

# 基于 Java 语言实现数据库的访问

谷庆华, 李成贵

(北京航空航天大学 仪器科学与光电工程学院, 北京 100083)

**摘要:** Java 语言是目前广泛使用的网络数据库编程语言, JDBC 给数据库应用开发人员提供了一种标准的应用程序设计接口, 使数据库开发人员可以用纯 Java 语言编写完整的数据库应用程序, Java 和 JDBC 结合真正实现一次编写, 处处运行。介绍 JDBC 技术, 阐述了 JDBC 接口技术 JDBC API 和 JDBC 驱动程序, 给出采用 JDBC-ODBC 桥来实现关系型数据库 SQL 查询的实现方法。例程中采用 Access2000 数据库, 说明了用 JDBC 编写访问数据库的一般步骤: 加载 JDBC-ODBC 桥驱动程序; 建立与数据库的连接; 执行 SQL 查询语句和处理对数据库的查询结果。所举案例对实际数据库项目的开发有一定的实用价值。

**关键词:** JDBC; JDBC-ODBC 桥; 数据库; SQL 语言

**中图分类号:** TP312

**文献标识码:** A

**文章编号:** 1673-629X(2008)02-0013-04

## Realizing Database Accessing Based on Java

GU Qing-hua, LI Cheng-gui

(School of Instrument Science and Opto-Electronics Engineering,  
Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

**Abstract:** Java is the most popular language used in developing net database system. JDBC provides a standard application programming interface for programmers, so they can develop database applications using pure Java language. Java connected with JDBC can realize write once, use everywhere. Firstly introduces JDBC technology, then JDBC API and JDBC driver are represented in details, finally a typical example of the implementation of SQL queries with JDBC-ODBC bridge is presented. Access2000 database is adopted by the example, the common methods of developing database system using JDBC are discussed which include fixing the JDBC-ODBC bridge driver programs, establishing the connection with database, executing SQL query language and dealing with required results. The example given is useful in developing actual database program.

**Key words:** JDBC; JDBC-ODBC bridge; database; SQL language

## 1 JDBC 技术概述

### 1.1 JDBC 产生背景

Java 语言具有健壮、安全、易于使用、易于理解和可从网络上自动下载等特性, 所以它成为了开发数据库应用程序的一种优秀的语言。随着越来越多的程序员开始使用 Java 语言, 许多 Java 应用开发者都希望能够编写独立于特定 DBMS (Data Base Management System, 数据库管理系统) 的程序, 使得与各种各样 DBMS 的连接变得更为便捷, 开发更加迅速。因此有必要定义一个通用的 SQL 数据库存取框架, 从而在各种各样

的数据库连接的模块上提供统一的界面, 使与数据库无关的 Java 工具和产品成为可能, 使得数据库连接的开发者可以提供各种各样的连接方案<sup>[1]</sup>。这一切所需要的只是一种 Java 应用程序与各种不同数据库之间进行对话的方法, 而 JDBC 正是实现此种对话的一种机制。

### 1.2 JDBC 简介

JDBC 是一种可用于执行 SQL 语句的 Java API (Application Programming Interface, 应用程序设计接口)。它由一组 Java 语言编写的类和接口组成。JDBC 给数据库应用开发人员、数据库前台工具开发人员提供了一种标准的应用程序设计接口, 使开发人员可以用纯 Java 语言编写完整的数据库应用程序。

通过使用 JDBC, 开发人员可以很方便地将 SQL 语句传送给几乎任何一种数据库。即开发人员可以不必写一个程序访问 Sybase, 另写一个程序访问 Oracle。

收稿日期: 2007-05-16

基金项目: 国防“十五”重大专项建设项目(995)

作者简介: 谷庆华(1980-), 女, 辽宁大连人, 硕士研究生, 研究方向为计算机通信、数据库技术、网络编程等; 李成贵, 副教授, 博士, 从事微纳米表面测量和表征技术、计算机测量和控制、智能仪器、虚拟仪器等方面的教学与研究工作。

用 JDBC 写的程序能够自动地将 SQL 语句传送给相应的数据库管理系统。

不但如此,使用 Java 编写的应用程序可以在任何支持 Java 的平台上运行,不必在不同的平台上编写不同的应用程序。Java 和 JDBC 的结合可以让开发人员在开发数据库应用时真正实现一次编写,处处运行。

### 1.3 JDBC 接口概貌

JDBC 接口分为两个层次:一个是面向程序开发人员的 JDBC API;另外一个底层的 JDBC Driver API (驱动程序)<sup>[2]</sup>。

#### 1.3.1 JDBC API

JDBC API 被描述成为抽象的 Java 接口,应用程序可以对某个数据库打开连接,执行 SQL 语句并且处理结果。最重要的接口有 4 个:

(1)java. sql. DriverManager, 处理驱动程序的调入并且对产生新的数据库连接提供支持。

(2)java. sql. Connection, 代表对特定数据库的连接。

(3)java. sql. Statement, 代表一个特定的容器,来对一个特定的数据库执行 S 语句。其中又有两个子类型:java. sql. PreparedStatement, 用于执行预编译的 Java 语句;java. sql. CallableStatement, 用于执行对一个数据库内嵌过程的调用。

(4)java. sql. ResultSet, 控制对一个特定语句的行数据的存取。

#### 1.3.2 JDBC Driver API(驱动程序)

在 JDBC 的发展经历中,产生了 4 种不同的驱动程序<sup>[3]</sup>,现分述如下:

(1)JDBC-ODBC 桥加 ODBC 驱动程序。

这类驱动程序利用类似桥接器的技术来连接数据库,通过 JDBC-ODBC 桥,开发人员可以使用 JDBC 来存取 ODBC 数据源。不足的是,它需要在客户端安装 ODBC 驱动程序,即必须安装 Microsoft Windows 的某个版本,因而最适合于企业网,或者是用 Java 编写的 3 层结构的应用程序服务器代码。连接过程如图 1 所示。

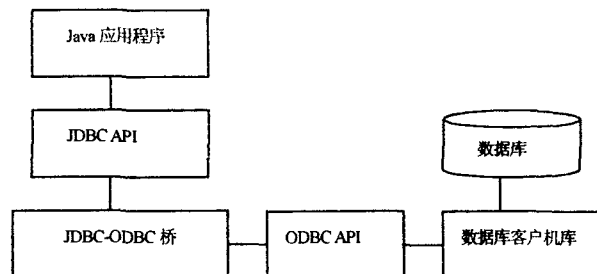


图 1 JDBC-ODBC 桥连接数据库过程图

(2)本地 API 部分用 Java 来编写的驱动程序。

JDBC 驱动程序将对数据库的 API 从标准的 JDBC 调用转换为本地调用。它们直接将 JDBC API 翻译成具体数据库的 API。它执行数据库开发商(如 Oracle, Sybase, Informix 等)所提供的 API 来存取数据库,要求在客户端安装一些本地代码,即数据库客户机应用程序必须有合适的数据库客户机库。连接过程如图 2 所示。

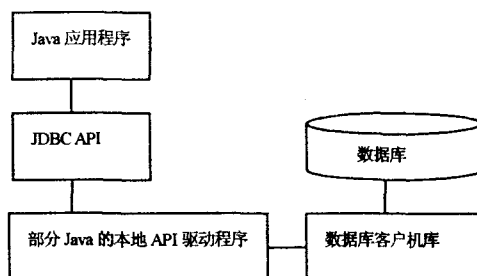


图 2 部分 Java 本地 API 驱动程序连接数据库过程图

(3)网络协议完全的 Java 驱动程序。

它将 JDBC API 转换成独立于数据库的协议。JDBC 驱动程序并没有直接和数据库进行通讯,它和一个中间件服务器通讯,然后这个中间件服务器和数据库进行通讯。由于中间件服务器隐藏了 Java 应用程序的细节,这种额外的中间层次更加灵活性,可以用相同的代码访问不同的数据库。连接过程如图 3 所示。

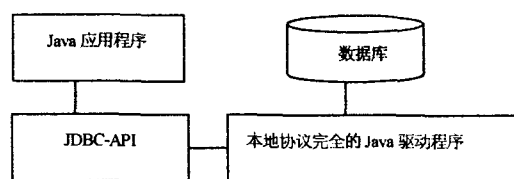


图 3 网络协议完全的 Java 驱动程序连接数据库过程图

(4)本地协议完全的 Java 驱动程序。

这种驱动程序是纯 Java 驱动程序,它直接与数据库进行通讯,执行数据库的直接访问。使用这类驱动程序可以说是纯 Java 的解决方案,它可直接通过网络协议,将客户端的请求直接送到服务器端处理,而执行结果也直接从服务器端取回,因此如果采用这类驱动程序来开发 Web 应用系统,不需要在客户端加装任何软件,也不需通过中介软件来进行任何转换工作。连接过程如图 4 所示。

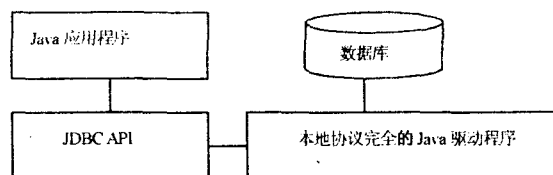


图 4 本地协议完全的 Java 驱动程序连接数据库过程图

## 2 数据库查询的Java实现

下面介绍利用 JDBC—ODBC 桥实现对 Microsoft Access 数据库的 SQL 查询的方法。

### 2.1 数据库建立和数据源设置

Microsoft Access 具备完善的数据库功能,可以作为独立的 DBMS 使用,是 PC 机上开发客户机/服务器型数据库应用的优秀工具。它将若干个相互关联的表(table)组成一个数据库(\*.mdb)。这里作为例子使用的数据库 dbinfo 包括 2 个表:tbinformation1 与 tbinformation2,其结构分别如表 1、表 2 所示。

表 1 tbinformation1 表结构

id	name	phone	address	email
----	------	-------	---------	-------

表 2 tbinformation2 表结构

id	userid	department	score	trial
----	--------	------------	-------	-------

在这 2 个数据表之间建立了一个关系,即在表 tbinformation1 的 id 字段与表 tbinformation2 的 userid 字段之间,它们内容相同。数据库制好后,打开 Windows 的“开始”菜单,选择“设置”菜单下的“控制面板”,从中启动微软的 ODBC 数据源,设置与数据库相应的微软 ODBC 驱动器和数据源。

ODBC 是用 C 语言写的在多种不同的 DBMS 中存取数据的标准应用程序接口。目前应用最广的是微软的 ODBC,它几乎可将所有平台的所有数据库连接起来。ODBC 在应用程序与特定数据库之间插入一个驱动程序管理器,每种数据库引擎都需要向驱动程序管理器注册它自己的 ODBC 驱动程序,驱动程序管理器将与 ODBC 兼容的 SQL 请求从应用程序传给 ODBC 驱动程序,并由 ODBC 驱动程序把 SQL 请求翻译为对数据库的固有调用,从而达到应用程序访问操作数据库的目的。JDBC 采用 JDBC—ODBC 桥接方式在 Java 应用程序中使用 ODBC。

### 2.2 访问数据库的Java应用程序的编写

用 JDBC 编写访问、操作数据库的 Java 应用程序一般完成下面 4 件事<sup>[4]</sup>:

#### (1) 加载 JDBC—ODBC 桥驱动程序。

为了与特定的数据源相连,JDBC 必须加载相应的驱动程序。加载的方式可以采用 Class.forName 方法显式加载,如下面语句加载 Sun 公司的 JDBC—ODBC 桥驱动:

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

#### (2) 建立与数据库的连接。

运用 DriverManager 类的 getConnection 方法建立与数据源的连接:

```
sourceURL="jdbc:odbc:";
```

```
databaseConnection = DriverManager.getConnection  
(sourceURL,"Administrator","password");
```

该语句与 sourceURL 对象指定的数据源建立连接。JDBC URL 提供了一种标识数据库驱动器的方式,其中用冒号分为 3 部分<sup>[5]</sup>,依次为:协议(在 JDBC 中只能为 jdbc)、子协议(用来标识一个数据库的连接机制,此处选用 odbc)和子名称(ODBC 数据源的名称,为 dbinfo)。

在 JDBC 中,java.sql.DriverManager 类的目的是提供在应用中使用的不同数据库驱动器的一个通用的访问层,使应用程序不用直接使用单独的 Driver 实现类,而是使用 DriverManager 类去得到连接。若连接成功,则返回一个 Connection 类的对象 databaseConnection,以后对这个数据源的操作都是基于 databaseConnection 对象的。

Connection 类代表与数据库的连接。连接过程包括所执行的 SQL 语句和在该连接上所返回的结果。一个应用程序可与单个数据库进行一个或多个连接,也可与多个数据库进行连接。

#### (3) 执行 SQL 查询语句。

文中介绍的是基于 Statement 对象的查询方法。执行 SQL 查询语句需要先建立一个 Statement 对象,下面的语句建立名为 myStatement 的 Statement 对象:

```
Statement myStatement  
= databaseConnection.createStatement();
```

Statement 对象用于将 SQL 语句发送到数据库中。实际上有 3 种 Statement 对象,它们都可作为在给定连接上执行 SQL 语句的容器:Statement,PreparedStatement(从 Statement 继承而来)和 CallableStatement(从 PreparedStatement 继承而来)<sup>[6]</sup>。它们专用于发送特定类型的 SQL 语句:Statement 对象用于执行不带参数的简单 SQL 语句;PreparedStatement 对象用于执行带或不带 IN 参数的预编译 SQL 语句;CallableStatement 对象用于执行对数据库已存储过程的调用。Statement 接口提供了执行语句和获取结果的基本方法;PreparedStatement 接口添加了处理 IN 参数的方法;而 CallableStatement 添加了处理 OUT 参数的方法。

通过创建 Statement 对象,可用这个方法执行 SQL 语句,并产生一个结果集对象,如下面的语句所示:

```
ResultSet authorResults  
= myStatement.executeQuery(queryWildcard);
```

该语句将在 authorResults 中返回查询结果,其中 queryWildcard 为预先设定的 SQL 查询语句。

Statement 接口提供了 3 种执行 SQL 语句的方法: executeQuery, executeUpdate 和 execute。使用哪一个方法由 SQL 语句所需产生的内容决定。方法 executeQuery 用于产生单个结果集的语句, 例如 SELECT 语句。方法 executeUpdate 用于执行 INSERT, UPDATE 或 DELETE 语句以及 SQL DDL(数据定义语言)语句, 例如 CREATE TABLE 和 DROP TABLE。INSERT, UPDATE 或 DELETE 语句的效果是修改表中零行或多行中的一列或多列。executeUpdate 的返回值是一个整数, 指示受影响的行数(即更新计数)。对于 CREATE TABLE 或 DROP TABLE 等不操作行的语句, executeUpdate 的返回值总为零。方法 execute 用于执行返回多个结果集、多个更新计数或二者组合的语句。执行语句的所有方法都将关闭所调用的 Statement 对象的当前打开结果集(如果存在)。这意味着在重新执行 Statement 对象之前, 需要完成对当前 ResultSet 对象的处理。应该注意, 继承了 Statement 接口中所有方法的 PreparedStatement 接口都有自己的 executeQuery, executeUpdate 和 execute 方法。Statement 对象本身不包含 SQL 语句, 因而必须给 Statement. execute 方法提供 SQL 语句作为参数。PreparedStatement 对象并不将 SQL 语句作为参数提供给这些方法, 因为它们已经包含预编译 SQL 语句。CallableStatement 对象继承这些方法的 PreparedStatement 形式。

#### (4) 处理对数据库的查询结果。

对 authorResults 对象进行处理后, 才能将查询结果显示给用户。authorResults 对象包括一个由查询语

句返回的一个表, 这个表中包含所有的查询结果。对 authorResults 对象的处理必须逐行进行, 而对每一行中的各个列, 可以按任何顺序进行处理。ResultSet 类的一套 get 方法(这些 get 方法可以访问当前行中的不同列)提供了对这些行中数据的访问, 并可结果集中的 SQL 数据类型转换为 Java 数据类型。

综上所述, 即可实现利用 JDBC-ODBC 桥在 Java 中进行数据库的查询。

### 3 结 语

讨论了利用 Java 语言的 JDBC API 和 JDBC-ODBC 桥完成数据库的 SQL 查询的方法。在此基础上可以构造更为复杂的查询, 以满足用户的不同需求。

#### 参考文献:

- [1] 李 诚, 王 兵. Java 2 简明教程[M]. 北京:清华大学出版社, 2004.
- [2] 王克宏, 张炳文. Java 语言 SQL 接口——JDBC 编程技术[M]. 北京:清华大学出版社, 2001:234-262.
- [3] 刘 欣. 基于 Servlet 和 JDBC 的 Web 数据库访问方案[J]. 山东电子, 2003(1):18-21.
- [4] 刘 巍, 唐学兵. 利用 Java 的多线程技术实现数据库的访问[J]. 计算机应用, 2002(12):121-123.
- [5] Java2 SDK. Standard edition documentation version 1.3.1[M]. US:SUN Microsystems, Inc, 2003.
- [6] Oram, Andy. Database programming with JDBC & Java paperback book[M]. [s.l.]:O'Reilly & Associates, Inc, 2000.

(上接第 12 页)

CT pObj)

```

{
    ECODE ec;
    if(NULL == m_pObj){
        m_pObj = pObj;
        pObj->AddRef();
    }
    else if(pObj != m_pObj){
        this->ObjDispose();
        m_pObj = pObj;
        pObj->AddRef();
    }
}

```

### 3 结 语

用户要使用智能指针, 只需采用宏定义 #define SMARTCLASS 将相关智能指针对象引入代码, 在实

现回调机制时, 只需要调用智能指针相关方法完成简单的注册工作就可以使用回调事件, 在使用完完后也只需调用智能指针的相关方法注销该回调事件即可。使用 CAR 智能指针可大大简化 Callback 机制的实现复杂度。

#### 参考文献:

- [1] Koretide. Elastos2.0 Manual[M/OL]. 2007. <http://www.koretide.com.cn/download/download.php?id=2>.
- [2] Pan A. COM's Principle and COM's Application[M]. Beijing: The Tsinghua Press, 1999.
- [3] Rogerson D. Inside COM: Microsoft's Component Object Model[M]. [s.l.]: Microsoft Press, 1999.
- [4] Eckel B. Thinking in C++ (Second Edition)[M]. [s.l.]: Prentice Hall, 2002.
- [5] Koretide. Website[EB/OL]. 2007. <http://www.koretide.com.cn>.