JDBC驱动程序研究和设计

Research and Design of JDBC Driver

王林松 （大连 辽宁师范大学计算机与信息技术学院 116029）

摘 要：介绍了JDBC架构，从实现者的角度详细介绍JDBC驱动程序的四种类型，最后以MySQL为例讲解驱动程序的设计与实现，对需要解决的关键性问题如静态结构、多线程安全、加载时注册等给出答案。

关键词：JDBC 驱动程序 JDBC MySQL

1 引言

由于Java健壮，安全，易于理解，且可以从网络自动下载，所以成为开发数据库应用的一种良好语言。许多Java应用开发者希望能够编写独立于特定DBMS的应用程序，因此需要一个独立于DBMS的Java接口，这个接口能让访问各种DBMS变得更为便捷，让应用开发更为迅速。JDBC就是这样一个“调用级”（call-level）的SQL接口。

JDBC由四部分构成[1]，分别是：

(1) JDBC驱动程序管理器，它记录了已经注册的驱动程序。当Java程序调用DriverManager.getConnection(url,...)获得到数据库连接时，管理器会在已经注册的驱动程序中查找能够接受这个url的驱动程序；

(2) JDBC驱动程序接口，例如java.sql.Connection,java.sql.ResultSet等；

(3) JDBC辅助类，其中包括一些数据类型如java.sql.Timestamp,java.sql.Date等，还包括异常支持类java.sql.SQLException和SQLWarning；

(4) JDBC驱动程序。前三部分已经包含在JDK中。JDBC驱动程序实现JDBC驱动程序接口中预定义的接口，是整个架构必不可少的一部分，本文将介绍JDBC驱动程序的设计和实现方法。

2 JDBC 驱动程序的类型

JDBC驱动程序提供下述四种类型的数据库驱动方式[1]，各有利弊如图1：

2.1 JDBC-ODBC 桥和 ODBC 驱动程序

JDBC-ODBC桥把对JDBC的调用转换成对ODBC的调用，通过ODBC驱动程序访问数据库。这种方法需要安装并配置ODBC驱动程序和JDBC-ODBC桥两种驱动程序。因为通过ODBC驱动程序间接调用DBMS，导致性能低，但是ODBC是一个通用的数据库编程接口，很多DBMS都提供了ODBC驱动程序，因此，在没有纯JAVA驱动程序时，不失为一个好的过渡方法。在JDK中，SUN内置了JDBC-ODBC桥。

2.2 本地API 部分Java驱动程序 (Native-API partly Java driver)

这种类型的驱动程序把对JDBC的调用转换成对Oracle、Sybase、DB2等数据库管理系统的专用客户库接口调用。由于这些客户库往往用C/C+语言编写，所以这种类型的驱动程序同第一种类型的驱动程序一样，也不是纯JAVA驱动程序，并且需要安装客户库在使用此JDBC驱动程序的客户机上。

2.3 JDBC-Net纯Java驱动程序(JDBC-Net pure Java driver)

这种类型的驱动程序的使用不依赖于具体DBMS的网络协议而是与中间服务器通信，由中间服务器根据要访问的DBMS将请求转换成针对具体的DBMS的访问协议。和中间服务器的通信协议由中间服务器的供应商提供。这种配置最灵活，能让Java客户端访问不同的DBMS。已经有厂商为他们现有的数据库中间件产品添加了JDBC驱动程序，如Weblogic。

2.4 专有协议纯 Java 驱动程序(Native-protocol pure Java driver)

这种类型的驱动程序把对JDBC的调用转化成对DBMS直接使用的网络协议，从而允许Java客户端直接访问DBMS。因为这些协议是专用的，因此DBMS厂商是首要的提供者。这类驱动程序性能最好，但是只能访问一种数据库。

在上述四类驱动程序中，第（1）类驱动程序调用ODBC库，第（2）类驱动程序调用DBMS专用库，他们都需要把对JDBC的调用转化成对已有的本地库调用，这些本地库往往用C/C++实现，因此一般采用JNI(Java Native Interface)调用实现；第（3）类和第（4）类驱动程序把对JDBC的调用转化成网络协议数据包发给中间服务器的监听器或者DBMS的监听器，同时接收返回的数据包并把它们转化成JDBC调用的返回值，这两种驱动程序的实现类似。下面以MySQL为例讲述第

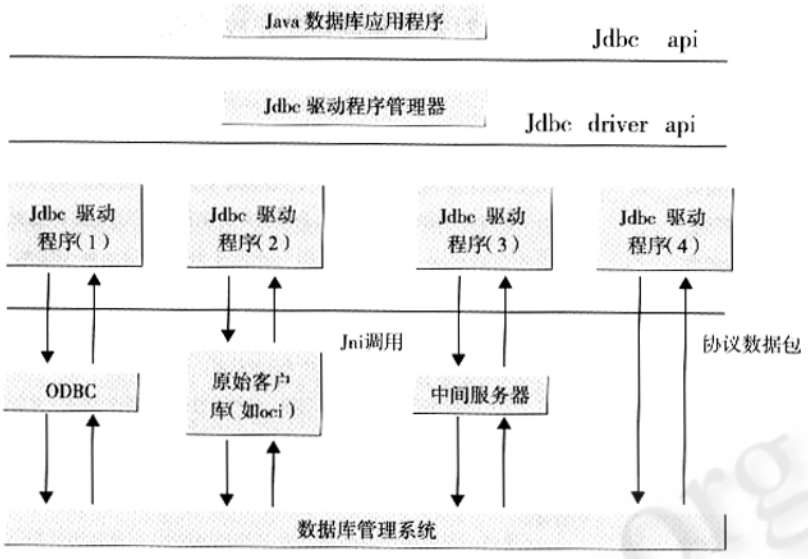

图 1 JDBC 驱动程序的四种类型以及在整个 JDBC 中的位置

(4)类驱动程序的设计与实现。

3 JDBC 驱动程序开发

每个JDBC驱动程序必须实现java.sql.Connection、java.sql.Statement、java.sql.PreparedStatement、java.sql.CallableStatement、java.sql.ResultSet等接口，还要提供一个实现java.sql.Driver接口的类。java.sql.DriverManager使用Driver类为一个数据库URL定位一个驱动程序，因此驱动程序必须使用Driver类在DriverManager中注册。同时java.sql对象的所有操作必须是多线程安全的，当有几个线程同时调用同一个对象时，应能够正确处理并发请求而不发生错误。

3.1 驱动程序的静态逻辑结构

驱动程序接口之间的主要关系如图2所示[1]，在驱动程序实现时，这些接口类会保持这些关系。DatabaseMetaData对象可以从Connection对象调用getMetaData方法得到，ResultSetMetaData对象可以从ResultSet调用getMetaData方法得到。为了能够通过Statement接口执行SQL语句，Statement对象应包含

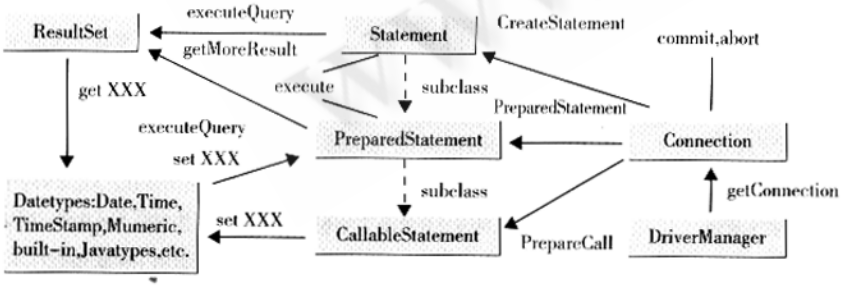

图 2 驱动程序接口之间的重要关系

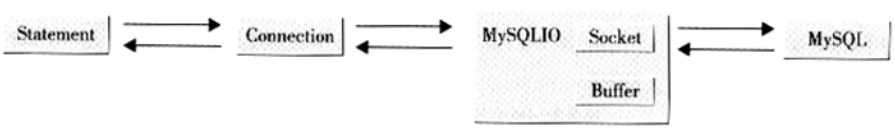

图 3 对象之间的重要关联

对Connection对象的引用，Connection类的实例包含对MySQLIO对象的引用。而只有MySQLIO类实例和MySQL直接交互。MySQLIO包含了一个Socket和一个Buffer，Socket用于和MySQL监听器通信，Buffer用于缓冲收发的数据包，还提供一些辅助函数用于打包、解包。其关系如图3所示。

3.2 Driver 类的加载和注册

当Java程序调用Class.forName()加载Driver类进解释器时，此Driver类使用静态块(static block)创建一个Driver类的实例并注册此驱动程序。代码如下：

```
public class Driver implements java.sql.Driver{
    static{
        try{
        java.sql.DriverManager.registerDriver(new Driver());
        }
        catch (java.sql.SQLException E){
        E.printStackTrace();
        }
    }
}
```

3.3 使用网络协议和 DBMS 通信

MySQLIO类负责和数据库监听器通信。它包含一个Socket，通过Socket向MySQL Server发送命令，并接收响应，一些重要数据成员和方法如下，和MySQL通信使用的包格式可以通过MySQL的源代码得到[2]。

```
class MySQLIO
{
private Socket _conn;
init();
final Buffer sendCommand(int command,
String ExtraData,
Buffer QueryPacket) throws Exception{...}
final byte[][]nextRow(int columnCount) throw
```

下转第 43 页 >>

```
Exception{...}
    final ResultSet sqlQuery(String Query,int
max_rows,String Encoding)
    throws Exception{...}
    final void SendPacket(Buffer packet) throw
IOException{...}
    final Buffer ReadPacket() throw IOException
{...}
    }
```

3.4 多线程同步问题

JDBC规范要求对象的所有操作必须是线程安全的,可以使用java的关键字synchronized实现正确的并发处理。例如，在一个到DBMS的连接上，不能同时执行两个命令，也就是说，对Connection类中MysqlIO对象的操作必须互斥进行。当synchronized(_IO)以后，这个对象加锁，在锁被释放以前，没有别的线程可以访问这个对象，从而使得访问互斥进行。也可以在需要互斥的对象的成员方法添加synchronized修饰符，这样在执行这个方法时，this对象上锁，使得只有一个线程可以在临界区中执行。

```
    public class Connection implements java.
sql.Connection{
    MySQLIO _IO;
    public ResultSet execSQL(String sql,int
max_rows,
    Buffer Packet) throws java.sql.SQLException
{
    synchronized(_IO){
    //use MySQLIO to send packet to MySQL
and read response,
    //then transform the response into ResultSet
    }
    }
    }
```

4 结束语

在开发JDBC驱动程序时，首先应根据数据库驱动方式决定驱动程序的类型，然后进行设计实现。本文以第四类驱动程序开发为例进行讲解，改变数据库驱动方式，例如通过JNI调用DBMS，可以很容易转化为相应类型的驱动程序。

参考文献

1 JavaSoft company. JDBC specification 3.0, http://java.sun.com/products/jdbc/download.html, 2003。

2 MySQL AB. MySQL3.23.36 Source Code, http://www.mysql.com/downloads/index.html, 2001。

3 Sun Microsystems,Inc. Java Tutorial[M], http://java.sun.com/docs/books/tutorial/, 2003。

4 Brain Jepson著,钱毅译，Java数据库编程指南，电子工业出版社，1998。