学生应知信息知识

Java知识手册（九）

狄登峰 主编

目　　录

使用 JDBC 创建数据库访问程序

什么是数据库？

数据库是以某种文件结构存储的一系列信息表，这种文件结构使您能够访问这些表、选择表中的列、对表进行排序以及根据各种标准选择行。数据库通常有多个索引与这些表中的许多列相关联，所以我们能尽可能快地访问这些表。

以员工记录为例，您可以设想一个含有员工姓名、地址、工资、扣税以及津贴等内容的表。让我们考虑一下这些内容可能如何组织在一起。您可以设想一个表包含员工姓名、地址和电话号码。您希望保存的其它信息可能包括工资、工资范围、上次加薪时间、下次加薪时间、员工业绩评定等内容。

这些内容是否应保存在一个表格中？几乎可以肯定不应该如此。不同类别的员工的工资范围可能没有区别；这样，您可以仅将员工类型储存在员工记录表中，而将工资范围储存在另一个表中，通过类型编号与这个表关联。考虑以下情况：

Key Lastname SalaryType SalaryType Min Max

1 Adams 2 1 30000 45000

2 Johnson 1 2 45000 60000

3 Smyth 3 3 60000 75000

4 Tully 1

5 Wolff 2

SalaryType 列中的数据引用第二个表。我们可以想象出许多种这样的表，如用于存储居住城市和每个城市的税值、健康计划扣除金额等的表。每个表都有一个主键列（如上面两个表中最左边的列）和若干数据列。在数据库中建立表格既是一门艺术，也是一门科学。这些表的结构由它们的范式指出。我们通常说表属于 1NF、2NF 或 3NF。

第一范式：表中的每个表元应该只有一个值（永远不可能是一个数组）。(1NF)

第二范式：满足 1NF，并且每一个非主键列完全依赖于主键列。这表示主键和该行中的剩余表元之间是 1 对 1 的关系。(2NF)

第三范式：满足 2NF，并且所有非主键列是互相独立的。任何一个数据列中包含的值都不能从其他列的数据计算得到。(3NF)

现在，几乎所有的数据库都是基于“第三范式 (3NF)”创建的。这意味着通常都有相当多的表，每个表中的信息列都相对较少。

从数据库中获取数据

假设我们希望生成一个包含员工及其工资范围的表，在我们设计的一个练习中将使用这个表。这个表格不是直接存在在数据库中，但可以通过向数据库发出一个查询来构建它。我们希望得到如下所示的一个表：

Name Min Max

Tully $30,000.00 $45,000.00

Johnson $30,000.00 $45,000.00

Wolff $45,000.00 $60,000.00

Adams $45,000.00 $60,000.00

Smyth $60,000.00 $75,000.00

我们发现，获得这些表的查询形式如下所示

SELECT DISTINCTROW Employees.Name, SalaryRanges.Min,

SalaryRanges.Max FROM Employees INNER JOIN SalaryRanges ON Employees.SalaryKey = SalaryRanges.SalaryKey

ORDER BY SalaryRanges.Min;

这种语言称为结构化查询语言，即 SQL，**而且它是几乎目前所有数据库都可以使用的一种语言。**SQL-92 **标准被认为是一种基础标准，而且已更新多次。**

数据库的种类

PC **上的数据库，如** dBase、Borland Paradox、Microsoft Access **和** FoxBase。

数据库服务器：IBM DB/2、Microsoft SQL Server、Oracle、Sybase、SQLBase **和** XDB。

所有这些数据库产品都支持多种相对类似的 SQL **方言，因此，所有数据库最初看起来好象可以互换。每种数据库都有不同的性能特征，而且每一种都有不同的用户界面和编程接口。**

ODBC

如果我们能够以某种方式编写不依赖于特定厂商的数据库的代码，并且能够不改变自己的调用程序即可从这些数据库中得到相同的结果，那将是一件很好的事。如果我们可以仅为所有这些数据库编写一些封装，使它们具有相似的编程接口，这种对数据库编程独立于供应商的特性将很容易实现。

什么是 JDBC**？**

JDBC **是对** ODBC API **进行的一种面向对象的封装和重新设计，它易于学习和使用，并且它真正能够使您编写不依赖厂商的代码，用以查询和操纵数据库。尽管它与所有** Java API **一样，都是面向对象的，但它并不是很高级别的对象集**.

除 Microsoft 之外，多数厂商都采用了 JDBC，并为其数据库提供了 JDBC 驱动程序；这使您可轻松地真正编写几乎完全不依赖数据库的代码。另外，JavaSoft 和 Intersolv 已开发了一种称为 JDBC-ODBC Bridge 的产品，可使您连接还没有直接的 JDBC 驱动程序的数据库。支持 JDBC 的所有数据库必须至少可以支持 SQL-92 标准。这在很大程度上实现了跨数据库和平台的可移植性。

安装和使用 JDBC

JDBC 的类都被归到 java.sql 包中，在安装 Java JDK 1.4 时会自动安装。然而，如果您想使用 JDBC-ODBC 桥。JDBC-ODBC 驱动程序可从 Sun 的 Java 网站 (http://java.sun.com/) 轻松地找到并下载。在您扩充并安装了这个驱动程序后，必须执行下列步骤：

将 \jdbc-odbc\classes; 路径添加到您的 PATH 环境变量中。

将 \jdbc-odbc\classes; 路径添加到您的 CLASSPATH 环境变量中。

JDBC 驱动程序的类型

Java 程序连接数据库的方法实际上有四种：

1. JDBC-ODBC 桥和 ODBC 驱动程序 -- 在这种方式下，这是一个本地解决方案，因为 ODBC 驱动程序和桥代码必须出现在用户的每台机器中。从根本上说

这是一个临时解决方案。

2. 本机代码和 Java 驱动程序 -- 它用另一个本地解决方案（该平台上的 Java 可调用的本机代码）取代 ODBC 和 JDBC-ODBC 桥。

3. JDBC 网络的纯 Java 驱动程序 -- 由 Java 驱动程序翻译的 JDBC 形成传送给服务器的独立协议。然后，服务器可连接任何数量的数据库。这种方法使您可能从客户机 Applet 中调用服务器，并将结果返回到您的 Applet。在这种情况下，中间件软件提供商可提供服务器。

4. 本机协议 Java 驱动程序 -- Java 驱动程序直接转换为该数据库的协议并进行调用。这种方法也可以通过网络使用，而且可以在 Web 浏览器的 Applet 中显示结果。在这种情况下，每个数据库厂商将提供驱动程序。

如果您希望编写代码来处理 PC 客户机数据库，如 dBase、Foxbase 或 Access，则您可能会使用第一种方法，并且拥有用户机器上的所有代码。更大的客户机-服务器数据库产品（如 IBM 的 DB2）已提供了第 3 级别的驱动程序。

两层模型和三层模型

当数据库和查询它的应用程序在同一台机器上，而且没有服务器代码的干预时，我们将生成的程序称为两层模型。一层是应用程序，而另一层是数据库。在

JDBC-ODBC 桥系统中通常是这种情况。

当一个应用程序或 applet 调用服务器,服务器再去调用数据库时,我们称其为三层模型。当您调用称为“ 服务器 ”的程序时通常是这种情况。

编写 JDBC 代码访问数据库

用 ODBC 注册您的数据库

连接数据库

所有与数据库有关的对象和方法都在 java.sql 包中,因此在使用 JDBC 的程序中必须加入 "import java.sql.* "。 JDBC 要连接 ODBC 数据库,您必须首先加载 JDBC-ODBC 桥驱动程序

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

该语句加载驱动程序,并创建该类的一个实例。然后,要连接一个特定的数据库,您必须创建 Connect 类的一个实例,并使用 URL 语法连接数据库。

```
String url = "jdbc:odbc:Northwind";

Connection con = DriverManager.getConnection(url);
```

请注意,您使用的数据库名是您在 ODBC 设置面板中输入的“ 数据源 ”名称。

URL 语法可能因数据库类型的不同而变化极大。

jdbc:subprotocol:subname

第一组字符代表连接协议，并且始终是 jdbc。还可能有一个子协议，在此处，子协议被指定为 odbc。它规定了一类数据库的连通性机制。如果您要连接其它机器上的数据库服务器,可能也要指定该机器和一个子目录：

jdbc:bark//doggie/elliott

最后，您可能要指定用户名和口令，作为连接字符串的一部分：

jdbc:bark//doggie/elliot;UID=GoodDog;PWD=woof

访问 MSSQL Server 方法：(驱动程序需要：msutil.jar,msbase.jar,mssqlServer.jar)

DBDriver=com.microsoft.jdbc.sqlserver.SQLServerDriver

URL=jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=demo

username=sa

password=

maxcon=10

mincon=1

poolName=SkyDev

利用我们开发的数据库类，使用方法如下：

DbObject DbO = new DbObject(new SqlServerConnectionFactory("loc

1433, "demo", "sa", ""));

Connection con = DbO.getConnection();

//**类代码（不含连接工厂实现）**

package skydev.modules.data;

public final class SqlServerConnectionFactory

extends ConnectionFactory {

private final String dbDriver =

"com.microsoft.jdbc.sqlserver.SQLServerDriver";

private String host;

private int port;

private String databaseName;

public SqlServerConnectionFactory() {

super.setDriverName(dbDriver);

```
}
/**
*
* @param host 数据库所在的主机名：如"localhost"
* @param port SQL 服务器运行的端口号，如果使用缺省值 1433，负数即可
* @param databaseName 数据库名称
* @param userName 用户名
* @param password 口令
*/
public SqlServerConnectionFactory(String host,
int port,
String databaseName,
String userName,
String password) {
this.setHost(host);
this.setPort(port);
```

```
this.setDatabaseName(databaseName);
this.setUserName(userName);
this.setPassword(password);
init();
}
private void init() {
super.setDriverName(dbDriver);
super.setUrl("jdbc:microsoft:sqlserver://" + host.trim() + ":" +
new Integer(port).toString() + ";DatabaseName=" +
databaseName.trim());
//super.setUrl("jdbc:microsoft:sqlserver://localhost:1433;DatabaseNam
}
……
//--------------------------------------------------------------------------
```

访问 MySQL **的方法：**

```
DBDriver=com.mysql.jdbc.Driver
```

URL=jdbc:mysql://localhost/demo

username=

password=

maxcon=5

mincon=1

poolName=zhengmao

访问数据库

一旦连接到数据库，就可以请求表名以及表列的名称和内容等信息，而且您可以运行 SQL **语句来查询数据库或者添加或修改其内容。可用来从数据库中获取信息的对象有：**

DatabaseMetaData **有关整个数据库的信息：表名、表的索引、数据库产品的名称和版本、数据库支持的操作。**

ResultSet **关于某个表的信息或一个查询的结果。您必须逐行访问数据行，但是您可以任何顺序访问列。**

ResultSetMetaData **有关** ResultSet **中列的名称和类型的信息。**

尽管每个对象都有大量的方法让您获得数据库元素的极为详细的信息，但在每个对象中都有几种主要的方

法使您可获得数据的最重要信息。然而，如果您希望看到比此处更多的信息，建议您学习文档以获得其余方法的说明。

ResultSet

ResultSet 对象是 JDBC 中最重要的单个对象。从本质上讲，它是对一个一般宽度和未知长度的表的一种抽象。几乎所有的方法和查询都将数据作为 ResultSet 返回。ResultSet 包含任意数量的命名列，您可以按名称访问这些列。它还包含一个或多个行，您可以按顺序自上而下逐一访问。在您使用 ResultSet 之前，必须查询它包含多少个列。此信息存储在 ResultSetMetaData 对象中。

```
//从元数据中获得列数
ResultSetMetaData rsmd;
rsmd = results.getMetaData();
numCols = rsmd.getColumnCount();
```

当您获得一个 ResultSet 时，它正好指向第一行之前的位置。您可以使用 next() 方法得到其他每一行，当没有更多行时，该方法会返回 false。由于从数据库中获取数据可能会导致错误，您必须始终将结果集处理语句包括在一个 try 块中。

您可以多种形式获取 ResultSet 中的数据，这取决于每个列中存储的数据类型。另外，您可以按列序号或列名获取列的内容。请注意，列序号从 1 开始，而不是从 0 开始。ResultSet 对象的一些最常用方法如下所示。

getInt(int); 将序号为 int 的列的内容作为整数返回。

getInt(String); 将名称为 String 的列的内容作为整数返回。

getFloat(int); 将序号为 int 的列的内容作为一个 float 型数返回。

getFloat(String); 将名称为 String 的列的内容作为 float 型数返回。

getDate(int); 将序号为 int 的列的内容作为日期返回。

getDate(String); 将名称为 String 的列的内容作为日期返回。

next(); 将行指针移到下一行。如果没有剩余行，则返回 false。

Close(); 关闭结果集。

getMetaData(); 返回 ResultSetMetaData 对象。

ResultSetMetaData

您使用 getMetaData() 方法从 ResultSet 中获取 ResultSetMetaData 对象。您可以使用此对象获得列的数目和类型以及每一列的名称。

getColumnCount(); 返回 ResultSet 中的列数。

getColumnName(int); 返回列序号为 int 的列名。

getColumnLabel(int); 返回此列暗含的标签。

isCurrency(int); 如果此列包含带有货币单位的一个数字，则返回 true。

isReadOnly(int); 如果此列为只读，则返回 true。

isAutoIncrement(int); 如果此列自动递增，则返回 true。这类列通常为键，而且始终是只读的。

getColumnType(int); 返回此列的 SQL 数据类型。这些数据类型包括

BIGINT

BINARY

BIT

CHAR

DATE

DECIMAL