Oracle *inter*Media Image Quick Start

Java Client Classes (Client Proxy Classes)

Introduction

Oracle *inter*Media ("*inter*Media") is a feature that enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information. Oracle *inter*Media extends Oracle Database reliability, availability, and data management to multimedia content in traditional, Internet, electronic commerce, and media-rich applications.

This article shows how to develop a JDBC Java application that stores and uses images in an Oracle database table. Storing images directly in Oracle Database has numerous benefits such as ease of management, centralized backup, automated replication, tight technology stack integration, ease of development, and so on. The benefits of storing images in a database are explained in detail in white papers on the Oracle Technology Network: http://www.oracle.com/technology/products/intermedia/index.html

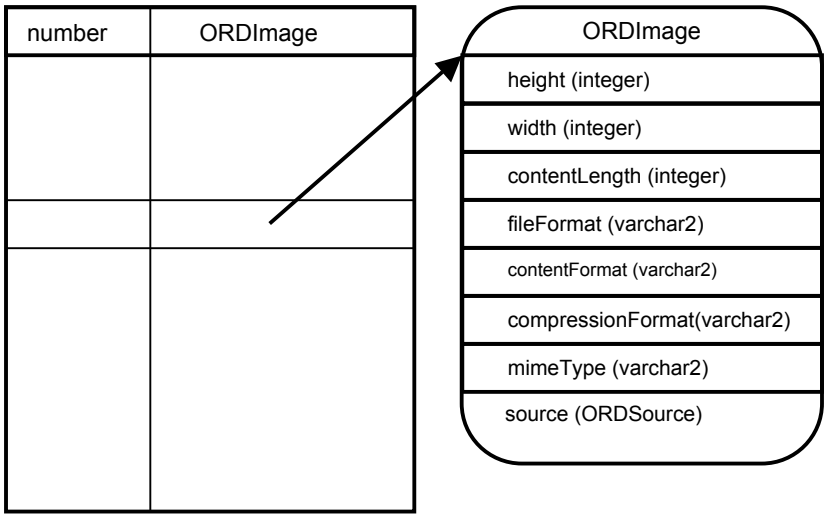
This article is a very brief introduction to the *inter*Media database objects and Java classes, and a code cookbook of how to use *inter*Media to perform common tasks – load images into the database, retrieve image properties, generate thumbnails, and download images from the database.

The examples presented below are provided to demonstrate the use of the *inter*Media Java Client in an easy to understand way. As such, readability sometimes takes precedence over performance and error handling. Places where an operation can be done differently for performance reasons are called out in the comments.

Overview of ORDSYS.ORDImage (Oracle Database) and OrdImage (Java) Objects

Java programmers are intimately familiar with Java objects, but are often unaware that Oracle Database is an object-relational database, and as such supports storage and retrieval of objects. Oracle *inter*Media provides the database type ORDImage which is used to store images in a database table just like any other relational data. Some *inter*Media functionality (such as thumbnail generation) may also be used if images are stored in BLOB (Binary Large OBject) columns, but Oracle Corporation recommends storing images in ORDImage columns.

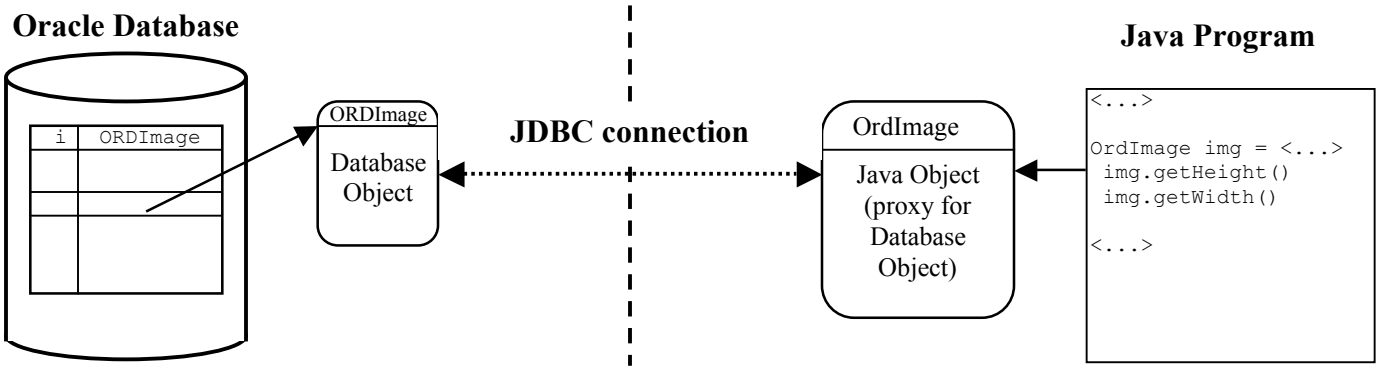
An example of an ORDImage object in a database table is illustrated in the following diagram.



Even though the JDBC specification does not support object-relational databases directly, Oracle *inter*Media **database** objects can be used in JDBC programs by means of the *inter*Media Java Client.

The *inter*Media Java Client contains high performance proxy Java objects that allow for quick object property retrieval and convenient upload/download. The proxies forward any requests for computation back to the database server for the ORDImage object to execute. These client objects are in the `oracle.ord.media.im` package (found in the `$ORACLE_HOME/ord/jlib/ordim.jar` file).

A schematic diagram of how a database ORDSYS.ORDImage object is related to the Java OrdImage object is shown below. It can't be stressed enough that OrdImage Java objects are merely proxies for database objects – they must be created from a database ORDImage object.



Setting Up the Required Java Environment—Imports and CLASSPATH

Section 1: Imports

To use the OrdImage class in your Java programs, the following import statement must be present.

```
import oracle.ord.im.OrdImage;
```

Note that the examples in this article also make use of several standard JDBC classes in the `java.sql` package and the Oracle JDBC extension classes `OracleResultSet` and `OraclePreparedStatement` that are included using the following import statements.

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import oracle.jdbc.OracleResultSet;
import oracle.jdbc.OraclePreparedStatement;
```

Section 2: CLASSPATH

To connect to the database and use OrdImage objects, the following jar files must be in your CLASSPATH.

1. The Oracle JDBC drivers
    a. $ORACLE_HOME/jdbc/lib/ojdbc14.jar (preferred)
    b. $ORACLE_HOME/jdbc/lib/classes12.jar (deprecated)
2. The SQLJ runtime
    a. $ORACLE_HOME/sqlj/lib/runtime12.jar
3. The Oracle *inter*Media Java Client library
    a. $ORACLE_HOME/ord/jlib/ordim.jar

Setting the CLASSPATH on the command line

- Windows (assumes %ORACLE_HOME% has been set)

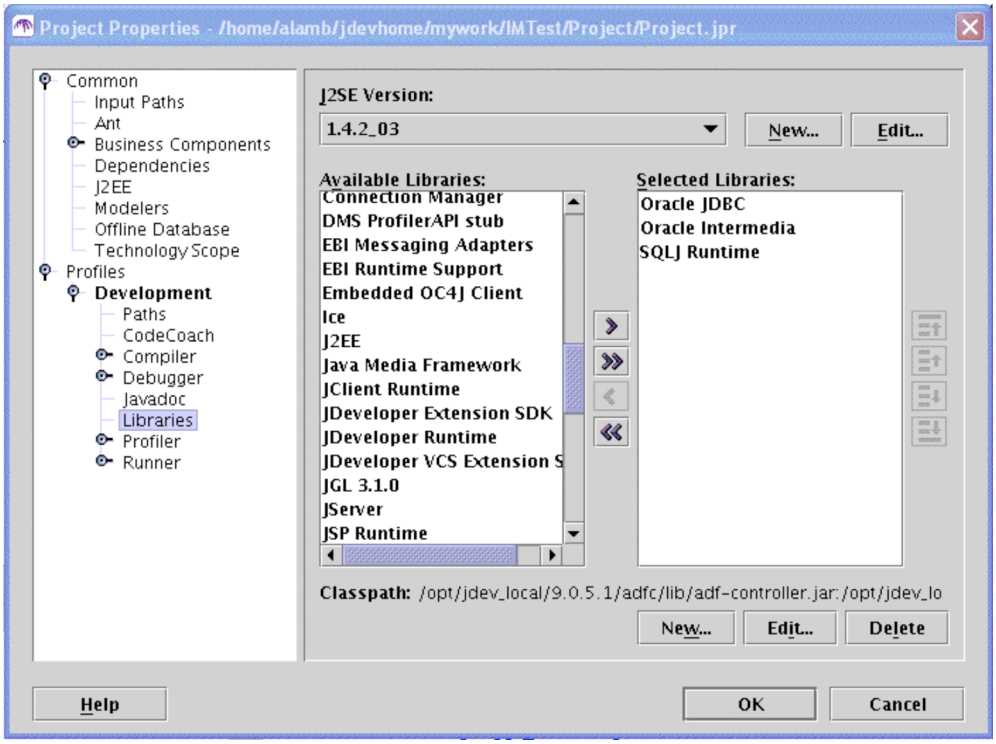
```
set CLASSPATH=%ORACLE_HOME%\jdbc\lib\ojdbc14.jar;%ORACLE_HOME%\ord\jlib\ordim.jar;%ORACLE_HOME%\sqlj\lib\runtime12.jar;.
```

- UNIX (assumes that $ORACLE_HOME has been set)

```
setenv CLASSPATH ${ORACLE_HOME}/jdbc/lib/ojdbc14.jar:$ORACLE_HOME/ord/jlib/ordim.jar:$ORACLE_HOME/sqlj/lib/runtime12.jar:.
```

Setting the CLASSPATH using JDeveloper 10*g* (9.0.5)

1. Go to the project properties (right click on your project and choose properties).
2. Navigate to the Profiles/Development/Libraries option and choose the three libraries shown below.
    a. Oracle JDBC
    b. Oracle Intermedia
    c. SQLJ Runtime



Documentation

The documentation (in javadoc format) for the *inter*Media Java classes can be found from the main list of Oracle Documentation Books (under the heading of "*inter*Media Java Classes Reference"). The Oracle documentation is included with the Database install, and it can be found on OTN at http://www.oracle.com/pls/db10g/db10g.homepage. (Click on the "Books" tab.)

JDBC/SQLJ Version Issues—getORAData versus getCustomDatum

You can skip this section unless you see the following compiler errors when compiling the code

```
setORAData(int,oracle.sql.ORAData) in oracle.jdbc.OraclePreparedStatement cannot be applied to
(int,oracle.ord.im.OrdImage)
cannot resolve symbol symbol  : method getORADataFactory  () location: class oracle.ord.im.OrdImage
method getORADataFactory not found in class oracle.ord.im.OrdImage
method setORAData(int, oracle.ord.im.OrdImage) not found in interface OraclePreparedStatement
```

There are two ways to get and set OrdImage Java proxy objects using the Oracle JDBC driver. With the Oracle *inter*Media 9*i* Java Client, the **deprecated** `OracleResultSet.getCustomDatum()` and `OraclePreparedStatement.setCustomDatum()` methods must be used ("getCustomDatum" syntax). Using the Oracle *inter*Media 10*g* Java Client, the `OracleResultSet.getORAData()` and `OraclePreparedStatement.setORAData()` methods may be used ("getORAData" syntax) instead.

In this article, we present the examples using the getORAData syntax, but we show how to use the getCustomDatum syntax below. The following table shows which syntax to use based on which software distribution that the *inter*Media Java Client is from:

Location of Oracle *inter*Media Java Client Libraries	**Syntax to use**
Oracle Database 10*g*	getORAData
Oracle Client 10g *	getORAData
JDeveloper 10*g* (9.0.5 and earlier)	getCustomDatum
All 9*i* clients	getCustomDatum

* The *inter*Media Java Client is only installed as part of the Oracle Client's "Administrator" install option. For a lightweight *inter*Media Java Client install, you can install the JDBC and SQLJ product components using the "Custom Install" option and download only the *inter*Media libraries from OTN at http://www.oracle.com/technology/products/intermedia/.

If you must use the getCustomDatum syntax, the example code in the rest of this article needs to be modified as follows (ignore the deprecated API compilation warnings).

To retrieve an imageProxy object from an OracleResultSet object, instead of the getORAData syntax:

```
OrdImage imageProxy = (OrdImage)rset.getORAData("image", OrdImage.getORADataFactory());
```

you should instead use the getCustomDatum syntax:

```
OrdImage imageProxy = (OrdImage)rset.getCustomDatum("image", OrdImage.getFactory());
```

Likewise, when updating a table with an OrdImage Java proxy object, instead of the getORAData syntax:

```
opstmt.setORAData(1, imageProxy);
```

you should instead use the getCustomDatum syntax:

```
opstmt.setCustomDatum(1, imageProxy);
```

Note that the supporting files included with this Quick Start guide contain two copies of the example code—one that uses the getCustomDatum syntax and one that uses the getORAData syntax.

Creating the JDBC Connection

Oracle *inter*Media can be used with all Oracle JDBC drivers (`thin` or `oci`) and the Connection object is created in the normal way, except that the autoCommit flag **must** be set to false. For example:

```
// register the oracle jdbc driver with the JDBC driver manager
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection conn = DriverManager.getConnection(connectString, username, password);

// Note: it is CRITICAL to set the autocommit to false so that
// two-phase select-commit of BLOBs can occur.
conn.setAutoCommit(false);

// create a JDBC Statement object to execute SQL in the database
Statement stmt = conn.createStatement();
```

As shown above, the Connection that is established **must** have the autoCommit flag set to false because *inter*Media uses BLOB columns internally to store data. Since BLOB updates in Oracle Database require a two-stage select-commit process, if the autoCommit flag is set to true (the default) then BLOB operations will fail with the exception:

```
java.sql.SQLException: ORA-01002: fetch out of sequence
```

Creating a Table With an ORDImage Column to Store Images

The first order of business is to create a simple table with two columns: a numeric identifier (`id`) and an ORDImage object (`image`). The table is created just like any other database table using the Java code shown below (`stmt` is the Statement that was created in the previous step). We also show a schematic of the table that is created.

```
String tableCreateSQL = "create table image_table " +
                        "(id number primary key, " +
                        "image ordsys.ordimage)";
stmt.execute(tableCreateSQL);
```

id	image

NOTE: All *inter*Media objects and procedures are defined in the ORDSYS schema. In Oracle 9*i* and earlier, you must use the fully qualified names (prefixed by `ORDSYS.`), but in Oracle 10*g* and later the prefix is not needed. We will use the fully qualified syntax in this document so that the examples work with *inter*Media version 9*i* and above.

Uploading Images from Files into Tables

This section shows how to upload images that are stored in disk files into the `image_table` table we made in the previous section.

1. Insert a new row into the table with `id` set to 1 and `image` initialized to a new ORDImage object.

```
// insert a row into image_table
String rowInsertSQL = ("insert into image_table (id, image) values (1,ordsys.ordimage.init())");
stmt.execute(rowInsertSQL);
```

2. Get a proxy for the ORDImage database object in row 1 in the OrdImage Java proxy object `imageProxy` (**NOTE** that since we will be uploading data into the ORDImage's underlying BLOB column, the row must be selected with the `FOR UPDATE` clause).

```
// select the new ORDImage into a java proxy OrdImage object (imageProxy)
String rowSelectSQL = "select image from image_table where id = 1 for update";
OracleResultSet rset = (OracleResultSet)stmt.executeQuery(rowSelectSQL);
rset.next();
OrdImage imageProxy = (OrdImage)rset.getORAData("image", OrdImage.getORADataFactory());
rset.close();
```

3. Load the image data from the goats.gif file into the ORDImage object (and by extension into the database) by calling the `loadDataFromFile` method on the Java proxy object.

```
imageProxy.loadDataFromFile("goats.gif");
```

4. Automatically detect the image's height, width, file format, and so on by calling `setProperties()` on the proxy object. Calling `setProperties()` on the proxy object forwards the request to the database to execute `ORDImage.setProperties()` on the server.

```
imageProxy.setProperties();
```

5. Update `image_table` to reflect the changes we have made to the ORDImage object (uploaded data and filled in properties).

```
String updateSQL = "update image_table set image=? where id=1";
OraclePreparedStatement opstmt = (OraclePreparedStatement)conn.prepareStatement(updateSQL);
opstmt.setORAData(1, imageProxy);
opstmt.execute();
opstmt.close();
```

NOTE: If you call `ORDimage.setProperties()` on an image that is not one of *inter*Media's supported formats (for example JPEG2000) a `java.sql.SQLException` that encapsulates an `IMG-00705` error such as the following is thrown:

```
java.sql.SQLException: ORA-29400: data cartridge error
IMG-00705: unsupported or corrupted input format
```

Retrieving Image Properties

Once images are in Oracle Database, you can access image metadata using either standard SQL queries or the Java proxy accessor methods. In the following examples, we demonstrate how to use the Java proxy accessor methods to access the properties of the goats.gif file that we uploaded in the previous section. Note that the

properties that may be selected are: `height`, `width`, `fileFormat` (JPEG, GIFF, and so on), `contentFormat` (monochrome, and so on), `contentLength` (number of bytes of image data), and `mimeType`.

One may access an image's height and width by calling the accessors `getHeight()` and `getWidth()` on the *inter*Media Java proxy objects. To do this, first the image is selected into a proxy object (`imageProxy`), and then the `getHeight()` and `getWidth()` methods are called.

```
String rowSelectSQL = "select image from image_table where id = 1";
OracleResultSet rset = (OracleResultSet)stmt.executeQuery(rowSelectSQL);
rset.next();
OrdImage imageProxy = (OrdImage)rset.getORAData("image", OrdImage.getORADataFactory());
rset.close();
int height = imageProxy.getHeight();
int width  = imageProxy.getWidth();
```

The above code results in `height = 375` and `width = 500` when using the example goats.gif file.

Creating Thumbnails and Changing Formats

We now illustrate how to create an ORDImage object that contains a thumbnail of an existing ORDImage object using the `processCopy()` method. To use the `processCopy()` method, the programmer describes the desired properties of the output image and provides the input image. For example, the following description generates a JPEG thumbnail image of size 75x100 pixels: "`fileformat=jfif fixedscale=75 100`".

Some image file extensions and the corresponding *inter*Media `fileformat` are as follows.

Extension	`fileformat`
.jpg	JFIF (9*i*, 10*g*), JPEG (10*g*)
.gif	GIFF(9*i*, 10*g*), GIF (10*g*)
.tif, .tiff	TIFF
.png	PNGF

The following example shows how to insert a new ORDImage object into a second row of `image_table`, and then shows how to generate a JPEG thumbnail of the goats.gif image in the new row with the "`maxscale=100 100 fileformat=jfif`" `processCopy` command.

```
// One could significantly reduce the number of round trip
// database communications in the following example.
String rowInsertSQL = ("insert into image_table (id, image) " +
                       "values (2, ordsys.ordimage.init())");
stmt.execute(rowInsertSQL);

// get the source ORDImage object
String srcSelectSQL = "select image from image_table where id=1";
OracleResultSet rset = (OracleResultSet)stmt.executeQuery(srcSelectSQL);
rset.next();
OrdImage srcImageProxy = (OrdImage)rset.getORAData("image", OrdImage.getORADataFactory());
rset.close();

// get the newly inserted destination ORDImage object
String dstSelectSQL = "select image from image_table where id=2 for update";
rset = (OracleResultSet)stmt.executeQuery(dstSelectSQL);
rset.next();
OrdImage dstImageProxy = (OrdImage)rset.getORAData("image", OrdImage.getORADataFactory());
rset.close();

// call the processCopy method (processing occurs on the SERVER)
```

```
srcImageProxy.processCopy("maxscale=100 100 fileformat=jfif", dstImageProxy);

// update the destination image in the second row
String dstUpdateSQL = "update image_table set image=? where id=2";
OraclePreparedStatement opstmt =
    (OraclePreparedStatement)conn.prepareStatement(dstUpdateSQL);
opstmt.setORAData(1, dstImageProxy);
opstmt.execute();
opstmt.close();
```

NOTE: The following error might be returned from `ORDImage.processCopy()` if the External Procedure Agent (extproc) is not configured correctly on the server.

```
java.sql.SQLException: ORA-29400: data cartridge error
IMG-00703: unable to read image data
ORA-28575: unable to open RPC connection to external procedure agent
```

In Oracle Database release 9*i*, JPEG (and some other less common formats) encoding and decoding requires extproc to run. To fix the preceding error, the Oracle Listener needs to be configured correctly to allow the use of extproc. Please see technical note 198099.1, *Configuration of the External Procedure Call for interMedia* at http://metalink.oracle.com for detailed instructions on setting up extproc for use with *inter*Media. Note that Oracle Database 10*g* does not require extproc for JPEG encoding and decoding.

If you do not want to or cannot change your Oracle Net configuration, try changing the file format to pngf as follows.

```
srcImageProxy.processCopy("maxscale=100 100 fileformat=pngf", dstImageProxy);
```

Downloading Image Data from Tables into Files

An ORDImage database object can be downloaded into a local disk file with the following steps.

1. Select the ORDImage object from the database into an OrdImage Java proxy.
2. Call the `getDataInFile()` method on the OrdImage Java proxy to download the image data into a file.

An example of these two steps to download the image in row 2 to “row2.jpg” is shown below.

```
// export the data in row 2
String exportSelectSQL = "select image from image_table where id = 2";

OracleResultSet rset = (OracleResultSet)stmt.executeQuery(exportSelectSQL);

// get the proxy for the image in row 2
rset.next();
OrdImage imageProxy = (OrdImage)rset.getORAData("image", OrdImage.getORADataFactory());
rset.close();

// call the getDataInFile method to write the ORDImage in row 2 to disk
imageProxy.getDataInFile("row2.jpg");
```

Cleaning Up

To restore your database to its original state, simply drop the `image_table` table as follows.

```
// drop the images table
stmt.executeQuery("drop table image_table");

// commit all our changes
conn.commit();
```

Conclusion

Using the *inter*Media Java proxy objects to work with ORDImage objects in database tables, we have shown how to import images into the database, retrieve image metadata (width, height, and so on), perform basic image processing, and export images to the file system. Hopefully, starting from these examples you can easily build and deploy your own Java solutions that take advantage of the unrivaled imaging capabilities that Oracle Database has to offer.

Oracle *inter*Media provides much more functionality than is covered in this Quick Start. Refer to the following documentation for more information: *Oracle interMedia User's Guide and Reference, Release 9.0.1*, *Oracle interMedia Reference, 10g Release 1 (10.1)*, and *Oracle interMedia User's Guide, 10g Release 1 (10.1)*. Additional examples and articles are available on the *inter*Media web page on the Oracle Technology Network at http://www.oracle.com/technology/products/intermedia/index.html.