

# Java 程序设计

## 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**
- **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，满足许多应用已经绰绰有余，而且 MySQL 的社区版是开源数据库管理系统、可以降低软件的开发和使用成本。**



# 1 下载

登录 [www.mysql.com](http://www.mysql.com) 后选择导航条上的 **products**，在出现的页面的左侧单击“**MySQL Community Edition**”或在出现的页面的右侧单击“**下载 MySQL 社区版**”超链接，如图 11.1 所示意。然后在出现的下载页面中选择适合相应平台的 MySQL

单击“**No thanks, just start my download**”超链接即可下载（可以忽略下载页上的注册 **Sign up**，如图 11.2 所示）。



图 11.1 选择 MySQL 社区版

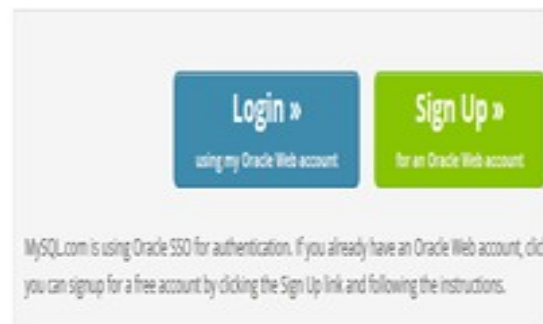


图 11.2 下载 MySQL 社区版



## 2 下载

将下载的 `mysql-5.7.15-winx64.zip` 解压缩到本地计算机即可，比如解压缩到 `D:\`。这里我们将下载的 `mysql-5.7.15-winx64.zip` 解压缩到 `D:\`，形成的目录结构如图 11.3。

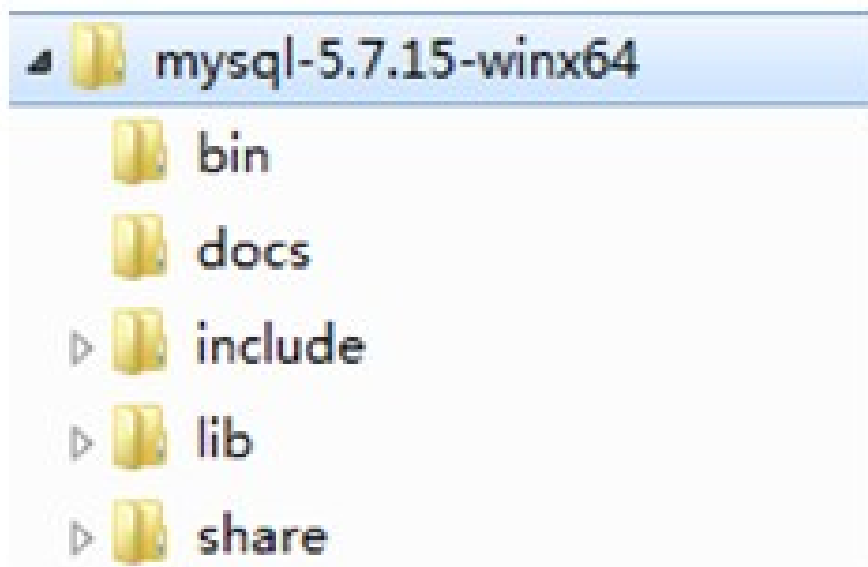


图 11.3 MySQL 的安装目录结构



## 11.2 启动 MySQL 数据库服务器

### 1. 启动

MySQL5.7 版本相对之前的 5.6 版本有所不同，在启动之前必须进行安全初始化。在命令行进入 MySQL 安装目录的 bin 子目录，键入 `mysqld --initialize-insecure` 命令：

```
D:\mysql-5.7.15-winx64\bin>mysql --initialize-insecure
```

其作用是初始化 data 目录，并授权一个无密码的 root 用户。执行成功后，MySQL 安装目录下多出一个 data 子目录（用于存放数据库，对于早期版本，安装后就有该目录）

初始化后，在 MySQL 安装目录的 bin 子目录下键入 `mysqld` 或 `mysqld -nt` 启动 MySQL 数据库服务器，MySQL 服务器占用的端口是 3306（3306 是 MySQL 服务器默认使用的端口号）。启动成功后，MySQL 数据库服务器将占用当前 MS-DOS 窗口，（和以前版本不同的是，启动成功后无任何提示）

# 11.2 启动 MySQL 数据库服务器

## 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。如果没有任何“用户”可以访问启动的 MySQL 数据库服务器，那么这个服务器就如同虚设、没有意义了。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）。关于建立新的用户的命令见稍后 11.3 节



## 11.3 MySQL 客户端管理工具

教材使用的是 Navicat for MySQL（比较盛行的），读者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（也可以到 <http://pan.baidu.com/s/1o79U6ds> 下载）

启动 navicat for MySQL 出现主界面，如图 11.8 所示。

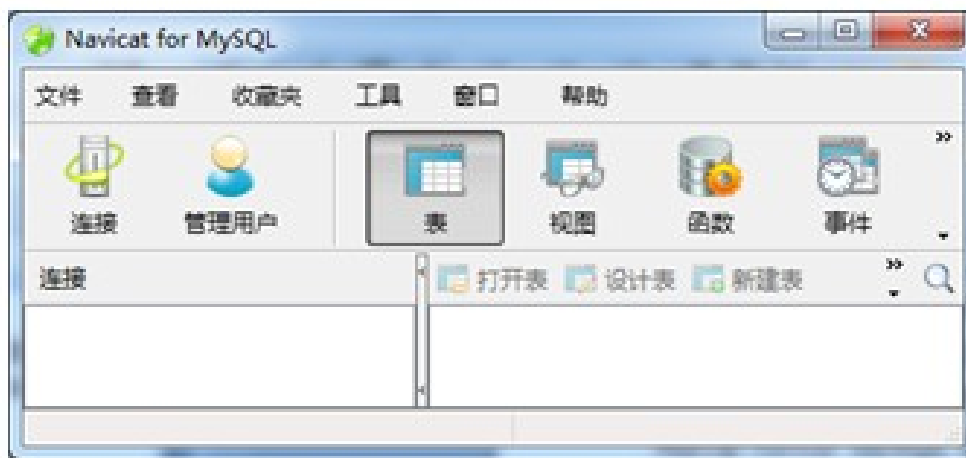


图 11.8 启动 navicat forMySQL 客户端管理工具





## 11.3 MySQL 客户端管理工具

### 1. 建立连接

启动 navicat for MySQL 后，单击主界面（图 11.8）上的“连接”选项卡，出现如图 11.9 所示意的建立连接对话框。在该对话框输入如下信息：

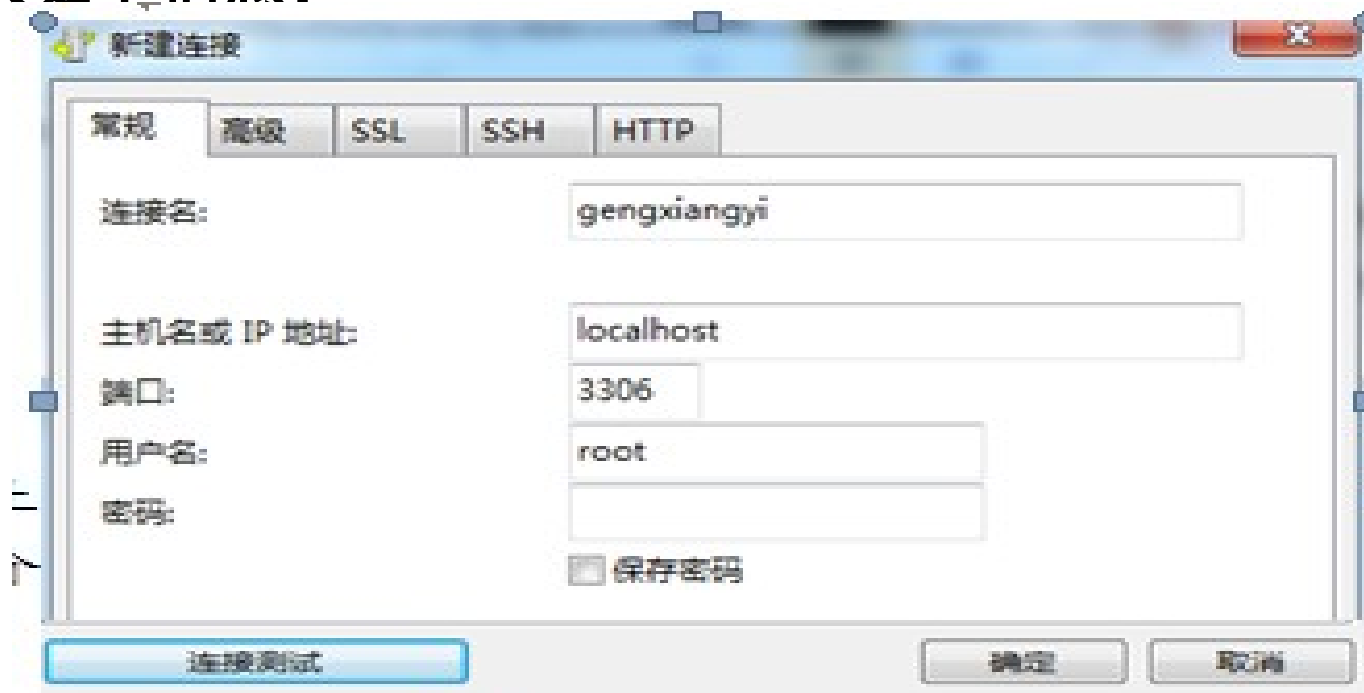


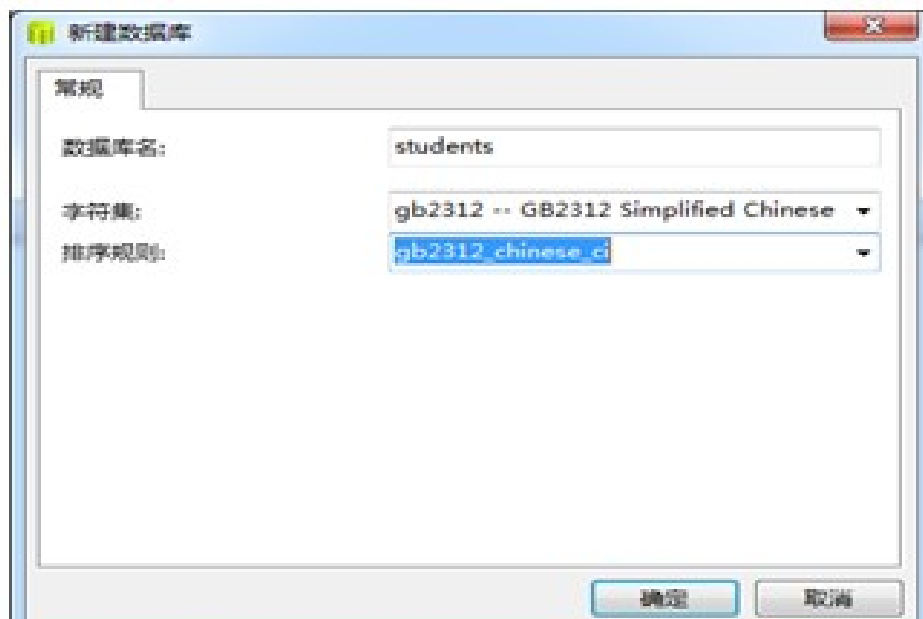
图 11.9 建立一个新连接



# 11.3 MySQL 客户端管理工具

## 2. 建立数据库

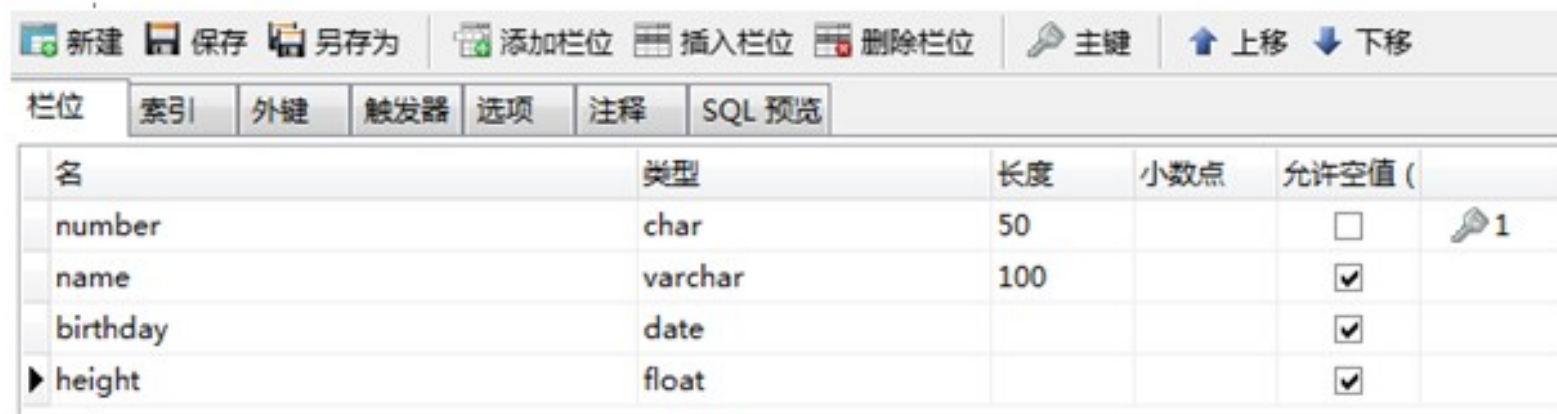
选择一个连接，比如 gengxiangyi，单击鼠标右键，选择“打开连接”，以便通过该连接在 MySQL 数据库服务器中建立数据库。打开 gengxiangyi 连接后，在 gengxiangyi 上单击鼠标右键，然后选择“新建数据库”，在弹出的新建数据库对话框中输入、选择有关信息，比如输入数据库的名称、选择使用的字符编码。这里建立的数据库的名字是 students，选择的的字符编码是 gb2312（GB2312 Simplified Chinese），如图 11.11 所示。



# 11.3 MySQL 客户端管理工具

## 3. 创建表

在主界面上，鼠标右键单击 gengxiangyi 连接下的数据库 students，选择“打开数据库”，主界面上 students 数据库将呈现打开（连接）状态，然后鼠标右键单击 students 下的“表”选项，选择新建表、弹出建表对话框（单击对话框上的添加栏位可以添加字段，即添加表中的列名），在该对话框中输入表的字段名（列名）与数据类型（如图 11.13 所示），其中 number 字段是主键，即要求记录的 number 的值必须互不相同。将该表保存为名字是 mess 的表。这时，数据库 students 的“表”下将有名字是 mess 的表。



新建	保存	另存为	添加栏位	插入栏位	删除栏位	主键	上移	下移
栏位	索引	外键	触发器	选项	注释	SQL 预览		
名	类型	长度	小数点	允许空值 (				
number	char	50		<input type="checkbox"/>		1		
name	varchar	100		<input checked="" type="checkbox"/>				
birthday	date			<input checked="" type="checkbox"/>				
height	float			<input checked="" type="checkbox"/>				

图 11.13 建立表



在“表”选择项单击鼠标左键，可以展开“表”，以便管理曾建立的表，比如管理曾建立的 mess 表。在 mess 表上单击鼠标右键，选择“打开表”，然后在弹出的对话框中向该表插入记录（单击 tab 键可以顺序地添加新记录，或单击界面下面的“+”或“-”号插入或删除记录，单击“√”保存当前的修改），如图 11.14 所示意。

number	name	birthday	height
R1001	张三	2000-12-12	1.78
R1002	李四	1999-10-09	1.68
* R1003	赵小五	1997-03-09	1.65

+

-

↶

✓

✕

↺

⌂

图 11.14 管理表



## 11.1.2 创建表

- 在 shop 数据库中创建名字为 goods 的表。在 shop 管理的“表”的界面上选择“使用设计器创建表”，然后单击界面中的“设计”菜单，将出现相应的建表界面，我们建立的表是 goods，该表的字段（属性）为：  
number(文本) name(文本) madeTime(日期) price(数字，双精度)。
- 在 shop 管理的“表”的界面上，用鼠标双击已创建的表可以为该表添加记录。

goods : 表		
	字段名称	数据类型
🔍	number	文本
	name	文本
	madeTime	日期/时间
	price	数字

图 11.2 goods 表及字段属性



# 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（如图 11.15）。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

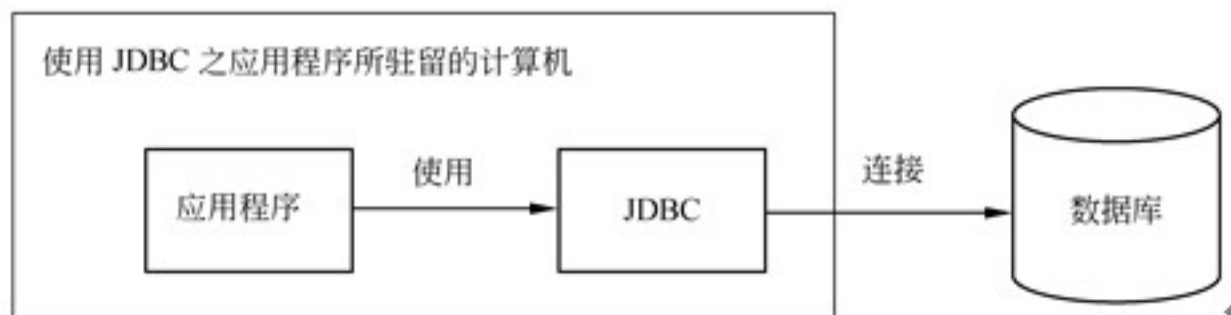


图 11.15 使用 JDBC 操作数据库

# 11.5 连接 MySQL 数据库

MySQL 数据库服务器启动后（见 11.2），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。目前在开发中常用的连接数据库的方式是加载 JDBC- 数据库驱动（连接器）（用 Java 语言编写的数据库驱动称作 JDBC- 数据库驱动），即 JDBC 调用本地的 JDBC- 数据库驱动和相应的数据库建立连接，如图 11.16 所示意。Java 运行环境将 JDBC- 数据库驱动转换为 DBMS（数据库管理系统）所使用的专用协议来实现和特定的 DBMS 交互信息。

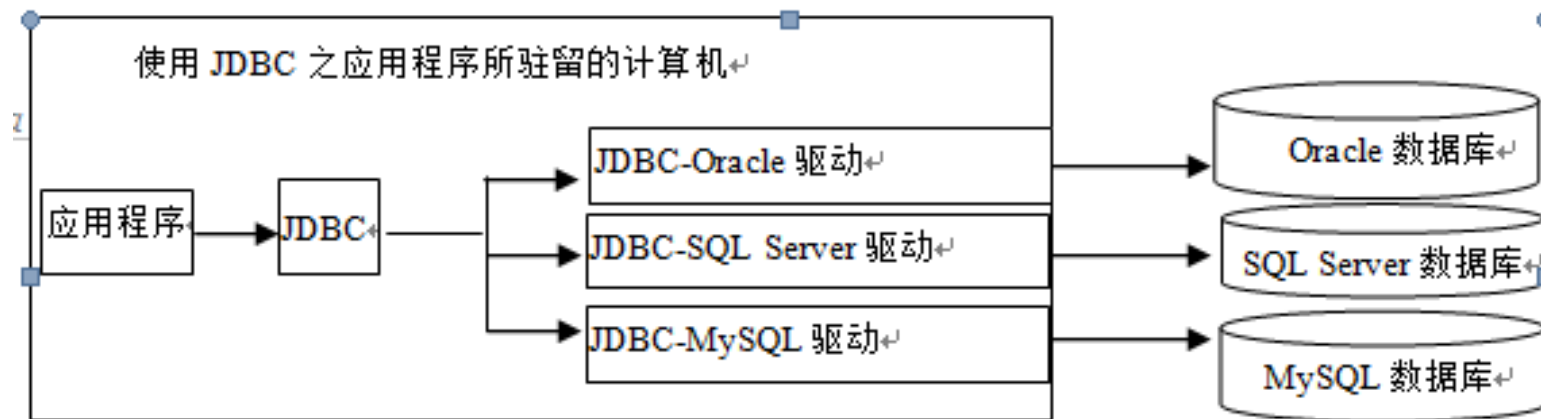


图 11.16 使用 JDBC-数据库驱动



# 1. 下载 JDBC-MySQL 数据库驱动

可以登录 MySQL 的官方网站：[www.mysql.com](http://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（即 `JAVA_HOME` 环境变量指定的 JDK，见第 1 章的 1.3.3），比如：`E:\jdk1.8\jre\lib\ext`。

作者将 `mysql-connector-java-5.1.40-bin.jar` 上传到了自己的网盘，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  
（连接 MySQL 数据库的 JDBC- 数据库驱动）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





## 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); }  
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 name, height FROM  
students");
```

内存的结果集对象 rs 列数只有两列，第一列是 name 列，第 2 列是 height 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

f: **例子1** 查询 students 数据库中 mess 表的全部记录（见 11.3 节建立的数据库）。效果如图 11.17（在后续的例子中，别忘记启动 MySQL 数据库服务器，见 11.2 节）

R1001	张三	2000-12-12	1.78
R1002	李四	1999-10-09	1.68
R1003	赵小五	1997-03-09	1.65

图 11.17 顺序查询





## 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

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

**例子2**将数据库连接的代码单独封装到一个[GetDatabaseConnection](#)类中。例子 2 随机查询 students 数据库中 mess 表的 2 条记录（见 11.3 节建立的数据库），首先将游标移动到最后一行，然后再获取最后一行的行号，以便获得表中的记录数目，本例子用到了第 8 章例子 18 中的[GetRandomNumber](#)类，该类的 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

表共有3条记录, 随机抽取2条记录:

R1002	李四	1999-10-09	1.68
R1003	赵小五	1997-03-09	1.65

图 11.18 随机抽取 2 条记录





## 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 name,height from mess where name='李四'
```

（2）字段值在某个区间范围，例如：


```
select * from mess where height>1.60 and height<=1.8
```

### 2. 排

序 **order by** 子语句对记录排序

```
select * from mess where name like '%林%' order by  
name
```





**例子3**查询 mess 表中姓张、身高大于 1.65，出生的年份在 2000 年或 2000 之前、月份在 7 月之后的学生，并按出生日期排序（在运行例子 2 程序前，我们使用 MySQL 客户端管理工具又向 mess 表添加了一些记录）。程序运行效果如图 11.19 所示（例子 3 中使用了[例子2中GetDBConnection类](#)）。

R1004	张常长	1999-12-10	1.76
R1001	张三	2000-12-12	1.78

图 11.19 条件查询与排序



# 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 mess 插入 2 条记录 ( 使用了 例子 2 中 GetDBConnection类)



# 11.8 使用预处理语句

## 11.8.1 预处理语句优

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 mess 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 mess 表添加记录并查询了姓张的记录（使用了例子2中[GetDBConnection](#)类）。





## 11.9 通用查询

本节的目的是编写一个类，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();
```

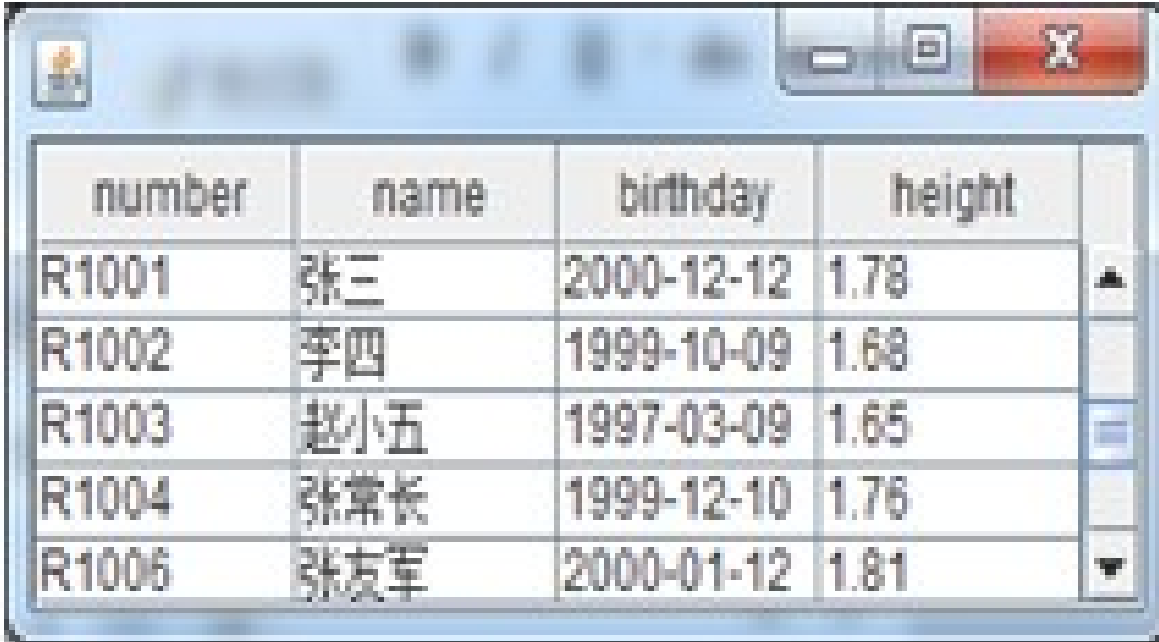
metaData 调用 getColumnName(int i) 方法就可以返回结果集 rs 中的第 i 列的名字：

```
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，见 9.7.2）显示查询到的记录。效果如

图 11.20



number	name	birthday	height
R1001	张三	2000-12-12	1.78
R1002	李四	1999-10-09	1.68
R1003	赵小五	1997-03-09	1.65
R1004	张常长	1999-12-10	1.76
R1006	张友军	2000-01-12	1.81

图 11.20 通用查询





# 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 `mess` 表中 `number` 字段是 R1001 的 `height` 的值减少 `n`，并将减少的 `n` 增加到字段是 R1002 的 `height` 上（使用了[例子2](#)中 `GetDBConnection` 类）。



# 11.11 连接 SQL Server 数据库

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);
}
```

链接：

<https://pan.baidu.com/s/1BRZiXIHP3VtdMiuOkrfj-Q>



## 11.12 连接 Derby 数据库

加载 Derby 数据库驱动程序的代码是：

```
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**使用了 Derby 数据库管理系统创建了名字是 students 的数据库，并在数据库中建立了 chengji 表，效果如图 11.22 。



张三	90.0
李斯	88.0
刘二	67.0

图 11.22 Derby 数据库



# 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：

(1) (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) 注册处理器 (2) 登录处理器

### 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。

