基于 Java 与 Oracle 数据库的图像处理技术

Image operating in Java and Oracle Database

张文东　刘培刚　（山东东营石油大学(华东)计算机与通信工程学院 257061）

摘要：Java 提供了处理图像和通过 JDBC 管理数据库的功能，与 Oracle 数据库结合良好。讨论了 Java 动态访问、显示文件图像和数据库图像的原理，给出了存取文件图像和数据库图像的基本方法，解决图像存储的问题，给出了部分实例代码。

关键词：Java Oracle 数据库 JDBC 图像处理

1 引言

作为纯面向对象的程序设计语言，Java 具有无可比拟的平台无关的优势，应用范围十分广泛。Java 提供了强大的图像处理功能，并且可以通过 JDBC 管理多种数据库。Oracle 对 JDBC 的支持使得 Java 与 Oracle 数据库的联合应用非常方便，可以顺利地实现图像的处理。

2 Java 程序操作 Oracle 数据库图像数据技术

2.1 图像在 Oracle 数据库中的存储

对于大型数据对象，Oracle 数据库提供了四种存储类型，分别是：BLOB、CLOB、NCLOB 和 BFILE。BLOB 类型存储二进制数据，CLOB 类型存储单字节字符数据，NCLOB 类型存储定宽的多字节国家字符集数据，BFILE 类型以外在方式存储大型二进制文件（即数据实际存储于数据库之外）。出于数据库管理方便和数据安全的考虑，图像数据通常采取 BLOB 存储类型。

2.2 JDBC (Java DataBase Connectivity)

JDBC 是一种用来进行数据库访问的简单 API，主要使用 SQL 语句，但 JDBC 可以在数据库和 Java 应用程序之间进行平稳转换。同时，将 JDBC 调用映射到 ODBC 调用的桥梁——JDBC－ODBC 桥，使得 Java 应用程序可以访问任何支持 ODBC 的数据库管理系统。JDBC 隐藏了每个数据库的细节，只需要考虑应用程序，不必考虑底层实现。

Oracle 为 JDBC 提供了两种类型的驱动程序。一种是标准的 API 驱动程序。Oracle 厂家提供进行数据库访问的标准 C 和 C++ 方法，驱动程序中则包含了它们的 Java 代码。这种方法需要在客户端系统上安装软件。另一种驱动程序利用 Java 套接字来直接与数据库通信。这是最纯粹、最直接的 Java 方法，推荐使用。

2.3 数据库连接

JDBC 使用一个类(java.sql.DriverManager)和两个接口(java.sql.Driver 和 java.sql.Connection)来对数据库进行连接，分为三步进行：装载 JDBC 驱动器、建立 Connection 对象、建立 Statement 对象。示例如下：

```
Class.forName("oracle.jdbc.driver.OracleDriver").newInstance(); //装载驱动器
Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@ip:port:serveName","userName","passWord"); //建立连接
Statement st = conn.createStatement(); //建立 Statement 对象
```

2.4 JDBC 与 SQL/Oracle 相对应的大型数据对象类型

JDBC 提供了与 SQL BLOB 相对应的数据类型 java.sql.Blob。同检索普通数据类型一样，可以从结果集中检索 Blob 数据，获得方法为：Blob b＝rs.GetBLOB()，rs 为结果集；不同的是，如此得到的 Blob 实例只是一个空壳，不存储任何东西，但是只要所在事务是打开的，就可以使用 Blob 对象的方法在任何时候存入数据。如果 Blob 的实现超出了事务的范围，JDBC 驱动程序就可以随意地改变它的生命周期。

java.sql.Blob 实际上是种接口，说明了操作大型数据对象的几个基本方法，被 oracle.sql.BLOB 实现。Blob 对象的功能不及 BLOB 对象，因此，在 Java 与 Oracle 程序设计过程中，经常将 Blob 转换为 BLOB 后使用，方法有两种，如下：

```
oracle.sql.BLOB blob = ((OracleResultSet)rs).getBLOB(1); //转换结果集，取得 BLOB
oracle.sql.BLOB blob = (oracle.sql.BLOB) rs.getBLOB(1); //将 Blob 转换为 BLOB
```

3 Java 动态加载、显示、保存图像文件

3.1 图像文件的动态加载

3.1.1 图像文件的动态选择

Java 提供了动态文件选择器 JFileChooser 和文件过滤器 FileFilter，用 addExtension(String type)方法将图像文件扩展名添加给文件过滤器，然后用 setFileFilter(FileFilter)方法将文件过滤器添加给文件选择器，最后，在事件响应中用 showOpenDialog(Component)方法显示文件选择器，就可以在所有目录下寻找指定扩展名的图像文件，选定后可得到图像文件的路径、名称，便可以加载了。

3.1.2 图像文件的加载

加载图像文件有三种方式，第一种是用 Toolkit 的 getImage(String path)方法通过文件路径取得图像，并用 MediaTracker 的 addImage(Image,int)方法和 waitForID(int)方法监视加载完成，这种方式得到 Image 类型的图像对象；第二种方式是通过文件路径建立图像文件的文件输入流(FileInputStream)，然后通过文件输入流用 JPEGCodec.createJPEGDecoder(InputStream)方法建立 JPEG 图像解码器(JPEGImageDecoder)，再用 JPEGImageDecoder 的 decodeAsBufferedImage()方法创建 BufferedImage 图像，关闭文件输入流，这样图像文件即被加载进程序中来。第三种方式也是通过文件路径建立图像文件的文件输入流(FileInputStream)，然后以字节方式把数据从文件输入流中读出。目前第一种方法可以加载使用 jpeg/jpg、gif 等多种格式图像；第二种方式仅支持 jpeg/jpg 格式图像；第三种方式支持所有类型的文件，并非仅限于图像文件，但不能直接生成图像对象。

3.2 图像文件的显示

相对于图像文件的加载，图像文件的显示则方便许多。图像加载完成之后，在缓冲区中生成 Image/BufferedImage 图像对象。Java2 程序实际显示的是缓冲区中的图像对象。使用容器的 Graphics 属性的绘图方法 drawImage(…)可以灵活地在指定的位置按指定的大小显示指定的图像。

3.3 图像文件的动态保存

在 Java2 程序中图像以对象的方式存在，将图像对象保存为图像文件时需对其根据文件格式编码。目前 Java2 支持效率比较高的 JPEG 编码方式，将图像保存为 jpeg/jpg 文件。

3.3.1 图像文件保存目录的动态选择

用 addExtension(String type)方法将图像文件扩展名添加给文件过滤器 FileFilter，然后用 setFileFilter(FileFilter)方法将文件过滤器添加给文件选择器 JFileChooser，再后，在事件响应中用 showSaveDialog(Component)方法显示文件选择器，用户可以选择保存目录，选择文件格式，指定图像文件的名称。

3.3.2 图像文件的保存

通过选定的图像文件路径和名称、格式建立新文件(如果文件已存在将被覆盖)File，通过 File 建立文件输出流(FileOutputStream)；如果在 Java 程序中图像以对象的形式存在，那么用 JPEGCodec.createJPEGEncoder(FileOutputStream)方法建立 JPEG 图像编码器 JPEGImageEncoder jpg2，用 jpg2 的 encode(BufferedImage)方法将图像对象编码后写入图像文件；如果在 Java 程序中图像以字节数组或者字符串的形式存在，则用文件输出流的 write(byte[],0,byte.length())方法将图像字节数组(字符串转化为字节数组)直接写入文件输出流，即写入图像文件；关闭文件输出流。

4 Java 操作 Oracle 数据库中的图像数据

4.1 存取 Oracle 数据库中 BLOB 类型图像数据的方法

与普通数据类型不同，BLOB 类型数据不能通过 SQL 语句直接存取，必须采用特殊方法。Java 采用数据流的方式操作 Oracle 的 BLOB 类型数据。

假定数据库中存在一个 image 表，表结构如下：

```
image(
imagename not null varchar2(50),
imagedata BLOB
)
```

当前目录存在一个 JPEG 图像文件 TIGER.JPG。

4.1.1 存储 BLOB 类型图像数据

第一步，向数据库中插入一条包括 BLOB 占位符的记录字段的记录，此时，BLOB 字段中仅仅是一个占位符，可以是一个普通字符串，如：

```
st.executeUpdate(insert into image(imagename,imagedata)values("TIGER.JPG","000"));
```

第二步，从数据库中查询取得该 BLOB 字段，并生成 BLOB 对象，以备更新，如：

```
ResultSet rs = st.executeQuery(select imagedata from image where imagename = "TIGER.JPG" for update);
BLOB lob = ((OracleResultSet)rs).getBLOB(1);
```

第三步，取得 BLOB 对象的输出流，向对象中写入数据，如：

```
OutputStream bos = lob.getBinaryOutputStream();
JPEGImageEncoder jpg = JPEGCodec.createJPEGEncoder(bos);
```

```
jpg.encode(bimage);
```

4.1.2 读取 BLOB 类型图像数据

第一步,从数据库中查询取得 BLOB 字段,生成 BLOB 对象,如:

```
ResultSet rs = st.executeQuery(select imagedata from image where imagename="TIGER.JPG");
BLOB lob = ((OracleResultSet) rs).getBLOB(1);
```

第二步,取得 BLOB 对象的输入流,从对象中读取数据,如:

```
InputStream inStream = lob.getBinaryStream();
JPEGImageDecoder jpg = JPEGCodec.createJPEGDecoder(binStream);
BufferedImage bimage = jpg.decodeAsBufferedImage();
```

4.2 存取 Oracle 数据库中 BLOB 类型数据的注意事项

(1) BLOB 字段从查出到写入数据的过程之中是不能提交事务的,因此需要将数据库连接对象(conn)的自动提交属性设为假 (false),在写入数据后提交,并将自动提交属性设为真(true)。

(2) 结果集和数据流使用完毕后,必须关闭,以节省资源。

5 Java 与 Oracle 数据库处理图像实例

5.1 将图像文件 TIGER.JPG 另存为新文件 LION.JPG

```
FileInputStream inStream = new FileInputStream("TIGER.JPG"); //建立文件输入流,将图片数据读入流
FileOutputStream outStream = new FileOutputStream(new File("LION.JPG")); //建立新文件"LION.JPG"及其输出流
byte[]buffer = new byte[1024];
int bytesRead = 0;
  while ((bytesRead = inStream.read(buffer)) != -1)
  {
    outStream.write(buffer, 0, bytesRead);
  } //将文件输入流中的内容写入文件输出流
inStream.close(); //关闭文件输入流
outStream.close(); //关闭文件输出流
```

5.2 将图像文件 TIGER.JPG 存入表 image 中,然后从表 image 中取出该图像数据,生成一个新的 JPEG 文件

```
FileInputStream inStream = new FileInputStream("TIGER.JPG"); //建立文件输入流,将图片数据读入流
JPEGImageDecoder jpg = JPEGCodec.createJPEGDecoder(inStream); //以文件输入流为参数建立 JPEG 图像文件解码器对象
BufferedImage bimage = jpg.decodeAsBufferedImage(); //用文件解码器对象生成缓存图像
inStream.close(); //关闭输入流
g.drawImage(bimage,0,0,100,100); //绘制图像(g 是绘制容器的 Graphics 环境属性)
st.executeUpdate (insert into image (imagename,imagedata)values("TIGER.JPG", "000")); //向数据库中插入一条记录
ResultSet rs = st.executeQuery(select imagedata from image where imagename = "TIGER.JPG" for update) //查询 BLOB 数据
BLOB lob = ((OracleResultSet) rs).getBLOB(1); //取得 BLOB 字段,生成 BLOB 对象,以备更新
OutputStream bos = lob.getBinaryOutputStream(); //取得 BLOB 对象的输出流
JPEGImageEncoder jpg = JPEGCodec.createJPEGEncoder(bos); //以 BLOB 对象的输出流为参数建立 JPEG 编码器对象
jpg.encode(bimage); //用 JPEG 编码器对象编码缓存图像
bos.close(); //关闭输出流
conn.commit(); //提交,图像数据存入数据库
rs = st.executeQuery(select imagedata from image where imagename="TIGER.JPG") //查询 BLOB 数据
BLOB lob = ((OracleResultSet) rs).getBLOB(1); //取得 BLOB 字段,生成 BLOB 对象
InputStream inStream = lob.getBinaryStream(); //取得 BLOB 对象的输入流
JPEGImageDecoder jpg = JPEGCodec.createJPEGDecoder(inStream); //以 BLOB 对象的输入流为参数建立 JPEG 解码器对象
BufferedImage bimage = jpg.decodeAsBufferedImage(); //用 JPEG 解码器对象建立缓存图像
binStream.close(); //关闭输入流
g.drawImage(bimage,100,100,100,100); //绘制图像(g 是绘制容器的 Graphics 环境属性)
File file = new File("LION.JPG"); // 以"LION.JPG"为参数建立新图像文件
```

(下转第 29 页)

(上接第 36 页)

```
FileOutputStream outStream = new FileInputStream(file); //以新图像文件为参数建立文件输出流
JPEGImageEncoder jpg = JPEGCodec.createJPEGEncoder(outStream); //以文件输出流为参数建立JPEG 编码器对象
jpg.encode(bImage); //用 JPEG 编码器对象编码缓存图像
outStream.close(); //关闭输出流,图像写入文件 LION.JPG
```

6 总结

Java 应用程序的设计与开发涉及多方面的内容,与 Oracle 数据库联合处理图像仅是其中一小部分。本文讨论了 Java 在图像处理方面的加载、显示与保存的部分问题,以及 Java 与 Oracle 数据库联合存储图像的部分内容。Java 的图像处理功能远不只如此,本文讨论的是图像处理的基本问题,是进一步处理图像的基础。

参考文献

1 George Reese, JDBCTM 与 JavaTM 数据库编程(第二版),石永鑫、宋隆 等译,中国电力出版社,2002。

2 Cay S. Horstmann、Gary Cornell, 最新 Java 2 核心技术,王建华、董志敏、杨宝明 等译, 机械工业出版社, 2003。

3 涂承胜, 基于 VB 的数据库的图像处理技术,计算机工程与设计, 2003,24(6).81－85。