

# 基于 JSP 的网络数据库连接技术

周彩兰, 孙琳, 李素芬

(武汉理工大学 计算机学院, 湖北 武汉 430070)

**摘要:**目前,数据库在 Web 服务中得到广泛应用。文中简要论述了网络数据库的三层结构及各层功能,对 JSP 技术进行了简单的论述。通过两个实例,详细论述了在 JSP 页面中利用 JDBC API 连接 Access 数据库和 MySQL 数据库的具体步骤及特点,证明了采用 JDBC 技术连接和操作各种数据库更加简便。

**关键词:**三层结构;网络数据库;JSP;JDBC;JDBC—ODBC 桥

**中图分类号:**TP311.138

**文献标识码:**A

**文章编号:**1005-3751(2006)04-0209-03

## Web Database Connection Technology Based on JSP

ZHOU Cai-lan, SUN Lin, LI Su-fen

(Wuhan Univ. of Tech., Wuhan 430070, China)

**Abstract:** Nowadays, the databases are widely used in Web services. This paper introduces the three-tier architecture of Web database, the functions of each tier and the JSP technology simply. And then by using two instances, the paper describes the steps and the characteristics of how to use JDBC API to make connection with Access database and MySQL database, and proofs that it is more simple for using JDBC technology to make connection with all kinds of databases.

**Key words:** three-tier architecture; Web database; JSP; JDBC; JDBC—ODBC bridge

### 0 引言

随着网络技术迅速发展,网络服务已经深入到社会发展的各个层面,尤其是数据库技术与网络的完美结合,使网络的服务范围、服务质量及性能都有了质的变化。在早期的数据库应用系统中,C/S(Client/Server)客户/服务器两层结构占据了主导地位。但随着业务量的增加和用户要求的提高,C/S两层结构逐渐在服务器的伸缩性、数据库的管理、客户端软件复杂性以及可移植性等方面表现出严重的缺陷。三层体系结构模式的出现则摆脱了种种限制,成功应用于 Internet 中。

### 1 网络数据库的三层结构

Java 语言具有平台无关性和强大的网络编程功能,因此 Java 已经成为网络数据库开发中最常用编程语言。基于 Java 的网络数据库三层结构通常分为客户端/Web 应用服务器/数据库服务器(Browser/Web Server/Database Server, B/W/D)三层。其结构如图 1 所示<sup>[1]</sup>。

第一层为客户层,通常是使用 Web 浏览器实现的与用户交互的最终界面。

第二层为 Web 应用服务器层,也称为中间层。主要负责安全控制、事务处理或应用逻辑。通常使用 Servlet

引擎或应用服务器实现。

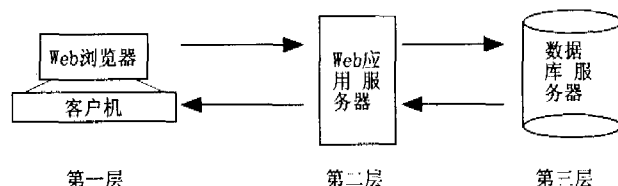


图 1 网络数据库三层结构

第三层为数据库服务器层,即数据层。主要是数据库操作系统(DBMS)。

在三层结构中,客户机调用 Java 应用程序,向 Web 应用服务器层提出数据库操作请求。Web 应用服务器通过调用 JDBC 与相应的数据库进行连接,数据操作在数据库服务器中进行,将结果返回给 Web 应用服务器,并最终发送到客户端并以 Web 形式显示给用户。

在网络数据库三层结构中,应用服务器与数据库服务器分离,因此数据库具有很高的物理独立性和逻辑独立性,从而提高了数据库的性能和安全保障。同时,用户可以使用简单的操作来访问数据库,而不必关心底层的调用细节。

### 2 JSP 技术简述

JSP(Java Server Pages)技术是一个纯 Java 平台技术,是由 Sun 公司倡导,多家公司共同参与制定的一种动态网页技术标准。JSP 将 Java 作为其脚本语言,在传统的 HTML 文件中加入 Java 程序片段和 JSP 标签,因此 JSP

收稿日期:2005-07-18

作者简介:周彩兰(1964-),女,湖南长沙人,副教授,研究方向为网络数据库及网络应用。

也继承了 Java 语言的各种优点。同时 JSP 技术将页面设计与商务逻辑分离,区分了 Web 设计者和 Web 开发者的角色。

JSP 技术以 Servlet 技术为基础。在三层结构中,JSP 工作在中间层。当客户端通过浏览器第一次发送 JSP 请求时,JSP 容器将 JSP 转译成 Servlet 代码,然后 Servlet 引擎加载 Servlet 执行,最后将结果以 HTML 的形式响应至客户端。当客户再次发出同样的处理请求时,Web 服务器就直接执行第一次产生的 Servlet,无需重新编译<sup>[2]</sup>。

### 3 调用 JDBC 连接数据库

#### 3.1 JDBC 的结构

JDBC 是一种用于执行 SQL 的 Java API,但 JDBC 常被误认为是 Java 数据库连接(Java DataBase Connectivity)的缩写,其实 JDBC 本身只是一个商标名。JDBC 允许开发人员将 SQL 语句传送给任何一种关系型数据库,而不必为不同的数据库连接编写不同的程序<sup>[3]</sup>。同时也允许用户从 Java 应用程序中访问任何表格化数据源,例如电子表格。

JDBC 的主要功能可总结为:建立与数据库或其他数据源的连接,向数据库发送 SQL 语句以及处理数据库返回结果。

JDBC API 可以分为两个层次:应用层和驱动程序层。JDBC API 与数据库之间的关系如图 2 所示。

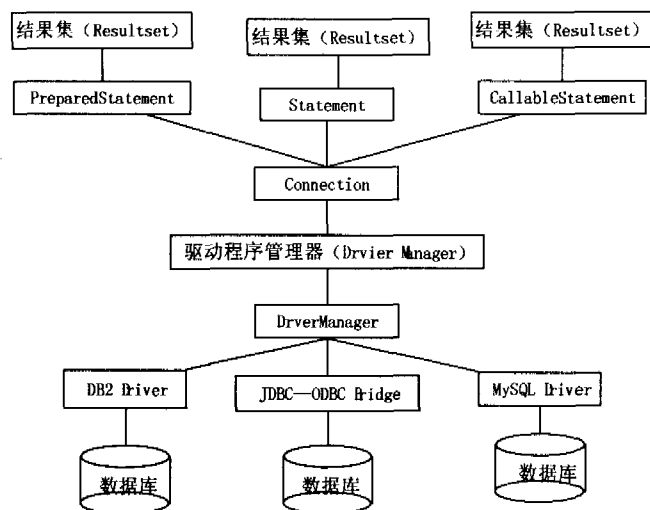


图 2 JDBC API 与数据库之间的关系

其中应用层 API 主要是提供给应用程序开发者使用的,以抽象的 Java 接口的形式描述。通过这些接口应用程序可以对数据库进行连接,执行 SQL 语句并且处理结果。Java 程序通过调用应用层 API,访问 JDBC 驱动程序管理器,JDBC 驱动程序管理器再通过 JDBC Driver API 访问不同的 JDBC 驱动程序,从而实现不同数据库系统的连接。驱动程序层 API 主要是针对数据库厂商开发数据库底层驱动程序使用的。

厂商提供的 JDBC 驱动程序可分为 4 类:

(1) JDBC—ODBC 桥接驱动程序(JDBC—ODBC

Bridge);

(2)原生协议以及纯 Java 驱动程序;

(3)原生 API 驱动程序(Native—API Driver);

(4)JDBC 通过网络的纯 Java 驱动程序。

#### 3.2 使用 JDBC—ODBC 桥连接 Access 数据库

ODBC API 是使用最广的访问数据库系统的编程接口,几乎所有的数据库系统都支持 ODBC 驱动程序。此外,JDBC 与 ODBC 都是以 X/Open SQL Call Level Interface 为基础的。因此利用 JDBC—ODBC 桥连接数据库可以作为连接没有提供 JDBC 驱动程序数据库系统(如:Access)的解决方案。

下面将以 Access 数据库的连接来说明 JDBC—ODBC 桥的应用<sup>[4]</sup>。

假设已经存在 Access 数据库 student.mdb,其中有表 grade 记录了学生的学号(Id)、姓名(Name)、英语(English)、数学(Math)等成绩信息。而且环境变量都已经配置完成。现在允许在一页面(research.html)中输入一分数,然后经过对数据库的检索,在另一页面(research.jsp)中显示出英语成绩高于此分数的学生姓名。其中 research.html 的代码如下:

```

<html>
<head>
<title>查询条件输入</title>
</head>

<body>
<h2>
请输入英语成绩,将查询大于此成绩的学生姓名
</h2>
<form name="form" method="post" action="research.jsp"> //规定参数传递的方法,以及传向的对象
<p>分数:<input name="English" type="text"></p>
<p><input type="submit" value="传送">
<input type="reset" value="取消">
</p>
</form>
</body>
</html>
  
```

research.html 页面允许用户输入一分数,并以 post 的方式将此参数传递给 research.jsp 页面,进行进一步的处理。

Research.jsp 页面主要负责通过 JDBC—ODBC 桥与 student.mdb 数据库进行连接,并显示查询结果<sup>[5]</sup>。其代码如下:

```

//引入 java.sql 包,并对整个页面进行设置
<%@ page import="java.sql.*"%>
  
```

```

<%@ page contentType="text/html;charset=gb2312"%>
<html>
<head>
<title>查询结果</title>
</head>
<body>
//接收 research.html 页面传来的分数参数,进行类型转换并传
递给 float 变量 englishGrade
<% float englishGrade = Float.parseFloat(request.getParameter
("English")); %>
<h2>您选择查询英语成绩在<% = englishGrade%>以上的学
生姓名</h2>
<%
String sql="SELECT * FROM grade WHERE English>" + eng-
lishGrade; //SQL 查询语句
try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //加载
注册 JDBC—ODBC 桥驱动程序
} catch(ClassNotFoundException e)
{out.print(e);}
%>
<%
try{
//建立连接对象 con,其中 jdbc 为主要通讯协议,odbc 为次
要通讯协议,student 为数据来源,由于 student 数据库没有设置
用户名和密码,因此这里省略
Connection con = DriverManager.getConnection("jdbc:odbc:stu-
dent","","");
//建立 SQL 陈述对象 stmt
Statement stmt = con.createStatement();
//利用 SQL 陈述对象执行 SQL 语句 sql 进行查询,并将结果返
回给结果集 rs
ResultSet rs = stmt.executeQuery(sql);
//如果查找到符合条件的纪录,则输出学生姓名和相应的英语
成绩;如果没有查找到,则提示信息
if(rs != null)
{
while(rs.next())
{
out.print(rs.getString("Name") + " - - - - ");
out.println(rs.getFloat("English"));
}
}
else{
out.println("没有此分数以上的学生");
}
//依次关闭结果集 rs、陈述对象 stmt、连接对象 con,释放资
源 rs.close();
stmt.close();
con.close();
}catch(SQLException ex) //如果抛出异常,则将异常信息进
行打印
{

```

```

System.out.println("sql=" + ex.getMessage());
}
%>
</body>
</html>

```

### 3.3 使用纯 Java 驱动程序连接 MySQL 数据库

这种驱动程序是由纯 Java 写成的,允许 Java 客户直接调用 DBMS 服务器,直接与数据库做沟通,不需要其他转换或中介软件。它与 JDBC—ODBC 桥相比,性能要好得多。

下面例子中 student 数据库改由 MySQL 创建,其中表及字段不作修改。research.html 页面以及 research.jsp 的部分代码与上例完全相同。要完成相同的任务只需要对 research.jsp 中相应代码作部分修改<sup>[5]</sup>。research.jsp 代码如下:

```

//引入 java.sql 包,并对整个页面进行设置
<%@ page import="java.sql.*"%>
<%@ page contentType="text/html;charset=gb2312"%>
.....
<body>
<%
String sql="SELECT * FROM grade WHERE English >"
+ englishGrade; //SQL 查询语句加载注册 MySQL 驱动程序,其
中 com.mysql.jdbc.Driver 为 MySQL 驱动程序所在路径
try{Class.forName("com.mysql.jdbc.Driver");
} catch(ClassNotFoundException e)
{out.print(e);}
%>
<%
try{
//建立连接对象 con,其中 jdbc 为主要通讯协议,mysql 为此
要通讯协议,localhost 是主机名称,这里代表本地主机,3306 是连
接数据库所用的端口,student 为数据来源,数据库用户名 ueser
为 root
//密码 password 为 browser
Connection con = null;
con = DriverManager.getConnection("jdbc:mysql://localhost:
3306/student? user=root&password=browser");
//建立 SQL 陈述对象 stmt
Statement stmt = con.createStatement();
//利用 SQL 陈述对象执行 SQL 语句 sql 进行查询,并将结果返
回给结果集 rs
ResultSet rs = stmt.executeQuery(sql);
.....
} catch(.....)
{.....}
%>
</body>
</html>

```

来,更好地保障网络系统的安全;采用异常检测技术的基础检测代理在检测过程中发现新的入侵形式时,就把审计记录传输到学习代理,通过计算得到一个可以检测此类入侵的更新了的分类器,然后将它分派给所有的基础检测代理,通过将异常检测和误用检测结合起来,可以提高系统的检测率,并提高了系统的可扩展性和自适应性。

### 3 提高入侵检测系统实时检测效率

目前,大部分审计机制都被设计成详尽记录系统和网络中的活动,这在保证没有入侵证据被遗漏的同时,鉴于网络数据的高速度和大流量,同样要求所构造的入侵检测系统在实时检测时具有高效率。否则,对审计数据(如:网络数据包)分析的延迟就可能给入侵成功提供机会。这就要求构造的入侵检测模型不仅要准确的,而且是高效的。

基于数据挖掘的入侵检测的效率是通过计算检测所必需的特征来测量的,因此这里提出了一个层叠的检测模型,如图 2 所示。底层的检测模块采用计算代价低的系统特征,而高层的检测模块采用计算代价比较高的系统特征,在图 1 的框架结构中反映为通过数据挖掘,选择不同的算法和系统特征,生成多个层叠的基础检测代理。

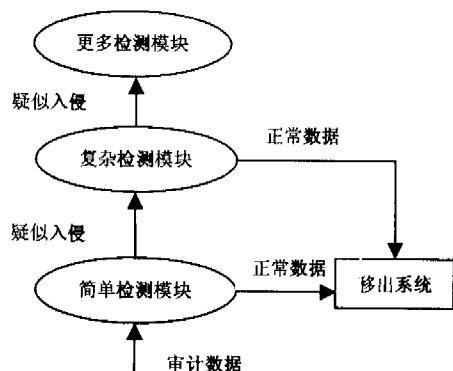


图 2 层叠检测模型

数据首先经过比较简单的检测模块,只有疑似为入侵的数据才被传递到下一层更为复杂的模块进行检测,这样只有一部分审计数据,也就是可能为攻击的那部分审计数据才被复杂的检测模块计算,这样大大提高了入侵检测系

统的实时运行效率。多个检测模块之间是独立的,而且通常采取不同的检测算法和特征。含有  $N$  个检测模块的层叠检测模型的检测率是各个检测模块检测率的乘积,为了使层叠检测模型有高检测率,就要求各个检测模块都要有高的检测率。而检测模块在拥有高检测率的同时,通常都会有高的误报率,采用层叠检测模块可以使得整个检测模型的误报率变得很低,例如:如果层叠模型中有 4 个检测模块,而每个模块的误报率是 10%,那么整个模型的误报率就是各个模块误报率的乘积:  $10^{-4}$ ,因此通过采用层叠检测模型,可以构造一个拥有高检测率和低误报率的入侵检测系统,而且这个系统在实时运行时是高效的。

### 4 结束语

近几年来,基于数据挖掘的入侵检测系统成为研究的热点。文中提出了一个基于数据挖掘和移动代理的分布式入侵检测系统框架和一个提高入侵检测模型运行效率的方法,提高了入侵检测系统的执行效率,改善了系统的可扩展性和自适应性。在实现时还存在一些需要解决的问题,如在提高检测效率的层叠检测模型方法中,如何选择适当的特征,才能使得复杂的检测模块比简单的模块更为广泛,更能有效地检测攻击等。今后,将针对这些问题展开进一步的研究。

#### 参考文献:

- [1] 罗守山.入侵检测[M].北京:北京邮电大学出版社,2004.
- [2] 范明,孟小峰.数据挖掘:概念与技术[M].北京:机械工业出版社,2001.
- [3] Lee W,Stolfo S J,Mok K W. A data mining framework for building intrusion detection models[A]. In Proceedings of the 1999 IEEE Symposium on Security and Privacy[C]. Oakland, CA:[s. n.],1999.
- [4] Lee W,Stolfo S J. Data mining approaches for intrusion detection[A]. In Proceedings of the 7th USENIX Security Symposium[C]. San Antonio, TX:[s. n.],1998.
- [5] Chan P K,Stolfo S J. Toward parallel and distributed learning by meta-learning[A]. In AAAI Workshop in Knowledge Discovery in Databases[C]. [s. l.]:[s. n.], 1993. 227-240.

(上接第 211 页)

### 4 结束语

通过以上的论述,可以看出 JDBC 为连接不同数据库提供了统一的接口,使 Java 开发人员更加容易连接并操作各种数据库,而不需要单独为数据库编写不同的连接程序,保证了 Java“一次编写,各处使用”的平台无关性原则。JSP 技术通过在 HTML 页面中嵌入 Java 代码的方式,极大地方便了网页的动态扩展功能,客户端只要提出数据库访问请求,具体的操作则由中间层完成。如果将 JSP 中数据库的连接操作代码写成专门的 JavaBean,将更加方便代码的复用。

#### 参考文献:

- [1] 周彩兰,陈才贤.基于 Java 的 Web 数据库连接池高效管理策略[J].武汉理工大学学报,2004(5):38-41.
- [2] 刘慧,李玉忱,苏鹏.基于 J2EE 架构的分布式 Web 应用的研究[J].计算机应用研究,2003(9):47-49.
- [3] 石振国.用 JSP 实现对 Web 数据库的访问[J].计算机应用,2001,21(5):91-93.
- [4] 李平,沈国民,李哲.基于 JSP 技术的 WEB 数据库设计[J].电脑与信息技术,2000(6):1-3.
- [5] 张维玉,李明东,陈劲.Web 数据库技术分析[J].西华师范大学学报,2004,25(2):219-222.