第十五章 与数据库通信功能的设计和实现

数据的存储和集中管理越来越成为 INTERNET 网络的重要内容。几乎每一个投入运行的商业软件系统都会和数据或数据库有着千丝万缕的联系。在本章中，我们将学习和讨论如何通过 JDBC 编程接口，实现 JAVA 应用程序和数据库系统之间交互通信的功能。这将是我们在下一章分析和讨论的网络购物应用程序的基础。

在本书的第六章和第七章之中，我们详细介绍了使用 JAVA 语言编写 JDBC 数据库编程接口。在本章的学习实践过程中，如果读者对 JDBC 数据库编程接口的基本知识还有不清楚的地方，请查阅本书的第六章和第七章。

让我们从如何创建数据库中的数据表开始本章的学习。

15．1 数据表的创建

数据库中的数据表的创建一般来说有两种方法：

一是使用各种数据库的图形界面创建；

二是采用标准的 SQL 语句，使用各种数据库的应用工具或自己编写专门的程序来创建。

下面我们就以实例的形式详细介绍这两种方法。

15．1．1 图形界面创建数据表

我们以 MICROSOFT ACCESS 数据库为例，看看如何通过数据库的图形界面来创建数据表。在下一章“网络购物功能的设计和实现”中，我们要使用到两个重要的数据表：已注册用户登录的用户数据表（USER 表）和在网络上定购商品的客户的客户数据表（CUSTOMER 表）。我们现在就先来创建这两个数据表。

创建 USER 表

在 MICROSOFT ACCESS 数据库中创建 USER 表的步骤如下：

打开 MICROSOFT ACCESS 应用程序，单击“新建”按钮，如图 15-1 所示。

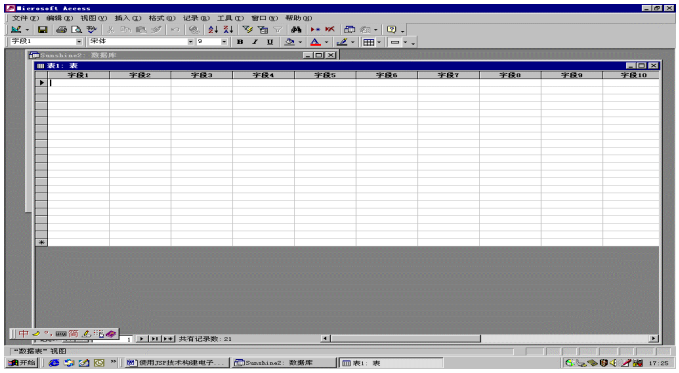

图 15-1　在 MICROSOFT ACCESS 数据库中创建新表 USER

1）在一个空表中填写和定义 USER 表的各项数据项，具体定义的各项数据项的内容如表 15-1 所示：

表 15-1　USER 表的数据项内容

数据项名	数据格式	数据长度	允许空字符	是否必填项
Name	文本	char(16)	NOT NULL	是
Password	文本	char(8)	NOT NULL	是
Email	文本	char(40)	NOT NULL	是
www	文本	char(50)	NULL	否
Tel	文本	char(20)	NULL	否
Address	文本	char(50)	NULL	否
Postcode	文本	char(10)	NULL	否

其中 name、password、email 不允许出现空字符，name、password、email 为必填项。

填写和定义完成 USER 表的各项数据项之后，界面显示情况如图 15-2 所示。

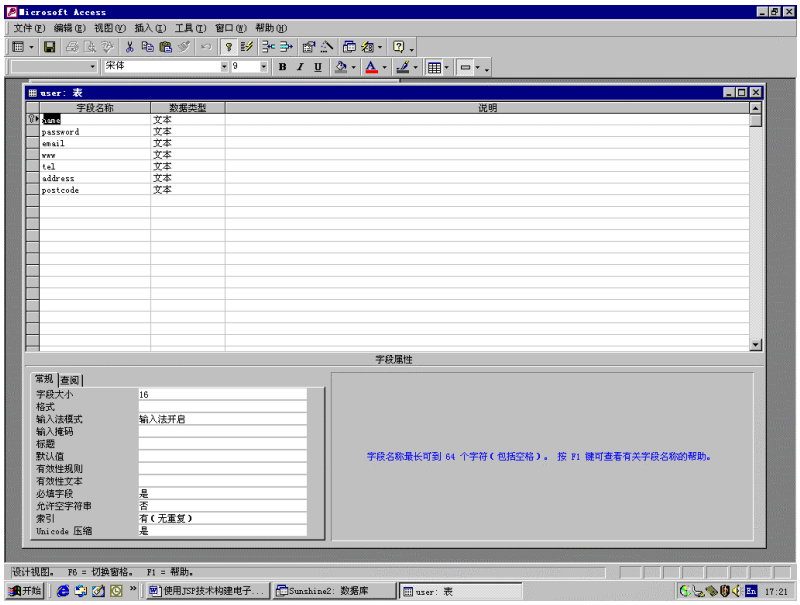

图 15-2　填写和定义完成 USER 表的后的界面显示情况

2）可以通过点击“设计”按钮，来查看和修改已生成的表，如图 15-3 所示。

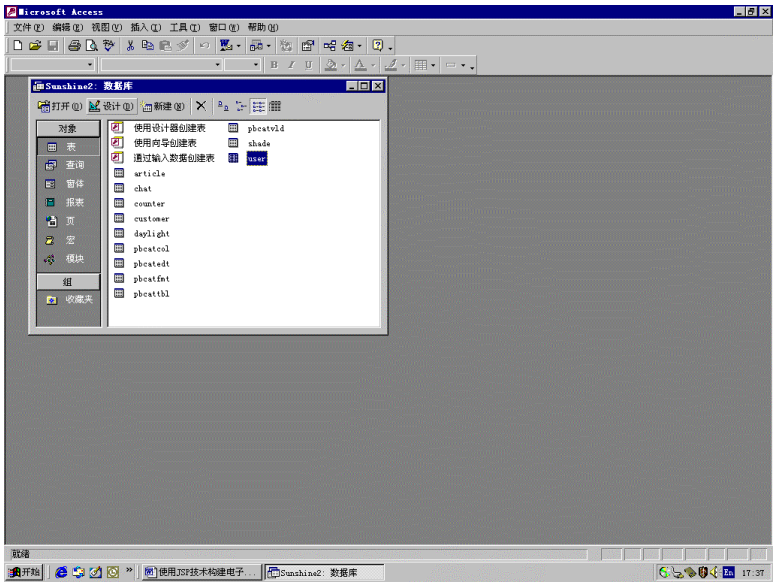

图 15-3　查看和修改已生成的旧表

比如说我们查看刚刚生成的 USER 表，查看的结果如图 15-4 所示：

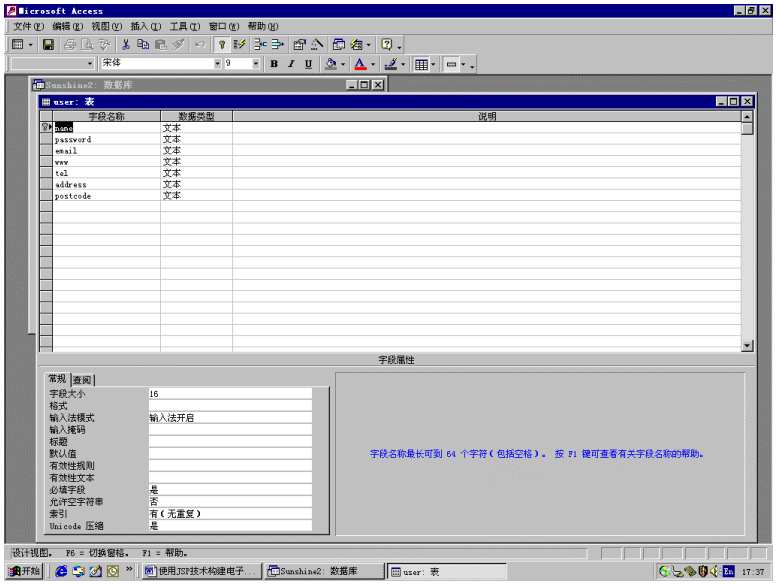

图 15-4　查看刚刚生成的 USER 表

创建 CUSTOMER 表

在下一章中我们将要使用到 CUSTOMER 表，用于保存网上购物的用户购物信息和个人信息等等。

在 MICROSOFT ACCESS 数据库中创建 CUSTOMER 表的步骤如下：

1）打开 MICROSOFT ACCESS 应用程序，单击“新建”按钮，新建表 CUSTOMER。如图 15-5 所示。

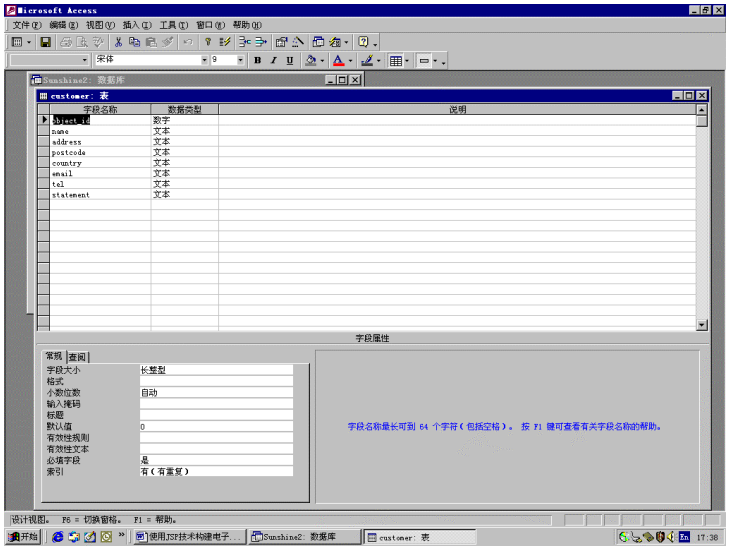

图 15-5　在 MICROSOFT ACCESS 数据库中创建新表 CUSTOMER

2）在一个空表中填写和定义 CUSTOMER 表的各项数据项，具体定义的各项数据项的内容如表 15-2 所示：

表 15-2　CUSTOMER 表的数据项内容

数据项名	数据格式	数据长度	允许空字符	是否必填项
object_id	长整型	long	NOT NULL	是
name	文本	char(16)	NOT NULL	是
address	文本	char(50)	NULL	否
postcode	文本	char(10)	NULL	否
country	文本	char(10)	NULL	否
email	文本	char(40)	NULL	否
tel	文本	char(20)	NULL	否
statement	文本	char(10)	NULL	否

其中 object_id 、name 不允许出现空字符，object_id 、name 为必填项。

3）可以通过点击“设计”按钮，来查看和修改已生成的表。

15．1．2　利用 SQL 创建数据表

利用 SQL 创建数据表又有两种方法可供选择：
一是使用各种数据库的应用工具来创建；
二是自己编写专门的程序来创建。
下面我们就来分别研究和讨论这两种方法。其中我们将重点介绍第二种方法。

15．1．2．1　使用数据库应用工具创建

对于使用数据库的应用工具的创建方法，我们需要事先使用标准 SQL 语言格式在数据库应用工具中编写好 SCRIPT，再使用数据库应用工具的相关功能来完成。如下所示的 SCRIPT

将创建两个表：User 表和 Customer 表：

```
CREATE TABLE User
    (name      char(16)  NOT NULL,
     password  char(8)   NOT NULL,
     email     char(40)  NULL,
     www       char(50)  NULL,
     tel       char(20)  NULL,
     address   char(50)  NULL,
     postcode  char(10)  NULL)

CREATE TABLE Customer
    (object_id long      NOT NULL,
     name      char(16)  NOT NULL,
     address   char(50)  NULL,
     postcode  char(10)  NULL,
     country   char(10)  NULL,
     email     char(40)  NULL,
     tel       char(20)  NULL,
     statement char(10)  NULL)
```

而为这些刚创建的新表添加新数据的 SQL SCRIPT 语句如下：

```
INSERT INTO Users
VALUES('Larry, Lin', 'Hekt47Gj', 'larry@yahoo.com', 'www.postgoods.com',
'020-83467793', '', '')
...

INSERT INTO Customer
VALUES('0001', 'Larry, Lin', 'West Street BOX 450,Dong Yuan Heng,Guangzhou',
'510110', 'China', 'larry@yahoo.com', '020-84543342', '购买一件')
...
```

15．1．2．2　编写专门程序创建

编写专门程序创建的方法是我们重点介绍和讨论的方法。这种方法的特点是有很高的灵活性。可以根据用户的需要和应用程序的要求自行定制。下面的例子中的 CreateDB.java 不但可以通用地创建各种新表的各个数据项，而且还能够方便地实现添加各数据项的具体数据的功能。

本节中涉及所有源程序在 SUN 公司的 JDK1.3 for Windows 上调试通过（调试时间：2001 年 1 月 8 日）。

CreateDB.java 的源代码：

```
001/**
```

```
002 * CreateDB.java
003 *
004 */
005
006import java.net.*;
007import java.sql.*;
```

第 7 行导入了包 java.sql.*，该包定义了进行 SQL 操作的相关方法。

```
008import java.io.*;
009import java.util.*;
010
011class CreateDB
012{  public static void main (String args[])
013   {  try
014      {  Connection con = getConnection();
```

第 14 行通过调用 DriverManager.getConnection()方法获得了一个 Connection 对象：con。

```
015         Statement stmt = con.createStatement();
```

第 15 行根据在第 14 行中获得的 Connection 对象（con），创建了一个 Statement 对象：stmt。这个 Statement 对象将在下面的程序中用来执行各种 SQL 语句。如下面的第 78 行和第 83 行所示：

```
stmt.executeUpdate(command);
```

```
016
017         String tableName = "";
018         if (args.length > 0)
019            tableName = args[0];
020         else
021         {  System.out.println("Usage: CreateDB TableName");
022            System.exit(0);
023         }
```

第 16-23 行引导用户输入表名。

本程序编译成功后，正确的使用格式是：

```
>java CreateDB <表格名>
```

第 16-23 行保证了当用户只输入：

```
>java CreateDB
```

会提醒用户按正确的使用格式输入。

```
024
025         BufferedReader in = new BufferedReader(new
026            FileReader(tableName + ".dat"));
027
```

第 24-27 行调用了 BufferedReader 中的 FileReader 方法，从用户输入的文件名中获取

数据表的资料。

```
028          createTable(tableName, in, stmt);
```

第 28 行调用 createTable 方法，创建新表，并对新表中的数据项赋值。我们在下面的程序中将详细讨论到。

```
029          showTable(tableName, stmt);
```

第 29 行调用 showTable 方法，显示创建的新表中的数据内容。我们在下面的程序中将详细讨论到。

```
030
031          in.close();
```

第 31 行关闭打开的数据文件*.dat。

```
032          stmt.close();
```

第 32 行关闭了用来执行各种 SQL 语句的 Statement 对象。

```
033          con.close();
```

第 33 行关闭了用 DriverManager.getConnection()方法获得的 Connection 对象。

```
034       }
035       catch (SQLException ex)
036       {  System.out.println ("SQLException:");
037          while (ex != null)
038          {  System.out.println ("SQLState: "
039                + ex.getSQLState());
040             System.out.println ("Message:  "
041                + ex.getMessage());
042             System.out.println ("Vendor:   "
043                + ex.getErrorCode());
044             ex = ex.getNextException();
045             System.out.println ("");
046           }
047       }
```

第 34-47 行显示有关 SQL 的异常信息。

```
048       catch (IOException ex)
049       {  System.out.println("Exception: " + ex);
050          ex.printStackTrace ();
051       }
052    }
053
```

第 48-53 行显示输入输出操作的异常信息。

```
054   public static Connection getConnection()
055      throws SQLException, IOException
056   {  Properties props = new Properties();
057      String fileName = "CreateDB.properties";
058      FileInputStream in = new FileInputStream(fileName);
059      props.load(in);
060
```

第 54-60 行先定义了 Properties 对象：props，并从文件 CreateDB.properties 中将相关的属性设置信息载入了 props 中。属性定义文件 CreateDB.properties 我们将在下面的内容中详细介绍。

```
061 String drivers = props.getProperty("jdbc.drivers");

062      if (drivers != null)
063         System.setProperty("jdbc.drivers",drivers);
```

第 61-63 行从属性文件 CreateDB.properties 中获得"jdbc.drivers"字段的内容。如果该"jdbc.drivers"属性不为空值，立即把该属性赋给应用程序系统。

```
064      String url = props.getProperty("jdbc.url");
```

第 64 行从属性文件 CreateDB.properties 中获得"jdbc.url"字段的内容。

```
065      String username = props.getProperty("jdbc.username");
```

第 65 行从属性文件 CreateDB.properties 中获得"jdbc.username"字段的内容。

```
066      String password = props.getProperty("jdbc.password");
```

第 66 行从属性文件 CreateDB.properties 中获得"jdbc.password "字段的内容。

```
067
068      return
069         DriverManager.getConnection(url, username, password);
070   }
```

第 69-70 行采用从属性文件 CreateDB.properties 中获得的 url、 username、password 等属性配置，在 DriverManager 中注册一个连接。

```
071
072   public static void createTable(String tableName,
073      BufferedReader in, Statement stmt)
```

第 72-73 行：方法 createTable 中有三个参数：

1）字符串型变量 tableName；

2）BufferedReader 类型的对象 in；

3）Statement 类型的对象 stmt。

```
074      throws SQLException, IOException
```

```
075   {  String line = in.readLine();
076      String command = "CREATE TABLE " + tableName
077         + "(" + line + ")";
078      stmt.executeUpdate(command);
079
```

第 74-79 行将完成创建新表的功能。其中第 78 行使用了在第 15 行中创建的 Statement 对象——stmt，来执行 SQL 的语句 Update，完成创建新表的功能。

```
080      while ((line = in.readLine()) != null)
081      {  command = "INSERT INTO " + tableName
082            + " VALUES (" + line + ")";
083         stmt.executeUpdate(command);
084      }
085   }
```

第 80-85 行将完成往创建的新表中插入和添加数据的功能。其中第 83 行使用了在第 15 行中创建的 Statement 对象——stmt，来执行 SQL 的语句 Update，完成在创建的新表中插入和添加数据的功能。

```
086
087   public static void showTable(String tableName,
088      Statement stmt) throws SQLException
```

第 87-88 行：方法 showTable 中有两个参数：

1）符串型变量 tableName；

2）Statement 类型的对象 stmt。

该方法将用来完成在 DOS 的命令行中显示生成的数据。

```
089   {  String query = "SELECT * FROM " + tableName;
090      ResultSet rs = stmt.executeQuery(query);
```

第 90 行使用了 executeQuery()方法，用来查询数据库的某一表值。executeQuery()方法查询表值通常是指使用简单的 SELECT 语句去查询的情形。

使用 executeQuery()方法的步骤为：

1.从一个连接中生成一个 statement 对象；

2.执行查询；

3.从 Statement 对象那里得到结果集。

而得到的结果集赋给了 ResultSet 类型的变量 rs。

ResultSet 的结构被组织成为逻辑上的行与列数据，并含有一个指向当前行的指针。通过调用 next()函数（如第 93 行所示），可以将指针移到下一行，直到把要求读取的数据全部读完为止。

```
091      ResultSetMetaData rsmd = rs.getMetaData();
092      int columnCount = rsmd.getColumnCount();
093      while (rs.next())
```

```
094       {  for (int i = 1; i <= columnCount; i++)
095          {  if (i > 1) System.out.print(", ");
096             System.out.print(rs.getString(i));
097          }
098          System.out.println();
099       }
```

第 91-99 行：我们可以先用 ResultSetMetaData 类型的变量连接上数据库，然后通过对 ResultSetMetaData 的 getColumnCount 操作得到列值总数（columnCount），再搜索全部列并输出每个列项；之后，搜索全部行，打印出每个数值。

```
100       rs.close();
```

第 100 行关闭对象 rs ，以回收宝贵的 ResultSet 资源。

```
101    }
102}
```

通过上面这个实例的学习和讨论，读者们可能已经体会到：使用 JDBC 处理数据远不是只得到数据的数值那么简单。还必须通过查询 metadata 结构的结果，来决定数据的有效性和格式的正确性。

CreateDB.properties 的源代码：

CreateDB.properties 是上例中 JDBC 编程接口配置的属性文件。

```
01 jdbc.drivers=sun.jdbc.odbc.JdbcOdbcDriver
```

第 1 行定义了使用 JDBC/ODBC 驱动桥作为应用程序和数据库之间的 JDBC 编程接口。

```
02 jdbc.url=jdbc:odbc:sunshine2
```

当用户连接数据库的时候，必须指定数据资源和一些参数，我们称之为 URL（通用资源定位器 Uniformed Resource Location）。

第 2 行中定义了该 JDBC 的 URL。其中“odbc”表示将访问的数据库类型是 ODBC 数据源，数据源的名称是 sunshine2。

```
03 jdbc.username=PUBLIC
04 jdbc.password=PUBLIC
```

第 3-4 行定义了访问数据库的用户名和密码。

需要添加的两张表（表 User 和表 Customer）的两个数据文件如下：

User.dat 的源代码：

```
name char(16), password char(8), email char(40), www char(50), tel char(20), address char(50), postcode char(10)
```

'Larry, Lin', 'Hekt47Gj', 'larry@yahoo.com', 'www.postgoods.com', '020-83467793',
'', ''

'James, Liu', 'Jeof69Hd', 'liu@263.net',
'www.skycat.com/~dawng/dawn.html','020-38639956', '', '510620'

'Wei, Peng', 'Jefldsfc', 'pengwei0731@163.net',
'www.postgoods.com','021-62626205','', '201103'

'Haowen, Huang', 'SkaU63Jd', 'vong@21cn.com', 'www.postgoods.com', '020-83788886',
'', '510110'

'vong', 'happy2001', 'vong@21cn.com', 'www.postgoods.com', '020-83788886', '',
'510110'

'Susan, Huang', 'Kdj77jHd', 'susanhuang@263.net',
'www.postgoods.com','010-62774302', '清华大学', '100086'

'Jiang, Long', 'D5kj3sfc', 'ceo@21b-b.com', 'www.21b-b.com','020-38659320','',
'510632'

'Hellen, Wang', 'Sk45hDdd', 'blueback@21cn.com', 'www.postgoods.com',
'020-84567765', '', '510120'

'Yixi, Bai', 'Afdfj5Hd', 'yi.xi.bai@yahoo.com',
'www.postgoods.com','021-69403224', '上海大学', ''

'James, Lu', 'Qtrg34tr', 'james.lu@yahoo.com',
'www.ceocio.com.cn','020-85239817','', '510621'

Customer.dat 的源代码：

object_id long, name char(16), address char(50), postcode char(10), country
char(10), email char(40), tel char(20), statement char(10)

'0001', 'Larry, Lin', 'West Street BOX 450,Dong Yuan Heng,Guangzhou',
'510110', 'China', 'larry@yahoo.com', '020-84543342', '购买一件'

'0002', 'James, Liu', 'NO.240 Tianhe Road,Guangzhou', '510620', 'China',
'liu@263.net', '020-38639120', '定购十件'

'0003', 'Wei, Peng', 'No.15 Shanghai Police Hospital', '210003', 'China',
'pengwei0731@163.net', '021-62626205', ''

源代码的主要内容介绍完后，下面我们就来编译和运行这个程序。

编译和运行

在 DOS 的命令行状态下编译该程序，如图 15-6 所示：

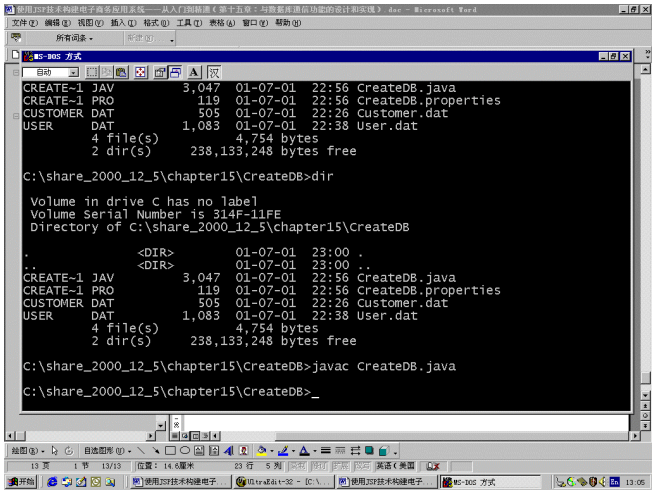

图 15-6　在 DOS 的命令行状态下编译程序 CreateDB.java

编译完成后，开始使用编译后生成的 CreateDB.class 创建新表并添加数据。首先创建新表 User 并添加数据，在 DOS 命令行状态下输入：

>java CreateDB User

如图 15-7 所示。

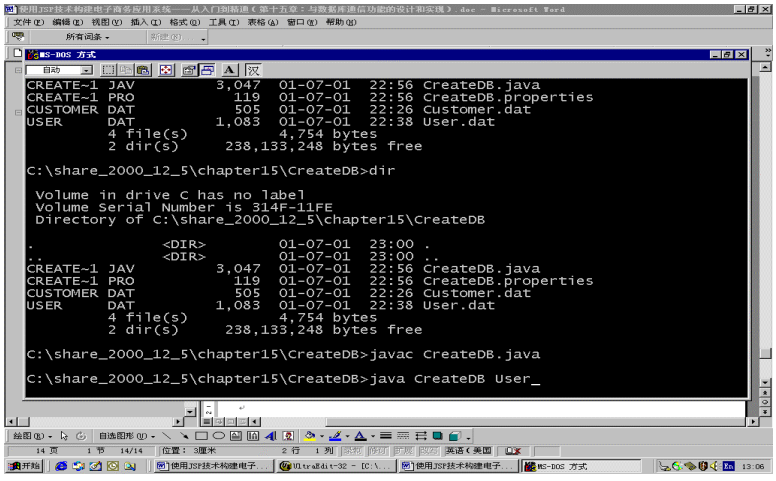

图 15-7　创建新表 User 并添加数据

如果创建和添加数据成功的话，会在数据库 sunshine2 中生成表 User（如图 15-8 所示），同时在屏幕上显示添加的全部表数据（如图 15-9 所示）。

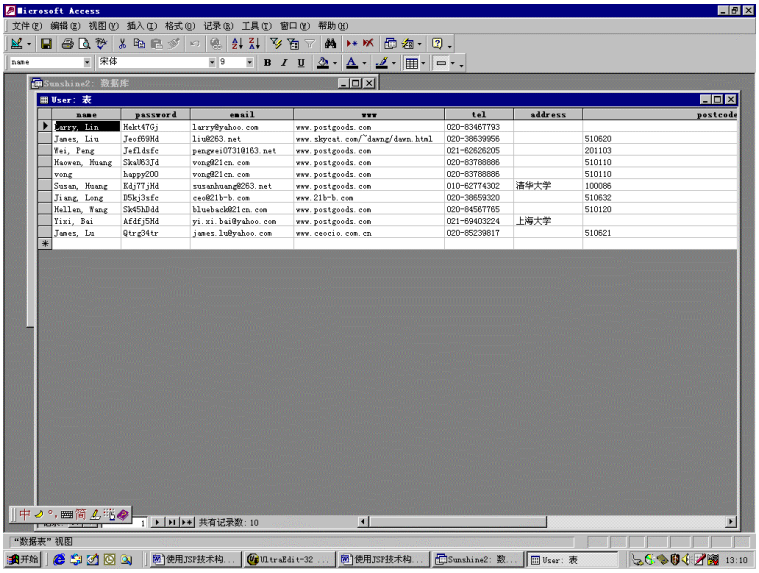

图 15-8　成功创建后在数据库 sunshine2 中生成表 User

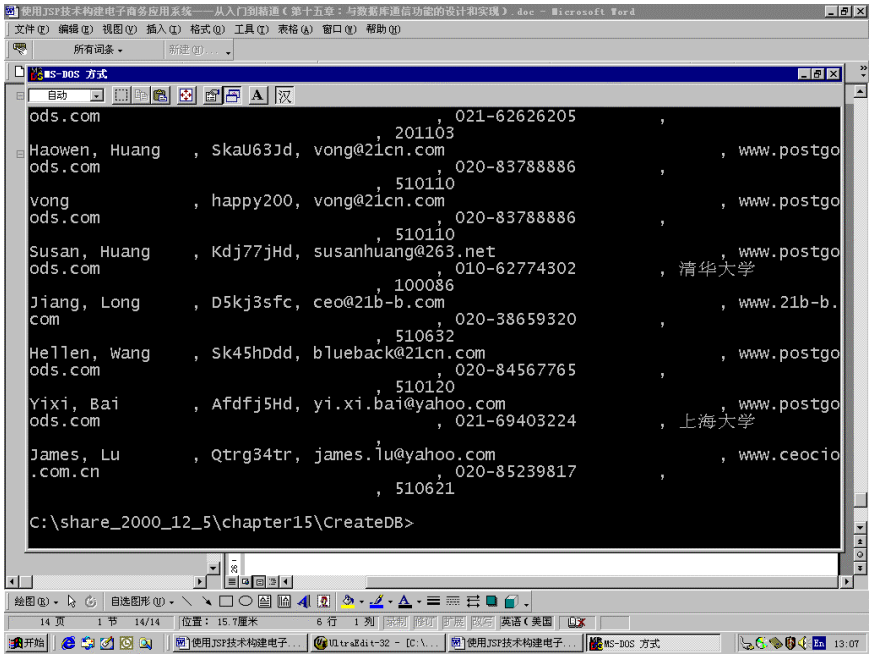

图 15-9　在屏幕上显示添加的表 User 数据

同理，我们可以在 DOS 命令行状态下输入：

>java CreateDB Customer

如图 15-10 所示。

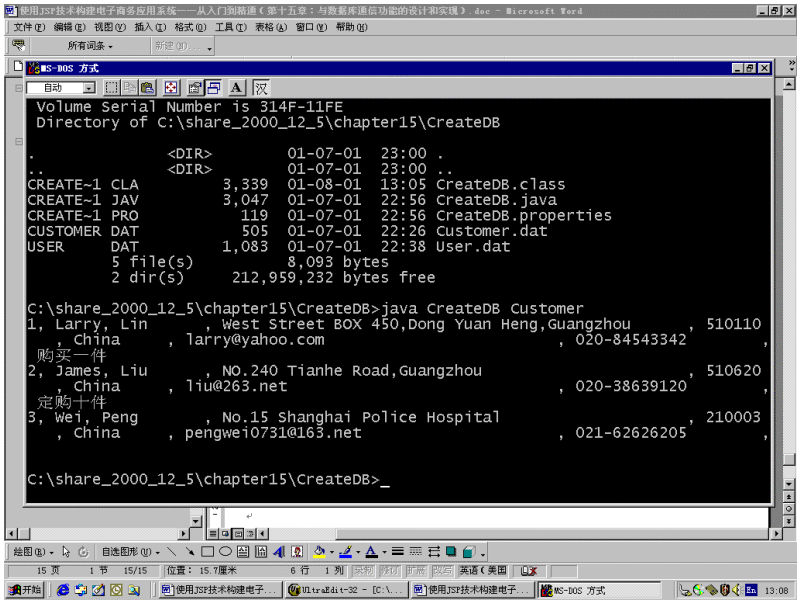

图 15-10　　创建新表 Customer 并添加数据

创建和添加数据成功的话，会在数据库 sunshine2 中生成表 Customer（如图 15-11 所示），同时在屏幕上显示添加的全部表数据（如图 15-10 所示）。

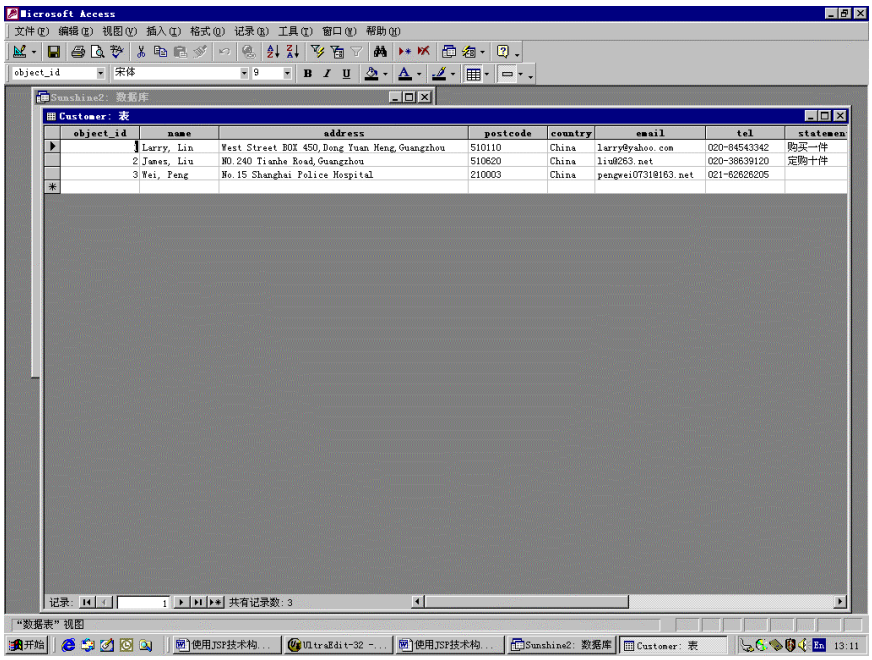

图 15-11　成功创建后在数据库 sunshine2 中生成表 Customer

15．2　与数据库通信的编程

本节中涉及所有源程序在 SUN 公司的 JDK1.3 for Windows 上调试通过（调试时间：2001 年 1 月 8 日）。

本节中我们将介绍和讨论如何通过 JDBC 编程接口设计和实现应用程序与数据库通信的功能。这种功能应该包括查询数据库中的表、根据各个不同的表查询各数据项的具体值等等。

通过该实例的学习，读者将会发现：JDBC 数据库编程接口不但可以关于数据库和其中的表的单独信息，还能够提供有关数据库和表的结构的信息。

下面我们就来分析以下的实例。

15．2．1　源代码分析和说明

本实例中包含两个文件：

WatchDB.java 和 WatchDB.properties。

WatchDB.java 是主程序，与数据库通信的所有功能由 WatchDB.java 完成。

WatchDB.properties 是一个属性文件。它定义了 JDBC 编程接口的重要配置。

WatchDB.java 程序的源代码：

```
001/**
002 * WatchDB.java
003 */
004
005import java.net.*;
006import java.sql.*;
```

第 6 行导入了包 java.sql.*，该包定义了进行 SQL 操作的相关方法。

```
007import java.awt.*;
008import java.awt.event.*;
009import java.io.*;
010import java.util.*;
011import javax.swing.*;
012
013public class WatchDB
014{  public static void main(String[] args)
015   {  JFrame frame = new WatchDBFrame();
016      frame.show();
017   }
018}
019
020class WatchDBFrame extends JFrame
021   implements ActionListener
022{  public WatchDBFrame()
023   {  setTitle("WatchDB");
024      setSize(300, 200);
025      addWindowListener(new WindowAdapter()
026         {  public void windowClosing(WindowEvent e)
027            {  System.exit(0);
028            }
```

```
029         } );
030
031       Container contentPane = getContentPane();
032
033       tableNames = new JComboBox();
034       tableNames.addActionListener(this);
035
036       dataPanel = new JPanel();
037       contentPane.add(dataPanel, "Center");
038
039       nextButton = new JButton("Next");
040       nextButton.addActionListener(this);
041       JPanel p = new JPanel();
042       p.add(nextButton);
043       contentPane.add(p, "South");
044
045       fields = new ArrayList();
046
047       try
048       {  con = getConnection();
```

第 48 行通过调用 DriverManager.getConnection()方法获得了一个 Connection 对象：con。

```
049          stmt = con.createStatement();
```

第 49 行根据在第 48 行中获得的 Connection 对象（con），创建了一个 Statement 对象：stmt。这个 Statement 对象将在下面的程序中用来执行各种 SQL 语句。

```
050          md = con.getMetaData();
```

第 50 行描述了数据库的 MetaData 数据结构。

MetaData 在 SQL 中是定义数据库和其组成部分数据结构的变量。之所以称之为 MetaData，是为了把它和数据库中的实际数据变量区分开。有两种 MetaData 数据类型：数据库的 MetaData 数据结构和 ResultSet 的 MetaData 数据结构。

为了获取数据库和其中表的数据结构等复杂信息，我们必须先从数据库的连接中获取一个 DatabaseMetaData 类型的对象；再对该对象进一步操作处理，最终得到我们需要的信息。

```
051          ResultSet mrs = md.getTables(null, null, null,
052             new String[] { "TABLE" });
053          while (mrs.next())
054             tableNames.addItem(mrs.getString(3));
055           mrs.close();
056       }
```

第 51-56 行：

第 51 行调用了类 DatabaseMetaData 中的一个方法：getTables。该方法的返回结果是

一个 ResultSet 类型的变量，其中包含了数据库中所有数据表的信息。

ResultSet 变量我们在第七章已经介绍过，每个 ResultSet 变量都可以返回一个表示 ResultSetMetaData 接口的对象，这个对象里的内容包括了各数据表中列项的总数、各列项的名和类型等的信息值。有了这些信息，就可以动态地显示查询输出的数值了。

```
057       catch(Exception e)
058       {  JOptionPane.showMessageDialog(this, e);
059       }
060
061       contentPane.add(tableNames, "North");
062    }
063
064    public static Connection getConnection()
065       throws SQLException, IOException
066    {  Properties props = new Properties();
067       String fileName = "WatchDB.properties";
068       FileInputStream in = new FileInputStream(fileName);
069       props.load(in);
```

第 66-69 行定义了 Properties 对象：props。并从文件 WatchDB.properties 中将相关的属性设置信息载入了 props 中。属性定义文件 WatchDB.properties 我们将在下面的内容中详细介绍。

```
070
071       String drivers = props.getProperty("jdbc.drivers");
072       if (drivers != null)
073          System.setProperty("jdbc.drivers", drivers);
```

第 71-73 行从属性文件 WatchDB.properties 中获得"jdbc.drivers"字段的内容。如果该"jdbc.drivers"属性不为空值，立即把该属性赋给应用程序系统。

```
074       String url = props.getProperty("jdbc.url");
```

第 74 行从属性文件 WatchDB.properties 中获得"jdbc.url"字段的内容。

```
075       String username = props.getProperty("jdbc.username");
```

第 75 行从属性文件 WatchDB.properties 中获得"jdbc.username"字段的内容。

```
076       String password = props.getProperty("jdbc.password");
```

第 76 行从属性文件 WatchDB.properties 中获得"jdbc.password "字段的内容。

```
077
078       return
079          DriverManager.getConnection(url, username, password);
```

第 79 行采用从属性文件 WatchDB.properties 中获得的 url、 username、password 等属性配置，在 DriverManager 中注册一个连接。

```
080   }
081
082   private void add(Container p, Component c,
083      GridBagConstraints gbc, int x, int y, int w, int h)
084   {  gbc.gridx = x;
085      gbc.gridy = y;
086      gbc.gridwidth = w;
087      gbc.gridheight = h;
088      p.add(c, gbc);
089   }
090
091   public void actionPerformed(ActionEvent evt)
092   {  if (evt.getSource() == nextButton)
093      {  showNextRow();
094      }
095      else if (evt.getSource() == tableNames)
096      {  remove(dataPanel);
097         dataPanel = new JPanel();
098         fields.clear();
099         dataPanel.setLayout(new GridBagLayout());
100         GridBagConstraints gbc = new GridBagConstraints();
101         gbc.weighty = 100;
102
103         try
104         {  String tableName
105               = (String)tableNames.getSelectedItem();
```

第105行从用户对数据表的选择中,将用户选到的数据表名赋给字符串变量tableName。

```
106            if (rs != null) rs.close();
107            rs = stmt.executeQuery("SELECT * FROM "
108               + tableName);
```

第107-108行根据在前面（第49行）创建的Statement对象：stmt，开始使用这个Statement对象来执行SQL语句。这里执行的是查询（SELECT）SQL语句。

```
109            ResultSetMetaData rsmd = rs.getMetaData();
110            for (int i = 1; i <= rsmd.getColumnCount(); i++)
111            {  String columnName = rsmd.getColumnLabel(i);
112               int columnWidth = rsmd.getColumnDisplaySize(i);
113               JTextField tb = new JTextField(columnWidth);
114               fields.add(tb);
```

第 109-114 行：我们可以先用 ResultSetMetaData 类型的变量连接上数据库，然后通过对 ResultSetMetaData 的 getColumnCount 操作得到列值总数(columnCount)；再搜索全部列，为每个数据项名和文本区域做一个标签(Label)，从而可以标识出不同的数据项名和文本区域。

第 113-114 行将保证各数据项在显示输出时，文本区域长度可以和该列数据项的列宽保持一致。

```
115
116                gbc.weightx = 0;
117                gbc.anchor = GridBagConstraints.EAST;
118                gbc.fill = GridBagConstraints.NONE;
119                add(dataPanel, new JLabel(columnName),
120                   gbc, 0, i - 1, 1, 1);
121
122                gbc.weightx = 100;
123                gbc.anchor = GridBagConstraints.WEST;
124                gbc.fill = GridBagConstraints.HORIZONTAL;
125                add(dataPanel, tb, gbc, 1, i - 1, 1, 1);
126             }
127          }
128          catch(Exception e)
129          {  JOptionPane.showMessageDialog(this, e);
130          }
131          getContentPane().add(dataPanel, "Center");
132          doLayout();
133          pack();
134
135          showNextRow();
136       }
137    }
138
139    public void showNextRow()
140    {  if (rs == null) return;
141       {  try
142          {  if (rs.next())
143             {  for (int i = 1; i <= fields.size(); i++)
144                {  String field = rs.getString(i);
145                   JTextField tb
146                      = (JTextField)fields.get(i - 1);
147                   tb.setText(field);
148                }
149             }
150             else
151             {  rs.close();
152                rs = null;
153             }
154          }
155          catch(Exception e)
```

```
156          {  System.out.println("Error " + e);
157          }
158       }
159    }
160
161    private JButton nextButton;
162    private JPanel dataPanel;
163    private JComboBox tableNames;
164    private ArrayList fields;
165
166    private Connection con;
167    private Statement stmt;
168    private DatabaseMetaData md;
169    private ResultSet rs;
170}
```

WatchDB.properties 的源代码：

WatchDB.properties 是上例中 JDBC 编程接口配置的属性文件。

```
01 jdbc.drivers=sun.jdbc.odbc.JdbcOdbcDriver
```

第 1 行定义了使用 JDBC/ODBC 驱动桥作为应用程序和数据库之间的 JDBC 编程接口。

```
02 jdbc.url=jdbc:odbc:sunshine2
```

当用户连接数据库的时候，必须指定数据资源和一些参数，我们称之为 URL（通用资源定位器 Uniformed Resource Location）。

第 2 行中定义了该 JDBC 的 URL。其中“odbc”表示将访问的数据库类型是 ODBC 数据源，数据源的名称是 sunshine2。

```
03 jdbc.username=PUBLIC
04 jdbc.password=PUBLIC
```

第 3-4 行定义了访问数据库的用户名和密码。

15．2．2　界面介绍

下面我们就来编译我们在 15.2.2 节中的介绍和讨论过的 WatchDB.java。如图 15-12 所示。

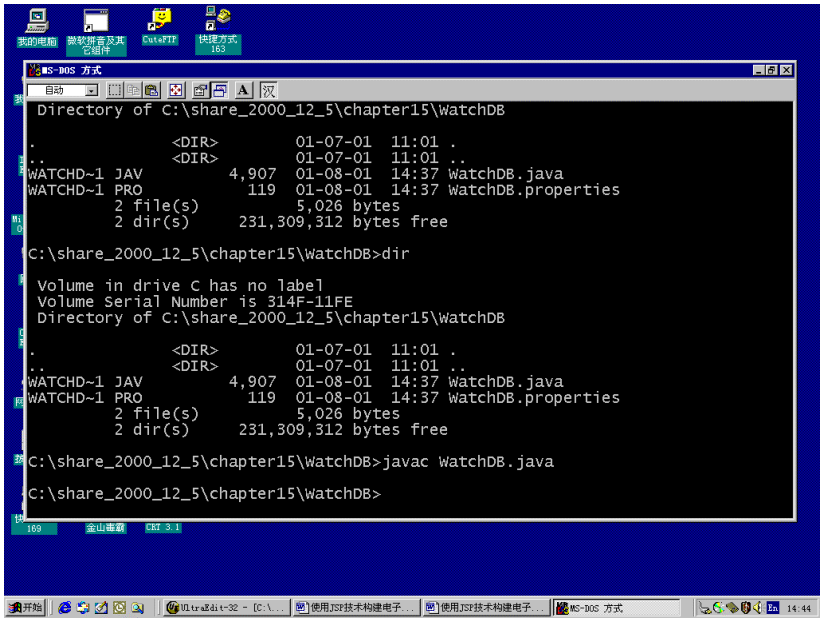

图 15-12　编译程序 WatchDB. java

细心的读者在程序编译完成之后，会发现编译中出现的一些现象和 15.1 节中的程序有所不同：

编译的结果之中出现了两个*. class 文件，分别是：WatchDB. class 和 WatchDBFrame. class。如图 15-13 所示。

这是因为我们在 WatchDB. java 中定义了类 WatchDBFrame，它从类 Jframe 中扩展而来：

```
class WatchDBFrame extends JFrame
```

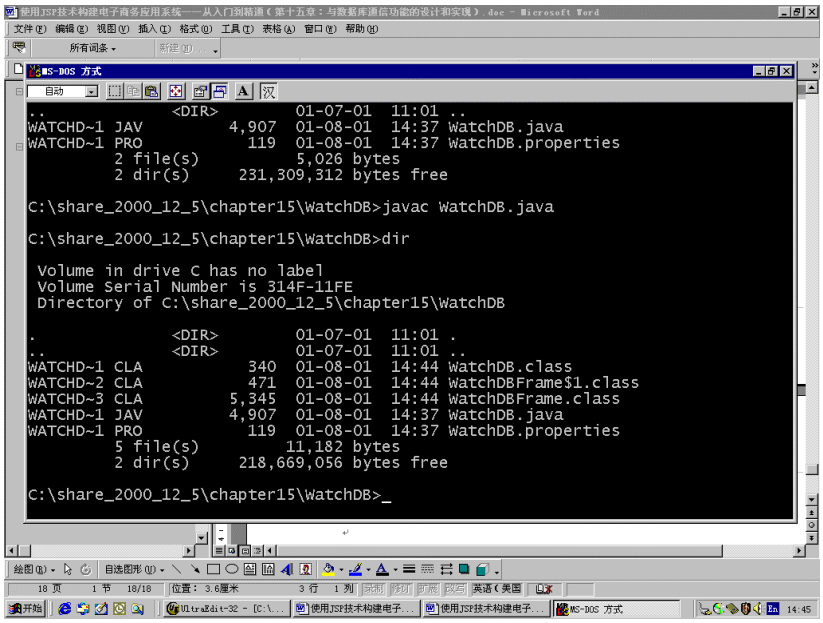

图 15-13 编译后生成类 WatchDB 和类 WatchDBFrame

编译完成后，我们可以运行该程序了。如图 15-14 所示。

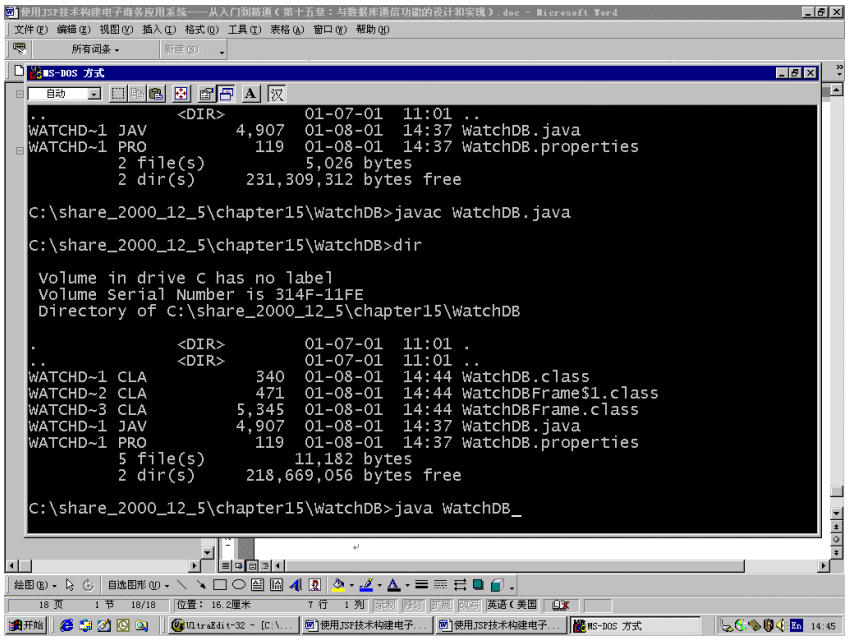

图 15-14　运行 JAVA 程序 WatchDB

程序运行后，会出现一个图形界面，如图 15-15 所示。当前的数据表名出现在界面上方的下拉菜单中。由于数据表是按照字母排序的，故一个以字母“A”开头的“Article”数据库先显示出来。可以发现，界面显示了该数据表的所有元素和一套对应的数据项的赋值。

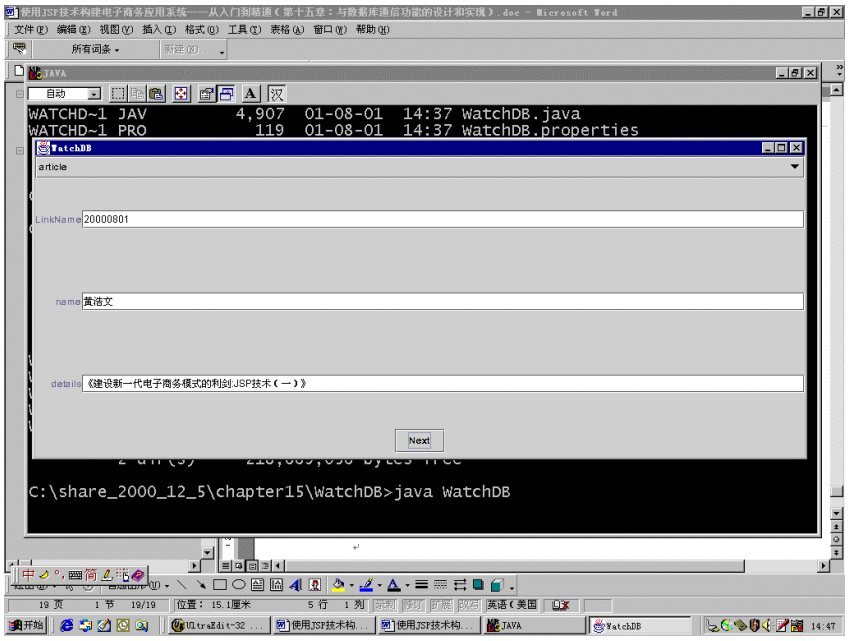

图 15-15　程序运行后出现的图形界面

在下拉菜单中选择我们在本章中定义的用户信息表：User 表。如图 15-16 所示。

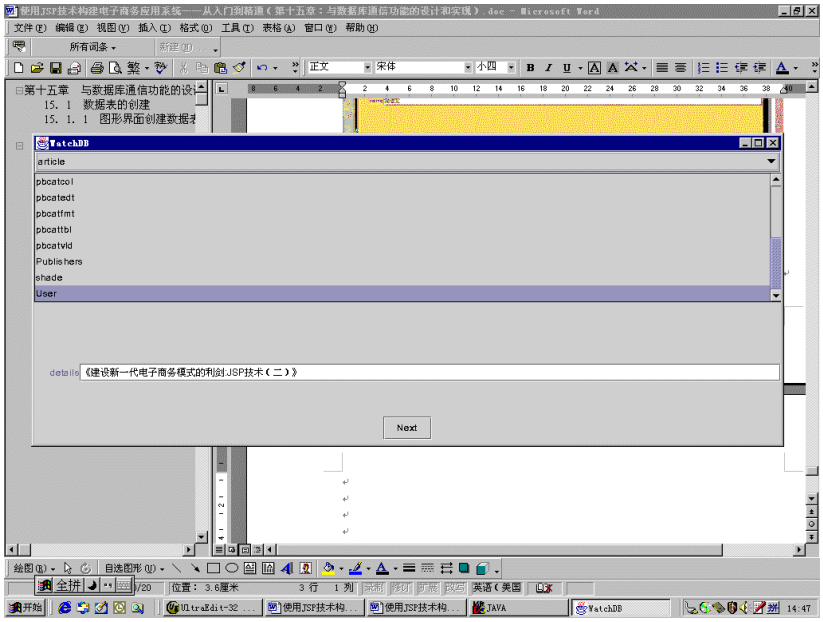

图 15-16　选择用户信息表：User 表

User 表中的用户信息就整齐地显示出来了。点击“NEXT”可以查看下一条记录。如图 15-17 所示。

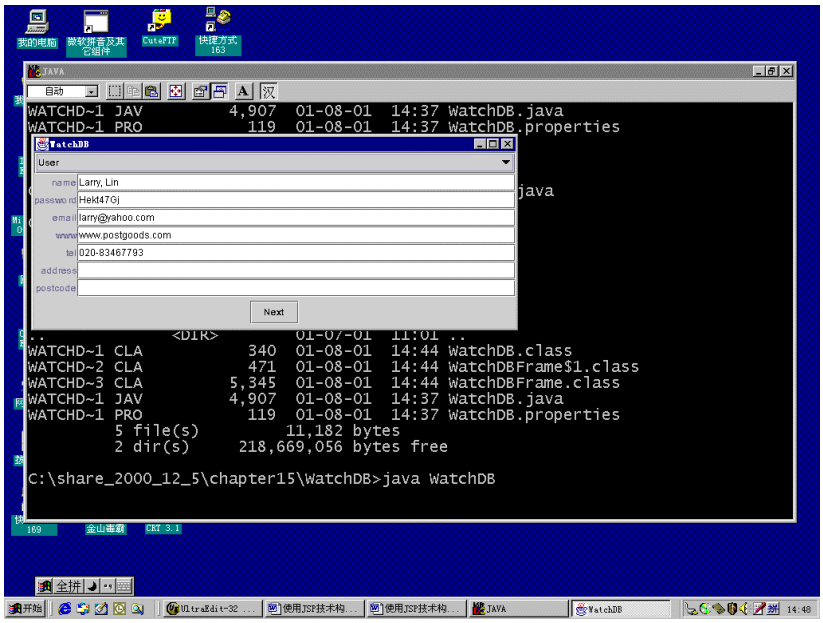

图 15-17　显示的 User 表中的用户信息

而在本章中定义的另一个数据表——Customer 表的显示情况如图 15-18 所示。

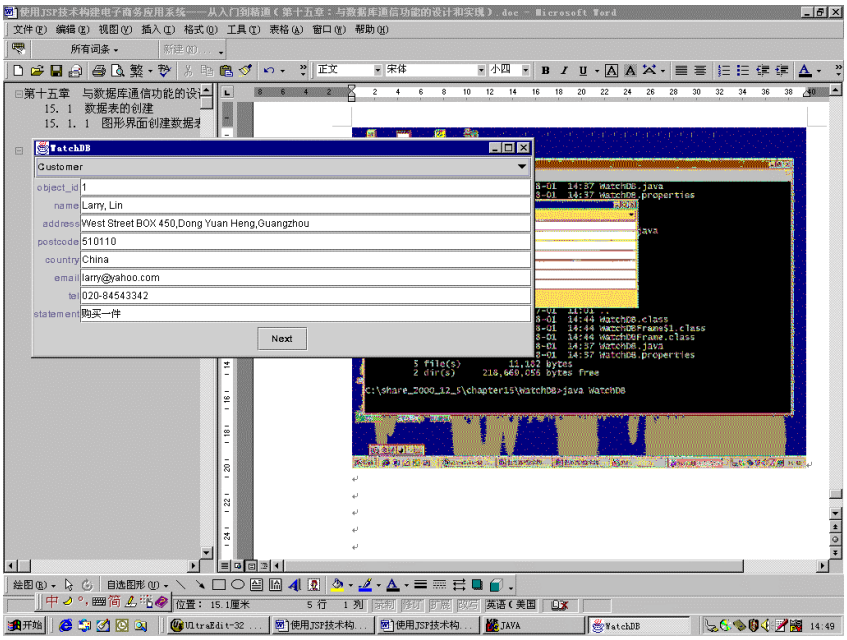

图 15-18　显示的 Customer 表中的用户信息

该实例是一个非常重要的例子，虽然程序不长，但基本覆盖了使用 JDBC 编程接口设计和实现数据库通信的主要知识点。读者只需要将本例稍加改动，就能够轻松地编写出基于电子商务系统的实用数据库和 JAVA 应用程序相互通信的软件。

本章小结：

本章学习和讨论如何通过 JDBC 编程接口，实现 JAVA 应用程序和数据库系统之间交互通信的功能。这将是我们在下一章分析和讨论的网络购物应用程序的基础。

本章首先介绍了数据库中的数据表的两种创建方法：

一是使用各种数据库的图形界面创建；

二是采用标准的 SQL 语句，使用各种数据库的应用工具或自己编写专门的程序来创建。本章以实例的形式重点介绍这方法二中的自己编写专门的程序来创建表的过程和实现。

本章还佐以实例，详细介绍和讨论如何通过 JDBC 编程接口，设计和实现应用程序与数据库通信的功能。这种功能应该包括查询数据库中的表、根据各个不同的表查询各数据项的具体值等等。通过该实例的学习，我们已经了解了 JDBC 的一个重要特性：JDBC 数据库编程接口不但可以关于数据库和其中的表的单独信息，还能够提供有关数据库和表的结构的信息（MetaData）。