# Java 语言与 DataBase 之间的接口——JDBC

罗东川 （中科院网络信息中心数据库室 100080）

**摘要**：本文全面分析了作为 Java 语言与 DataBase 之间接口的 JDBC，包括它的工作原理，构成接口的组成部分，以及各部分之间的工作关系。并介绍了作为 JDBC 工作基础的 Driver 的原理和分类。例举了 JDBC 与 ODBC 之间的几点差异，描述了应用 JDBC 之后，给网络环境下的信息系统所带来的变化。

**关键词**：Java JDBC

JDBC(Java DataBase Connectivity)是 Java 语言与数据库管理系统之间的接口，通过它，一个 Java 应用程序可以访问任何一个关系型数据库。这就将数据库技术引入了 Internet/WWW 环境，为开发基于这个环境的信息系统带来了极大的便利。

## 一、JDBC 介绍

构成 Java 语言基础的是一系列类库，而 JDBC 正是定义了 Java 与 Database 之间的接口类库(java.sql 包)。它是 java 语言中执行 SQL 语句的 API，它由一系列类(classes)和接口(interfaces)所组成，这些类和接口均是由 java 语言写的。通过它，一个程序开发者可以在 java 语言中建立与 database 的链接，执行 SQL 语句，处理 SQL 语句返回的结果。

JDBC 的一个最大特点就是它的虚拟关系数据库机制，程序开发者在 JDBC 中所使用的 SQL 语句，与 Database 进行链接的语句均不是针对某一个具体的关系型数据库管理系统。它是针对一个虚拟的一般关系型数据库管理系统的。所以用它编写的应用与具体的数据库管理系统无关，它可以访问任何一个关系型数据库系统，换句话说，你不用针对不同的库编写不同的应用程序。以前我们为了访问 Oracle，需要编写一个应用程序，而这个应用是不能访问 Sybase 的，除非我们再针对 Sybase 编写另一个应用程序。而采用 JDBC 编写的应用，就可以访问各类关系型数据库管理系统，包括 Oracle、Sybase 和 Informix 等。并且，由于它是用 Java 编写的，所以它也可以运行于任何平台上。

使用 JDBC，可以增强 Java 程序的能力，进而丰富在 Internet/web 环境下的应用。比如：可以在你的 Web 主页上增加一个 applet 程序，这个 applet 程序可以实时、动态地从数据库中取得数据，并且这个数据库可以是位于本地机器上，也可以是位于网络上的某一台机器上。

我们知道，用 HTML 语言编制的文档从本质上来说是一种文本文件，也就是说它是一种固定格式的文件，它提供的信息内容是死的、静止的。而通过 CGI 接口编程可以实现与数据库管理系统的链接，使用户可以通过 WWW 浏览器直接查询数据库，这样就可以向 Web 主页提供动态、变化的信息内容。但是 CGI 程序的一个缺点是：程序的编译、链接是与某一个具体的数据库管理系统相联系的，也就是说这个应用程序只能访问某一个具体的数据库管理系统。而使用 JDBC 编制的程序，却能访问各种关系型数据库管理系统。而不需要针对不同的库编制不同的应用程序。

## 二、JDBC 的工作原理

完成一个基于 JDBC 的应用需要一组层次结构的软件产品，它包括：JDBC API，JDBC Driver Manager 和针对某一个特定数据库的 Driver。它的一般工作原理如图 1 所示：

其中的 JDBC Driver Manager 是 JDBC 的管理层，它负责建立应用程序与 Driver 之间的联系，因为针对不同的数据库系统要采用不同的 Driver，这种判断和选择就由 Driver Manager 来完成。它能根据用户应用程序中所提供的信息自动选择不同的 Driver. 因为用户应用程序与某一个数据库系统之间的最终联接是要靠 Driver 来完成的。特定的 Driver 与特定的数据库建立联系，并将应用程序中的相关 SQL 语句调用转换成那个数据库管理系统所能理解的形式，以完成对数据库的操作，并将结果返回给应用程序。而数据库管理系统可以与应用程序位于同一台机器，也可以位于网络中的其他机器上。

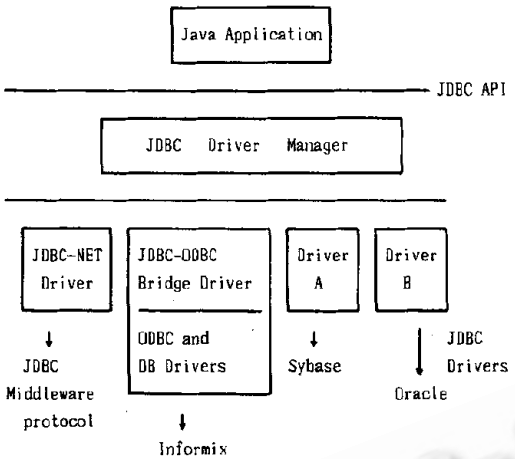


图 1

简单地说:JDBC 的功能包括以下三个方面:

1. **建立与数据库的链接**
2. **发送 SQL 语句**
3. **处理返回结果**

下述代码程序是一个简单的例子,它反映了 JDBC 的上述三个主要功能:

```
Connection con = DriverManager.getConnection(
        "jdbc:odbc:wombat","login","password");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM
Table1");
while (rs.next()){
    int x = getInt("a");
    String s = getString("b");
    float f = getFloat("c");}
```

## 三、JDBC 模式

JDBC 可以以两种模式存取数据库系统:

1. **两层模式**

在这种模式中,应用程序是直接访问数据库的,它要求有一个能够直接访问特定数据库的 JDBC Driver。用户的 SQL 语句被发送给数据库,数据库处理完这个 SQL 语句后,将结果发回给用户。它的原理如图 2 所示:

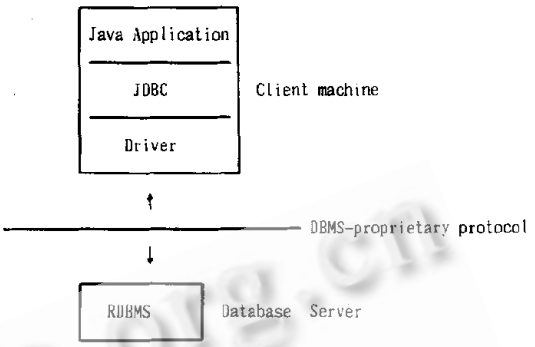


图 2

2. **三层模式**

在这种模式中,应用程序的命令先是被发送到一个被称为 application server 的中间层,然后由中间层将 SQL 语句发送给数据库,而结果也是先返回到中间层,然后由中间层再发送给应用程序。这种模式有两个好处:一是它可以由中间层对数据的存取进行控制并对返回的数据进行加工;另一个好处是,有了这个中间层,使得用户可以开发一些更加高层的应用界面,这些界面是位于这个中间层之上的,它可以是一些用户更乐于使用的界面,如图形用户界面(GUI),它的原理如图 3 所示:

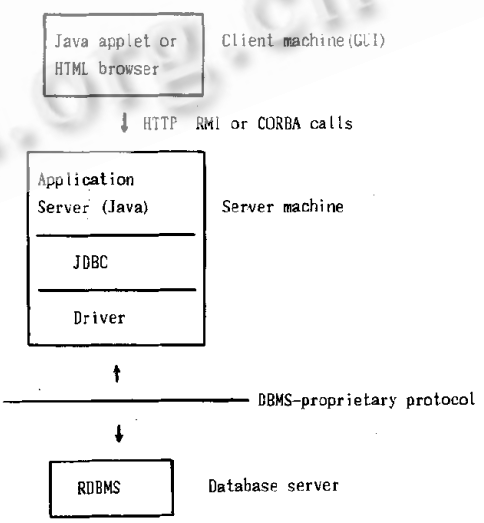


图 3

## 四、JDBC Driver 的分类

通过上面的分析，我们可以看到，基于 JDBC 的应用程序最后是通过一个个不同的 Driver 程序来存取数据库的，并且 Driver 也不是 JDBC 的标准产品组成部分，也就是说，JDBC 并不包括 Driver，这些 Driver 都是由数据库厂家或第三方软件商自行开发的。就目前所见到的类似产品来说，大概可以分成四类：

1. JDBC－ODBC bridge plus ODBC driver

这类 Driver 的实现机制是提供了 JDBC 与 ODBC Driver 之间的接口，一个基于 JDBC 的应用程序通过这个接口与 ODBC Driver 建立联系，进而通过 ODBC Driver 访问数据库管理系统。但这类 Drivers 要求在每一台运行这个应用程序的 client 端机器上都装有 ODBC Driver。

2. Native－API partly－java driver

这类 driver 是针对这样一种情况而设计的：在运行 JDBC 应用的 Client 端机器上，存在有针对某一个数据库管理系统的 API。这类 driver 的作用是建立 JDBC 与这些 API 之间的联系，而最终与数据库打交道的是这些 API。这类 driver 同样也要求在 client 端机器上要装有这种 API。

3. JDBC－Net all－Java driver

这类 driver 包括一个与 DBMS 无关的网络协议层（Net protocol），在这一层中还包括有与不同数据库管理系统进行通信的特殊协议。这样用户应用程序就可以通过它访问各种数据库系统，并且它是完全用 Java 语言编写的，所以它可以运行于任何一台 client 端机器上。

4. Native－protocol all－Java driver

这种 driver 是将 JDBC 应用直接转换成访问某一个特殊数据库管理系统的专门网络协议。它允许 client 端的应用程序直接调用 server 端的数据库系统。

在上述四类 driver 中，前两类应该说只是一种权宜之计，因为它们都不是完全由 Java 语言编写的，都或多或少地包括有一些非 Java 程序，比如：ODBC driver，这些用 C 语言编写的程序，移植性不好，并且要针对每一台 client 端机器进行安装。使用起来很不方便。第三、第四类 driver 应该是今后发展的方向，因为它们是全 Java 程序，可以通过网络自动下载到任一台 clinet 端机器上执行。不存在任何移植上的问题，也不需要在 clinet 端机器上事先安装任何软件。

## 五、JDBC 与 ODBC

Microsoft 公司的 ODBC（Open DataBase Connectivity）API 目前被广泛用于访问各种关系型数据库管理系统，它几乎可以运行于任一种平台上，并可以存取任一种数据库。从上面的分析中我们也可看出，JDBC 的工作原理与 ODBC 是极为相似的。那么为什么我们不在 Java 中直接使用 ODBC 呢？下面几点说明了这个问题：

1. 在 Java 中直接使用 ODBC 是不合适的，因为 ODBC 使用的是 C 语言接口，从 Java 中调用 C 代码在程序安全性、可靠性和可移植性方面是一种退步。

2. 因为 ODBC API 是用 C 语言写的，所以当 ODBC API 与 Java API 之间进行转换时，就存在很大的问题，比如在数据类型方面，Java 中就没有指针。

3. ODBC 很难掌握，它包括了许多复杂的特性。既使是一个简单的查询，也需要设置许多复杂的选项。而 JDBC 就相对简单多了，它是为广大的编程者而设计的。简单明了，易学易用。

4. ODBC 是用 C 语言写的，所以 ODBC driver manager 和 drivers 必须人工地安装在每一台 client 端机器上。而完全用 Java 编写的 JDBC，由于 Java 程序本身的特点，它可以通过网络自动地下载到每一台 client 端机器上。可以说不存在任何安装、配置问题。

## 六、结束语

Java 与 JDBC API 相结合，给一个信息系统的应用带来了极大的便利。让我们设想一下，在一个企业内部存在多种机型，多种操作系统和多种数据库管理系统，你不可能要求企业内各部门使用同一种机器和同一种数据库，这对实际应用是不可能的。那么在这种情况下，要开发一个企业内部的信息系统，就必须受这些软、硬件条件的制约，结果往往是要开发多个版本的应用程序，要针对不同的数据库、不同的硬件平台，开发不同的应用程序。而使用 Java 和 JDBC，情况就完全不同了，Java 的一个最大特点就是移植性好，它的原码经编译后，可通过网络传至各种硬件平台和操作系统中去执行，并不需要做任何改动，而通过 JDBC 接口编程访问数据库，又屏蔽掉了不同数据库管理系统的差异，这样，一个应用程序就可以在任何一台机器上运行。并可查询多种不同的数据库系统。这给应用程序的开发和维护带来了极大的便利。

### 参考文献

[1] 《Java API Overview》Doug Kramer 1996

[2] 《JDBC Guide》Sun Microsystems, Inc. 1996

（来稿时间：1997 年 9 月）