在 JTable 中实现数据库的动态排序显示

Dynamic Sort Display of Database in JTable

崔尚卿　（广州中山大学计算机系　510275）

摘要：JTable 是 Java2 中与数据库的显示有着紧密联系的一个 Swing 组件。本文详细论述了在 JTable 中对数据库按任意列进行动态排序显示的方法，并给出了具体的实现类。

关键词：JTable　数据库　动态　排序　显示

在数据库应用系统中，经常要把对数据库的操作结果按某种次序显示出来，但是如果每当用户要求按特定的列进行排序显示时，就利用各种各样的排序查询进行重操作，然后显示新的结果，则无论对查询如何优化，效率都是很低的。一种较好的方法是只进行一次查询，每次排序都动态生成一个排序索引，然后按照索引进行显示。在 Java2 中，JTable 是一个与数据库的显示有着密切联系的 Swing 组件，但是 JTable 并没有提供按某一列动态排序显示数据库的功能。本文讨论了在 JTable 中对数据库进行排序显示的方法。

1　JTable 和 TableModel

JTable 是一个二维表格，基于代理模式的体系结构，是存储在 TableModel 实现中的表格数据的代理，也就是说 JTable 并不存储数据，而是通过 TableModel 获取数据。TableModel 是一个接口，声明了相关的获取和修改数据的方法，而且 JTable 代理通过调用 TableModel 中的方法建立自己的视图。图 1 显示了 JTable 和 TableModel 的代理模式关系。

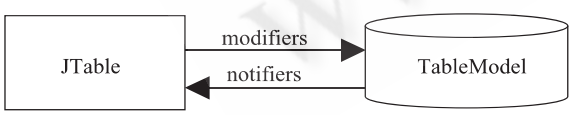

图 1

2　连接和操作数据库的方法

在 Java2 中，使用 JDBC（Java Database Connectity）对数据库进行访问。JDBC 是一个低层的、支持基本 SQL 功能的通用 Java API，其和驱动程序之间是相互独立的，这就使得在改变数据库时不需要修改访问数据库的代码，保证了代码的可移植性。JDBC API 提供了四种方式和数据库建立联系：通过 JDBC 网络驱动程序直接访问数据库；通过 JDBC－ODBC 连接桥访问 ODBC 接口间接访问数据库；通过 JDBC 驱动程序访问原始驱动程序来访问数据库；通过 JDBC 驱动程序直接访问数据库。下面以对 MySQL 数据库的连接为例，说明利用 JDBC 通过 mysql 驱动程序来访问数据库的方法。

2.1　注册数据库驱动程序和建立连接

```
try{ Class. forName ( " org. gjt. mm. mysql. Driver" ) . newInstance( ) ;
     //数据库为 test
     Connection con = java. sql. DriverManager. getConnection
        ( " jdbc: mysql://localhost/test " ," test" ," " ) ;
     ……
}catch……//处理异常
```

2.2　操作数据库

在 Java2 中，对数据库的操作都是使用 Statement 进行 SQL 语句的执行，使用 ResultSet 存储返回结果：

```
Statement stmt = con. createStatement( ) ;
DatabaseMetaData md = con. getMetaData( ) ;
ResultSet mrs = md. getTables
  ( null, null, null, new String[ ]{ " TABLE" } ) ;
String query = " select * from " + tableName;
rs = stmt. executeQuery( query) ;
```

3　动态排序显示数据库

在建立了对数据库的连接后，就可以对数据库进行操作，把结果记录显示出来。通常都是按照结果中记录的先后顺序进行显示，但是如果想要实现动态地对记录按列排序显示，就需要使用所谓的排序 TableModel，实现一个索引缓冲，把对结果某一列排序后的记录的真实索引存储起来，当 JTable 需要查找某个值时，通过显示的索引查找缓冲，取得真实的索引，最后根据真实索引返回结果。图 2 显示了 JTable、真实 TableModel 和排序 TableModel 三者的关系：

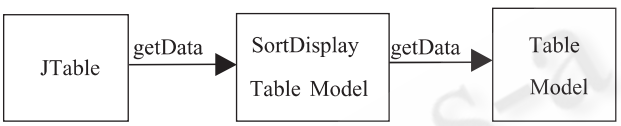

图 2

下面是实现类的代码：

```
class ResultSetTableModel extends AbstractTableModel
{ public ResultSetTableModel(ResultSet aResultSet)
  { rs = aResultSet;
    try{ rsmd = rs.getMetaData();
         cache = new ArrayList();
         int cols = getColumnCount();
         while (rs.next())//初始化 cache
         { Object[]row = new Object[cols];
           for(int j = 0;j < row.length;j++)
           row[j] = rs.getObject(j + 1);
           cache.add(row);
         }
    }catch(SQLException e)
    { System.out.println("Error " + e);}
  }
  //下面重定义和实现 AbstractTableModel 的方法
  public Object getValueAt(int r, int c)
  { if(r < cache.size())
      return ((Object[])cache.get(r))[c];
    else return null;
  }
  public String getColumnName(int c)
  { try{ return rsmd.getColumnName(c + 1);
    }catch(SQLException e)
    { System.out.println("Error " + e);
      return null;
    }
  }
  public int getColumnCount()
  { try{ return rsmd.getColumnCount();
    }catch(SQLException e)
    { System.out.println("Error " + e);
      return 0;
    }
  }
  public int getRowCount()
  { return cache.size();}
  private ArrayList cache;
  private ResultSet rs;
  private ResultSetMetaData rsmd;
}
class SortDisplayTableModel extends ResultSetTableModel
{ public SortDisplayTableModel(ResultSet aResultSet)
  { super(aResultSet);
    //初始化 rows 数组，rows 就是排序索引
    rows = new Row[getRowCount()];
    for (int i = 0; i < rows.length; i++)
    { rows[i] = new Row();
      rows[i].index = i;
    }
}
//排序函数
public void sort(int c)
{ sortColumn = c;
  Arrays.sort(rows);
  fireTableDataChanged();
}
```

```
public void addMouseListener(final JTable table)
{  table. getTableHeader ( ). addMouseListener
   (new MouseAdapter( )
   { public void mouseClicked(MouseEvent event)
     { if (event. getClickCount( ) < 2) return;
       int tableColumn
          = table. columnAtPoint(event. getPoint
       ( ));
       int modelColumn
          = table. convertColumnIndexToModel
       (tableColumn);
       sort(modelColumn); }}});
}
/*下面重定义 ResultSetTableModel 的函数,
  利用了 rows 排序索引*/
public Object getValueAt(int r, int c)
{ return super. getValueAt(rows[r]. index, c);}
private Object getValueRealAt(int r,int c)
{ return super. getValueAt(r,c);}
public boolean isCellEditable(int r, int c)
{ return super. isCellEditable(rows[r]. index,
c);}
public void setValueAt(Object aValue,int r,int c)
{ super. setValueAt(aValue, rows[r]. index,
c);}
//内部类
private class Row implements Comparable
{ public int index;
  //在这里定义排序标准
  public int compareTo(Object other)
  { Row otherRow = (Row)other; Object a =
    getValueRealAt(index, sortColumn);
    Object b = getValueRealAt(otherRow. index,
    sortColumn);
    if (a instanceof Comparable)
      return ((Comparable)a). compareTo(b);
    else return index - otherRow. index;
  }
}
private int sortColumn;
//排序缓冲
private Row[] rows;
}
```

4 结束语

通过在排序 TableModel 中实现一个索引缓冲,把真实索引进行排序存储,就可以实现 JTable 中数据库记录的动态排序显示。虽然在集成开发环境(如 JBuilder)里一般都提供了方便的组件来访问和显示数据库,但是它们都是针对一张关系表进行的操作,当需要一次显示多张表时,它们就无能为力了,而利用 JTable 却可以轻易的实现,因此本文讨论的技术是很有实用价值的。

参考文献

1 Cay S. Horstmann , Gary Cornell . Core Java2 Volume Ⅱ – Advanced Features . Sun Microsystems Press . 2000.

2 Harvey M. Deitel ,Paul J. Deitel ,Sean E. Santry. Advanced Java2 Platform How to Program. Prentice Hall . 2000.

3 侯晓强、刘艳慧、郭英丽,精通 Java2 ,科学出版社,2003。