.jar`

加载SQL Server驱动程序代码如下：

```
try { Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
}
```

```
catch(Exception e){  
}
```

连接的代码如下：

```
try{  
    String uri=  
        "jdbc:sqlserver://192.168.100.1:1433;DatabaseName=warehouse";  
    String user="sa";  
    String password="dog123456";  
    con=DriverManager.getConnection(uri,user,password);  
}  
catch(SQLException e){  
    System.out.println(e);  
}
```



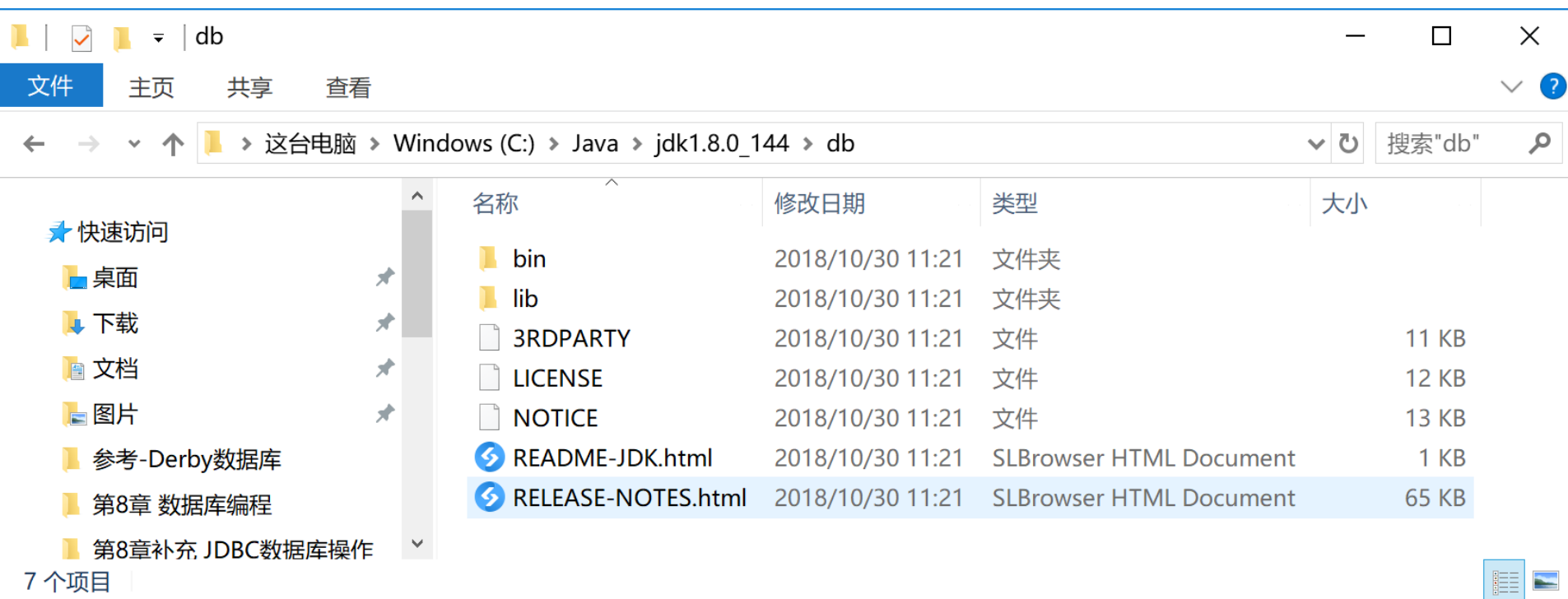
```

public class SequenceQuery {
    public static void main(String args[]) {
        Connection con=null;    Statement sql;    ResultSet rs;
        try { Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        }
        catch(Exception e){    }
        try{
String uri= "jdbc:sqlserver://127.0.0.1\\SQLEXPRESS:1433;DatabaseName=学生_教学";
            String user="sa";    String password="123456";
            con=DriverManager.getConnection(uri,user,password);
        }catch(SQLException e){    System.out.println(e);    }
        try { sql=con.createStatement();
            rs=sql.executeQuery("SELECT * FROM student"); //查询student表
            while(rs.next()) {
                String sno=rs.getString(1);        String sname=rs.getString(2);
                String ssex=rs.getString(3);        int sage=rs.getInt(4);
                String sdept=rs.getString(5);
                System.out.printf("%s\t",sno);        System.out.printf("%s\t",sname);
                System.out.printf("%s\t",ssex);        System.out.printf("%d\t",sage);
                System.out.printf("%s\n",sdept);
            } con.close();
        } catch(SQLException e) { System.out.println(e);
        }
    }
}

```

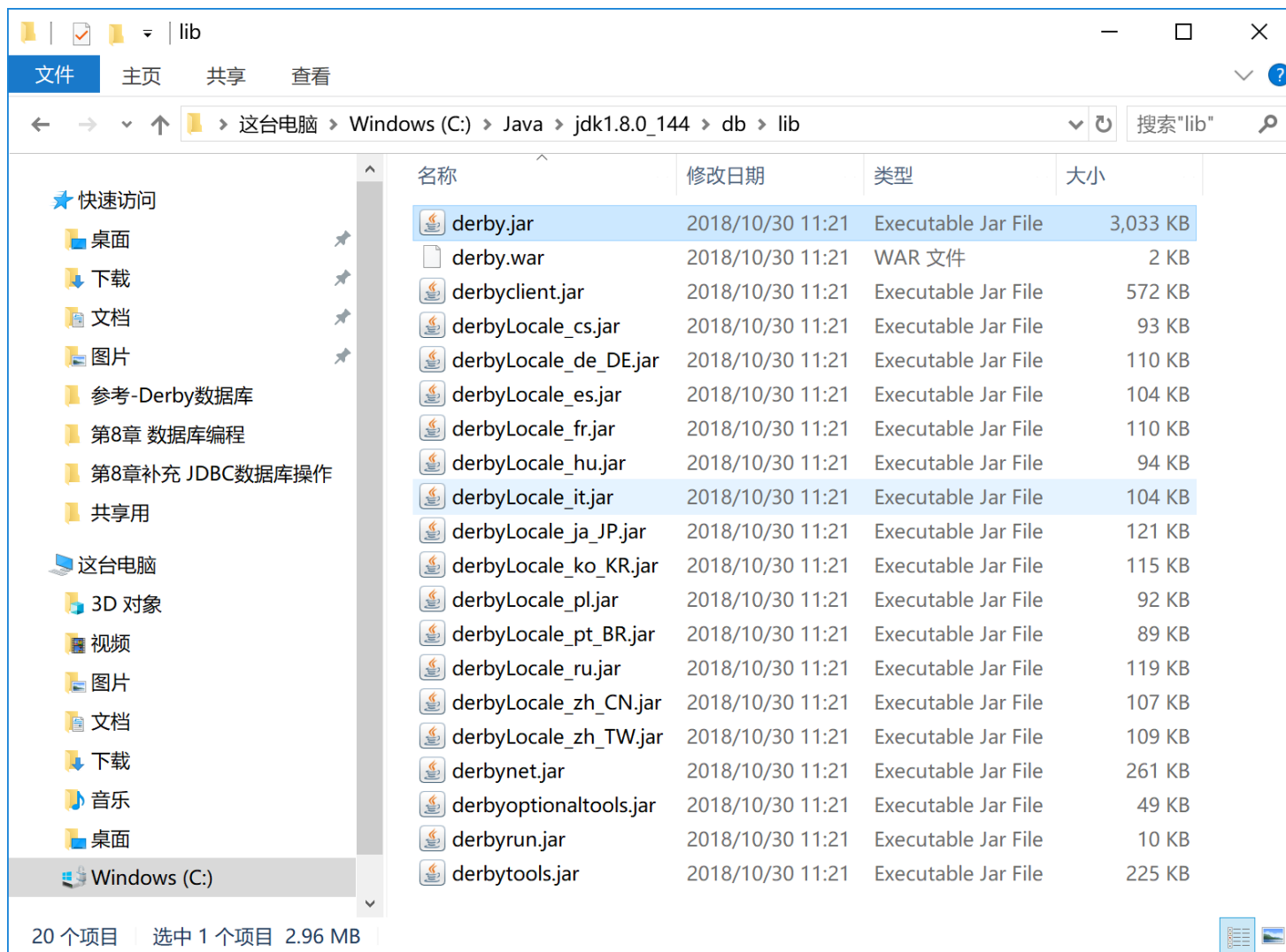
11.12 连接Derby 数据库

- Apache Derby是一个完全用java编写的数据库，Derby是一个Open source的产品。
- Apache Derby非常小巧，核心部分derby.jar只有2M，所以既可以做为单独的数据库服务器使用，也可以内嵌在应用程序中使用。
- JDK自带的derby数据库所在目录：



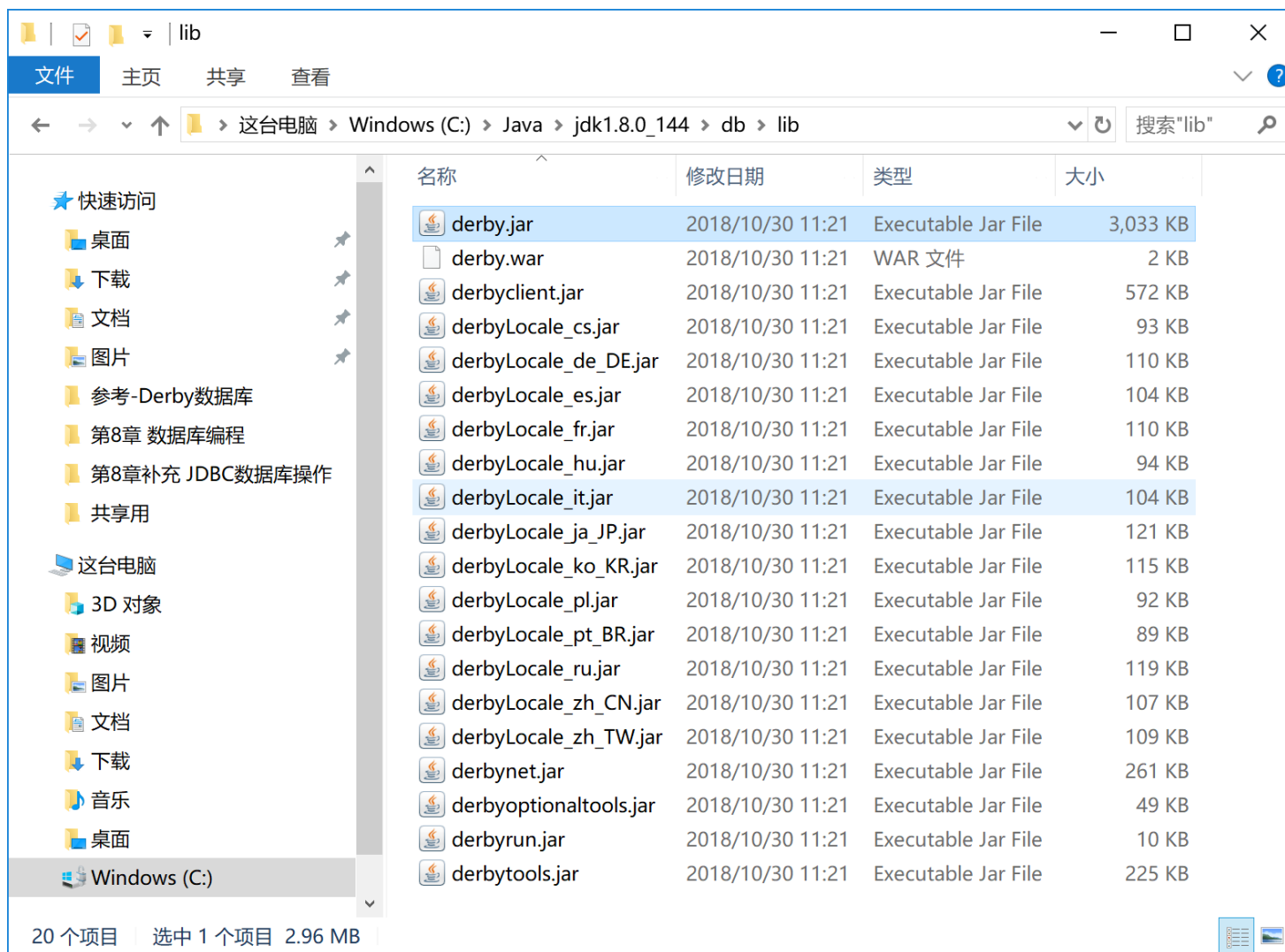
11.12 连接Derby 数据库

➤ 操作derby数据库的jar包所在位置:



11.12 连接Derby 数据库

➤ 操作derby数据库的jar包所在位置:



11.12 连接Derby 数据库

加载Derby数据库驱动程序的方法是：

```
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

连接（create取值是true）的代码是：

```
Connection con =  
DriverManager.getConnection("jdbc:derby:students;create=true");
```

例8使用Derby数据库管理系统创建了名字是students的数据库，并建立chengji表、插入3条数据。



张三	90.0
李斯	88.0
刘二	67.0

图 11.22 Derby 数据库

```

import java.sql.*;
public class Example11_8 {
    public static void main(String[] args) {
        Connection con =null;    Statement sta = null;    ResultSet rs;    String SQL;
        try {    Class.forName("org.apache.derby.jdbc.EmbeddedDriver");//加载驱动
        }    catch(Exception e) {        System.out.println(e);        return;
        }
        try {String uri ="jdbc:derby:students;create=true";
            con=DriverManager.getConnection(uri); //连接数据库
            sta = con.createStatement();
        }    catch(Exception e) {        System.out.println(e);        return;    }
        try { SQL = "create table chengji(name varchar(40),score float)";
            sta.execute(SQL);//创建表
        }    catch(SQLException e) {        //System.out.println("该表已经存在");    }
        SQL="insert into chengji values"+ "('张三', 90),('李斯', 88),('刘二', 67)";
        try {sta.execute(SQL);    rs = sta.executeQuery("select * from chengji "); // 查询表中的记录
            while(rs.next()) {
                String name=rs.getString(1);        System.out.print(name+"\t");
                float score=rs.getFloat(2);        System.out.println(score);
            }
            con.close();
        } catch(SQLException e) {        System.out.println(e);    }
    }
}

```

11.13 应用举例

11.13.1 设计思路

1. 数据库设计

在清楚了用户的需求之后，就需要进行数据库设计。数据库设计好之后才能进入软件的设计阶段，因此当一个应用问题的需求比较复杂时，数据库的设计（主要是数据库中各个表的设计）就显得尤为重要

2. 数据模型

程序应当将某些密切相关的数据封装到一个类中，例如，把数据库的表的结构封装到一个类中，即为表建立数据模型。其目的是用面向对象的方法来处理数据

3. 数据处理者

程序应尽可能将数据的存储与处理分开，即使用不同的类。数据模型仅仅存储数据，数据处理者根据数据模型和需求处理数据，比如当用户需要注册时，数据处理者将数据模型中的数据写入到数据库的表中

4. 视图

程序尽可能提供给用户交互方便的视图，用户可以使用该视图修改模型中的数据。并利用视图提供的交互事件（例如ActionEvent事件），将模型交给数据处理者

11.13.2 具体设计

1. user数据库和register表

使用MySQL客户端管理工具（见11.3）创建名字是**user**的数据库，在该库中新建名字是**register**的表，表的设计结构为：
(id char(20) primary key,password varchar(30),birth date)

2. 模型 (1) 注册模型 (2) 登录模型

3. 数据处理 (1) 注册处理者 (2) 登录处理者

4. 视图 (1) 注册视图 (2) 登录视图 (3) 集成视图


11.13.3 用户程序

下列程序提供一个华容道游戏（见第9章例子25），但希望用户登录后才可以玩游戏。因此，程序决定引入geng.view包中的RegisterAndLoginView类，以便提示用户登录或注册

（RegisterAndLoginView就可以满足用户的这个需求）。应用程序的主类没有包名，将主类[MainWindow.java](#)保存到c:\ch11中即可（但需要把第9章例子25中相关的[类Hua Rong road](#)和[Person类](#)与主类保存到同一目录中），运行效果如图11.24，11.25。



总结

- 
- （1）JDBC技术在数据库开发中占有很重要的地位，JDBC操作不同的数据库仅仅是连接方式上的差异而已，使用JDBC的应用程序一旦和数据库建立连接，就可以使用JDBC提供的API操作数据库。
 - （2）当查询ResultSet对象中的数据时，不可以关闭和数据库的连接。
 - （3）使用PreparedStatement对象可以提高操作数据库的效率。