软件开发环境

主讲教师 刘凡

fanliu@hhu.edu.cn

第九章

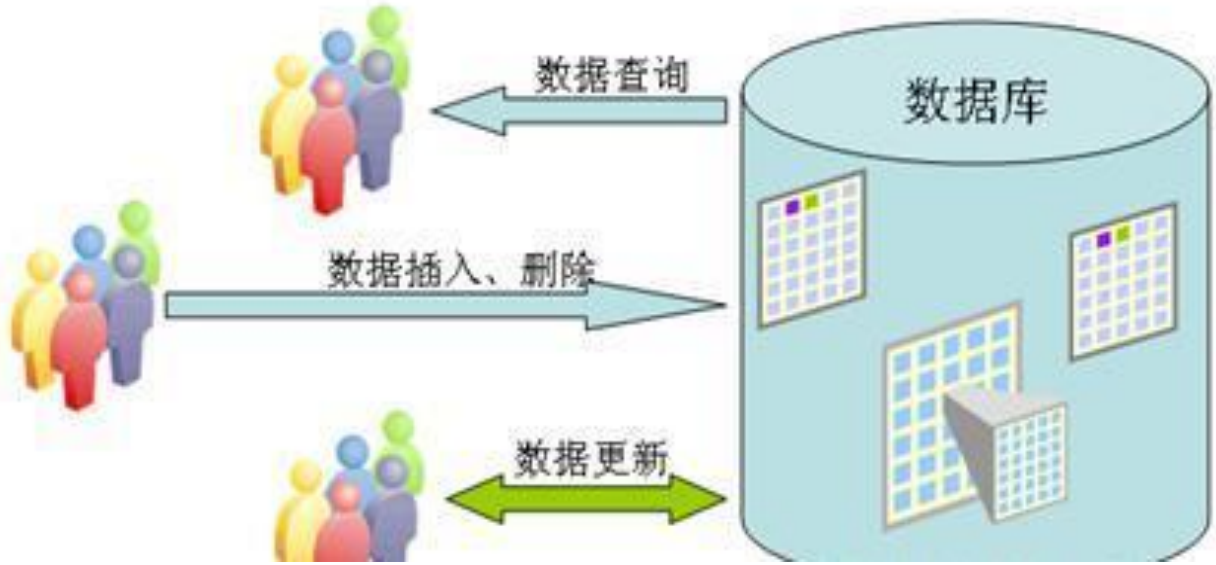
在JSP中使用数据库

本章主要内容

- 9.1 MySQL数据库管理系统
- 9.2 JDBC
- 9.3 连接MySQL数据库
- 9.4 查询记录
- 9.5 更新、添加、删除记录
- 9.6 用结果集操作数据库中的表
- 9.7 预处理语句
- 9.8 事务
- 9.9 分页显示记录
- 9.10 常见数据库连接
- 9.11 标准化考试

概述-数据库

数据库是提供数据的基地，它能保存数据并能使用户方便的访问数据。



概述-数据库

为了方便地管理数据，数据库中的数据按照一定的结构进行存储。

实时信息数据库表结构

测站基本属性表

测站编码	所在河流编码	行政区划编码
100	F00S	320115000000
101	H00A	320924000000

库站关系表

水库编码	入库标志	关联站码	测站编码
13123902	1	200	100
11212211	0	201	101

库站防洪指标表

水库类型	总库容（m3）	防洪库容（m3）	测站编码
1	1100.0	800.0	100
2	2200.0	1600.0	101

概述-数据库管理系统

- **DBMS 是 Data Base Management System) 的缩写。**
- **DBMS 是管理数据库软件的集合。**
- **DBMS 包含面向用户接口功能和面向系统维护功能。**
  - 面向用户接口功能是提供用户访问数据库的一些必要手段（如处理能力）。
  - 面向系统维护功能是为数据库管理者提供数据库的维护工具。（具体为数据库定义，数据装入，数据库操作、控制、监督、维护、恢复、通信等）

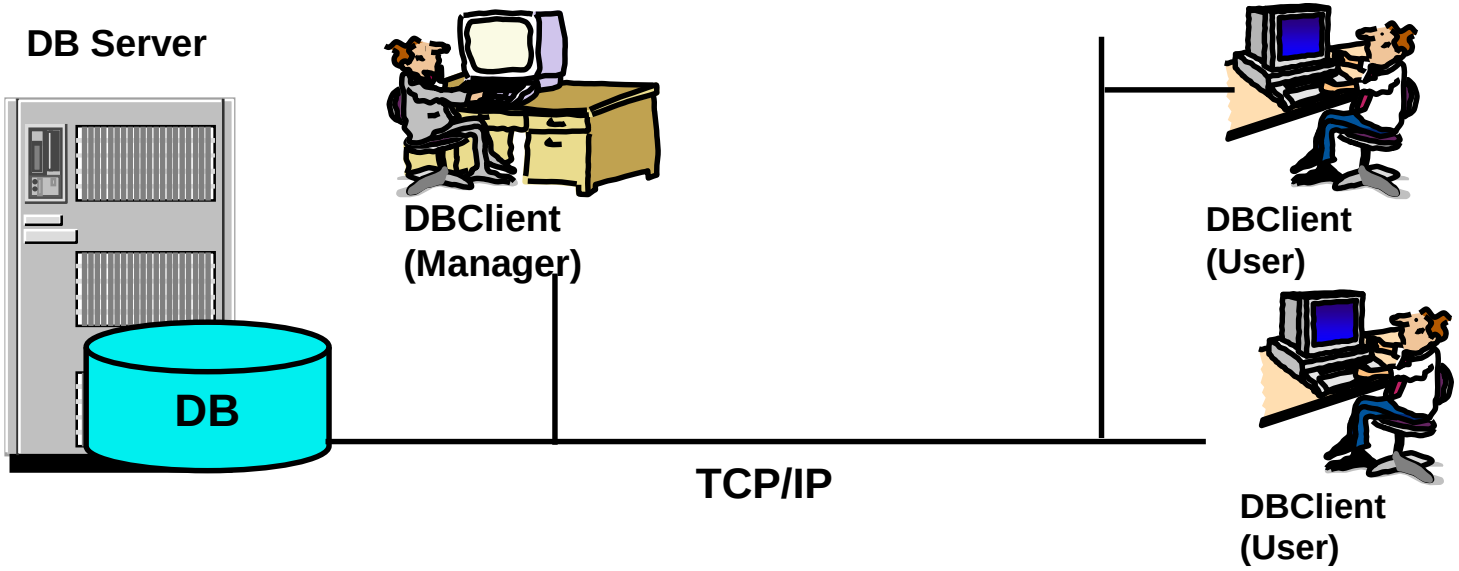
概述-常见的DBMS

目前常见的数据库管理系统 有：

- **Oracle**
- **Sybase**
- **Informix**
- **Microsoft SQL Server**
- **MySQL**

概述-DBMS的结构

- 通常通过两层结构来访问数据库，称之为 **Client/Server** 结构；
- 客户端软件发送数据操作请求到数据库服务器端，数据库服务器端将结果返回给客户端软件。
- 两层之间的语言用的是**SQL**语句。



概述-常用SQL语句

(1) 增：

sql="insert into 数据表(字段1,字段2,字段3…) values(值1,值2,值3…) "

(2) 删：

sql=" delete from 数据表 where 条件表达式"

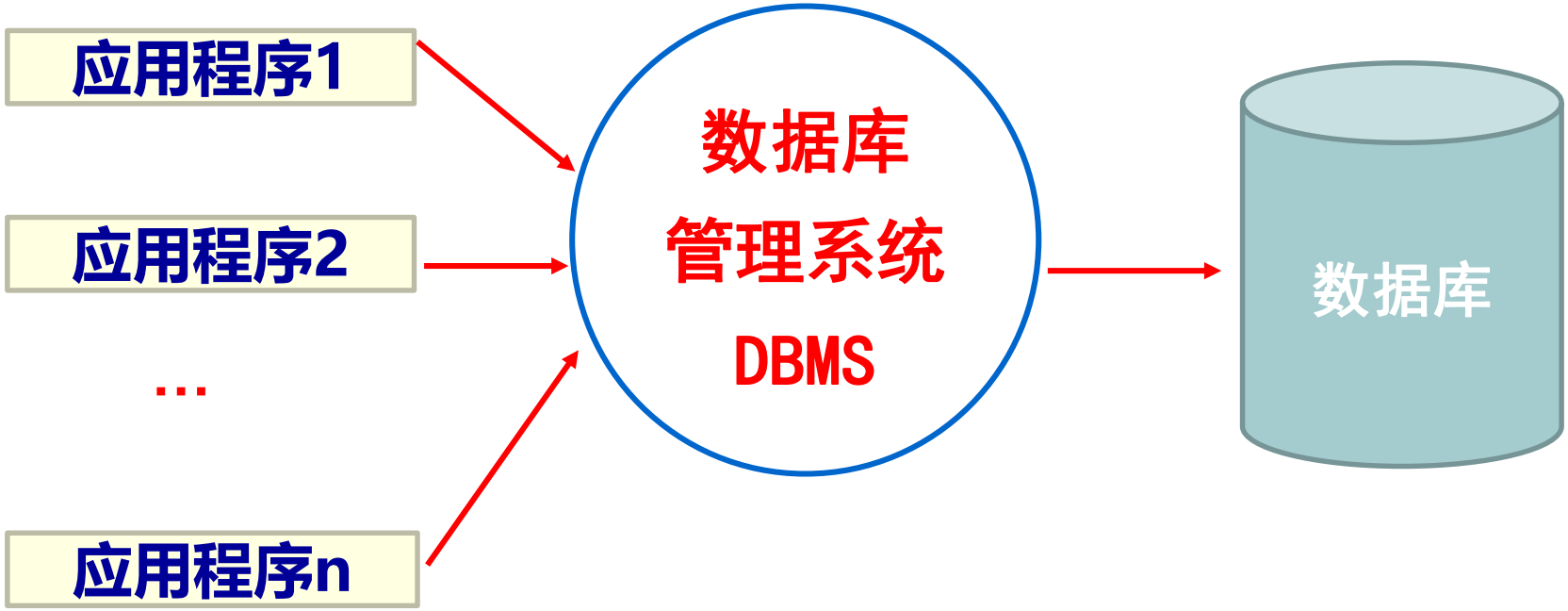
(3) 改：

sql="update 数据表 set 字段名=字段值 where 条件表达式"

(4) 查：

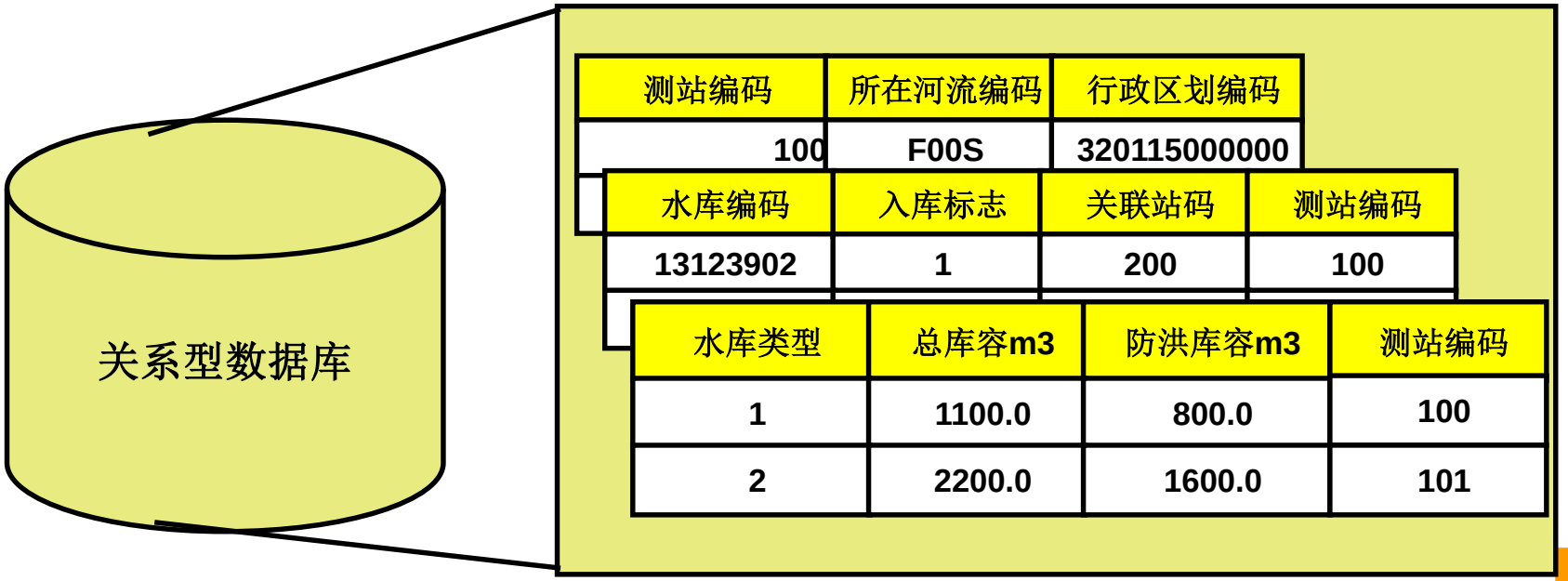
sql="select * from 数据表 where 字段名=字段值 orderby 字段名"

概述-概述-DBMS与数据库的关系



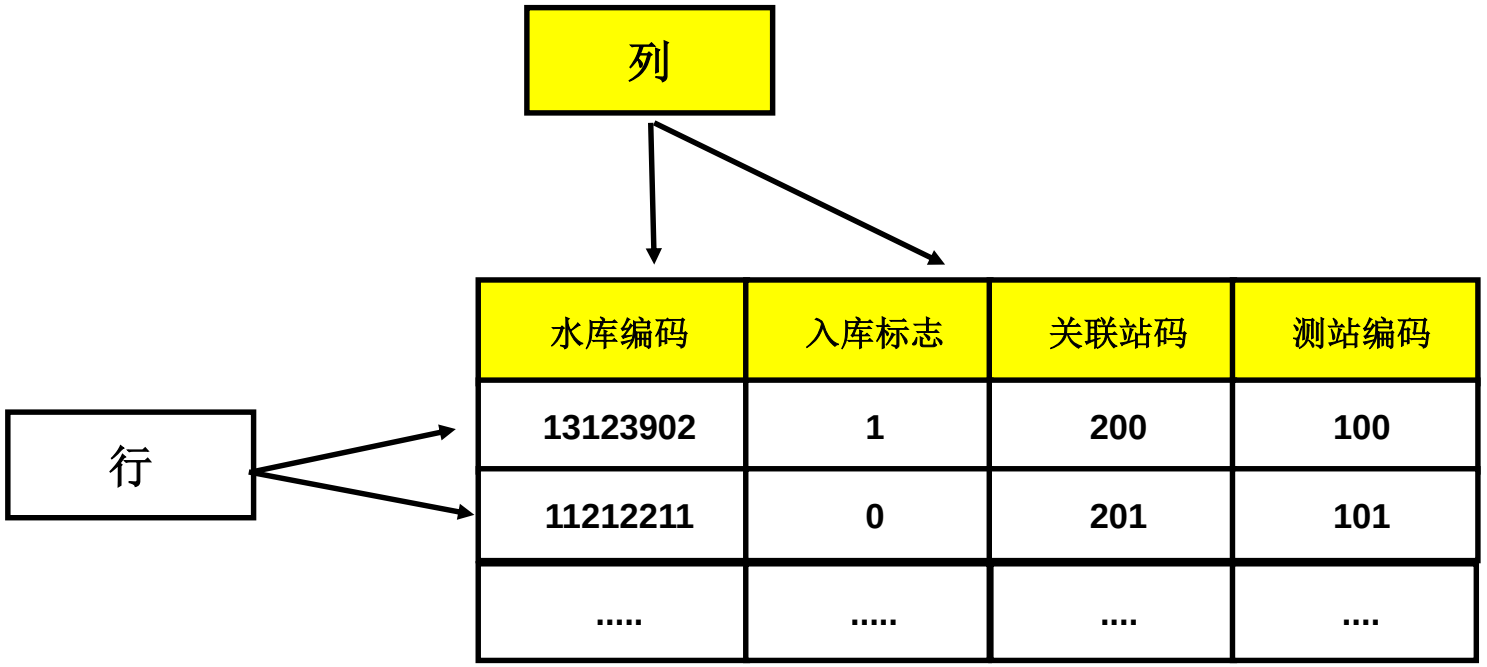
概述-关系型数据库

- 关系型数据库是基于**E.F. Codd**教授发明的关系型数学模型组织数据的数据库；
- 关系型数据库中所有的数据都是有表的形式来组织的，而表又是有列和行组成的。



概述-表

- 表是用两维关系来反映现实中的实体及它的属性。
- 表是由列和行组成的。



概述-列

- 表的每一列都代表了实体的一项属性。
- 每一列又叫表的一个字段。
- 每一个字段都有一个字段名。
- 每一个字段都只能包含同样数据类型的数据。
- 每一个字段长度是有限的。

入库标志为布尔类型

每一行水库编码字段是长度为20的字符串

水库编码	入库标志	关联站码	测站编码
13123902	1	200	100
11212211	0	201	101
.....	.....	....	....

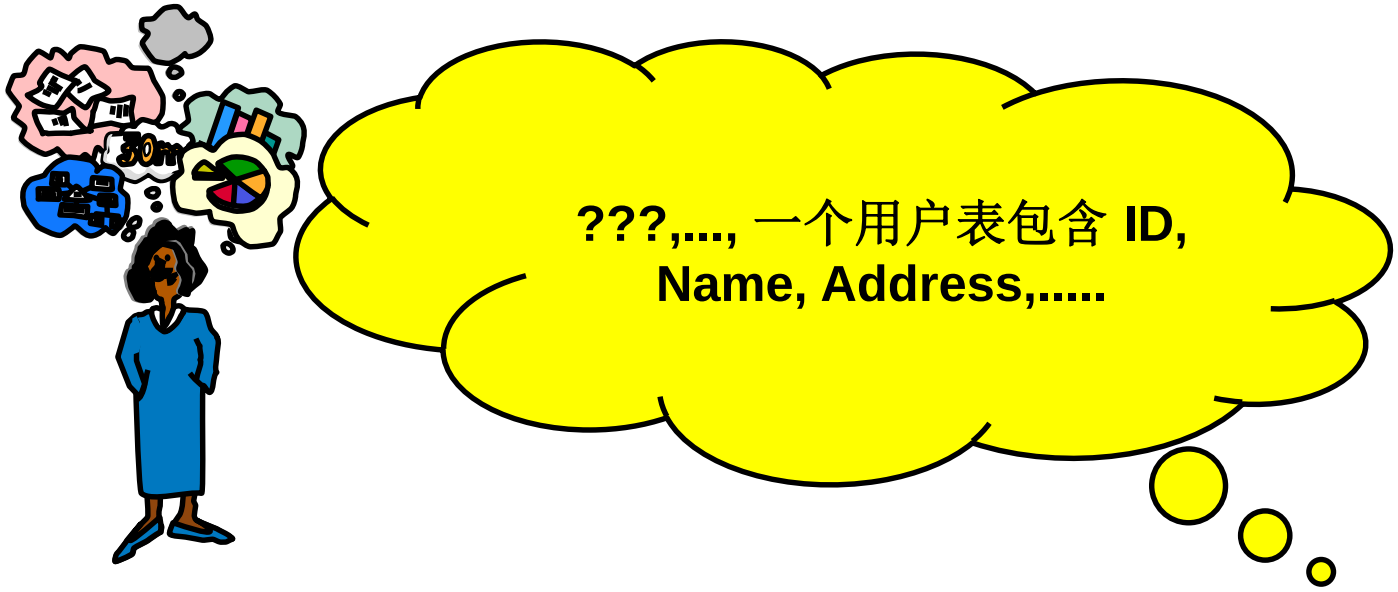
概述-行

- 表的每一行都代表实体的一个实例。

水库类型	总库容（m3）	防洪库容（m3）	测站编码	....
1	1100.0	800.0	100	...
2	2200.0	1600.0	101	...
.....	.....	....	....	...

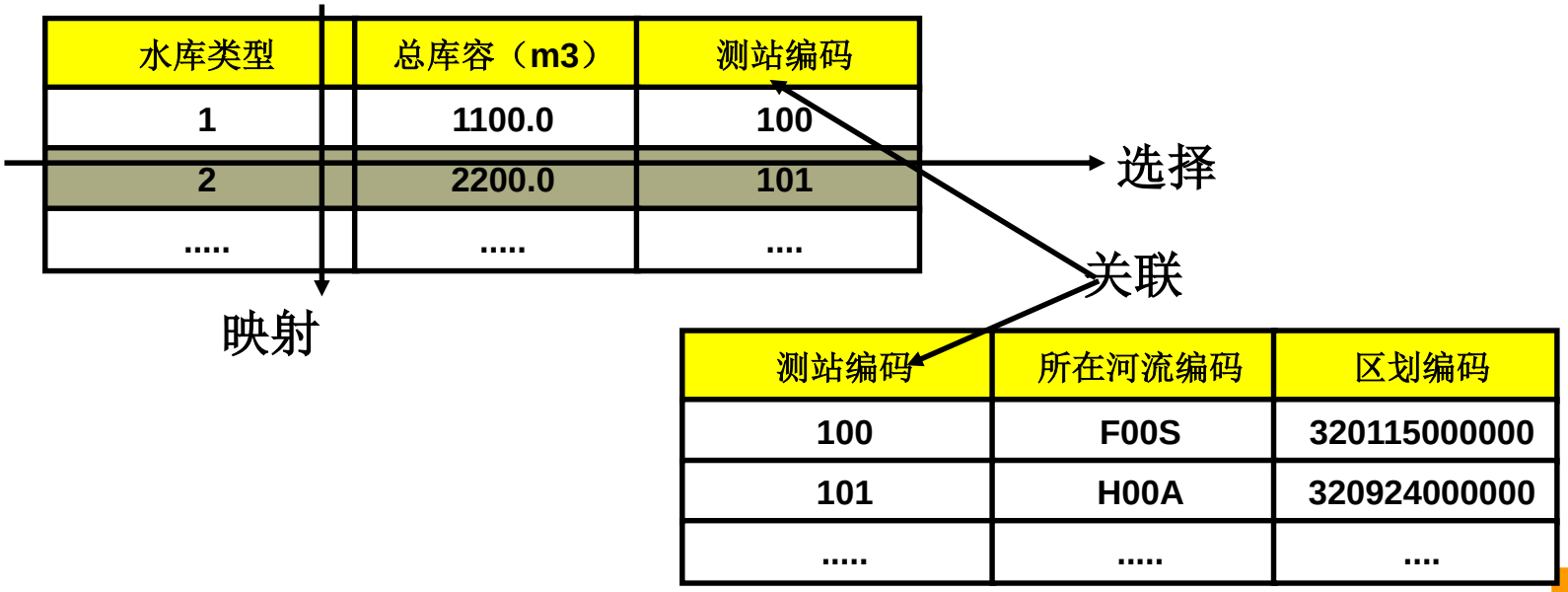
概述-怎么样定义一个表

- **创建一个表你必需考虑以下事情：**
  - 为表取一个名字并且名字不能重复；
  - 这个表有多少个字段；
  - 为每一个字段取一个名字；
  - 为每一个字段选取相应的数据类型和长度；



概述-表的操作

- 通俗的说，数据库是表的集合；数据库的操作主要是对表进行操作。表的操作有三种类型：
  - 选择
  - 映射
  - 关联



9.1 数据库系统-MySQL

- 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）是世界上最流行的免费下载的开源数据库管理系统。
- 目前许多Web开发项目都选用MySQL，其主要原因是MySQL的社区版（MySQL Community Edition）性能卓越，满足许多Web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.

9.1 数据库系统-MySQL

- 下载 登录www.mysql.com后选择导航条上的products，在出现的页面的左侧选择“MySQL Community Edition”或在出现的页面的右侧选择“下载MySQL社区版”。
- 安装将下载的mysql-5.6.16-win32.zip解压缩到本地计算机即可，比如解压缩到D:\。
- 启动打开MS-DOS命令行窗口，进入到bin目录中，执行MySQL安装目录的bin子目录中的mysqld.exe文件:

  即在命令行键入：

  mysqld 或 mysqld -nt

9.1 数据库系统-MySQL

启动MySQL数据库服务器后，就可以建立数据库，并在数据库中创建表。

- 可以下载图形界面的**MySQL**管理工具，并使用该工具进行创建数据库、在数据库中创建表等操作，**MySQL**管理工具有免费的也有需要购买的。
- 也可以使用**MySQL**提供的命令行工具进行创建数据库、在数据库中创建表等操作

9.1 数据库系统-MySQL

可以登录：http://www.navicat.com.cn/download下载试用版或购买商业版。下载navicat9_mysql_cs.exe后，安装即可

- 建立连接 启动navicat for MySQL 。我们建立的连接名称是gengxiangyi，用户名取root，密码是空，MySQL服务器占用的端口是3306.
- 建立数据库 在新建的连接（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.
- 创建表 在其“表”选择项上单击鼠标右键，选择“新建表”，将出现创建表的对话框。

9.1 数据库系统-MySQL

- MySQL提供的监视器（MySQL monitor），允许用户使用命令行方式管理数据库。如果有比较好的数据库知识，特别是SQL语句的知识，那么使用命令行方式管理MySQL数据库也是很方便的.
- 需要再打开一个MS-DOS命令行窗口，并使用MS-DOS命令进入到bin目录中，然后使用默认的root用户启动MySQL监视器（在安装MySQL时root用户是默认的一个用户，没有密码）。命令如下：

mysql –u root

9.1 数据库系统-MySQL

- 启动MySQL监视器后就可以使用SQL语句进行创建数据库、建表等操作。在MS-DOS命令行窗口输入SQL语句需要用“;”号结束，在编辑SQL语句的过程中可以使用\c终止当前SQL语句的编辑。可以把一个完整的SQL语句命令分成几行来输入，最后用分号作结束标志即可。
- 使用MySQL监视器创建一个名字为Book的数据库，在当前MySQL监视器占用的命令行窗口输入创建数据库的SQL语句:

```
create database Book;
```

9.1 数据库系统-MySQL

- 首先进入该数据库(即使用数据库)： use Book

- 在数据库Book建立一个名字为bookList表

```
CREATE TABLE bookList (
 ISBN varchar(100) not null ,
 name varchar(100) CHARACTER SET gb2312,
 price float ,
 PRIMARY KEY (ISBN)
);
```

- 插入记录的SQL语句:

```
insert into bookList values('7-302-01465-5','高等数学',28.67);
```

- 查询记录的SQL语句

```
select * from bookList;
```

9.1 数据库系统-MySQL

可以事先将需要的SQL语句保存在一个扩展名是.sql的文本文件中，然后在MySQL监视器占用的命令行窗口使用source命令导入.sql的文本文件中的SQL语句. group.sql文本文件的内容如下：

```
drop table carList ;
create table carList(
number char(60) CHARACTER SET gb2312 not null,
name char(50) CHARACTER SET gb2312 ,
price float,
year date,
PRIMARY KEY(number)
);
insert into carList values('加A89CQ8','奔驰','820000','2015-12-26');
insert into carList values('洲C12456','宝马','620000','2015-10-10');
select * from carList;
```

然后在当前MySQL监视器占用的命令行窗口键入如下命令：

source d:/1000/group.sql

9.1 数据库系统-MySQL

➢删除数据库的命令：drop database <数据库名>，例如：删除名为tiger的数据库：

```
drop database tiger;
```

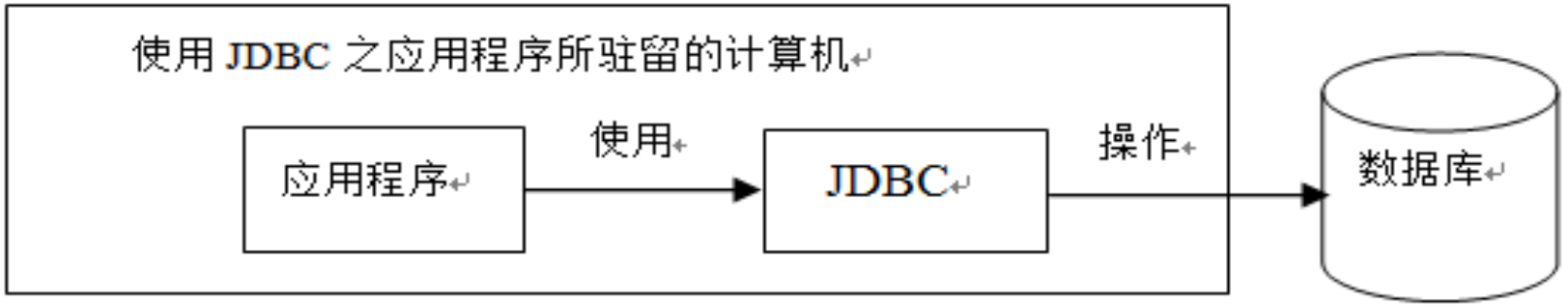
➢删除表的命令：drop table <表名>，例如，使用book数据库后，执行

```
drop table booklist;
```

将删除book数据库中的bookList表。

9.2 JDBC

JDBC（Java DataBase Connectivity）提供了访问数据库的API，即由一些Java类和接口组成，是Java运行平台的核心类库中的一部分。在JSP中可以使用JDBC实现对数据库中表的记录的查询、修改和删除等操作。



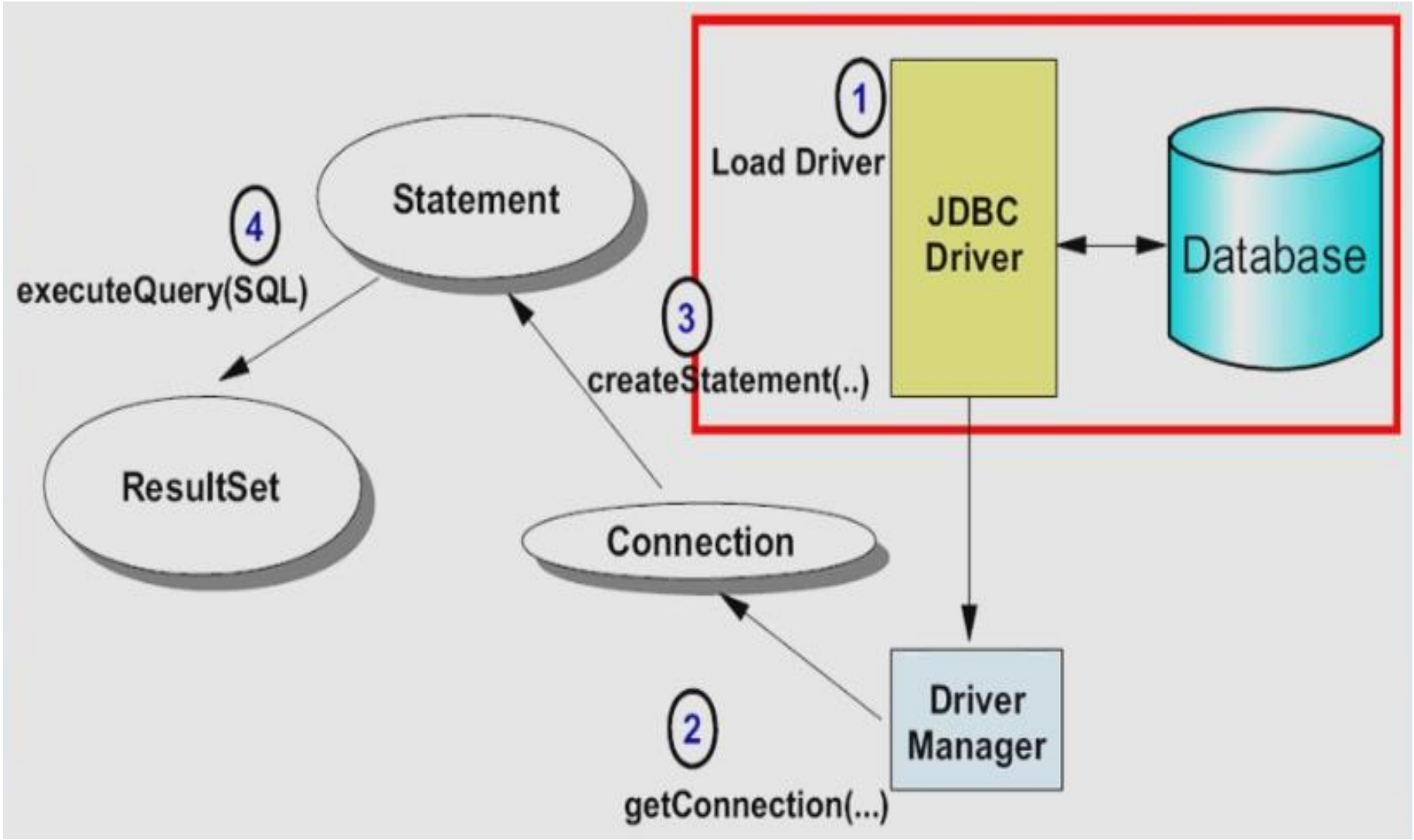
我们经常使用JDBC进行如下的操作：

1. 与一个数据库建立连接。
2. 向已连接的数据库发送SQL语句。
3. 处理SQL语句返回的结果。

9.2 JDBC-API中重要的接口和类(java.sql包中)

名称	描述
DriverManager类	依据数据库的不同, 管理JDBC驱动
Connection接口	负责连接数据库, 并担任传送数据的任务
Statement接口	由Connection产生, 负责执行SQL语句
PreparedStatement接口	创建一个可以编译的SQL语句对象, 该对象可以被多次执行, 以提高执行的效率,
ResultSet接口	负责保存Statement执行后所产生的查询结果

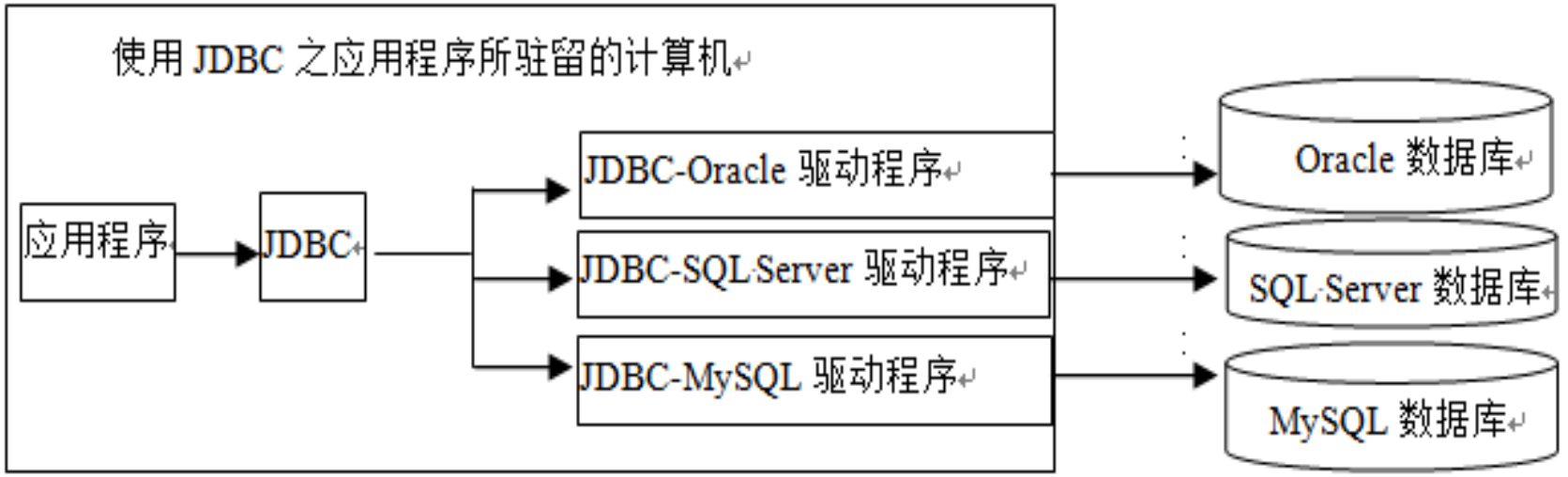
9.2 JDBC-通过JDBC操作数据库步骤图



9.3 连接MySQL数据库

使用JDBC-数据库驱动程序方式和数据库建立连接需要经过2个步骤：

- 加载JDBC-数据库驱动程序
- 和指定的数据库建立连接



9.3 连接MySQL数据库

1.加载JDBC-数据库驱动程序

- 教材下载的是mysql-connector-java-5.1.28.zip，解压后获得mysql-connector-java-5.1.28-bin.jar文件就是连接MySQL数据库的JDBC-数据库驱动程序。
- 将该驱动程序复制到Tomcat服务器所使用的JDK的扩展目录中**D:\jdk1.7\jre\lib\ext**
- 或Tomcat服务器安装目录的\common\lib文件夹中

加载MySQL的JDBC-数据库驱动程序代码如下：

```
try{

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

9.3 连接MySQL数据库

2. 建立连接-方式1

- 为了能和MySQL数据库服务器管理的数据库建立连接，必须保证该MySQL数据库服务器已经启动，如果没有更改过MySQL数据库服务器的配置，那么该数据库服务器占用的端口是3306。假设MySQL数据库服务器所驻留的计算机的IP地址是192.168.100.1
- 使用Connection getConnection(String,String,String)建立连接

```
try{  String uri = "jdbc:mysql:// 192.168.100.1:3306/warehouse";
      String user ="root";
      String password ="99";
      con = DriverManager.getConnection(uri,user,password);
   }
catch(SQLException e){
     System.out.println(e);}
```

9.3 连接MySQL数据库

2. 建立连接-方式2

- 使用Connection getConnection(String)方法建立连接

```
try{
 String uri =
 "
jdbc:mysql://192.168.100.1:3306/warehouse?user=root&password=99
";
      con = DriverManager.getConnection(uri);
   }
 catch(SQLException e){
      System.out.println(e);
 }
```

如果root用户没有设置密码，那么将上述uri中的&password=99更改为：

```
&password=
```

9.3 连接MySQL数据库

2. 建立连接-方式3

- 避免操作数据库出现中文乱码，需要使用
- Connection getConnection(String) 方法建立连接，连接代码是（假设用户是root，其密码是99）：

```
String uri = "jdbc:mysql://127.0.0.1/warehouse?"+
"user=root&password=99&characterEncoding=gb2312";
con = DriverManager.getConnection(uri);
```

- 用户要和连接MySQL驻留在同一计算机上，使用的IP地址可以是127.0.0.1或localhost。另外，由于3306是MySQL数据库服务器的默认端口号，连接数据库时允许应用程序省略默认的3306端口号.

9.3 连接MySQL数据库

3. MySQL乱码解决方案_1

- 数据库和表使用支持中文的字符编码，在创建数据库时指定数据库使用的字符编码：

  create 数据库名 CHARACTER SET字符编码

- 创建表时，可以指定某个字段使用的字符编码：

  字段名 类型 CHARACTER SET字符编码

```
create people CHARACTER SET gb2312
create table myList (
id int,
name  varchar(100) CHARACTER SET gb2312,
PRIMARY KEY (id)
);
```

9.3 连接MySQL数据库

3. MySQL乱码解决方案_2:连接数据库支持中文编码

- JSP中连接MySQL数据库时，需要使用

  Connection getConnection(java.lang.String) 建立连接，

- 而且向该方法参数传递的字符串是：

- **"jdbc:mysql://地址/数据库?user=用户&password=密码** &characterEncoding=gb2312";

9.3 连接MySQL数据库

example7_1.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.sql.*" %>
<HTML><body bgcolor=#EEDDFF>
 <% Connection con;
    Statement sql;
    ResultSet rs;
    try{  Class.forName("com.mysql.jdbc.Driver");
    }
    catch(Exception e){
      out.println("忘记把MySQL数据库的JDBC-数据库驱动程序复制到
    JDK的扩展目录中");
    }
try { String uri= "jdbc:mysql://127.0.0.1/warehouse";
          String user="root";
          String password="";
con=DriverManager.getConnection(uri,user,password);
          sql=con.createStatement();
          rs=sql.executeQuery("SELECT * FROM product ");
```

9.3 连接MySQL数据库

example7_1.jsp

```
out.print("<table border=2>");
        out.print("<tr>");
          out.print("<th width=100>"+"产品号");
          out.print("<th width=100>"+"名称");
          out.print("<th width=50>"+"生产日期");
          out.print("<th width=50>"+"价格");
        out.print("</TR>");
        while(rs.next()){
          out.print("<tr>");
            out.print("<td >"+rs.getString(1)+"</td>");
            out.print("<td >"+rs.getString(2)+"</td>");
            out.print("<td >"+rs.getDate("madeTime")+"</td>");
            out.print("<td >"+rs.getFloat("price")+"</td>");
          out.print("</tr>") ;
        }
        out.print("</table>");
        con.close();
    }
    catch(SQLException e){
        out.print(e);}
 %>
</body></HTML>
```

9.4 查询记录

- 和数据库建立连接后，就可以使用JDBC提供的API和数据库交互信息。比如查询、修改和更新数据库中的表等
- JDBC和数据库表进行交互的主要方式是使用SQL语句，JDBC提供的API可以将标准的SQL语句发送给数据库，实现和数据库的交互。

9.4 查询记录

1. 结果集与查询:

- 让连接对象con调用方法createStatement()创建执行SQL语句的Statement对象：

  Statement sql=con.createStatement();

- sql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：

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

9.4 查询记录

1. 结果集与查询：

- 对于：**ResultSet rs= sql.executeQuery("SELECT name,price FROM product");**
- 内存的结果集对象rs只有两列，第1列是name列、第2列是price列。
- ResultSet结果集一次只能看到一个数据行，使用next()方法走到下一数据行，获得一行数据后，ResultSet结果集可以使用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等等）或列名传递给getXxx方法的参数即可

9.4 查询记录

表 7.1　ResultSet 类的若干方法

返回类型	方法名称
boolean	next()
byte	getByte(int columnIndex)
Date	getDate(int columnIndex)
double	getDouble(int columnIndex)
float	getFloat(int columnIndex)
int	getInt(int columnIndex)
long	getLong(int columnIndex)
String	getString(int columnIndex)
byte	getByte(String columnName)
Date	getDate(String columnName)
double	getDouble(String columnName)
float	getFloat(String columnName)
int	getInt(String columnName)
long	getLong(String columnName)
String	getString(String columnName)

9.4 查询记录

```
int id = rs.getInt("id") ;
String name = rs.getString("name") ;
String password = rs.getString("password") ;
int age = rs.getInt("age") ;
```

通过 ResultSet 接口中的 getXxx()方法，可以取出数据，按类型取

getInt、getString、getFloat…

在开发中往往使用表格对数据显示进行处理

✓总是可以使用**getString()**返回字段值的串表示

9.4 查询记录

2. 结果集的列名与列的数目

- 程序查询的时候，为了代码更加容易维护，希望知道数据库表的字段（列）的名字以及表的字段的个数，那么一个办法是使用返回到程序中的结果集来获取相关的信息。
- (1) 得到元数据对象metaData

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

- (2)得到结果集的列的个数，即共有几列

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

- (3)结果集rs中的第i列的名字：

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

9.4 查询记录

3. 随机查询

- 使用Result的next()方法顺序地查询数据，但有时候我们需要在结果集中前后移动、或显示结果集指定的一条记录等等。这时，必须要返回一个可滚动的结果集。为了得到一个可滚动的结果集，必须使用下述方法先获得一个Statement对象：

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

- 然后，根据参数的type、concurrency的取值情况，stmt返回相应类型的结果集：

ResultSet re=stmt.executeQuery(SQL语句);

9.4 查询记录

3. 随机查询

type的取值决定滚动方式，取值可以是：

- **ResultSet.TYPE_FORWORD_ONLY ： 结果集的游标只能向下滚动。**
- **ResultSet.TYPE_SCROLL_INSENSITIVE ： 结果集的游标可以上下移动, 当数据库变化时，当前结果集不变。**
- **ResultSet.TYPE_SCROLL_SENSITIVE ： 返回可滚动的结果集，当数据库变化时，当前结果集同步改变。**

9.4 查询记录

3. 随机查询

concurrency取值决定是否可以用结果集更新数据库

- **ResultSet.CONCUR_READ_ONLY**：不能用结果集更新数据库中的表。
- **ResultSet.CONCUR_UPDATETABLE**：能用结果集更新数据库中的表。

9.4 查询记录

3. 随机查询

滚动查询经常用到ResultSet的下述方法：

✧**public boolean previous()**：将游标向上移动，当移到结果集第一行之前时返回**false.**

✧**public void beforeFirst**：将游标移动到结果集的初始位置，即在第一行之前。

✧**public void afterLast()**：将游标移到结果集最后一行之后。

✧**public void first()**：将游标移到结果集的第一行。

✧**public void last()**：将游标移到结果集的最后一行。

✧**public boolean isAfterLast()**：判断游标是否在最后一行之后。

✧**public boolean isBeforeFirst()**：判断游标是否在第一行之前

✧**public boolean ifFirst()**：判断游标是否指向结果集的第一行。

✧**public boolean isLast()**：判断游标是否指向结果集的最后一行。

✧**public int getRow()**：得到当前游标所指行的行号，行号从**1**开始，如果没有返回**0**

✧**public boolean absolute(int row)**：将游标移到参数**row**指定的行号。注意：如果**row**取负值，就是倒数的行数，**absolute(-1)**表示移到最后一行。当移动到第一行前面或最后一行的后面时返回**false**。

9.4 查询记录

4. 条件查询

- select… from 表 where 字段 满足的条件
- select * from product where price > 2000 and price<5000
- select * from product where name = ‘java’
- 模糊查询，使用“%”表示零个或多个字符，用“_”表示任意一个字符：
- "select * from product where name like ‘%里%’ "

9.4 查询记录

5. 排序查询

- 可以在SQL语句中使用ORDER BY子语句，对记录排序。在下面的例子中，使用SQL语句的ORDER BY子语句查询所有同学的成绩，可以选择按3科的总分从低到高排列记录、按姓氏拼音排序或英语成绩排序。例如，按总成绩排序查询的SQL语句：
- SELECT * FROM student ORDER BY 总分

9.4 查询记录

example7_2.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=cyan><font size=2>
<form
   action="queryByConditionServlet?dataBase=warehouse&tableName=
   product"
   method="post" >
  查询warehouse数据库product表中
  <br>price值大于<input type=text name="price" value =-1 size=6>的记
   录
  <br>输入用户名：<input type=text name="user" value=root
   size=5>(默认root)
  <br>输入密码：  <input type="password" name="password"
   size=3>(默认空)
 <br><input type=submit name="sub" value="提交">
</form>
</font></body></HTML>
```

9.4 查询记录

Example7_2_bean.java

```
package mybean.data;
public class Example7_2_Bean{
   String []columnName ;          //存放列名
   String [][] tableRecord=null;   //存放查询到的记录
   public Example7_2_Bean() {
      tableRecord = new String[1][1];
      columnName = new String[1];
   }
   public void setTableRecord(String [][] s){
      tableRecord=s;
   }
   public String [][] getTableRecord(){
      return tableRecord;
   }
   public void setColumnName(String [] s) {
      columnName = s;
   }
   public String [] getColumnName() {
      return columnName;
   }
}
```

9.4 查询记录

web.xml

```
<servlet>
    <servlet-name>queryByConditionServlet</servlet-name>
   <servlet-class>myservlet.control.Example7_2_Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>queryByConditionServlet</servlet-name>
  <url-pattern>/queryByConditionServlet</url-pattern>
</servlet-mapping>
```

9.4 查询记录

Example7_2_servlet.java

```
public void doPost(HttpServletRequest request,HttpServletResponse
    response)throws ServletException,IOException{
     Example7_2_Bean resultBean=null;
    try{  resultBean=(Example7_2_Bean)request.getAttribute("resultBean");
          if(resultBean==null){
             resultBean=new Example7_2_Bean(); //创建Javabean对象
             request.setAttribute("resultBean",resultBean);
          }
     } catch(Exception exp){
          resultBean=new Example7_2_Bean();  //创建Javabean对象
          request.setAttribute("resultBean",resultBean);
     }
    try{  Class.forName("com.mysql.jdbc.Driver");catch(Exception e){}
    String number = request.getParameter("price");
    if(number==null||number.length()==0) return;
    String dataBase = request.getParameter("dataBase");
    String tableName = request.getParameter("tableName");
    String user = request.getParameter("user");
    String password = request.getParameter("password");
    float p = Float.parseFloat(number);
```

9.4 查询记录

Example7_2_servlet.java

```
try{
        String uri="jdbc:mysql://127.0.0.1/"+dataBase;
        con=DriverManager.getConnection(uri,user,password);
        sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                ResultSet.CONCUR_READ_ONLY);
        String condition ="SELECT * FROM "+tableName+" where price > "+p;
        rs=sql.executeQuery(condition);
        ResultSetMetaData metaData = rs.getMetaData();
        int columnCount = metaData.getColumnCount(); //得到结果集的列数
        String []columnName = new String[columnCount];
        for(int i=0;i<columnName.length;i++) {
          columnName[i] = metaData.getColumnName(i+1); //得到列名
        }
        resultBean.setColumnName(columnName);  //更新Javabean数据模型
        rs.last();
        int rowNumber=rs.getRow();  //得到记录数
        String [][] tableRecord=resultBean.getTableRecord();
```

9.4 查询记录

Example7_2_servlet.java

```
tableRecord = new String[rowNumber][columnCount];
        rs.beforeFirst();
        int i=0;
        while(rs.next()){
          for(int k=0;k<columnCount;k++)
            tableRecord[i][k] = rs.getString(k+1);
          i++;
        }
        resultBean.setTableRecord(tableRecord); //更新Javabean数据模型
        con.close();
        RequestDispatcher dispatcher=
        request.getRequestDispatcher("showRecord.jsp");
        dispatcher.forward(request,response);
    }
    catch(SQLException e){
        System.out.println(e);
    }
  }
```

9.4 查询记录

showRecod.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="resultBean"
    class="mybean.data.Example7_2_Bean" scope="request"/>
<HTML><body bgcolor=#DEEFF9><font size=2>
 <table border=1>
 <%  String []columnName=resultBean.getColumnName();
 %>
    <tr>
 <%  for(String s:columnName) {
 %>    <th><%= s %></th>
 <%  }
 %>  </tr>
 <%  String [][] record = resultBean.getTableRecord();
    for(int i=0;i<record.length;i++) {
 %>    <tr>
 <%    for(int j=0;j<record[i].length;j++) {
 %>       <td><%= record[i][j] %> </td>
 <%    }
 %>    </tr>
 <%  }
 %>
 </table></font></body></HTML>
```

9.5 更新、添加、删除记录

- **Statement对象调用方法：**

public int executeUpdate（String sqlStatement）；

实现对数据库表中记录的字段值的更新、添加和删除记录。

- **executeUpdate("UPDATE product SET price = 6866 WHERE name='海尔电视机'");**
- **executeUpdate("INSERT INTO students VALUES ('012','神通手机’,'2015-2-26',2687)");**
- **executeUpdate("DELETE FROM product WHERE number = '888' ");**

9.5 更新、添加、删除记录

example7_3.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=pink ><font size=2>
<form
   action="insertServlet?dataBase=warehouse&tableName=product"
   method=post>
添加新记录:
<table border=1>
<tr><td> 产品号：</td><td><Input type="text"
   name="number"></td></tr>
<tr><td>名称：</td><td><Input type="text"
   name="name"></td></tr>
<tr><td>生产日期：</td><td><Input type="text"
   name="madeTime"></td></tr>
<tr><td>价格：</td><td><Input type="text"
   name="price"></td></tr>
</table>
<br><input type="submit" name="b" value="提交">
</font></body></HTML>
```

9.5 更新、添加、删除记录

- 模型（Javabean）

Javabean 模型例子 2 中的 Example7_2_Bean.java 完全相同，创建的 Javabean 模型的 id 是 resultBean，scope取值是request。

9.5 更新、添加、删除记录

web.xml

```
<servlet>
    <servlet-name>insertServlet</servlet-name>
   <servlet-class>myservlet.control.Example7_3_Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>insertServlet</servlet-name>
  <url-pattern>/insertServlet</url-pattern>
</servlet-mapping>
```

9.5 更新、添加、删除记录

Example7_3_servlet.java

```
public void doPost(HttpServletRequest request,HttpServletResponse
    response) throws ServletException,IOException{
     Example7_2_Bean resultBean=null;
 try{  resultBean=(Example7_2_Bean)request.getAttribute("resultBean");
        if(resultBean==null){
           resultBean=new Example7_2_Bean(); //创建Javabean对象
           request.setAttribute("resultBean",resultBean);
        }
     }
     catch(Exception exp){
        resultBean=new Example7_2_Bean();  //创建Javabean对象
        request.setAttribute("resultBean",resultBean);
     }
    try{  Class.forName("com.mysql.jdbc.Driver");}catch(Exception e){}
    request.setCharacterEncoding("gb2312");
    String dataBase = request.getParameter("dataBase");
    String tableName = request.getParameter("tableName");
    String nu=request.getParameter("number");
    String na=request.getParameter("name");
    String mT=request.getParameter("madeTime");
```

9.5 更新、添加、删除记录

Example7_3_servlet.java

```
String pr=request.getParameter("price");
    if(nu==null||nu.length()==0) {
      fail(request,response,"添加记录失败,必须给出记录");
      return;}
    float p=Float.parseFloat(pr);
    String condition = "INSERT INTO "+tableName+" VALUES"+
     "("+"'"+nu+"','"+na+"','"+mT+"',"+p+")";
    Connection con;
    Statement sql;
    ResultSet rs;
    try{
        String uri="jdbc:mysql://127.0.0.1/"+dataBase+"?"+
                "user=root&password=&characterEncoding=gb2312";
        con=DriverManager.getConnection(uri);
        sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                        ResultSet.CONCUR_READ_ONLY);
        sql.executeUpdate(condition);
        rs=sql.executeQuery("SELECT * FROM "+tableName);
        ResultSetMetaData metaData = rs.getMetaData();
        int columnCount = metaData.getColumnCount(); //得到结果集的列数
```

9.5 更新、添加、删除记录

Example7_3_servlet.java

```
String []columnName = new String[columnCount];
        for(int i=0;i<columnName.length;i++) {
          columnName[i] = metaData.getColumnName(i+1); //得到列名}
        resultBean.setColumnName(columnName);  //更新Javabean数据模型
        rs.last();
        int rowNumber=rs.getRow();  //得到记录数
        String [][] tableRecord=resultBean.getTableRecord();
        tableRecord = new String[rowNumber][columnCount];
        rs.beforeFirst();
        int i=0;
        while(rs.next()){
         for(int k=0;k<columnCount;k++)
           tableRecord[i][k] = rs.getString(k+1);
         i++; }
        resultBean.setTableRecord(tableRecord); //更新Javabean数据模型
        con.close();
RequestDispatcher dispatcher=request.getRequestDispatcher("showRecord.jsp");
        dispatcher.forward(request,response);
    }catch(SQLException e){
        System.out.println(e);
        fail(request,response,"添加记录失败:"+e.toString());
    }   }
```

9.5 更新、添加、删除记录

Example7_3_servlet.java

```
public  void  doGet(HttpServletRequest request,HttpServletResponse
    response)
        throws ServletException,IOException{
     doPost(request,response);
  }
  public void fail(HttpServletRequest request,HttpServletResponse
    response,
                  String backNews) {
      response.setContentType("text/html;charset=GB2312");
      try {
       PrintWriter out=response.getWriter();
       out.println("<html><body>");
       out.println("<h2>"+backNews+"</h2>") ;
       out.println("返回");
       out.println("<a href =example7_3.jsp>输入记录</a>");
       out.println("</body></html>");
      }
      catch(IOException exp){}
   }
}
```

9.5 更新、添加、删除记录

showRecod.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="resultBean"
    class="mybean.data.Example7_2_Bean" scope="request"/>
<HTML><body bgcolor=#DEEFF9><font size=2>
 <table border=1>
 <%  String []columnName=resultBean.getColumnName();
 %>
    <tr>
 <%  for(String s:columnName) {
 %>    <th><%= s %></th>
 <%  }
 %>  </tr>
 <%  String [][] record = resultBean.getTableRecord();
    for(int i=0;i<record.length;i++) {
 %>    <tr>
 <%    for(int j=0;j<record[i].length;j++) {
 %>       <td><%= record[i][j] %> </td>
 <%    }
 %>    </tr>
 <%  }
 %>
 </table></font></body></HTML>
```

9.6 用结果集操作数据库中的表

- 尽管可以用SQL语句对数据库中表进行更新、插入操作，但也可以使用内存中ResultSet结果集对底层数据库表进行更新和插入操作（这些操作由系统自动转化为相应的SQL语句），优点是不必熟悉有关更新、插入的SQL语句，而且方便编写代码，缺点是必须要事先返回结果集。

9.6 用结果集操作数据库中的表

使用结果集更新数据库表中第n行记录中某列的值的步骤是：

1.结果集rs的游标移动到第n行

```
rs.absolute(n);
```

2.结果集将第n行的某列的列值更新

例如 更新列名是columnName的日期值是x指定的值：

```
updateDate(String columnName, Date x)
```

3.更新数据库中的表

最后，结果集调用updateRow()方法用结果集中的第n行更新数据库表中的第n行记录。

以下代码片段更新product表中的第3行记录的name列（字段）的值。

```
rs.absolute(3);
rs.updateString("name", "IBM PC");
rs.updateRow();
```

9.6 用结果集操作数据库中的表

使用结果集向数据库表中插入（添加）一行记录步骤是：

1.结果集rs的游标移动到插入行（用于构建要插入的行的暂存区域）

```
rs.moveToInsertRow();
```

2.更新插入行的列值

例如：

```
rs.updateString(1, "c002");
rs.updateString(2, "IBM iPad");
rs.updateDate(3,Date());
rs.updateDouble(4, 5356);
```

3. 插入记录

最后，结果集调用**insertRow()**方法用结果集中的插入行向数据库表中插入一行新记录。

9.6 用结果集操作数据库中的表

example7_4.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=#AAFFEE><font size=2>
<form
    action="insertBySetServlet?dataBase=warehouse&tableName=produ
    ct" method=post>
<b>添加新记录:<br>
产品号：<input type="text" name="number" size=20>
<br>名称：<input type="text" name="name" size=22>
<br>生产日期(日期必须用-或/格式)：
<br><input type="text" name="madeTime" size=18>
<br>价格：<input type="text" name="price" size=12>
<br><input type="submit" name="b" value="提交">
</font></body></HTML>
```

9.6 用结果集操作数据库中的表

web.xml

```
<servlet>
    <servlet-name>insertBySetServlet</servlet-name>
   <servlet-class>myservlet.control.Example7_4_Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>insertBySetServlet</servlet-name>
  <url-pattern>/insertBySetServlet</url-pattern>
</servlet-mapping>
```

9.6 用结果集操作数据库中的表

- 模型（Javabean）

Javabean 模型与例子 2 中的 Example7_2_Bean.java 完全相同，创建的 Javabean 模型的 id 是 resultBean，scope取值是request。

9.6 用结果集操作数据库中的表

Example7_4_servlet.java

```
public void doPost(HttpServletRequest request,HttpServletResponse
    response) throws ServletException,IOException{
     Example7_2_Bean resultBean=null;
 try{  resultBean=(Example7_2_Bean)request.getAttribute("resultBean");
         if(resultBean==null){
            resultBean=new Example7_2_Bean(); //创建Javabean对象
            request.setAttribute("resultBean",resultBean);
         }
     }catch(Exception exp){
         resultBean=new Example7_2_Bean();  //创建Javabean对象
         request.setAttribute("resultBean",resultBean);}
    try{  Class.forName("com.mysql.jdbc.Driver");}
    catch(Exception e){}
    request.setCharacterEncoding("gb2312");
    String dataBase = request.getParameter("dataBase");
    String tableName = request.getParameter("tableName");
    String number=request.getParameter("number");
    String name=request.getParameter("name");
    String madeTime=request.getParameter("madeTime");
    String pr=request.getParameter("price");
```

9.6 用结果集操作数据库中的表

Example7_4_servlet.java

```
if(number==null||number.length()==0) {
      fail(request,response,"添加记录失败,必须给出记录");
      return;}
   float price=Float.parseFloat(pr);
   Connection con;
   Statement sql;
   ResultSet rs;
   try{
      String uri="jdbc:mysql://127.0.0.1/"+dataBase+"?"+
                 "user=root&password=&characterEncoding=gb2312";
      con=DriverManager.getConnection(uri);
      sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                              ResultSet.CONCUR_READ_ONLY);
      rs=sql.executeQuery("SELECT * FROM "+tableName);
      rs.moveToInsertRow();     //rs的游标移动到插入行
      rs.updateString(1,number); //更新结果集
      rs.updateString(2, name); //更新结果集
      String [] str=madeTime.split("[-/]");//输入的日期必须用-或/格式
      int year=Integer.parseInt(str[0]);
      int month=Integer.parseInt(str[1]);
      int day=Integer.parseInt(str[2]);
```

9.6 用结果集操作数据库中的表

Example7_4_servlet.java

```
Calendar calendar =Calendar.getInstance();
calendar.set(year,month-1,day);
Date date=new java.sql.Date(calendar.getTimeInMillis());
rs.updateDate(3,date);     //更新结果集
rs.updateDouble(4,price);  //更新结果集
rs.insertRow();     //向表插入一行记录
rs=sql.executeQuery("SELECT * FROM "+tableName);
ResultSetMetaData metaData = rs.getMetaData();
int columnCount = metaData.getColumnCount(); //得到结果集的列数
String []columnName = new String[columnCount];
for(int i=0;i<columnName.length;i++) {
  columnName[i] = metaData.getColumnName(i+1); //得到列名
}
resultBean.setColumnName(columnName);  //更新Javabean数据模型
rs.last();
int rowNumber=rs.getRow();  //得到记录数
String [][] tableRecord=resultBean.getTableRecord();
tableRecord = new String[rowNumber][columnCount];
rs.beforeFirst();
int i=0;
```

9.6 用结果集操作数据库中的表

Example7_4_servlet.java

```
while(rs.next()){
        for(int k=0;k<columnCount;k++)
          tableRecord[i][k] = rs.getString(k+1);
        i++;
      }
      resultBean.setTableRecord(tableRecord); //更新Javabean数据模型
      con.close();
      RequestDispatcher dispatcher=
      request.getRequestDispatcher("showRecord.jsp");
      dispatcher.forward(request,response);  //转发
   }
   catch(SQLException e){
      System.out.println(e);
      fail(request,response,"添加记录失败:"+e.toString());
   }
```

9.7 预处理语句

- 对于JDBC，如果使用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。

9.7 预处理语句

- 对于JDBC，如果使用Connection和某个数据库建立了连接对象con，那么 con就可以调用

  PreparedStatement pre=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pre 对象可以随时调用下列方法都可以使得该底层内部命令被数据库执行，提提高了数据库的访问速度：
  - **boolean execute()**
  - **int executeUpdate()**
  - **ResultSet executeQuery()**

9.7 预处理语句

example7_5.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.sql.*" %>
<HTML><body bgcolor=#EEDDFF>
 <% Connection con;
   PreparedStatement sql; //预处理语句
   ResultSet rs;
   try{  Class.forName("com.mysql.jdbc.Driver");
   }catch(Exception e){
     out.println("忘记把MySQL数据库的JDBC-数据库驱动程序复制到JDK的扩展
    目录中");}
   try { String uri= "jdbc:mysql://127.0.0.1/warehouse";
         String user="root";
         String password="";
         con=DriverManager.getConnection(uri,user,password);
         sql=con.prepareStatement("SELECT * FROM product");
         rs=sql.executeQuery();
         out.print("<table border=2>");
         out.print("<tr>");
```

9.7 预处理语句

example7_5.jsp

```
out.print("<table border=2>");
        out.print("<tr>");
          out.print("<th width=100>"+"产品号");
          out.print("<th width=100>"+"名称");
          out.print("<th width=50>"+"生产日期");
          out.print("<th width=50>"+"价格");
        out.print("</TR>");
        while(rs.next()){
          out.print("<tr>");
            out.print("<td >"+rs.getString(1)+"</td>");
            out.print("<td >"+rs.getString(2)+"</td>");
            out.print("<td >"+rs.getDate("madeTime")+"</td>");
            out.print("<td >"+rs.getFloat("price")+"</td>");
          out.print("</tr>") ; }
        out.print("</table>");
        con.close(); }
   catch(SQLException e){ out.print(e);}
 %>
</body></HTML>
```

9.7 预处理语句

- 在对SQL进行预处理时可以使用通配符“？”来代替字段的值

```
prepareStatement pre=
con.prepareStatement("SELECT * FROM product WHERE price < ? ");
```

- 调用相应的方法设置通配符“？”，代表具体的值

```
pre.setDouble(1,6565);//指定上述预处理语句pre中第1个通配符“？”代表的值是6565
```

- 通配符按着它们在预处理的“SQL语句”中从左至右依次出现的顺序分别被称做第1个、第2个… 第m个通配符。
- 预处理语句设置通配符“？”的值的常用方法有：

```
void setDate(int parameterIndex,Date x)
void setDouble(int parameterIndex,double x)
void setFloat(int parameterIndex,float x)
```

9.7 预处理语句

example7_6.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=#AAFFEE><font size=2>
<form action="preparedServlet" method=post>
<b>输入主键number是<input type="text" name="number"
    size=10><br>
  的name,madeTime和price的更新值：<br>
  name:<input type="text" name="name" size=7><br>
  madeTime:<input type="text" name="madeTime" size=10><br>
  price:<input type="text" name="price" size=8><br>
 <input type="submit" name="b" value="提交">
</font></body></HTML>
```

9.7 预处理语句

```
<servlet>
    <servlet-name>preparedServlet</servlet-name>
    <servlet-class>myservlet.control.Example7_6_Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>preparedServlet</servlet-name>
  <url-pattern>/preparedServlet</url-pattern>
</servlet-mapping>
```

9.7 预处理语句

- 模型（Javabean）

Javabean 模型与例子 2 中的 Example7_2_Bean.java 完全相同，创建的 Javabean 模型的 id 是 resultBean，scope取值是request。

9.7 预处理语句

example7_6_servlet.java

```
package myservlet.control;
import mybean.data.Example7_2_Bean; //引入例子2中的Javabean模型
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Calendar;
public class Example7_6_Servlet extends HttpServlet{
  public void init(ServletConfig config) throws ServletException{
    super.init(config);}
  public void doPost(HttpServletRequest request,HttpServletResponse
   response) throws ServletException,IOException{
    Example7_2_Bean resultBean=null;
 try{  resultBean=(Example7_2_Bean)request.getAttribute("resultBean");
       if(resultBean==null){
          resultBean=new Example7_2_Bean(); //创建Javabean对象
          request.setAttribute("resultBean",resultBean);
       }
    }catch(Exception exp){
       resultBean=new Example7_2_Bean();  //创建Javabean对象
       request.setAttribute("resultBean",resultBean);
    }
```

9.7 预处理语句

example7_6_servlet.java

```
try{  Class.forName("com.mysql.jdbc.Driver");}catch(Exception e){}
    request.setCharacterEncoding("gb2312");
    String number=request.getParameter("number");
    String name=request.getParameter("name");
    String madeTime=request.getParameter("madeTime");
    String pr=request.getParameter("price");
    if(number==null||number.length()==0) {
      fail(request,response,"更新记录失败,必须给出记录");
      return; }
    float price=Float.parseFloat(pr);
    String [] str=madeTime.split("[-/]");//输入的日期必须用-或/格式
    int year=Integer.parseInt(str[0]);
    int month=Integer.parseInt(str[1]);
    int day=Integer.parseInt(str[2]);
    Calendar calendar =Calendar.getInstance();
    calendar.set(year,month-1,day);
    Date date=new Date(calendar.getTimeInMillis());
    Connection con;
    PreparedStatement sql; //预处理语句 ;
    ResultSet rs;
```

9.7 预处理语句

example7_6_servlet.java

```
try{
        String uri="jdbc:mysql://127.0.0.1/warehouse?"+
                   "user=root&password=&characterEncoding=gb2312";
        con=DriverManager.getConnection(uri);
       String condition =
     "UPDATE product SET name = ?,price = ?,madeTime = ?  WHERE number=?";
        sql=con.prepareStatement(condition);
        sql.setString(1,name);   //设置第1个统配符“？”代表的具体值
        sql.setFloat(2,price);   //设置第2个统配符“？”代表的具体值
        sql.setDate(3,date);     //设置第3个统配符“？”代表的具体值
        sql.setString(4,number); //设置第4个统配符“？”代表的具体值
        sql.executeUpdate();
        sql=con.prepareStatement("select * from product");
        rs=sql.executeQuery();
        ResultSetMetaData metaData = rs.getMetaData();
        int columnCount = metaData.getColumnCount(); //得到结果集的列数
        String []columnName = new String[columnCount];
        for(int i=0;i<columnName.length;i++) {
          columnName[i] = metaData.getColumnName(i+1); //得到列名
        }
```

9.7 预处理语句

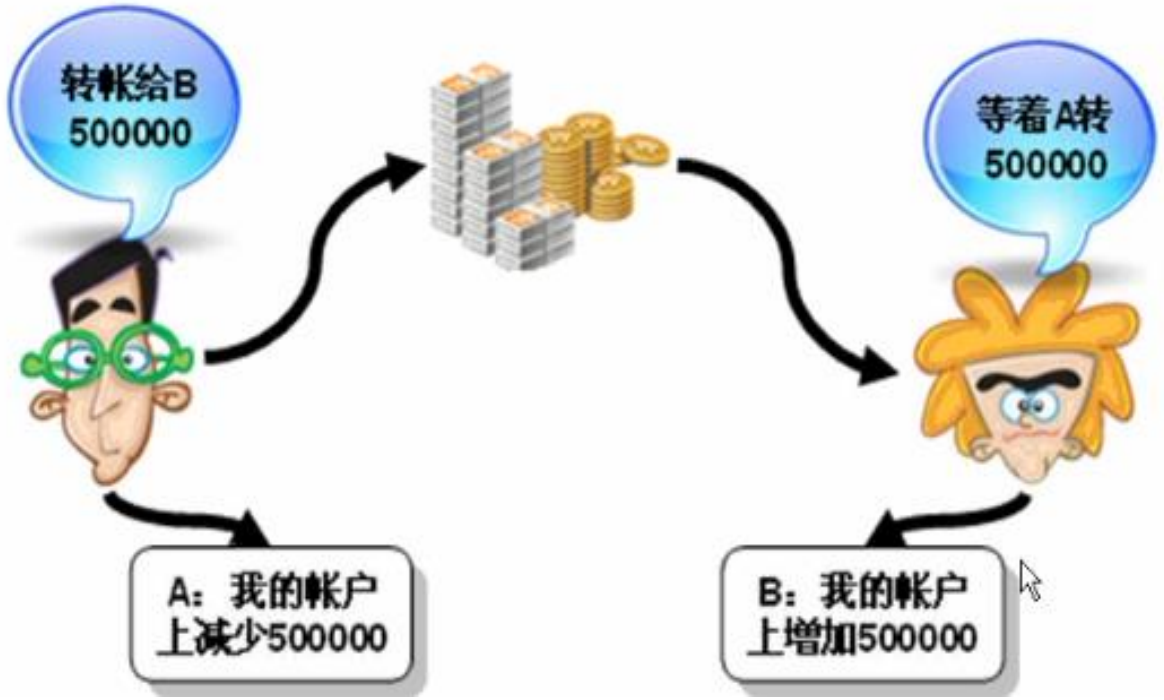
example7_6_servlet.java

```
resultBean.setColumnName(columnName);  //更新Javabean数据模型
        rs.last();
        int rowNumber=rs.getRow();  //得到记录数
        String [][] tableRecord=resultBean.getTableRecord();
        tableRecord = new String[rowNumber][columnCount];
        rs.beforeFirst();
        int i=0;
        while(rs.next()){
          for(int k=0;k<columnCount;k++)
            tableRecord[i][k] = rs.getString(k+1);
          i++;  }
        resultBean.setTableRecord(tableRecord); //更新Javabean数据模型
        con.close();
        RequestDispatcher dispatcher=
        request.getRequestDispatcher("showRecord.jsp");
        dispatcher.forward(request,response);  //转发
    }catch(SQLException e){
        System.out.println(e);
        fail(request,response,"更新记录失败:"+e.toString());
    } }
```

9.8 事务

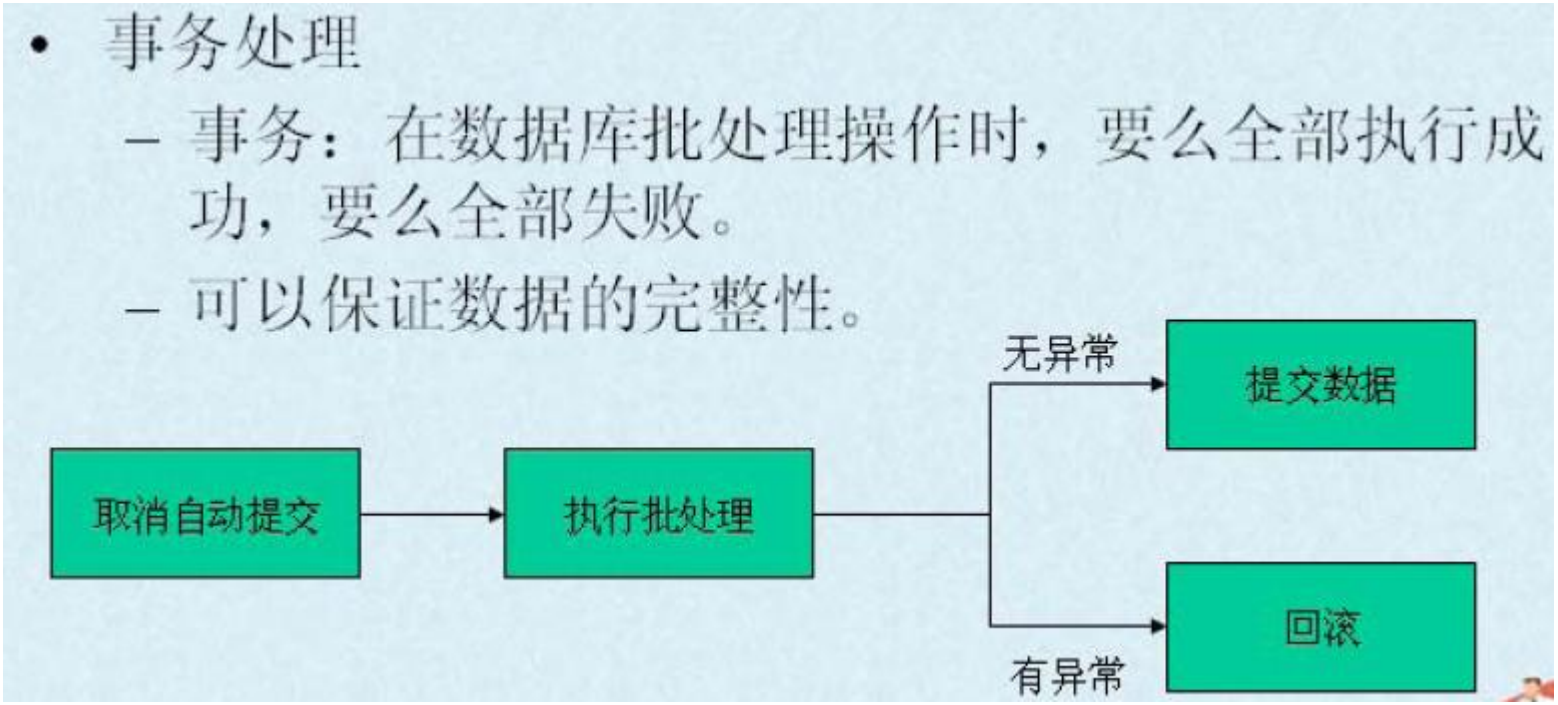
- 事务由一组SQL语句组成，所谓“事务处理”是指：应用程序保证事务中的SQL语句要么全部都执行，要么一个都不执行。

如图所示，A转帐和B接帐分别是两个不可再分的操作，但是如果A的转帐失败，则B的操作也肯定无法成功。



9.8 事务

- 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。



9.8 事务

- 事务是保证数据库中数据完整性与一致性的重要机制。事务处理步骤如下：

1. 连接对象使用setAutoCommit(boolean autoCommit)方法将参数autoCommit取值为false来关闭自动提交模式：

```
con.setAutoCommit(false);
```

2. commit()方法

con调用commit()方法就是让事务中的SQL语句全部生效。

3. rollback()方法

只要事务中任何一个SQL语句没有生效,就抛出SQLException异常。在处理SQLException异常时,必须让con调用rollback()方法,其作用是撤消事务中成功执行过的SQL语句对数据库数据所做的更新、插入或删除操作.

9.8 事务

example7_7.jsp

```
<%@ page import="java.sql.*" %>
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=AAEF9E><font size=2>
<%  Connection con=null;
   Statement sql;
   ResultSet rs;
   try { Class.forName("com.mysql.jdbc.Driver");
   }
   catch(ClassNotFoundException e){}
   try{  int n=50;
         String uri=
        "jdbc:mysql://127.0.0.1/bank?"+
        "user=root&password=&characterEncoding=gb2312";
         con=DriverManager.getConnection(uri);
         con.setAutoCommit(false);      //关闭自动提交模式
         sql=con.createStatement();
      rs=sql.executeQuery("SELECT userMoney FROM user WHERE name='geng'");
         rs.next();
         double gengMoney=rs.getDouble("userMoney");
    rs=sql.executeQuery("SELECT userMoney FROM user WHERE name='zhang'");
      rs.next();
```

9.8 事务

example7_7.jsp

```
double zhangMoney=rs.getDouble("userMoney");
        out.print("转账前geng的userMoney的值是"+gengMoney+"<br>");
        out.print("转账前zhang的userMoney的值是"+zhangMoney+"<br>");
        gengMoney=gengMoney-n;
        if(gengMoney>=0) {
          zhangMoney=zhangMoney+n;
sql.executeUpdate("UPDATE user SET userMoney ="+gengMoney+" WHERE name='geng'");
sql.executeUpdate("UPDATE user SET userMoney="+zhangMoney+" WHERE name='zhang'");
        con.commit(); //开始事务处理}
       rs=sql.executeQuery("SELECT * FROM user WHERE name='zhang'||name='geng'");
        out.println("<b>转账后的情况如下:<br>");
        while(rs.next()) {
           out.print(rs.getString(1)+" ");
           out.print(rs.getString(2));
           out.print("<br>");}
        con.close();}
    catch(SQLException e){
        try{ con.rollback();     //撤消事务所做的操作}
        catch(SQLException exp){}
        out.println(e);}
%>
</font></body></HTML>
```

9.9 分页显示记录

- 为避免长时间占用数据库的连接，可以使用二维数组table存放表的记录，即用二维数组table中的行（一维数组table[i]）存放一条记录，然后关闭服务器连接。
- 假设table存放了m行记录，准备每页显示n行，那么，总页数的计算公式：

  总页数=(m%n)==0?(m/n):(m/n+1);

1. 如果m除以n的余数大于0,总页数等于m除以n的商加1
2. 如果m除以n的余数等于0，总页数等于m除以n的商

- 如果准备显示第p页的内容，应当从table第(p-1)*n行开始，连续输出n行（最后一页可能不足n行）。

9.9 分页显示

example7_8.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=#AAFFEE><font size=2>
<form action="queryAllServlet" method=post>
<b>数据库:<input type="text" name="dataBase" size=22 value
   =warehouse>
<br>表名：<input type="text" name="tableName" size=23
   value=product>
<br>用户名(默认root)：<input type="text" name="user" size=10
   value=root>
<br>用户密码(默认空)：<input type="text" name="password" size=10>
<br><input type="submit" name="b" value="提交">
</form>
</font></body></HTML>
```

9.9 分页显示

```
<servlet>
    <servlet-name>queryAllServlet</servlet-name>
    <servlet-class>myservlet.control.Example7_8_Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>queryAllServlet</servlet-name>
  <url-pattern>/queryAllServlet</url-pattern>
</servlet-mapping>
```

9.9 分页显示

Example7_8_bean.java

```
package mybean.data;
public class Example7_8_Bean{
   String []columnName ;          //存放列名
   String [][] tableRecord=null;   //存放查询到的记录
   int pageSize=1;                //每页显示的记录数
   int totalPages=1;               //分页后的总页数
   int currentPage =1   ;           //当前显示页
   public void setTableRecord(String [][] s){tableRecord=s;}
   public String [][] getTableRecord(){return tableRecord;}
   public void setColumnName(String [] s) {columnName = s;}
   public String [] getColumnName() {return columnName;}
   public void setPageSize(int size){pageSize=size;}
   public int getPageSize(){return pageSize;}
   public int getTotalPages(){return totalPages;}
   public void setTotalPages(int n){totalPages=n; }
   public void setCurrentPage(int n){currentPage =n;}
   public int getCurrentPage(){return currentPage ;}
}
```

9.9 分页显示

Example7_8_servlet.java

```
package myservlet.control;
import mybean.data.Example7_8_Bean;
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Example7_8_Servlet extends HttpServlet{
  public void init(ServletConfig config) throws ServletException{
    super.init(config);
    try {  Class.forName("com.mysql.jdbc.Driver");}catch(Exception e){}
  }
  public void doPost(HttpServletRequest request,HttpServletResponse
   response) throws ServletException,IOException{
    request.setCharacterEncoding("gb2312");
    String dataBase= request.getParameter("dataBase");
    String tableName= request.getParameter("tableName");
    String user= request.getParameter("user");
    String password= request.getParameter("password");
    boolean boo =( dataBase==null||dataBase.length()==0);
    boo = boo||( tableName==null||tableName.length()==0);
    boo = boo||( user==null||user.length()==0);
```

9.9 分页显示

Example7_8_servlet.java

```
if(boo) {fail(request,response,"查询失败");}
    HttpSession session=request.getSession(true);
    Connection con=null;
    Example7_8_Bean pageBean=null;
    try{
        pageBean=(Example7_8_Bean)session.getAttribute("pageBean");
        if(pageBean==null){
          pageBean=new Example7_8_Bean();  //创建Javabean对象
          session.setAttribute("pageBean",pageBean);}
    }catch(Exception exp){
        pageBean=new Example7_8_Bean();
        session.setAttribute("pageBean",pageBean);}
    String uri="jdbc:mysql://127.0.0.1/"+dataBase;
    try{
      con=DriverManager.getConnection(uri,user,password);
      Statement sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                        ResultSet.CONCUR_READ_ONLY);
       ResultSet rs=sql.executeQuery("SELECT * FROM "+tableName);
       ResultSetMetaData metaData = rs.getMetaData();
       int columnCount = metaData.getColumnCount(); //得到结果集的列数
       String []columnName = new String[columnCount];
```

9.9 分页显示

Example7_8_servlet.java

```
for(int i=0;i<columnName.length;i++) {
          columnName[i] = metaData.getColumnName(i+1); //得到列名
        }
        pageBean.setColumnName(columnName);   //更新Javabean数据模型
        rs.last();
        int rowNumber=rs.getRow();  //得到记录数
        String [][] tableRecord=pageBean.getTableRecord();
        tableRecord = new String[rowNumber][columnCount];
        rs.beforeFirst();
        int i=0;
        while(rs.next()){
         for(int k=0;k<columnCount;k++)
           tableRecord[i][k] = rs.getString(k+1);
           i++;
        }
        pageBean.setTableRecord(tableRecord); //更新Javabean数据模型
        con.close();
        response.sendRedirect("example7_8_pageShow.jsp");  //重定向
    }
    catch(SQLException e){
        System.out.println(e);}
  }
```

9.9 分页显示

example7_8_pageShow.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="pageBean" class="mybean.data.Example7_8_Bean" scope="session"/>
<HTML><body bgcolor=#9FEFDF><center>
<br>当前显示的内容是:
<jsp:setProperty name="pageBean" property="pageSize" param="pageSize"/>
<jsp:setProperty name="pageBean" property="currentPage" param="currentPage"/>
<table border=1>
 <%
    String [][] table=pageBean.getTableRecord();
    if(table==null) {
       out.print("没有记录");
       return;}
    String []columnName=pageBean.getColumnName();
    if(columnName!=null) {
      out.print("<tr>");
      for(int i=0;i<columnName.length;i++){
        out.print("<th>"+columnName[i]+"</th>");
      }
      out.print("</tr>");
    }
    int totalRecord = table.length;
```

9.9 分页显示

```
out.println("全部记录数"+totalRecord); //全部记录数
    int pageSize=pageBean.getPageSize();  //每页显示的记录数
    int totalPages = pageBean.getTotalPages();
    if(totalRecord%pageSize==0)
        totalPages = totalRecord/pageSize;//总页数
    else
        totalPages = totalRecord/pageSize+1;
    pageBean.setPageSize(pageSize);
   pageBean.setTotalPages(totalPages);
    if(totalPages>=1) {
      if(pageBean.getCurrentPage()<1)
          pageBean.setCurrentPage(pageBean.getTotalPages());
      if(pageBean.getCurrentPage()>pageBean.getTotalPages())
          pageBean.setCurrentPage(1);
      int index=(pageBean.getCurrentPage()-1)*pageSize;
      int start=index;  //table的currentPage页起始位置
      for(int i=index;i<pageSize+index;i++) {
        if(i==totalRecord) break;
        out.print("<tr>");
        for(int j=0;j<columnName.length;j++) {
           out.print("<td>"+table[i][j]+"</td>");}
        out.print("</tr>");}
   }%></table>
```

9.9 分页显示

```
<br>每页最多显示<jsp:getProperty name="pageBean" property="pageSize"/>条信息
 <br>当前显示第<font color=blue>
   <jsp:getProperty name="pageBean" property="currentPage"/>
  </font>页,共有
  <font color=blue><jsp:getProperty name="pageBean" property="totalPages"/>
  </font>页。
<table>
 <tr><td><form action="" method=post>
       <input type=hidden name="currentPage" value=
      "<%=pageBean.getCurrentPage()-1 %>">
       <input type=submit name="g" value="上一页"></form></td>
    <td><form action="" method=post>
       <input type=hidden name="currentPage"
       value="<%=pageBean.getCurrentPage()+1 %>">
       <input type=submit name="g" value="下一页"></form></td></tr>
 <tr><td> <form action="" method=post>
       每页显示<input type=text name="pageSize" value =1 size=3>
       条记录<input type=submit name="g" value="确定"></form></td>
    <td> <form action="" method=post>
       输入页码：<input type=text name="currentPage" size=2 >
       <input type=submit name="g" value="提交"></form></td></tr>
</table>
</center></body></HTML>
```

9.10 数据库连接的常用方式

1）.一种使用纯Java数据库驱动程序

加载MySQL驱动程序代码如下：

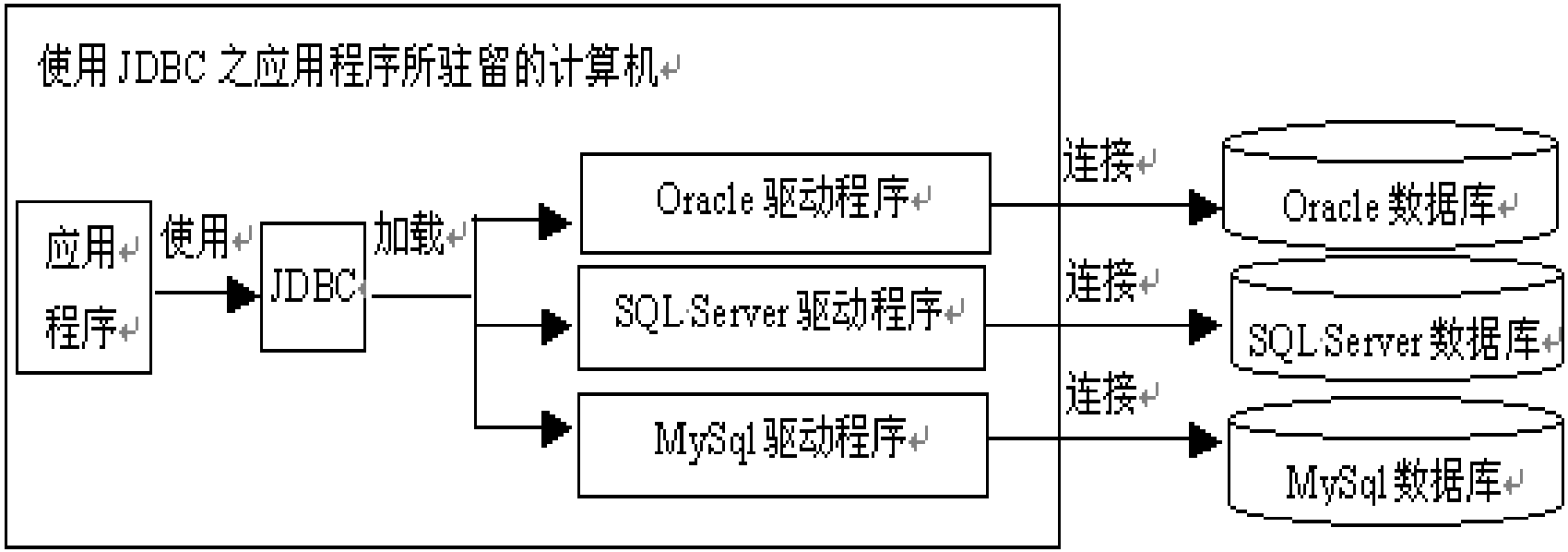
```
Class.forName("com.mysql.jdbc.Driver");
```

2）.一种是建立起一个JDBC-ODBC桥接器：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

9.10 数据库连接的常用方式

- 用Java语言编写的数据库驱动程序称作Java数据库驱动程序。



9.10 数据库连接的常用方式

- 配置驱动程序（以连接SQL Server 2012为例）
- 连接SQL Server 2012用的纯Java驱动程序(从www.microsoft.com下载)

  sqljdbc_1.1.1501.101_enu.exe
- 安装此文件后，在enu子目录中找到驱动文件sqljdbc.jar,将其复制到Tomcat所用的JDK的\jre\lib\ext文件夹中或Tomcat的安装目录\common\lib中 。
- 加载驱动程序

```
try
{ Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  }
    catch(Exception e){ out.print(e); }
```

9.10 数据库连接的常用方式

- 调用DriverManager类的getConnection方法建立连接，该方法有三个参数：

- 第一个字符串是JDBC URL，格式为：

  Jdbc:子协议:子名称

  Jdbc表示协议，子协议是驱动程序类的名称，子名称为数据库的名称，如果是远程数据库，还应该包括网络地址，格式如下：

  //主机名:端口;数据库名

- 第二个参数是用户名
- 第三个参数是密码

```
String uri="jdbc:sqlserver://IP地址:端口;DatabaseName=数据库名";
String user="testuser";
String password="testuser";
Connection con=
DriverManager.getConnection(uri,user,password);
```

9.10 数据库连接的常用方式

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

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);
}
```

9.10 数据库连接的常用方式

- 通过直接加载Oracle的Java数据库驱动程序来连接数据库
- 安装Oracle后，找到目录jdbc/lib中的classes12.jar，即用java编写Oracle数据库驱动程序。将classes12.jar复制到Tomcat引擎所使用的JDK的扩展目录中。通过如下的两个步骤和一个Oracle数据库建立连接。

1. 加载驱动程序

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

2. 建立连接

```
Connection con=
DriverManager.getConnection( “jdbc:oracle:thin:@主机host:端口号:数据库名 “, ”用户名 “, ”密码 “);
```

9.10 数据库连接的常用方式

使用JDBC-ODBC桥接器方式的机制是：应用程序只需建立JDBC和ODBC之间的连接，即所谓的建立JDBC-ODBC桥接器，而和数据库的连接由ODBC去完成。

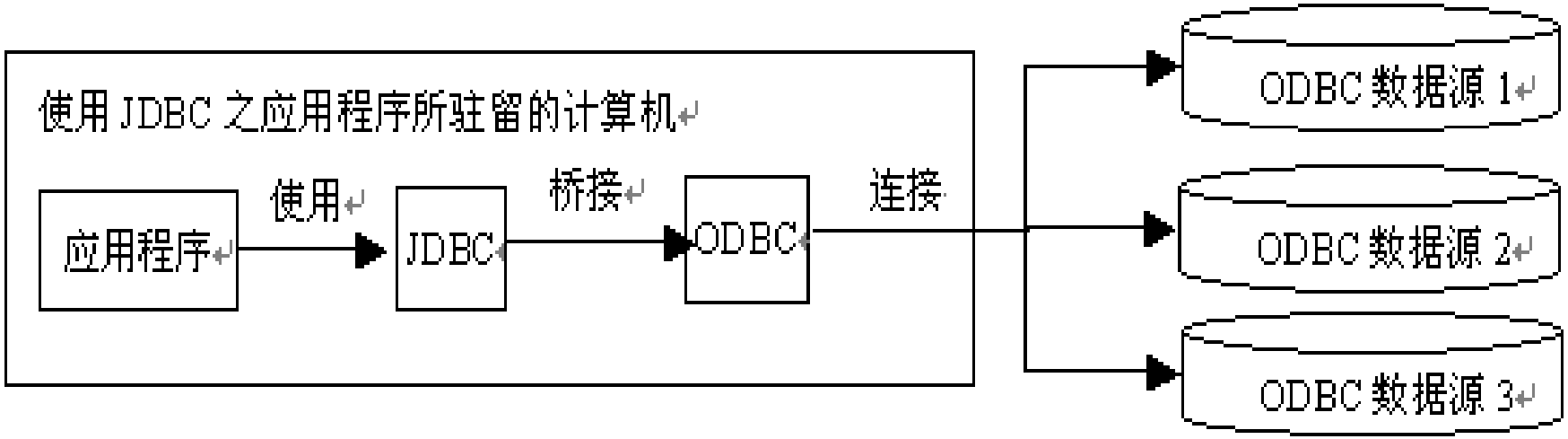


图 6.6 JDBC—ODBC 桥接器

9.10 数据库连接的常用方式

◆ 使用JDBC-ODBC桥接器访问数据库的步骤：

（1）建立JDBC-ODBC桥接器

（2）创建ODBC数据源

I. Windows控制面板_管理工具_ ODBC数据源

（3）和ODBC数据源建立连接

9.10 数据库连接的常用方式

第一步:加载驱动程序

JDBC 使用

- 数据库驱动程序
  - JDBC：各数据库厂商提供
  - ODBC：SUN提供
    （sun.jdbc.odbc.JdbcOdbcDrvier）
- 加载驱动程序：
  - 通过Class.forName("....")
  - 例：

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

```
<%@page import="java.sql.*"%>
<%!
        String driver = "sun.jdbc.odbc.JdbcOdbcDrvier" ;
%>
<%
        try
        {
                Class.forName(drvier) ;
        }
        catch(Exception e)
        {
                System.out.println("驱动程序加载异常");
        }
%>
```

9.10 数据库连接的常用方式

第二步:连接数据库

JDBC 使用

- 连接数据库:
  - Connection
    con=DriverManager.getConnection(url,id,password);
  - ODBC URL:
    - jdbc:odbc:ODBC数据源

Connection 表示连接数据库的连接接口，必须通过DriverManager类进行实例化

如果使用JDBC-ODBC连接数据库，则方式固定:

jdbc:odbc:数据源名称,“登录名”,“密码”

9.10 数据库连接的常用方式

第三步:操作数据库

JDBC 使用

- 创建语句对象：
  - Statement
  - PreparedStatement

例：

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

- 执行SQL命令：
  - stmt.executeUpdate(sql);
    - INSERT
    - UPDATE
    - DELETE
  - stmt.executeQuery(sql);
    - SELECT使用， 返回ResultSet实例化对象

9.10 数据库连接的常用方式

Statement接口常用方法:

No.	方法	类型	描述
1	int executeUpdate(String sql) throws SQLException	普通	执行数据库更新的SQL语句，例如：INSERT、UPDATE、DELETE等语句，返回更新的记录数
2	ResultSet executeQuery(String sql) throws SQLException	普通	执行数据库查询操作，返回一个结果集对象
3	void addBatch(String sql) throws SQLException	普通	增加一个待执行的SQL语句
4	int[] executeBatch() throws SQLException	普通	批量执行SQL语句
5	void close() throws SQLException	普通	关闭Statement操作
6	boolean execute(String sql) throws SQLException	普通	执行SQL语句

9.10 数据库连接的常用方式

第四步：关闭数据库

关闭数据库

- 数据库在每次使用之后都必须关闭
- 关闭数据库操作的顺序与打开数据库操作的顺序相反
  - 先关闭结果集（ResultSet）
  - 再关闭操作（Statement）
  - 最后关闭连接（Connection）

所有的数据库在操作之后都必须关闭！！！

9.10 数据库连接的常用方式

- Access也是比较流行的一种数据库管理系统，操作简单、使用方便。一些规模不大的Web应用经常使用Access数据库数据库管理系统。
- 下面介绍使用JDBC-ODBC桥接器方式连接Access数据库。

1. 建立JDBC-ODBC桥接器

```
try{  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}catch(ClassNotFoundException e){  out.print(e);    }
```

2. 创建ODBC数据源时选择的数据库驱动程序为：Microsoft Access Driver(*.mdb),设置的数据源的名字为redsun

3. 和ODBC数据源建立连接

```
Connection con=
DriverManager.getConnection("jdbc:odbc:redsun","登 录名","密码");
```

9.10 数据库连接的常用方式

example7_9.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.sql.*" %>
<HTML><BODY bgcolor=cyan>
 <% Connection con;
   Statement sql;
   ResultSet rs;
   try{  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");}
   catch(ClassNotFoundException e){out.print(e);}
   try { con=DriverManager.getConnection("jdbc:odbc:myData","","");
        sql=con.createStatement();
        rs=sql.executeQuery("SELECT * FROM goods ");
        out.print("<table border=2>");
        out.print("<tr>");
          out.print("<th width=100>"+"产品号");
          out.print("<th width=100>"+"名称");
          out.print("<th width=50>"+"生产日期");
          out.print("<th width=50>"+"价格");
       out.print("</TR>");
```

9.10 数据库连接的常用方式

example7_9.jsp

```
while(rs.next()){
          out.print("<tr>");
          out.print("<td >"+rs.getString(1)+"</td>");
          out.print("<td >"+rs.getString(2)+"</td>");
          out.print("<td >"+rs.getDate("madeTime")+"</td>");
          out.print("<td >"+rs.getFloat("price")+"</td>");
        out.print("</tr>") ;
       }
       out.print("</table>");
       con.close();
  }
  catch(SQLException e){
      out.print(e);
  }
 %>
</BODY></HTML>
```

9.10 数据库连接的常用方式

通过JDBC-ODBC桥接器访问Excel电子表格. 步骤：

1. 设置数据源

为数据源选择的驱动程序是Microsoft Excel Driver。

2. 选择表

必须在电子表格中选出一工作区作为连接时使用的表。在Excel电子表格中拖动鼠标选出范围。然后在Excel菜单中选择“插入”→“名称”→“定义”，给选中的工作区命名（这一工作区的名称将作为连接时使用的表名）。

9.10 数据库连接的常用方式

showByJdbcOdbc.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.sql.*" %>
<HTML><BODY bgcolor=cyan>
<% Connection con; Statement sql;  ResultSet rs;
   try{  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");}
   catch(ClassNotFoundException e) {  out.print(e); }
try { con=DriverManager.getConnection("jdbc:odbc:mymoon", "sa", "sa");
        sql=con.createStatement();
rs=sql.executeQuery("SELECT * FROM employee WHERE     salary>3000");
       out.print("<table border=2>");
        out.print("<tr>");
        out.print("<th width=100>"+"雇员号");
        out.print("<th width=100>"+"姓名");
        out.print("<th width=50>"+"出生日期");
        out.print("<th width=50>"+"薪水");
        out.print("</tr>");
```

9.10 数据库连接的常用方式

showByJdbcOdbc.jsp

```
while(rs.next())
      {  out.print("<tr>");
          out.print("<td >"+rs.getString(1)+"</td>");
          out.print("<td >"+rs.getString(2)+"</td>");
          out.print("<td >"+rs.getDate("birthday")+"</td>");
          out.print("<td >"+rs.getFloat("salary")+"</td>");
        out.print("</tr>") ;   }
      out.print("</table>");
      con.close(); } catch(SQLException e)  {  out.print(e);   }
 %>
</BODY></HTML>
```

9.11 标准化考试

我们很熟悉标准化考试，就是只需在给出的选择中选出正确的答案。在本节的标准化考试中，用户只能顺序的回答每个随机抽取到的题目，即回答一个题目，然后读取下一个题目后，用户就不能再回到上一个题目（类似新的驾驶员交通理论考试规则）。

9.11 标准化考试

设计要求

1. 考生可以在输入考号的页面输入考号，单击确认提交键开始考试。
2. 考生单击确认提交键后，可以在答题页面看到随机抽取到的第1题。
3. 考生回答一个题目后，可以继续随机抽取下一题目。
4. 考生在答题页面单击交卷提交键，完成考试，系统将给出考生的分数。

9.11 标准化考试

- 创建一个名字为**school**的数据库，在数据库中使用**test**表存放试题。**test**表的各个字段及意义如下：

number(int) ： 存放题号,

content(char) ： 存放试题内容 ,

a(char) ： 存放试题提供的a选择 ,

b(char) ： 存放试题提供的b选择,

c(char) ： 存放试题提供的c选择,

d(char) ： 存放试题提供的d选择,

pic(char) ： 存放试题示意图的图像文件的名字 ,

answer(char) ： 存放试题的答案

- 使用**student**表存放考生的学号和分数。**student**表的各个字段及意义如下：

id(char) ： 存放考号,

score(float) ： 存放分数

9.11 标准化考试

Example7_10_Bean.java

```
package mybean.data;
public class Example7_10_Bean{
   String id ;          //存放考号
   float score;         //存放分数
   String questions;    //存放题目
   int number;          //存放题号
   int textAmount=3 ; //题目数量
   String choiceA,choiceB,choiceC,choiceD;//存放选择
   String image;   //题目的示意图的图像文件名
   String answer;       //存放用户给出的答案
   String correctAnswer; //存放正确答案
   String mess;       //存放提示信息
   public String getCorrectAnswer() {return correctAnswer;}
   public void setCorrectAnswer(String s){correctAnswer =s;}
   public void setId(String s){id=s;}
   public String getId(){return id;}
   public void setScore(float s) {score = s;}
   public float getScore() {return score;}
   public void setQuestions(String s){questions=s;}
   public String getQuestions(){return questions;}
   public void setNumber(int s){number=s;}
```

9.11 标准化考试

Example7_10_Bean.java

```
package mybean.data;
public class Example7_10_Bean{
   public int getNumber(){return number;}
   public void setChoiceA(String s){choiceA=s;}
   public String getChoiceA(){return choiceA;}
    public void setChoiceB(String s){choiceB=s;}
   public String getChoiceB(){return choiceB;}
    public void setChoiceC(String s){choiceC=s;}
   public String getChoiceC(){return choiceC;}
   public void setChoiceD(String s){choiceD=s;}
   public String getChoiceD(){return choiceD;}
   public void setImage(String s){image=s;}
   public String getImage(){return image;}
   public void setAnswer(String s){answer=s;}
   public String getAnswer(){return answer;}
   public void setMess(String s){mess=s;}
   public String getMess(){return mess;}
   public void setTestAmount(int s){textAmount=s;}
   public int getTestAmount(){return textAmount;}
}
```

9.11 标准化考试

example7_10.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="testBean"
    class="mybean.data.Example7_10_Bean" scope="session"/>
<HTML><body bgcolor=#EAFFCF><font size=2>
<jsp:setProperty name="testBean" property="score" value="0"/>
<jsp:setProperty name="testBean" property="number" value="0"/>
<h2> 考题数量是<jsp:getProperty name="testBean"
    property="testAmount"/>
<form action="readTestServlet?amount=3"  method="post" >
  <br>输入考号<input type=text name="id" value =-1 size=16 >
  <br><input type=submit name="sub" value="开始考试">
</form>
</font></body></HTML>
```

9.11 标准化考试

根据例子10中使用的servlet的名字及相关类，Web服务目录ch7的WEB-INF下的web.xml文件需包含如下内容（有关web.xml文件的编辑与保存见5.1.2）：

```
<servlet>
<servlet-name>readTestServlet</servlet-name>
<servlet-class>myservlet.control.Example7_10_Servlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>readTestServlet</servlet-name>
<url-pattern>/readTestServlet</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>endTestServlet</servlet-name>
<servlet-class>myservlet.control.Example7_10_End_Servlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>endTestServlet</servlet-name>
<url-pattern>/endTestServlet</url-pattern>
</servlet-mapping>
```

9.11 标准化考试

Example7_10_Servlet.java

```
public void doPost(HttpServletRequest request,HttpServletResponse
    response) throws ServletException,IOException{
     Example7_10_Bean testBean=null;
     HttpSession session=request.getSession(true);
     try{  testBean=(Example7_10_Bean)session.getAttribute("testBean");
          if(testBean==null){
             testBean=new Example7_10_Bean(); //创建Javabean对象
             session.setAttribute("testBean",testBean);}
     }catch(Exception exp){
          testBean=new Example7_10_Bean();  //创建Javabean对象
          session.setAttribute("testBean",testBean);}
    try{  Class.forName("com.mysql.jdbc.Driver");} catch(Exception e){}
    request.setCharacterEncoding("gb2312");
    String id=request.getParameter("id");
    if(id==null||id.length()==0) {
       notify(request,response,"必须给出学号");
       return;
    }
    testBean.setId(id);
    int testAmount = testBean.getTestAmount();   //考题数量
    Connection con;
    Statement sql;
    ResultSet rs;
```

9.11 标准化考试

Example7_10_Servlet.java

```
try{
        String uri="jdbc:mysql://127.0.0.1/school?"+
                   "user=root&password=&characterEncoding=gb2312";
        con=DriverManager.getConnection(uri);
        sql=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                ResultSet.CONCUR_READ_ONLY);
        rs=sql.executeQuery("SELECT * FROM test");
        rs.last();
        int recordAmount=rs.getRow();  //得到记录数
        testAmount = Math.min(recordAmount,testAmount);
        LinkedList<Integer> list=(LinkedList<Integer>)session.getAttribute("list");
        if(list==null||list.size()==0){
          list = new LinkedList<Integer>();
          for(int i=1;i<=recordAmount;i++) {
            list.add(i);
          }
          session.setAttribute("list",list);
        }
        int m= -1;
        int index=-1;
```

9.11 标准化考试

Example7_10_Servlet.java

```
if(list.size()>=1) {
        m= (int)(Math.random()*list.size());
        index=list.get(m);
        list.remove(m);
        session.setAttribute("list",list);
        int tihao=testBean.getNumber();
        if(tihao<testAmount) {
           //首先判断上一题是否正确，给出分数：
          String studentAnswer=testBean.getAnswer();
          if(studentAnswer!=null&&studentAnswer.length()>=1) {
        if(studentAnswer.equalsIgnoreCase(testBean.getCorrectAnswer())){
             float score= testBean.getScore();
             score++;
            testBean.setScore(score);
          }
        }
        //随机抽取下一题目：
        tihao++;
        testBean.setNumber(tihao); //题号
        rs.absolute(index); //随机抽取题目
        testBean.setQuestions(rs.getString(1));//题目内容
```

9.11 标准化考试

Example7_10_Servlet.java

```
            testBean.setChoiceA(rs.getString(2));  //题目的选择a
             testBean.setChoiceB(rs.getString(3));  //题目的选择b
             testBean.setChoiceC(rs.getString(4));  //题目的选择c
             testBean.setChoiceD(rs.getString(5));  //题目的选择d
             testBean.setImage(rs.getString(6));  //题目的示意图名称
             testBean.setCorrectAnswer(rs.getString(7).trim());//题目的答案
             testBean.setMess("现在是第"+tihao+"题");
             con.close();
          }
          else {
             testBean.setMess("答题结束，单击交卷查看分数");
             String studentAnswer=testBean.getAnswer(); //判断最后一题
             if(studentAnswer!=null&&studentAnswer.length()>=1) {
        if(studentAnswer.equalsIgnoreCase(testBean.getCorrectAnswer())){
                  float score= testBean.getScore();
                  score++;
                 testBean.setScore(score);
               }
             }
             testBean.setAnswer(null);
             testBean.setNumber(0);
             testBean.setQuestions(null);
```

9.11 标准化考试

Example7_10_Servlet.java

```
            testBean.setChoiceA(null);
             testBean.setChoiceB(null);
             testBean.setChoiceC(null);
             testBean.setChoiceD(null);
             testBean.setImage(null);
          }
        }
        else {
          testBean.setMess("没有抽到题目");
        }
        response.sendRedirect("example7_10_examination.jsp");
    }
    catch(SQLException e){
        notify(request,response,e.toString());
    }
  }
```

9.11 标准化考试

Example7_10_Servlet.java

```
public void notify(HttpServletRequest request,HttpServletResponse
    response,String backNews) {
      response.setContentType("text/html;charset=GB2312");
      try {
       PrintWriter out=response.getWriter();
       out.println("<html><body>");
       out.println("<h2>"+backNews+"</h2>") ;
       out.println("返回");
       out.println("<a href =example7_10.jsp>返回</a>");
       out.println("</body></html>");
      }
      catch(IOException exp){}
   }
}
```

9.11 标准化考试

example7_10_examination.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="testBean"
   class="mybean.data.Example7_10_Bean" scope="session"/>
<HTML><body bgcolor=#DEEFF9><font size=2>
 <br><b> <jsp:getProperty name="testBean"
   property="questions"/></b>
 <br> <br><jsp:getProperty name="testBean" property="choiceA"/>
 <br> <br><jsp:getProperty name="testBean" property="choiceB"/>
 <br> <br><jsp:getProperty name="testBean" property="choiceC"/>
 <br> <br><jsp:getProperty name="testBean" property="choiceD"/>
 <% String pic=testBean.getImage();
   if(pic!=null&&pic.length()>=1) {
 %>  <br> <image src =image/<%=pic%> width=100
   height=60></image>
 <% }
 %>
 <br>
 <% String studentAnswer = request.getParameter("R");
   if(studentAnswer!=null&&studentAnswer.length()>=1){
      testBean.setAnswer(studentAnswer.trim());
   }
 %>
```

9.11 标准化考试

example7_10_examination.jsp

```
<b>  目前分数： <jsp:getProperty name="testBean" property="score"/>,
  消息： <jsp:getProperty name="testBean" property="mess"/><br>
 <form action="" method=post name=form>
  <br>选择:<input type="radio" name="R" value=A>A
        <input type="radio" name="R" value=B>B
        <input type="radio" name="R" value=C>C
        <input type="radio" name="R" value=D>D
  <br><input type="submit" value="确认(再读取下一题之前，可反复确认)"
    name="submit">
 </form>
 <b>你目前给出的选择是<%=studentAnswer %>
 <form action="readTestServlet" method=post name=form>
  <br><input type= "hidden"
      value="<%=testBean.getId()%>" name ="id">
    <br><input type="submit" value="下一题" name="submit">
 </form>
 <form action="endTestServlet" method=post name=form>
   <input type="submit" value="交卷" name="submit">
 </form>
</font></body></HTML>
```

9.11 标准化考试

Example7_10_End_Servlet.java

```
public void doPost(HttpServletRequest request,HttpServletResponse
    response)throws ServletException,IOException{
     Example7_10_Bean testBean=null;
     HttpSession session=request.getSession(true);
     try{  testBean=(Example7_10_Bean)session.getAttribute("testBean");
     }catch(Exception exp){
       response.sendRedirect("example7_10.jsp");
     }
     try{  Class.forName("com.mysql.jdbc.Driver");
     }
     catch(Exception e){}
     request.setCharacterEncoding("gb2312");
     String id=testBean.getId();
     Connection con;
     Statement sql;
```

9.11 标准化考试

Example7_10_End_Servlet.java

```
String condition = "INSERT INTO student VALUES"+
                   "("+"'"+id+"',"+testBean.getScore()+")";
    try{
       String uri="jdbc:mysql://127.0.0.1/school?"+
                  "user=root&password=&characterEncoding=gb2312";
       con=DriverManager.getConnection(uri);
       sql=con.createStatement();
       sql.executeUpdate(condition);
       float score = testBean.getScore();
       notify(request,response,id+"最后得分:"+score);
       session.invalidate();            //销毁用户的session对象
    }
    catch(SQLException exp){}
  }
```

小结

- JSP使用JDBC提供的API和数据库进行交互信息。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。
- 当查询ResultSet对象中的数据时，不可以关闭和数据库的连接。
- 使用PreparedStatement对象可以提高操作数据库的效率