Oracle8™ Visual Information Retrieval Cartridge

User's Guide

Release 1.0.1

November 1997

Part No. A55255-02

Visual Information Retrieval technology licensed
from Virage, Inc.

ORACLE®

Enabling the Information Age™

Oracle8 Visual Information Retrieval Cartridge User's Guide

Part No. A55255-02

Release 1.0.1

Primary Author: Jeff Hebert

Contributors: Cathy Trezza, Chuck Murray, Susan Shepard

Contents

Send Us Your Comments vii

Preface ix

1 Introduction

1.1 If You Already Understand Oracle8 Image Cartridge 1-2
1.2 Data Cartridges 1-3
1.3 Image Concepts 1-4
1.3.1 Digital Images 1-4
1.3.2 Image Components 1-4
1.3.3 Interchange Formats 1-5
1.4 Object Relational Technology 1-5
1.4.1 Storing Images 1-5
1.4.2 Querying Images 1-6
1.4.3 Accessing Images 1-6
1.5 Cartridge Methods and Operations 1-7
1.5.1 Image Property Extraction 1-7
1.5.2 Image Analysis and Comparison 1-7
1.5.3 Image Modification 1-7
1.5.4 Image Copy 1-7

2 Content-Based Retrieval Concepts

2.1 Overview and Benefits 2-1
2.2 How Content-Based Retrieval Works 2-2

2.2.1 Global Color and Local Color ... 2-3
2.2.2 Texture and Structure ... 2-5
2.2.3 Face Recognition ... 2-6
2.3 How Matching Works ... 2-6
2.3.2 Score ... 2-7
2.3.3 Similarity Calculation ... 2-8
2.3.4 Threshold Value ... 2-10
2.3.5 Example: Medical X-Ray Screening ... 2-10
2.4 Preparing or Selecting Images for Useful Matching ... 2-12

3 Visual Information Retrieval Examples

3.1 Create a New Table Containing an Image ... 3-1
3.2 Add an Image Column to an Existing Table ... 3-2
3.3 Load Images into BLOBs From External Files ... 3-2
3.4 Retrieve an Image ... 3-4
3.5 Retrieve Images Similar to a Comparison Image ... 3-4
3.6 Convert an Image to a Different Format ... 3-6
3.7 Extend the Cartridge With a New Type ... 3-7
3.8 Use Image Types With Object Views ... 3-8

4 Visual Information Retrieval Reference

4.1 Object Data Types (ODTs) ... 4-2
ORDVirB Object Data Type ... 4-3
ORDVirF Object Data Type ... 4-4
4.2 Methods ... 4-5
CopyContent() Method ... 4-6
Process( ) Method ... 4-7
ProcessCopy( ) Method ... 4-9
SetProperties( ) Method ... 4-10
4.3 Operators ... 4-12
Analyze( ) Operator ... 4-13
Convert( ) Operator ... 4-15
Score( ) Operator ... 4-17

Similar( ) Operator 4-20

A File and Compression Formats

B Sample Program

C Frequently Asked Questions

C.1 Errors During Installation C-1
C.2 Errors Using the Image Types C-2
C.3 RPC Errors After Installation C-2
C.4 SQL Select Statement Does Not Work C-5
C.5 Extract the File Name from a BFILE Image C-5

D Error Messages

Send Us Your Comments

Oracle8 Visual Information Retrieval Cartridge User's Guide

Part No. A55255-02

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available).

You can send comments to us in the following ways:

- e-mail: nedc_doc@us.oracle.com
- FAX - 603.897.3269. Attn: Visual Information Retrieval Documentation
- postal service:
  Oracle Corporation
  Visual Information Retrieval Cartridge Documentation
  One Oracle Drive
  Nashua, NH 03062
  USA

If you would like a reply, please include your name, and a postal or e-mail address.

Preface

This guide describes how to use Oracle8 Visual Information Retrieval Cartridge. This cartridge, based on technology licensed from Virage, Inc., extends the capabilities of Oracle8 Image Cartridge by allowing content-based retrieval of images.

Visual Information Retrieval Cartridge requires Oracle8 Enterprise Edition. Oracle8 and Oracle8 Enterprise Edition have the same basic features. However, several advanced features, such as data cartridges, are available only with the Enterprise Edition, and some of these features are optional. For example, to extend Visual Information Retrieval Cartridge, you must have the Enterprise Edition and the Objects Option.

For information about the differences between Oracle8 and the Oracle8 Enterprise Edition and the features and options that are available to you, see *Getting to Know Oracle8*.

Intended Audience

This guide is intended for anyone who is interested in storing, retrieving, and manipulating image data in an Oracle database, including developers of image specialization cartridges.

Structure

This guide contains the following chapters and appendixes:

Chapter 1	Introduces image information retrieval and data cartridges; explains image-related concepts.
Chapter 2	Explains concepts, operations, and techniques related to content-based retrieval.

Chapter 3	Provides basic examples of using Visual Information Retrieval Cartridge types and methods.
Chapter 4	Provides reference information on Visual Information Retrieval object data types, procedures, and operators.
Appendix A	Describes the supported image data formats.
Appendix B	Describes how to run the sample application and includes a source listing of that program.
Appendix C	Emphasizes several entries from the online FAQ.
Appendix D	Lists potential errors, their causes, and user actions to correct them.
Glossary	Defines important terms related to data cartridges and image information retrieval.

Related Documents

Note: For information added after the release of this guide, refer to the online README.TXT file in your ORACLE_HOME directory. Depending on your operating system, this file may be in:

`ORACLE_HOME/ord/vir/admin/README.TXT`

Please see your operating-system specific installation guide for more information.

For more information about using this data cartridge in a development environment, see the following documents in the Release 8.0 Oracle Server documentation set:

- *Oracle Call Interface Programmer's Guide*
- *Oracle8 Application Developers Guide*
- *Oracle8 Concepts*
- *PL/SQL User's Guide and Reference*

For information about the Oracle8 Image Cartridge, see the *Oracle8 Image Cartridge User's Guide*.

Visual Information Retrieval Cartridge is based on technology licensed from Virage, Inc. Visit the Virage web site for additional information about this ongoing relationship: http://www.virage.com/oracle.

Conventions

In this guide, Oracle8 Visual Information Retrieval Cartridge is sometimes referred to as Visual Information Retrieval Cartridge. Oracle Corporation's Network Computing Architecture is referred to as "the architecture."

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this guide:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
italic text	Italic text is used for emphasis and for book titles.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1
Introduction

Oracle8 Visual Information Retrieval Cartridge is an extension to Oracle8 Enterprise Edition that provides image storage, content-based retrieval, and format conversion capabilities through an object data type (ODT). The capabilities of this cartridge encompass the storage, retrieval, and manipulation of image data managed by the Oracle8 Enterprise Edition universal data server. This cartridge supports image storage using binary large objects (BLOBs) and references to image data residing in external files (BFILEs).

Visual Information Retrieval Cartridge is a building block for various imaging applications rather than being an end-user application in itself. It consists of ODTs along with related methods for managing and processing image data. Some example applications for this cartridge are:

- digital art galleries or museums
- real estate marketing
- document imaging
- stock photograph collections (for example, for fashion designers or architects)

These applications have certain distinct requirements and some degree of commonality. The image ODTs accommodate the commonality and support extensions that address application-specific requirements. With Visual Information Retrieval Cartridge, images can be managed as easily as standard attribute data.

Visual Information Retrieval Cartridge supports static, two-dimensional images in Oracle databases. The images may be bitonal (black and white) document images, grayscale photographs, or three-color photographic images. The cartridge provides the means to add image columns / objects to existing tables, insert and retrieve images, and convert between popular application formats. This enables database designers to extend existing application databases with images or to build new end-

user image database applications. Cartridge developers can use the basic functions provided here to build specialized image data cartridges.

1.1 If You Already Understand Oracle8 Image Cartridge

If you are already familiar with Oracle8 Image Cartridge, you can skim much of the conceptual information in this chapter. Image Cartridge and Visual Information Retrieval Cartridge both let you store an image as an object in the database or as a reference to an external file. Both cartridges let you store and query on the following attributes:

- image height
- image width
- image size
- file type or format (such as TIFF)
- compression type or format (such as JPEG)
- image type (such as monochrome)
- annotation or description

The main difference between the two cartridges is that Visual Information Retrieval Cartridge also lets you perform **content-based retrieval**, that is, to perform queries based on intrinsic visual attributes of the image (color, structure, texture) rather than being limited to keyword searches in textual annotations or descriptions. The underlying technology was developed by Virage, Inc., a leader in content-based retrieval.

For an example of a query using content-based retrieval, consider a database containing images of many automobiles. If you want to retrieve information on all the red automobiles, you would specify an image of a red automobile for comparison and request all records where the image looks like your picture. To increase the accuracy of the query (because all the images are of automobiles and you are interested only in red ones), you specify that the greatest relative weight is to be given to the global color attribute, with no weight given to the structure and texture attributes.

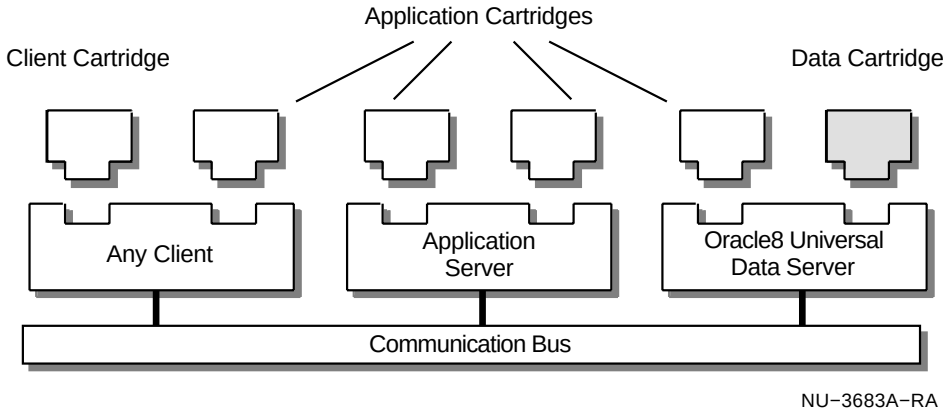
For further information on content-based retrieval, including how and why to specify relative weights (importance) for different visual attributes, see Chapter 2.

1.2 Data Cartridges

Within the Oracle Network Computing Architecture (NCA), data cartridges facilitate the storage and retrieval of complex data types required by nontraditional database applications, such as geographic information systems, imaging, workflow, document management, and digital libraries. These applications are built using components or modules that support the capture, input, processing, analysis, storage, retrieval, and display of the complex data types. Figure 1–1 shows the relationship between data cartridges and NCA.

A **data cartridge** is the mechanism by which clients, application-specific servers, and database servers can be easily and reliably extended. Visual Information Retrieval Cartridge plugs into Oracle8 Enterprise Edition and provides support for image domain-specific types, methods, and interfaces. The cartridge focuses on a set of image data representation and access mechanisms sufficient to support many image applications, and the development of more specialized image cartridges.

Figure 1–1 Cartridges and the Network Computing Architecture



The primary focus of Visual Information Retrieval Cartridge is to provide storage, retrieval, reformatting, compression, decompression, and a minimal set of data manipulation services, including scaling and cropping.

It is possible to extend this data cartridge by creating a new object type or a new composite object type based on the provided Visual Information Retrieval types. Extending a data cartridge requires the Objects Option of Oracle8 Enterprise Edition. See the example in Section 3.7 for more information.

1.3 Image Concepts

This section contains conceptual material about digital images. Chapter 2 contains conceptual information about content-based retrieval and using Visual Information Retrieval Cartridge to build image applications or specialized image cartridges.

1.3.1 Digital Images

Visual Information Retrieval Cartridge supports two-dimensional, static, digital images stored as binary representations of real-world objects or scenes. Images may be produced by a document or photograph scanner, a video source such as a camera or video tape recorder connected to a video digitizer or frame grabber, other specialized image capture devices, or even by program algorithms. Capture devices take an analog or continuous signal, such as the light that falls onto the film in a camera, and convert it into digital values on a two-dimensional grid of data points known as pixels.

Visual Information Retrieval Cartridge provides the mechanism to integrate the storage and retrieval of images in Oracle databases using the Oracle8 Enterprise Edition universal data server.

1.3.2 Image Components

A digital image can be thought of as consisting of the image data (digitized bits) and attributes that describe the characteristics of the image. Image applications sometimes associate application-specific information, such as the name of the person whose image a photograph represents, with an image by storing descriptive text in an attribute or column in the database table.

The minimal attributes carried along with an image may include such things as its size (height in scan lines and width in pixels), the resolution at which it was sampled, and the number of bits per pixel in each of the colors that were sampled. The data attributes describe the image as it was produced by the capture device.

The image data (pixels) can have varying depths (bits per pixel) depending upon how the image was captured, and they can be organized in various ways. The organization of the image data, known as the data format, is crucial to access and represent the image.

The size of digital images (number of bytes) tends to be large compared to traditional computer objects such as numbers and text. Therefore, many compression schemes are in use which squeeze an image into fewer bytes, thus putting a smaller load on storage devices and networks. *Lossless* compression schemes squeeze an image in such a fashion that when it is decompressed, the resulting image is bit-for-

bit identical with the original. *Lossy* compression schemes do not result in a bit-wise identical image when decompressed, but one in which the changes may be imperceptible to the human eye, or at worst, tolerable.

Visual Information Retrieval Cartridge also provides a signature attribute that permits content-based retrieval. The **signature** is a vector containing detailed information about the visual attributes of the image. The signature is created when the image is processed by Visual Information Retrieval Cartridge, and is used in all content-based queries. For more information about the signature attribute, see Section 2.2.

1.3.3 Interchange Formats

Image interchange format describes a well-defined organization and use of image attributes, data, and often compression scheme, allowing different applications to create, interchange, and use images. Interchange formats are often stored in or as disk files, but may also be exchanged in a sequential fashion over a network and be referred to as a protocol. There are many application subdomains within the digital imaging world and many applications that create or use digital images within these. To assist application developers, Visual Information Retrieval Cartridge supports many popular interchange formats (see Appendix A, “File and Compression Formats”.)

1.4 Object Relational Technology

The Oracle8 Enterprise Edition universal data server is an *object relational* database management system. That means that in addition to its traditional role in the safe and efficient management of relational data, it now provides support for the definition of object types including the data involved in an object and the operations that can be performed on it (methods). This powerful mechanism, well established in the object-oriented world, includes integral support for binary large objects (BLOBs) to provide the basis for adding complex objects, such as digital images, to Oracle databases.

See the following for extensive information on using BLOBs and BFILEs:

- *Oracle8 Application Developers Guide, Chapter 6: Large Objects (LOBs)*
- *Oracle8 Concepts, Chapter 12: Object Views*

1.4.1 Storing Images

Visual Information Retrieval Cartridge can store digital images within the Oracle database under transactional control through the BLOB mechanism. It can also

externally reference digital images stored in flat files through the BFILE mechanism. Although this latter mechanism is particularly convenient for integrating pre-existing sets of flat-file images with an Oracle database, these images will not be under transactional control. BLOB stored images have an object relational type known as **ORDVirB**, while BFILE stored images have an object relational type known as **ORDVirF**.

1.4.2 Querying Images

Once stored within an Oracle database, an image can be queried and retrieved by finding a row in a table that contains the image using the various alphanumeric columns (attributes) of the table. For example, select a photograph from the Employee table where the employee name is Jane Doe. An example of content-based retrieval might be as follows: using the photograph of Jane Doe for comparison, find all other photographs similar to Jane Doe's in the Employee table or some other table. (Moreover, to refine the meaning of "similar" in this case, you can experiment with different weight values for the visual attributes of global color, local color, texture, and structure.)

The collection of digital images in the database can be related to some set of attributes or keywords that describe the associated content. The image content can be described with text and numbers such as dates and identification numbers. For Oracle8, image attributes can now reside in the same table as the image ODT. Alternatively, the application designer could define a composite ODT that contains one of the cartridge ODTs along with other attributes.

1.4.3 Accessing Images

Applications access and manipulate images using SQL or PL/SQL through the object relational image types ORDVirB and ORDVirF. The object syntax for accessing attributes within a complex object, such as an image, is the dot notation:

```
variable.data_attribute
```

The syntax for invoking methods of a complex object, such as an image, is also the dot notation:

```
variable.function(parameter1, parameter2, ...)
```

See the *Oracle8 Server Concepts* manual for information on this and other SQL syntax.

1.5 Cartridge Methods and Operations

Visual Information Retrieval Cartridge provides several functions for performing format conversion, compression, and data manipulation operations on image data. It also provides the ability to extract image properties and to compare images based on their content. This section presents a conceptual overview of the Visual Information Retrieval Cartridge methods and operations.

1.5.1 Image Property Extraction

The SetProperties( ) method is used to extract important properties from image data, including the height and width in pixels, total size in bytes, file format (such as TIFF and BMP), compression format (such as JPEG and LZW), content format (such as monochrome and 8-bit grayscale), and signature.

The size of the image data may be machine dependent. Importing and exporting images may require a recalculation to determine the current properties.

1.5.2 Image Analysis and Comparison

The Analyze( ) operator examines an image and creates a signature based on the global and local colors, texture, and structure of the image content. Two additional operators, Score( ) and Similar( ), compare the signatures of two images to determine if the images match based on a set of user-supplied criteria.

1.5.3 Image Modification

The Process( ) and ProcessCopy( ) methods are used for image format conversion, compression, and basic manipulation functions including scaling and cutting. The image may be compressed using an algorithm from the set of supported image formats and compression schemes. For example, image data in the TIFF format may be compressed using Packbits, Huffman, JPEG, LZW, or one of the other supported schemes. Appendix A, “File and Compression Formats” lists the supported image formats and related compression schemes.

For the initial release of this cartridge, the output of any image manipulation function must be directed to a BLOB. In-place modification is supported for BLOBs, not for BFILEs.

1.5.4 Image Copy

The CopyContent( ) method makes a copy of an image into an empty BLOB selected for update. You can copy from a BLOB or BFILE, but can only write to a BLOB.

2

Content-Based Retrieval Concepts

This chapter explains, at a high level, why and how to use content-based retrieval. It covers the following topics:

- overview and benefits of content-based retrieval
- how content-based retrieval works, including definitions and explanations of the visual attributes (global color, local color, texture, structure) and why you might emphasize specific attributes in certain situations
- image matching using a specified comparison image, including comparing how the weights of visual attributes determine the degree of similarity between images
- image preparation or selection to maximize the usefulness of comparisons

2.1 Overview and Benefits

Inexpensive image-capture and storage technologies have allowed massive collections of digital images to be created. However, as a database grows, the difficulty of finding relevant images increases. Two general approaches to this problem have been developed, both of which use metadata for image retrieval:

- using information manually entered or included in the table design, such as titles, descriptive keywords from a limited vocabulary, and predetermined classification schemes
- using automated image feature extraction and object recognition to classify image content -- that is, using capabilities unique to content-based retrieval

With Visual Information Retrieval Cartridge, you can combine both approaches in designing a table to accommodate images: use traditional text columns to describe the semantic significance of the image (for example, that the pictured automobile won a particular award, or that its engine has six or eight cylinders), and use the

Visual Information Retrieval types for the image, to permit content-based queries based on intrinsic attributes of the image (for example, how closely its color and shape match a picture of a specific automobile).

As an alternative to defining image-related attributes in columns separate from the image, a database designer could create a specialized composite data type that combines Visual Information Retrieval Cartridge and the appropriate text, numeric, and date attributes.

The primary benefit of using content-based retrieval is reduced time and effort required to obtain image-based information. With frequent adding and updating of images in massive databases, it is often not practical to require manual entry of all attributes that might be needed for queries, and content-based retrieval provides increased flexibility and practical value.

Examples of database applications where content-based retrieval is useful -- where the query is semantically of the form, “find objects that look like this one” -- include:

- medical imaging
- trademarks and copyrights
- art galleries and museums
- retailing
- fashion and fabric design
- interior design or decorating
- law enforcement and criminal investigation

For example, a web-based interface to a retail clothing catalog might allow users to search by traditional categories (such as style or a price range) and also by image properties (such as color or texture). Thus, a user might ask for formal shirts in a particular price range that are off-white with pin stripes. Similarly, fashion designers could use a database with images of fabric swatches, designs, concept sketches, and actual garments to facilitate their creative processes.

2.2 How Content-Based Retrieval Works

A content-based retrieval system processes the information contained in image data and creates an abstraction of its content in terms of visual attributes. Any query operations deal solely with this abstraction rather than with the image itself. Thus, every image inserted into the database is analyzed, and a compact representation of its content is stored in a feature vector, or **signature**.

The signature contains information about the following visual attributes:

- **Global color** represents the distribution of colors within the entire image. This distribution includes the amounts of each color, but not the locations of colors.
- **Local color** represents color distributions *and where they occur* in an image, such as the fact that an RGB vector for sky blue occurs in the upper half of an image.
- **Texture** represents the low-level patterns and textures within the image, such as graininess or smoothness. Unlike structure, texture is very sensitive to features that appear with great frequency in the image.
- **Structure** represents the shapes that appear in the image, as determined by shape-characterization techniques such as edge detection.
- **Facial** represents unique characteristics of human faces. For example, characteristics include the size and shape of the nose, the distance between the eyes, and various other attributes that cannot easily be disguised. Facial signatures are generated using separately purchasable software from Viisage Technology, Inc.

Feature data for all these visual attributes is stored in the signature, whose size typically ranges from 1000 to 2000 bytes.

Images in the database can be retrieved by matching them with a comparison image. The comparison image can be any image: inside or outside the current database, a sketch, an algorithmically generated image, and so forth.

The matching process requires that signatures be generated for the comparison image and each image to be compared with it. Images are seldom identical, and therefore matching is based on a similarity-measuring function for the visual attributes and a set of weights for each attribute. The **score** is the relative distance between two images being compared. The score for each attribute is used to determine the degree of similarity when images are compared, with a smaller distance reflecting a closer match, as explained in Section 2.3.3.

2.2.1 Global Color and Local Color

Global color reflects the distribution of colors within the entire image, whereas local color reflects color distributions *and where they occur* in an image. To illustrate the difference between global color and local color, consider Figure 2–1.

Figure 2–1 Image Comparison: Global Color and Local Color

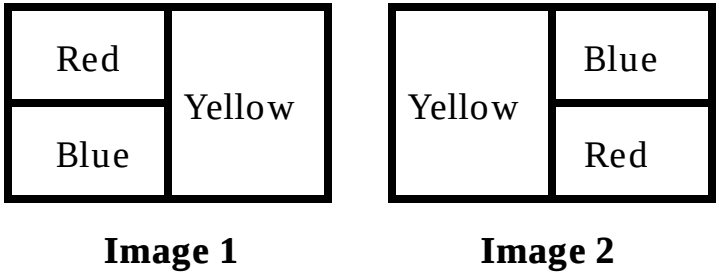


Image 1 and Image 2 are the same size and are filled with solid colors. In Image 1, the top left quarter (25%) is red, the bottom left quarter (25%) is blue, and the right half (50%) is yellow. In Image 2, the top right quarter (25%) is blue, the bottom right quarter (25%) is red, and the left half (50%) is yellow.

If the two images are compared first solely on global color and then solely on local color, the following are the similarity results:

- global color: complete similarity (score = 0.0), because each color (red, blue, yellow) occupies the same percentage of the total image in each one
- local color: no similarity (score = 100), because there is no overlap in the placement of any of the colors between the two images

Thus, if you need to select images based on the dominant color or colors (for example, to find apartments with blue interiors), give greater relative weight to global color. If you need to find images with common colors in common locations (for example, red dominant in the upper portion to find sunsets), give greater relative weight to local color.

Figure 2–2 shows two images very close (score = 0.0) in global color. Figure 2–3 shows two images very close (score = 0.02461) in local color.

Figure 2–2 Images Very Similar in Global Color

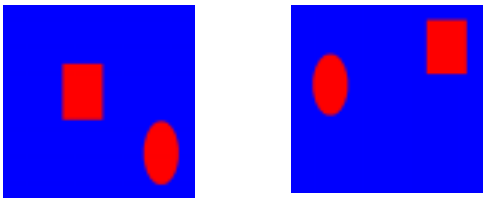


Figure 2–3 Images Very Similar in Local Color

2.2.2 Texture and Structure

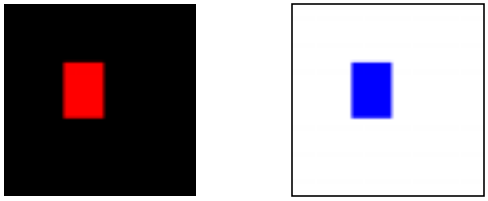
Texture is most useful for full images of textures, such as catalogs of wood grains, marble, sand, or stones. These images are generally hard to categorize using key-words alone because our vocabulary for textures is limited. Texture can be used effectively alone (without color) for pure textures, but also with a little bit of global color for some kinds of textures, like wood or fabrics. Figure 2–4 shows two similar fabric samples (score = 4.1).

Figure 2–4 Fabric Images with Similar Texture

Structure is not strictly confined to certain sizes or positions. However, when objects are of the same size or position, they have a lower score (greater similarity) than objects of different sizes. Structure is useful to capture objects such as horizon

lines in landscapes, rectangular structures in buildings, and organic structures like trees. Structure is very useful for querying on simple shapes (like circles, polygons, or diagonal lines) especially when the query image is drawn by hand and color is not considered important when the drawing is made. Figure 2–5 shows two images very close (score = 0.61939) in structure.

Figure 2–5 Images with Very Similar Structure



2.2.3 Face Recognition

Visual Information Retrieval Cartridge supports face recognition software developed by Viisage Technology, Inc. This third-party software analyzes images of faces and generates a *facial signature* based on various unique biometric characteristics.

After you have generated facial signatures with the Viisage software, you can use Visual Information Retrieval Cartridge Convert( ), Score( ), and Similar( ) operators to compare the images.

See the Viisage product documentation for more details.

2.3 How Matching Works

When you match images, you assign an importance measure, or weight, to each of the visual attributes, and the cartridge calculates a similarity measure for each visual attribute.

2.3.1 Weight

Each *weight* value can range from 0.0 (no importance) to 1.0 (highest importance). Each weight value reflects how sensitive the matching process should be to the degree of similarity or dissimilarity between two images. For example, if you want global color to be completely ignored in matching, assign a weight of 0.0 to global color; in this case, any similarity or difference between the color of the two images is totally irrelevant in matching. On the other hand, if global color is extremely important, assign it a weight near or equal to 1.0; this will cause any similarity or

dissimilarity between the two images with respect to global color to contribute greatly to whether or not the two images match.

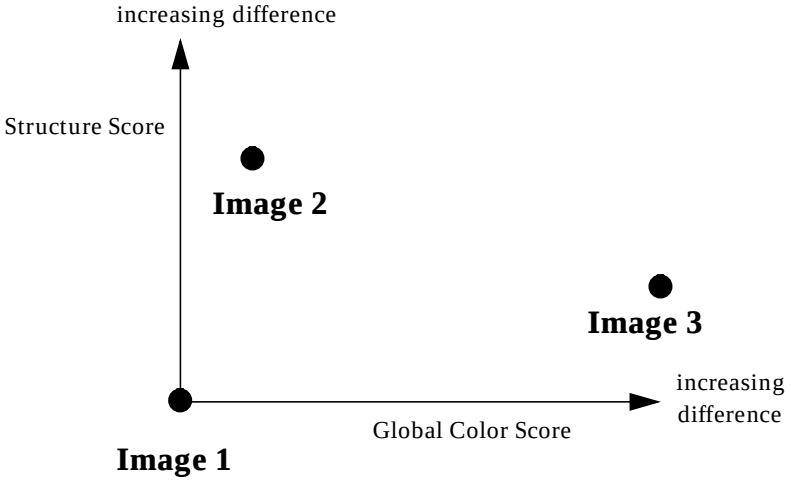
You should give at least one visual attribute a weight significantly greater than 0.0, otherwise you may get too many matches. As an extreme example, if you assign a weight of 0.0 to all four visual attributes, then all image comparisons will result in matches. This will occur because the weighted sum of all attributes will be zero and thus less than or equal to any threshold value; see Section 2.3.3 for details of the calculation. Such a result, of course, defeats the purpose of content-based retrieval.

2.3.2 Score

The similarity measure for each visual attribute is calculated as the **score** or distance between the two images with respect to that attribute. The score can range from 0 (no difference) to 100 (maximum possible difference). Thus, the more similar two images are with respect to a visual attribute, the *smaller* the score will be for that attribute.

As an example of how distance is determined, assume that the dots in Figure 2–6 represent scores for three images with respect to two visual attributes, such as global color and structure, plotted along the X and Y axes of a graph.

Figure 2–6 Score and Distance Relationship



For matching, assume Image 1 is the comparison image, and Image 2 and Image 3 are each being compared with Image 1. With respect to the global color attribute

plotted on the X axis, the distance between Image 1 and Image 2 is relatively small (for example, 15), whereas the distance between Image 1 and Image 3 is much greater (for example, 75). If the global color attribute is given more weight, then the fact that the two distance values differ by a great deal will probably be very important in determining whether or not Image 2 and Image 3 match Image 1. However, if global color is minimized and the structure attribute is emphasized instead, then Image 3 will match Image 1 better than Image 2 matches Image 1.

2.3.3 Similarity Calculation

In Section 2.3.2, Figure 2–6 showed a graph of only two of the attributes that Visual Information Retrieval Cartridge can consider. In reality, when images are matched, the degree of similarity depends on a weighted sum reflecting the weight and distance of all four of the visual attributes of the comparison image and the test image.

For example, assume that for the comparison image (Image 1) and one of the images being tested for matching (Image 2), Table 2–1 lists the relative distances between the two images for each attribute. Note that you would never see these individual numbers unless you computed four separate scores, each time highlighting one attribute and setting the others to zero.

Table 2–1 Distances for Visual Attributes Between Image1 and Image2

Visual Attribute	Distance
Global color	15
Local color	90
Texture	5
Structure	50

In this example, the two images are most similar with respect to texture (distance = 5) and most different with respect to local color (distance = 90).

Assume that for the matching process, the following weights have been assigned to each visual attribute:

- global color =0.1
- local color = 0.6
- texture = 0.2
- structure = 0.1

The weights are typically supplied in the range of 0.0 to 1.0. Within this range, a weight of 1 indicates the strongest emphasis, and a weight of 0 means the attribute should be ignored. You can use a different range (such as 0 to 100), but be careful not to accidently combine different ranges. The values you supply are automatically be normalized such that the weights total 100 percent, still maintaining the ratios you have supplied. In this example, the weights were specified such that normalization was not necessary.

The following formula is used to calculate the weighted sum of the distances, which is used to determine the degree of similarity between two images:

```
weighted_sum = global_color_weight * global_color_distance +
               local_color_weight  * local_color_distance +
               texture_weight      * texture_distance +
               structure_weight    * structure_distance
```

The degree of similarity between two images in this case is computed as:

0.15*gc_distance + 0.55*lc_distance + 0.2*tex_distance + 0.1*struc_distance

That is:

(0.1*15 + 0.6*90 + 0.2*5 + 0.1*50) = (1.5 + 54.0 + 1.0 + 5.0) = 61.5

To illustrate the effect of different weights in this case, assume that the weights for global color and local color were reversed. In this case, the degree of similarity between two images is computed as:

0.6*gc_distance + 0.1*lc_distance +0.2*tex_distance + 0.1*struc_distance

That is:

(0.6*15 + 0.1*90 + 0.2*5 + 0.1*50) = (9.0 +9.0 + 1.0 + 5.0) = 24.0

In this second case, the images are considered to be more similar than in the first case, because the overall score (24.0) is smaller than in the first case (61.5). Whether or not the two images are considered matching depends on the threshold value (explained in Section 2.3.4): if the weighted sum is less than or equal to the threshold, the images match; if the weighted sum is greater than the threshold, the images do not match.

In these two cases, the *correct* weight assignments depend on what you are looking for in the images. If local color is extremely important, then the first set of weights is a better choice than the second, because the first set of weights grants greater significance to the disparity between these two specific images with respect to local color (weighted sum of 24 versus 61.5). Thus, with the first set of weights, these two images are less likely to match -- and a key goal of content-based retrieval is to

eliminate uninteresting images so that you can focus on images containing what you are looking for.

2.3.4 Threshold Value

When you match images, you assign a *threshold* value. If the weighted sum of the distances for the visual attributes is less than or equal to the threshold, the images match; if the weighted sum is greater than the threshold, the images do not match.

Using the examples in Section 2.3.3, if you assign a threshold of 60, the images do not match when the weighted sum is 81.5, but they do match when the weighted sum is 32. If the threshold is 30, the images do not match in either case; and if the threshold is 81.5 or greater, the images match in both cases.

The following example shows a cursor (getphotos) that selects the photo_id, annotation, and photograph from the Pictures table where the threshold value is 20 for comparing photographs with a comparison image:

```
CURSOR getphotos IS
 SELECT photo_id, annotation, photo FROM Pictures WHERE
   ORDSYS.VIR.Similar(photo.ImgSignature, comparison_sig, 'globalcolor="1.0",
   localcolor="0.7", texture="0.1", structure="0.9"', 20)=1;
```

Before the cursor executes, the Analyze( ) operator must be used to compute the signature of the comparison image (comparison_sig), and to compute signatures for each image in the table. Chapter 4 describes all the operators, including Analyze( ) and Similar( ).

You will probably want to experiment with different weights for the visual attributes and different threshold values, to see which combinations retrieve the kinds and approximate number of matches you want.

2.3.5 Example: Medical X-Ray Screening

In cancer screenings, image comparisons can help to detect suspicious areas of x-ray images. X-rays of patients being screened are compared with one or more x-rays that show cancerous cells. For each comparison, the levels of magnification are the same, and the images are comparable in every respect possible (see Section 2.4 for suggestions on preparing images for comparison).

In a medical application, you may never want the computerized comparisons to interfere with the ability of a trained professional (in this case, the cytologist) to make judgments. Thus, you might consider one or both of the following uses of content-based matching to maximize the productivity of the screening process:

- Images are compared by content-based retrieval before the cytologist sees any images.

  All matches are passed on to the cytologist for immediate and careful review; non-matches are to be reviewed later. In this case, it is very important that any image even remotely suspicious be passed on for immediate review (to reduce the possibility of cancer going undetected for a significant time); therefore, it would be wise to set a relatively high threshold value (thus causing more images to be passed on to the cytologist than would occur with a lower threshold value).

- Images that the cytologist, on visual inspection, determines are noncancerous are compared by content-based retrieval.

  In this case, it may be useful to set a lower threshold value than in the preceding item, because human judgment has already been exercised. The chosen threshold value can help, however, to detect obvious cases of cancer that were missed due to human error.

For the cancer screening process, you might consider the following guidelines in selecting weights for the visual attributes:

- global color: A high value is best, because the darker cancerous areas contrast with the light noncancerous areas, and the presence of any cancerous areas in the image being screened will increase its overall color similarity to the cancer-present comparison image.
- local color: A low value is probably best if the darker cancerous areas of the image being screened can appear in any part of the image; however, a high value is probably best if the cancerous areas always or usually appear in the same part of the image as in the comparison image.
- texture: A high value is best if cancerous areas are relatively grainy (light / dark patterns of uneven size, shape, and shade) compared to the noncancerous areas. If this is the case, a high weight for texture will increase the likelihood of a match between a grainy screened image and the comparison image.
- structure: A low value is best if the cancerous areas have random shapes or seem amorphous; in this case, the shapes of cancerous areas on screened and comparison images would not be a useful basis for content-based retrieval. However, if the cancerous areas always or often have similar shapes, a high weight value for structure is best.

2.4 Preparing or Selecting Images for Useful Matching

The human mind is infinitely smarter than a computer in matching images. If we are near a street and want to identify all red automobiles, we can easily do so because our minds rapidly adjust for the following factors:

- whether the automobile is stopped or moving
- the absolute size of the automobile, as well as its relative size in our field of vision (because of its distance from us)
- the location of the automobile in our field of vision (center, left, right, top, bottom)
- the direction in which the automobile is pointing or traveling (left or right, toward us, or away from us)

However, for a computer to find red automobiles (retrieving all red automobiles and no or very few images that are not red or not automobiles), it is helpful if all the automobile images have the automobile occupy almost the entire image, have no extraneous elements (people, plants, decorations, and so on), and have the automobiles pointing in the same direction. In this case, a match emphasizing global color and structure would produce useful results. However, if the pictures show automobiles in different locations, with different relative sizes in the image, pointing in different directions, and with different backgrounds, it will be difficult to perform content-based retrieval with these images.

The following are some suggestions for selecting images or preparing images for comparison. The list in not exhaustive, but the basic principle to keep in mind is this: Know what you are looking for, and use common sense.

- Have what you expect to be looking for occupy almost all of the image space, or at least occupy the same size and position on each image. For example, if you want to find suspected criminals with blue eyes, each image should show only the face and should have the eyes in approximately the same position within the overall image.
- Minimize any extraneous elements that might prevent desired matches or cause unwanted matches. For example, if you want to match red automobiles and if each automobile has a person standing in front of it, the color, shape, and position of the person (skin and clothing) may cause color and shape similarities to be detected, and might reduce the importance of color and shape similarities between automobiles (because part of the automobile is behind the person and thus not visible).

If possible, crop and edit images in accordance with these suggestions before performing content-based retrieval.

Note: Visual Information Retrieval Cartridge operates as a fuzzy search engine, and is not designed to do correlations. For example, the cartridge cannot find a face in a crowd, but if you crop an individual face from a picture of a crowd, you can then compare it to known images.

3

Visual Information Retrieval Examples

This chapter provides examples of common operations with Visual Information Retrieval Cartridge. These operations include:

- creating a new table with a column based on the cartridge
- modifying an existing table to add a column based on the cartridge
- loading images into the table (includes generating the signatures)
- retrieving images (simple read operation; no content-based retrieval)
- retrieving images similar to a comparison image (content-based retrieval)
- converting an image from one format to another
- extending the cartridge with new object types
- using the cartridge with object views

The examples in this chapter use a table of photographs. For each photograph, a photo ID, the photographer's name, a descriptive annotation, and the photographic image are stored.

Reference information on the functions used in these examples is presented in Chapter 4.

3.1 Create a New Table Containing an Image

This example creates a new table of photographic images that includes, for each photograph, the following information:

- photo ID number
- photographer's name (variable text, up to 64 characters)
- descriptive annotation (variable text, up to 255 characters)

- photographic image (defined using the cartridge, stored in the database rather than referred to as an external file)

Example 3–1 creates the table.

Example 3–1 Create a New Table Containing an Image

```
CREATE TABLE stockphotos (photo_id NUMBER, photographer VARCHAR2(64),
            annotation VARCHAR2(255), photo ORDSYS.ORDVIRB);
```

The SQL DESCRIBE TABLE statement shows the following description:

```
Column Name                    Null?    Type
------------------------------ -------- ----
PHOTO_ID                                NUMBER
PHOTOGRAPHER                            VARCHAR2(64)
ANNOTATION                              VARCHAR2(255)
PHOTO                                   ADT(2880)
```

3.2 Add an Image Column to an Existing Table

This example modifies an existing table to store an image with each row. Assume that the table in Section 3.1 already exists, but does not include the actual photographic images.

Example 3–2 adds the `photo` column to the stockphotos table, storing the photographic image in the database itself rather than as a reference to the external file.

Example 3–2 Add an Image Column to an Existing Table

```
ALTER TABLE stockphotos ADD (photo ORDSYS.ORDVIRB);
```

3.3 Load Images into BLOBs From External Files

Loading images into BFILEs is easy. Loading images into BLOBs requires a few extra steps. First, create an empty BLOB, then load the image into a BFILE, and finally, copy the image from the BFILE to the BLOB.

Create a table to use as a temporary location for storing BFILEs as follows:

```
CREATE TABLE imgfiles (id NUMBER, photof ORDVIRF);
```

Next, create an alias for the directory where the images are stored:

```
CREATE DIRECTORY tempdir as 'C:\Images';
```

Example 3–3 loads a file image into a BLOB in the stockphotos table. The program segment performs these operations:

1. Generates an ID number for the row.
2. Inserts a row into the database with the intended values for all columns except `photo`, which is initialized as an empty BLOB.
3. Selects the newly inserted row for update.
4. Inserts the image into a BFILE.
5. Copies the image from the BFILE to the empty BLOB.
6. Sets the property attributes for the image.
7. Generates the signature.
8. Updates the `photo` column with content of the new image.

Example 3–3 Load Images into a Table

```
DECLARE
   buffer    RAW(2000);
   offset    NUMBER;
   amount    NUMBER;
   IdNum     NUMBER
   image     ORDSYS.ORDVIRB;
   imagef    ORDSYS.ORDVIRF;
BEGIN
   -- Generate a photo ID and insert a row into the table.
   -- Note: empty_blob() is the initializer for the BLOB attribute.
   IdNum := 1;
   INSERT INTO stockphotos VALUES (IdNum, 'Janice Gray',
          'Living room, full-length drapes, modern furniture',
          ORDSYS.ORDVIRB(empty_blob(),NULL,NULL,NULL,NULL,NULL,NULL,NULL));

   -- Move image into a temporary BFILE.
   INSERT INTO imgfiles VALUES (IdNum,
      ORDSYS.ORDVIRF(bfilename('TEMPDIR','1097_SAMP.JPG'),NULL,NULL,NULL,NULL,
       NULL,NULL,NULL));

   -- Select the BFILE image.
   SELECT photof INTO imagef FROM imgfiles a WHERE a.id=IdNum;
```

```
   -- Select the empty BLOB row for update.
   SELECT photo INTO image FROM stockphotos b WHERE b.photo_id = IdNum
      FOR UPDATE;

   -- Move image from the BFILE.
      imagef.CopyContent(image.photo);

   -- Set property attributes for the image data.
      image.SetProperties();

   -- Generate the image signature.
      ORDSYS.VIR.Analyze(image.content, image.signature);

   -- Update the photo column with the contents of image.
   -- This also stores the signature and other image-specific attributes.
   UPDATE stockphotos SET photo = image WHERE photo_id = IdNum;
END;
```

3.4 Retrieve an Image

Example 3–4 reads an image from the table and prepares it to be passed along, either directly to the end user or to the application for further processing. The program segment performs these operations:

1. Selects the desired photograph (where `photo_id = myid`) and places it in an image storage area.
2. Prepares the image for output.

Example 3–4 Retrieve an Image (Simple Read)

```
DECLARE
   image     ORDSYS.ORDVIRB;
   myid      INTEGER;
BEGIN

   -- Select the desired photograph from the stockphotos table.
   SELECT photo INTO image FROM stockphotos
          WHERE photo_id = myid;
END;
```

3.5 Retrieve Images Similar to a Comparison Image

Example 3–5 performs content-based retrieval: it finds images that are similar to an image chosen for comparison. The program segment performs these operations:

1. Defines a cursor to perform the matching. The cursor sets the following weight values:
   - global color: 0.5
   - local color: 0.7
   - texture: 0.1
   - structure: 0.9
2. Generates the signature (`compare_sig`) of the comparison image (`compare_img`). Note: The program must have previously placed the comparison image content into `compare_img`.
3. Sets the threshold value at 75.
4. Selects the matching images, using the cursor.

Example 3–5 Retrieve Images Similar to a Comparison Image

```
DECLARE
   threshold    NUMBER;
   compare_sig  RAW(2000);
   compare_img  BLOB;
   photo_id     NUMBER;
   photographer VARCHAR2(64);
   annotation   VARCHAR2(255);
   photo        ORDSYS.ORDVIRB;

-- Define cursor for matching. Set weights for the visual attributes.
CURSOR getphotos IS
   SELECT photo_id, photographer, annotation, photo FROM stockphotos T
   WHERE ORDSYS.VIR.Similar(T.photo.Signature, compare_sig,
                'globalcolor="0.5" localcolor="0.7" texture="0.1"
                   structure="0.9"', threshold)=1;

BEGIN
   -- Generate signature of comparison image, which resides
   -- in compare_img.
   ORDSYS.VIR.Analyze(compare_img, compare_sig);

   -- Set the threshold value.
   threshold := 75;

   -- Retrieve rows for matching images.
   SELECT OPEN getphotos
```

```
   LOOP
      FETCH getphotos INTO photo_id, photographer, annotation, photo;
      EXIT WHEN getphotos%NOTFOUND;
      -- Display or store the results.
      .
      .
   END LOOP;
   CLOSE getphotos;
END;
```

3.6 Convert an Image to a Different Format

Example 3–6 converts an image from its current format to Windows bitmap (BMP) format. The program segment performs these operations:

1. Selects the desired photograph (where `photo_id = 1234`) and places it in an image storage area.
2. Uses the Process method to convert the format to BMP. (You do not need to know the current image format.)
3. Updates the `photo` column with content of the converted image.

Example 3–6 Convert an Image to a Different Format

```
DECLARE
   image     ORDSYS.ORDVIRB;

BEGIN

   -- Select the desired photograph from the stockphotos table.
   SELECT photo INTO image FROM stockphotos WHERE photo_id = 1234;

   -- Use Process method to perform the conversion.
   image.Process('fileFormat=BMPF');

   -- Update the photo column with the contents of image.
   -- This also stores the signature and other image-specific attributes.
   UPDATE stockphotos SET photo = image WHERE photo_id = 1234;

END;
```

3.7 Extend the Cartridge With a New Type

You can use the ORDVirF and ORDVirB types as the basis for a new type of your own creation. This task requires the Objects Option of Oracle8 Enterprise Edition.

```
create type AnnotatedImage as object
   (  image ordsys.ordvirF,
       description varchar2(2000),

       MEMBER PROCEDURE SetProperties(SELF IN OUT AnnotatedImage),
       MEMBER PROCEDURE CopyContent(dest IN OUT BLOB),
       MEMBER PROCEDURE ProcessCopy(command in VARCHAR2, dest IN OUT BLOB)
    );
/

create type body AnnotatedImage as
  MEMBER PROCEDURE SetProperties(SELF IN OUT AnnotatedImage) IS
  BEGIN
    SELF.image.setProperties;
    SELF.description :=
 'This is an example of using the VIR cartridge as a subtype';
  END SetProperties;

  MEMBER PROCEDURE CopyContent(dest IN OUT BLOB) IS
  BEGIN
    SELF.image.copyContent(dest);
  END CopyContent;

  MEMBER PROCEDURE ProcessCopy(command in VARCHAR2, dest IN OUT BLOB) IS
  BEGIN
    SELF.image.processCopy(command,dest);
  END ProcessCopy;
END;
/
```

After creating the new type, you can use it as you would any other type. For example:

```
create or replace directory TEST_DIR as 'C:\TESTS';

create table my_example (id number,an_image AnnotatedImage);

insert into my_example values ( 1,
    AnnotatedImage(
      ordsys.ordvirf(
```

```
        bfilename('TEST_DIR','test1.jpg'),
        NULL,NULL,NULL,NULL,NULL,NULL,NULL),
      NULL)
    );
commit;

declare
  myimage AnnotatedImage;
begin
  select an_image into myimage from my_example;

  myimage.setProperties;

  dbms_output.put_line('This image has a description of ');
  dbms_output.put_line( myimage.description);

  update my_example set an_image=myimage;
end;
```

3.8 Use Image Types With Object Views

Just as a view is a virtual table, an object view is a virtual object table.

Oracle provides object views as an extension of the basic relational view mechanism. By using object views, you can create virtual object tables from data-- of either built-in or user-defined types -- stored in the columns of relational or object tables in the database.

Object views provide the ability to offer specialized or restricted access to the data and objects in a database. For example, you might use an object view to provide a version of an employee object table that doesn't have attributes containing sensitive-data and doesn't have a deletion method. Object views also allow you to try object-oriented programming without permanently converting your tables. Using object views, you can convert data gradually and transparently from relational tables to object-relational tables.

Consider the following non-object image table:

```
create table flat (
   id            number,
   content       bfile,
   height        number,
   width         number,
   contentLength number,
```

```
   fileFormat    varchar2(64),
   contentFormat varchar2(64),
   compressionFormat varchar2(64),
   signature     raw(2000)
   );
```

You can create an object view on the flat table as follows:

```
create or replace view object_images_v as
  select
      id,
      ordsys.ORDVirF(
         T.content,
         T.height,
         T.width,
         T.contentLength,
         T.compressionFormat,
         T.signature) IMAGE
   from flat T;
```

Object views provide the flexibility of looking at the same relational or object data in more than one way. Thus you can use different in-memory object representations for different applications without changing the way you store the data in the database. See the *Oracle8 Concepts* manual for more information on defining, using, and updating object views.

4

Visual Information Retrieval Reference

The Visual Information Retrieval Cartridge library consists of:

- object data types (ODTs) -- see Section 4.1.
- methods -- see Section 4.2.
- operators -- see Section 4.3.

The examples in this chapter assume that the stockphotos table, referred to in Chapter 3, has been created and filled with some photographic images. The table was created using the following SQL statement:

```
CREATE TABLE stockphotos (photo_id NUMBER, photographer (VARCHAR2(64),
            annotation (VARCHAR2(255), photo ORDSYS.ORDVIRB);
```

When you are storing or copying images, you must first create an empty BLOB in the table. Use the empty_blob( ) function, which has the following format:

```
ORDSYS.ORDVIRB(empty_blob(),NULL,NULL,NULL,NULL,NULL,NULL,NULL)
```

4.1 Object Data Types (ODTs)

The Visual Information Retrieval Cartridge object data types (ODTs) are as follows:

- ORDVirB: supports images stored in an Oracle8 BLOB
- ORDVirF: supports images stored in an Oracle8 external file (BFILE)

This section presents reference information on the object data types.

For more information on BLOBs and BFILEs, see *Oracle8 Application Developers Guide, Chapter 6: Large Objects (LOBs).*

ORDVirB Object Data Type

The ORDVirB ODT supports storage and retrieval of image data in a BLOB within an Oracle database. This object data type is defined as follows:

```
CREATE TYPE ORDVIRB
(
-- TYPE ATTRIBUTES
content             BLOB,
height              INTEGER,
width               INTEGER,
contentLength       INTEGER,
fileFormat          VARCHAR2(64),
contentFormat       VARCHAR2(64),
compressionFormat   VARCHAR2(64),
signature           RAW(2000),
-- METHOD DECLARATION
  MEMBER PROCEDURE copyContent(dest IN OUT ORDVIRB),
  MEMBER PROCEDURE setProperties(SELF IN OUT ORDVIRB),
  MEMBER PROCEDURE process    (SELF    IN OUT ORDVIRB,
                               command  IN     VARCHAR2)
  MEMBER PROCEDURE processCopy(command  IN     VARCHAR2,
                               dest     IN OUT BLOB)
);
```

Where:

- content: Stored image
- height: Height of the image in pixels
- width: Width of image in pixels
- contentLength: Size of the *on-disk* image file in bytes
- fileFormat: File type of image (such as, TIFF, JFIF)
- contentFormat: Type of image (such as, monochrome, 8-bit gray scale)
- compressionFormat: Compression type of image
- signature: Feature vector (signature) for content-based retrieval

In PL/SQL data is moved with the DBMS LOB package. From the client, data is moved via OCI LOB calls. The ORDVirB ODT does not supply piece-wise routines for moving data.

ORDVirF Object Data Type

The ORDVirF ODT supports storage and retrieval of image data in external files (BFILEs, which are not stored in the database). BFILEs are assumed to be read-only, and this is reflected in the member procedures. This object data type is defined as follows:

```
CREATE TYPE ORDVIRF
(
  -- TYPE ATTRIBUTES
content             BFILE,
height              INTEGER,
width               INTEGER,
contentLength       INTEGER,
fileFormat          VARCHAR2(64),
contentFormat       VARCHAR2(64),
compressionFormat   VARCHAR2(64),
signature           RAW(2000),
-- METHOD DECLARATION
MEMBER PROCEDURE copyContent(dest IN OUT ORDVIRB),

MEMBER PROCEDURE setProperties(SELF IN OUT ORDVIRF),

MEMBER PROCEDURE processCopy(command IN VARCHAR2,
                             dest    IN OUT BLOB)
);
```

Where:

- content: Stored image
- height: Height of the image in pixels
- width: Width of image in pixels
- contentLength: Size of the *on-disk* image file in bytes
- fileFormat: File type of image (such as, TIFF, JFIF)
- contentFormat: Type of image (such as, monochrome, 8-bit gray scale)
- compressionFormat: Compression type of image
- signature: Feature vector (signature) for content-based retrieval

4.2 Methods

This section presents reference information on the methods used for image manipulation.

The Visual Information Retrieval Cartridge methods are as follows:

- CopyContent( ): copies only the image portion of a BLOB or BFILE to a BLOB.
- Process( ): processes an image in place (for example, modifies it or converts it to another format); available for BLOBs only.
- ProcessCopy( ): copies an image and processes the copy (for example, modifies it or converts it to another format); available for BLOBs and BFILEs.
- SetProperties( ): obtains and stores the attributes of an image.

For more information on object types and methods, see the *Oracle8 Concepts* manual.

CopyContent() Method

Format

CopyContent (dest IN OUT BLOB);

Description

Copy an image without changing it.

Parameter

dest
The destination of the new image.

Usage

Copies the image data into the supplied BLOB.

Example

Create a copy of the image in image1 into myblob:

```
image1.CopyContent(myblob);
```

Process( ) Method

Format

Process (command IN VARCHAR2);

Description

Performs one or more image processing techniques on a BLOB, writing the image back on itself.

Parameter

command
A list of image processing changes to make for the image.

Usage

You can change one or more of the image attributes shown in Table 4–1. See Appendix A, “File and Compression Formats” for information on the supported format combinations.

Table 4–1 Image Processing Parameters

Parameter Name	Usage	Values
fileFormat	file format of the image	GIFF, TIFF, PCXF, PICT, RASF, CALS, BMPF, TGAF, JFIF
contentFormat	imagetype / pixel / data format	MONOCHROME, RAW, 4BITGRAYSCALE, 4BITGREYSCALE, 8BITGRAYSCALE, 8BITGREYSCALE, 1BITLUT, 2BITLUT, 4BITLUT, 8BITLUT, 16BITRGB, 24BITRGB, 32BITRGB, 24BITPLANAR
scale	scale factor (for example, 0.5 or 2.0); preserves aspect ratio	<FLOAT> positive
xscale	X-axis scale factor (default is 1)	<FLOAT> positive
yscale	Y-axis scale factor (default is 1)	<FLOAT> positive

Table 4–1 Image Processing Parameters

Parameter Name	Usage	Values
compressionFormat	compression type/format	JPEG, SUNRLE, BMPRLE, TARGARLE, LZW, LZWHDIFF, FAX3, FAX4, HUFFMAN3, Packbits, GIFLZW
compressionQuality	compression quality	MAXCOMPRATIO, MAXINTEGRIT, LOWCOMP, MEDCOMP, HIGHCOMP
cut	window to cut (origin.x origin.y width height)	(Integer Integer Integer Integer) maximum value is 65535

Note: When specifying parameter values that include floating-point numbers, you must use double quotation marks (" ") around the value. If you do not, this may result in incorrect values being passed and you will get incorrect results.

Examples

Change the file format of image1 to GIF:

```
image1.process('fileFormat=GIFF');
```

Change image1 to use lower quality JPEG compression and double the size of the image, preserving the aspect ratio:

```
image1.process('compressionFormat=JPEG, scale="2.0"');
```

Note that changing the length on only one axis (for example, xscale=2.0) does not affect the length on the other axis, and would result in image distortion.

ProcessCopy( ) Method

Format

ProcessCopy (command IN VARCHAR2,

dest IN OUT BLOB);

Description

Copies an image BLOB to another BLOB.

Parameters

command
A list of image processing changes to make for the image in the new copy.

dest
The destination of the new image.

Usage

See Table 4–1, “Image Processing Parameters”.

Example

Copy an image, changing the file format, compression format, and data format in the destination image:

```
create or replace procedure copyit is
 imgB1     ORDSYS.ORDVIRB;
 imgB4     ORDSYS.ORDVIRF;
 mycommand   VARCHAR2(400);
begin
  select col2 into imgB1 from ordimgtab  where col1 = 1;
  select col2 into imgB4 from ordimgtab  where col1 = 4 for update;
  mycommand:= 'fileFormat=tiff compressionFormat = packbits
  contentFormat = 8bitlut';
  imgB1.processcopy(mycommand,imgB4.content);
  imgB4.setproperties;
  update ordimgtab set col2 = imgB4 where col1 = 4;
end;
```

SetProperties( ) Method

Format

SetProperties();

Description

Writes the characteristics of an image (BLOB or BFILE) into the appropriate attribute fields.

Parameters

None

Usage

After you have copied or stored an image, call this method to set the characteristics of the new image content.

This procedure sets the following information about an image:

- height in pixels
- width in pixels
- data size of the *on-disk* image in bytes
- file type (TIFF, JFIF, and so forth)
- image type (monochrome, 8-bit gray scale, and so forth)
- compression type (JPEG, LZW, and so forth)

Note that SetProperties( ) does not create the signature required for visual information retrieval. See the Analyze( ) operator in Section 4.3 for details.

Example

Select the type, and then set the attributes using the SetProperties procedure.

```
imgB1 ORDSYS.ORDVIRB;
.
.
.
select col2 into imgB1 from ordimgtab where col1 = 1 for update;
imgB1.setProperties;
```

```
dbms_output.put_line('image width = '|| imgB1.width );
dbms_output.put_line('image height = '|| imgB1.height );
dbms_output.put_line('image size = '|| imgB1.contentLength );
dbms_output.put_line('image file type = '|| imgB1.fileFormat );
dbms_output.put_line('image type = '|| imgB1.contentFormat );
dbms_output.put_line('image compression = '|| imgB1.compressionFormat );
-- Note: signature not meaningful as displayed output.
```

Example output:

```
image width = 360
image height = 490
image size = 59650
image file type = JFIF
image type = 24BITRGB
image compression = JPEG
```

4.3 Operators

The Visual Information Retrieval Cartridge operators are located in the ORDSYS.VIR package. The operators, which are specific to content-based retrieval, are as follows:

- Analyze( ): Produces the signature of an image.
- Convert( ): Converts the signature to either big or little endian byte order based on the architecture of the host machine. This operator can also convert a Viisage facial signature to a signature usable by the Score( ) and Similar( ) operators.
- Score( ): Compares two signatures, considers the weights for the visual attributes, and computes an overall score that is the weighted sum of the distances. For example, in Section 2.3.3, overall scores (weighted sums) of 61.5 and 24 were computed for comparisons of the same two imags using two different sets of weights.
- Similar( ): Compares two signatures and determines whether or not the images match, based on the weights and threshold; returns 1 if the computed distance measure (weighted average) is less than or equal to the threshold value, and otherwise returns 0.

For ease of use, you can create a local synonym for the ORDSYS.VIR package. Connect to your schema and issue the following command:

```
SVRMGR> CREATE SYNONYM VIR FOR ORDSYS.VIR;
```

After creating the synonym, you can use it in calls to the operators: VIR.Analyze( ), VIR.Score( ), and so forth. Note that you must have the default CREATE SYNONYM privilege.

This section presents reference information on the operators.

Analyze( ) Operator

Format

Analyze(image IN BLOB, signature OUT RAW);

or

Analyze(image IN BFILE, signature OUT RAW);

Description

Analyzes an image BLOB or BFILE, derives information relating to the visual attributes (including a score for each), and creates the image signature.

Parameters

image
The BLOB or BFILE to be analyzed.

signature
The vector to contain the signature.

Usage

The Analyze( ) operator creates the image attribute signature, which is necessary for any content-based retrieval. Whenever you are working with a new or changed image, you should also use the SetProperties( ) method to set the other image characteristics.

Signatures for facial images can be created using an optional third-party software package from Viisage Technology, Inc. After creating a facial signature, Visual Information Retrieval cartridge can convert the signature to a standard format and then compare the signatures using the Score( ) and Similar( ) operators.

Example

Create the signatures for all images in the stockphotos table.

```
DECLARE
   temp_image   ORDSYS.ORDVirB;
   temp_id      INTEGER;
   cursor c1 is select id, image from stockphotos;
BEGIN
   OPEN c1;
   LOOP
      fetch c1 into temp_id, temp_image;
      EXIT WHEN c1%NOTFOUND;

 -- Generate signature and set the properties for the image.
      ORDSYS.VIR.Analyze(temp_image.content, temp_image.signature);
      temp_image.setProperties;
      UPDATE stockphotos SET photo = temp_image WHERE photo_id = temp_id;
   END LOOP;
   CLOSE c1;
END;
```

Convert( ) Operator

Format

Convert(signature IN OUT RAW, operation IN VARCHAR2);

Description

Converts the image signature to a format usable by the host machine.

Parameters

signature
The signature of the image, as created by the Analyze( ) operator. Data type is raw(2000).

operation
The operation specifies what processing is done to the image signature. The following operations are available:

Operation Keyword	Description
BYTEORDER	Converts the signature to the natural byte order of the host machine.
VIISAGE	Converts the signature from the format used for Viisage face-recognition to a signature usable by the Score( ) and Similar( ) operators.

Usage

The signature is converted to the format of the host platform regardless of its initial state.

This procedure is useful if the database is stored on a remote system, but you want to do your processing locally. If your host machine is from Sun Microsystems, Inc., the Convert( ) operator sets the signature to the big endian byte order. On an Intel Corporation machine, the operator converts the signature to the little endian byte order. Note that the images themselves are platform-independent; only the signatures need to be converted.

Example

The following example converts the signature of the image with `photo_id=1` to the platform of the host system:

```
DECLARE
   myimage ORDSYS.ORDVirB;
   myid      INTEGER;
BEGIN
   SELECT photo INTO myimage FROM stockphotos WHERE photo_id=1;
   ORDSYS.VIR.Convert(myimage.signature,'BYTEORDER');
   UPDATE stockphotos SET photo=myimage;
END;
```

Score( ) Operator

Format

```
Score(signature1   IN RAW,
      signature2   IN RAW,
      weightstring IN VARCHAR2);
```

where weightstring is: 'globalcolor="val", localcolor="val", texture="val", structure="val"'

or: 'facial=1'

Description

Compares the signatures of two images and returns a number representing the weighted sum of the distances for the visual attributes.

Parameters

signature1
The signature of the comparison image (the image with which other images are being compared to test for matches). Data type is raw(2000).

signature2
The signature of the image being compared with the comparison image. Data type is raw(2000).

weightstring
A list of weights to apply to each visual attribute. Data type is VARCHAR2. The following attributes can be specified:

Attribute	Description
globalcolor	The weight value (0.0 to 1.0) assigned to the global color visual attribute. Data type is number. Default is 0.0.
localcolor	The weight value (0.0 to 1.0) assigned to the local color visual attribute. Data type is number. Default is 0.0.
texture	The weight value (0.0 to 1.0) assigned to the texture visual attribute. Data type is number. Default is 0.0.

Attribute	Description
structure	The weight value (0.0 to 1.0) assigned to the structure visual attribute. Data type is number. Default is 0.0.
facial	The two signatures are Viisage facial signatures. When comparing facial signatures, no other attributes can be included in the weight-string. Data type is number, and must be set to 1 if used.

Note: When specifying parameter values that include floating-point numbers, you should use double quotation marks (" ") around the value. If you do not, this may result in incorrect values being passed and you will get incorrect results.

Returns

This function returns a FLOAT value between 0.0 and 100.0.

Usage

Before the Score( ) operator can be used, the image signatures must be created with the Analyze( ) operator.

The Score( ) operator can be useful when an application wants to make finer distinctions about matching than the simple "Yes" or "No" returned by Similar( ). For example, using the number for weighted sum returned by Score( ), the application might assign each image being compared to one of several categories, such as Definite Matches, Probable Matches, Possible Matches, and Non-Matches. The Score( ) operator can also be useful if the application needs to perform special processing based on the degree of similarity between images.

The weights supplied for the four visual attributes are normalized prior to processing such that they add up to 100 percent. To avoid confusion and meaningless results, you should develop a habit of always using the same scale, whether 0 to 100 or 0.0 to 1.0.

You must specify at least one of the four image attributes or the facial attribute, in the weightstring. You cannot combine the facial attribute with any of the other attributes.

Example

The following example finds the weighted sum of the distances between image 1 and the other images in the stockphotos table, using the following weights for the visual attributes:

- global color: 0.2
- local color: 0.2
- texture: 0.1
- structure: 0.5

This example assumes that the signatures have already been created using the Analyze( ) operator and they are stored in the database.

```
DECLARE
   weighted_sum     NUMBER;
BEGIN
   SELECT Q.photo_id,
          ORDSYS.VIR.Score(S.photo.signature,
                                Q.photo.signature,
                                'globalcolor="0.2"
                                 localcolor="0.2"
                                 texture="0.1"
                                  structure="0.5"') weighted_sum
   FROM stockphotos Q, stockphotos S
     WHERE S.photo_id=1 and Q.photo_id !=S.photo_id;
END;
```

The following shows possible output from this example. The last image has the highest score, and therefore is the best match of the test image (`photo_id=1`). Changing the weights used in the scoring would lead to different results.

```
PHOTO_ID   WEIGHTED_SUM
---------- ------------
         2    3.79988
         3    76.0807
         4    47.8139
         5    80.451
         6    91.2473
5 rows selected.
```

Similar( ) Operator

Format

```
Similar(signature1 IN RAW
        signature2 IN RAW,
        weightstring IN VARCHAR2
        threshold   IN FLOAT);
```

where weightstring is: 'globalcolor="val", localcolor="val", texture="val", structure="val"'

or: 'facial=1'

Description

Determines whether or not two images match. Specifically, compares the signatures of two images, computes a weighted sum of the distance between the two images using weight values for the visual attributes, compares the weighted sum with the threshold value, and returns the integer value 1 if the weighted sum is less than or equal to the threshold value, and otherwise returns 0.

Parameters

signature1
The signature of the comparison image (the image with which other images are being compared to test for matches).

signature2
The signature of the image being compared with the comparison image.

weightstring
A list of weights to apply to each visual attribute. Data type is VARCHAR2. The following attributes can be specified:

Attribute	Description
globalcolor	The weight value (0.0 to 1.0) assigned to the global color visual attribute. Data type is number. Default is 0.0.
localcolor	The weight value (0.0 to 1.0) assigned to the local color visual attribute. Data type is number. Default is 0.0.

Attribute	Description
texture	The weight value (0.0 to 1.0) assigned to the texture visual attribute. Data type is number. Default is 0.0.
structure	The weight value (0.0 to 1.0) assigned to the structure visual attribute. Data type is number. Default is 0.0.
facial	The two signatures are Viisage facial signatures. When comparing facial signatures, no other attributes can be included in the weight-string. Data type is number, and must be set to 1 if used.

Note: When specifying parameter values that include floating-point numbers, you should use double quotation marks (" ") around the value. If you do not, this may result in incorrect values being passed and you will get incorrect results.

threshold
The threshold value with which the weighted sum of the distances is to be compared. If the weighted sum is less than or equal to the threshold value, the images are considered to match. This range of this parameter is from 0.0 to 100.0.

Returns

This operator returns an integer value of 0 (not similar) or 1 (match).

Usage

Before the Similar( ) operator can be used, the image signatures must be created with the Analyze( ) operator.

The Similar( ) operator is useful when the application needs a simple "Yes" or "No" for whether or not two images match. The Score( ) operator is useful when an application wants to make finer distinctions about matching or to perform special processing based on the degree of similarity between images.

The weights supplied for the four visual attributes are normalized prior to processing such that they add up to 100 percent. To avoid confusion and meaningless results, you should develop a habit of always using the same scale, whether 0 to 100 or 0.0 to 1.0.

You must specify at least one of the four image attributes, or the facial attribute in the weightstring. You cannot combine the facial attribute with any of the other attributes.

Example

This example checks the first image against all of the other images in the table and determines if there are any matches, using a threshold value of 75 and the following weights for the visual attributes:

- global color: 20
- local color: 20
- texture: 10
- structure: 50

```
DECLARE

BEGIN
   SELECT Q.photo_id FROM stockphotos Q, stockphotos S
      WHERE S.photo_id=1 and Q.photo_id != S.photo_id
      AND  ORDSYS.VIR.Similar(S.photo.signature,
                                    Q.photo.signature,
                                   'globalcolor="20"
                                    localcolor="20"
                                    texture="10"
                                        structure="50"', 75)=1;
END;
```

The following shows a possible output from this example. See the Example section of the Score( ) operator for a different result using the same images and weights.

```
PHOTO_ID
----------
         3
         5
         6
3 rows selected.
```

A

File and Compression Formats

A.1 Supported File and Compression Formats

The following tables describe the file and compression formats supported by Visual Information Retrieval Cartridge.

To use these tables, find the data format you are interested in, and then determine the supported formats. For example, Table A–1 shows that Visual Information Retrieval Cartridge supports BMP format in monochrome, for read and write access, and in 32-bit RGB for read access.

Table A–1 BMP Data Format

Format	Pixel Format	Support
BMP	Monochrome	Read / Write
	4-bit LUT	Read
File Format: 'BMPF'	8-bit LUT	Read / Write
File Ext: .bmp	16-bit RGB	Read
	24-bit RGB	Read / Write
	32-bit RGB	Read
	Compression Format	**Support**
Choose one of these compression formats:	uncompressed	Read / Write
	BMPRLE (for 8-bit LUT)	Read / Write

	Data Description	Support
Choose one or more of these content formats:	Inverse DIB	Read
	OS/2 format	Read
	Byte Order	**Support**
	NA	NA

Table A–2 CALS Raster Data Format

Format	Pixel Format	Support
CALS Raster File Format: 'CALS' File Ext: .cal	Monochrome	Read / Write
	Compression Format	**Support**
	FAX4 (CCITT G4)	Read / Write
	Data Description	**Support**
	NA	NA
	Byte Order	**Support**
	NA	NA

Table A–3 GIF Data Format

Format	Pixel Format	Support
GIF File Format: 'GIFF' File Ext: .gif	Monochrome 8-bit LUT	Read Read / Write
	Compression Format	**Support**
	GIFLZW (LZW)	Read / Write
	Data Description	**Support**
	NA	NA
	Byte Order	**Support**
	NA	NA

Table A–4 JFIF Data Format

Format	Pixel Format	Support
JFIF File Format: 'JFIF' File Ext: .jpg	8-bit grayscale 24-bit RGB	Read / Write Read / Write
	Compression	**Support**
	JPEG	Read / Write
	Data Description	**Support**
	NA	NA
	Byte Order	**Support**
	NA	NA

Table A–5 PCX Data Format

Format	Pixel Format	Support
PCX v 5	Monochrome	Read
	2-bit LUT	Read
File Format: 'PCXF'	4-bit LUT	Read
File Ext: .pcx	8-bit LUT	Read
	1-bit RGB	Read
	2-bit RGB	Read
	4-bit RGB	Read
	8-bit RGB	Read
	24-bit RGB	Read
	Compression Format	**Support**
	RLE	Read
	Data Description	**Support**
	NA	NA
	Byte Order	**Support**
	NA	NA

Table A–6 PICT Data Format

Format	Pixel Format	Support
PICT v. 1 & 2	Monochrome	Read / Write
	2-bit LUT	Read
File Format: 'PICT'	4-bit LUT	Read
File Ext: .pct	8-bit LUT	Read / Write
	16-bit RGB	Read
	24-bit RGB	Read / Write

	Compression Format	Support
Choose one of these compression formats:	Packbits	Read / Write
	JPEG (8-bit gray & RGB)	Read / Write
	Data Description	**Support**
Choose one or more of these content formats:	vector / object graphics	Not supported
	Byte Order	**Support**
	NA	NA

Table A–7 Sun Raster Data Format

Format	Pixel Format	Support
Sun Raster	Monochrome	Read / Write
	8-bit grayscale	Read / Write
File Format: 'RASF'	8-bit LUT	Read / Write
File Ext: .ras	24-bit RGB	Read / Write
	Compression Format	**Support**
Choose one of these compression formats:	(uncompressed)	Read / Write
	SUNRLE (RLE)	Read / Write
	Data Description	**Support**
	NA	NA
	Byte Order	**Support**
	NA	NA

Table A–8 Targa Data Format

Format	Pixel Format	Support
Targa	8-bit grayscale	Read / Write
	8-bit LUT	Read / Write
File Format:'TGAF'	16-bit RGB	Read
File Ext: .tga	24-bit RGB	Read / Write
	32-bit RGB	Read
	Compression Format	**Support**
Choose one of these compression formats:	(uncompressed)	Read / Write
	TARGARLE (RLE)	Read / Write
	Data Description	**Support**
	NA	NA
	Byte Order	**Support**
	NA	NA

Table A–9 TIFF Data Format

Format	Pixel Format	Support
TIFF v.4/5/6	Monochrome	Read / Write
	8-bit grayscale	Read / Write
File Format: 'TIFF'	4 bit LUT	Read
File Ext: .tif	8-bit LUT	Read / Write
	24-bit RGB	Read / Write
	Compression Format	**Support**
Choose one of these compression formats:	(uncompressed)	Read / Write
	Packbits	Read / Write
	Huffman	Read / Write
	FAX3 (CCITT G3)	Read / Write
	FAX4 (CCITT G4)	Read / Write
	LZW	Read / Write
	LZWHDIFF	Read / Write
	JPEG (8-bit gray & RGB)	Read / Write
	Data Description	**Support**
Choose one or more of these content formats:	Planar data	Not supported
	Tiled data	Not supported
	Photometric interpretation	Read / Write
	MSB / LSB	Read / Write
	Byte Order	**Support**
Choose one of these orders:	Intel byte order	Read
	Motorola byte order	Read / Write

B
Sample Program

A sample program is included with Visual Information Retrieval Cartridge to demonstrate how to load two images into the database, generate their signatures, and then compare their signatures using a weighted similarity function.

This program uses two data files, virdemo1.dat and virdemo2.dat, as its input. No other input or parameters are required.

Environment

The following assumptions are made:

- Visual Information Retrieval Cartridge has been installed and PUBLIC has EXECUTE privilege on it.
- Install script has been run. This creates the VIRDEMODIR directory and grants PUBLIC READ access in order that the image data file can be read into the database.
- virdemo1.dat and virdemo2.dat are valid image files that reside in the VIRDEMODIR directory and the user has read / write access to the directory.
- User SCOTT has the default "TIGER" password.

Running the Sample Program

There are two ways to run the sample program: using the included sample images, or using your own images.

Example B-1 runs the sample program using the included image files. The images are compared using the following attribute weights:

- globalcolor = 1.0
- localcolor = 1.0
- texture = 1.0
- structure = 1.0

Example B-1

```
% virdemo
Image 1 and 2 have a similarity score of 0.0
```

Example B-2 shows how to specify your own images on the command line. The images must reside in the VIRDEMODIR directory.

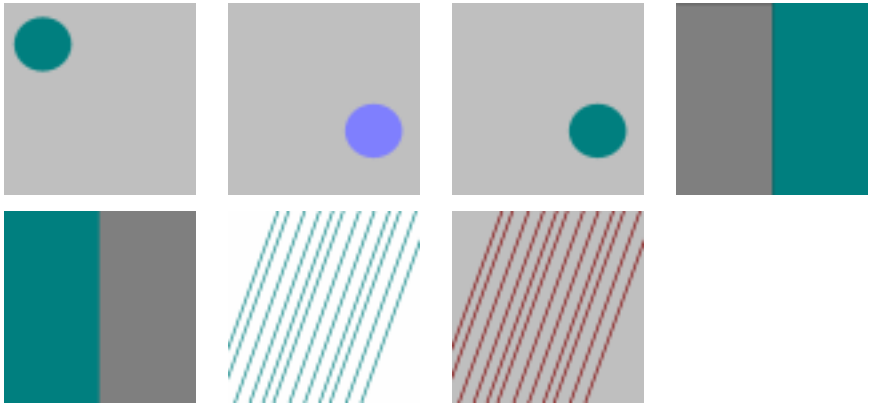
Example B-2

```
% virdemo <image1> <image2> <global_color> <local_color> <texture> <structure>
```

All six parameters: the two file names and four attribute weights (ranging from 0.0 to 1.0) must be specified in this sample program. Note that when using the Score( ) operator in your own applications, it is only necessary to provide at least one attribute weight.

Several other sample image files have been provided in the VIRDEMODIR directory to demonstrate the effects of emphasizing the different visual attributes. You can use an image viewer (such as xv) to display the images, and then compare them using the sample program, experimenting with different weights. Figure B-1 shows the sample images.

Figure B-1 Sample Images in VIRDEMODIR



Understanding the Results

The relative distance between the images being compared is called the score and is output during the execution of the sample program. A lower score indicates a closer match. A score of zero would indicate a perfect match.

A score is only valid for a given set of weights for the four visual attributes. If you change the weights, the results will change. For example, consider a comparison of national flags. The flag of Cote d'Ivoire (the Ivory Coast) is composed of three equal vertical color stripes: orange, white, and green. The flag of Ireland is composed of three vertical stripes in the reverse order: green, white, and orange. A comparison of flag images based on global color or structure would consider these two flags identical. However, a comparison emphasizing local color would return a much larger distance between the two images, indicating a poor match[1].

Sample Program Source Code

```
#ifdef RCSID
static char *RCSid =
"$Header: virdemo.c 05-nov-97 jhebert Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1997. All Rights Reserved. */

/*

   NAME
     virdemo.c - out-of-the-box demo using the visual information retrieval
     cartridge

   DESCRIPTION
      This program demonstrates how to use Visual Information Retrieval
      Cartridge to load two images into the database, generate their signatures,
      and then compare their signatures using a weighted similarity function.
      The program will use two data files provided with the cartridge named
      virdemo1.dat and virdemo2.dat as its input so no parameter is required.

      The relative distance between the images being compared is called the
      score and is output during the execution of the sample program.

      An image viewer (such as xv) may be used to view the input images.

      Sample usage1:    virdemo
```

[1] In the case of using Visual Information Retrieval for flag recognition, you would obviously need to know which direction the flag was facing in the test images.

```
      virdemo1.dat, virdemo2.dat    --  images to be compared with default
weights as:
globalcolor = 1.0
localcolor  = 1.0
texture     = 1.0
structure   = 1.0

      Sample usage2:    virdemo <datafile.dat> <datafile.dat> <gc> <lc> <tx>
<st>

      gc,lc,tx,st--globalcolor,localcolor,texture,structure.

      Optionally, user-provided image files can be specified on
      the command line as shown above, provided that they reside
      in the VIRDEMODIR directory (see assumptions below).
      Note: All 6 parameters (2 file names and 4 values in the range [0.0,1.0])
      must be given for sample usage 2.

   ENVIRONMENT:

      The following assumptions are made:

       1) Visual Information Retrieval Cartridge has been installed and PUBLIC
          has EXECUTE privilege on it.

       2) Install script has been run and thus created VIRDEMODIR
          directory and granted PUBLIC READ access in order that
          the image data file can be read into the database.

       3) virdemo1.dat and virdemo2.dat are valid image files that reside in
           the VIRDEMODIR directory and the user has read/write access
           to the directory.

       4) User SCOTT has the default password.

   PUBLIC FUNCTION(S)
```

```
   PRIVATE FUNCTION(S)

   RETURNS

   NOTES
      The table VIRDEMOTAB is left in the SCOTT account in the default
      database for the user's viewing as well.  The VIRDEMODIR directory
      will likewise be left in the system account.

   MODIFIED   (MM/DD/YY)
   jhebert     11/05/97 - Edit comments in sample program
   rchopra     09/02/97 - Remove redundant HandleFree stmts
   rchopra     07/21/97 - Creation (modified from imgdemo.c by svivian)

*/


#include <stdio.h>

#ifndef OCI_ORACLE
#include <oci.h>
#endif

/* local routines */
static sb4 init_handles();
static sb4 attach_server();
static sb4 log_on();
static void logout();
static sb4 create_table();
static void drop_table();
static sb4 load_cart();
static sb4 test_1_vir(char *, char *, char *, char *);
static void report_error();

#define TRUE       1
#define FALSE      0

#define MAXBUFLEN  16384
```

```
#define STMTLEN    512
#define FNAMELEN   80
#define ARGLEN     10
#define NUMIMGS    2

static OCIEnv        *envhp;
static OCIServer     *srvhp;
static OCISvcCtx     *svchp;
static OCIError      *errhp;
static OCISession    *authp;
static OCIStmt       *stmthp;
static OCILobLocator *blob, *bfile;
static OCIDefine     *defnp1, *defnp2;
static OCIBind       *bndhp, *bndhp1,*bndhp2,*bndhp3,*bndhp4;


static text *user = (text *)"SCOTT";
static text *upwd = (text *)"TIGER";


int main(int argc, char *argv[])
{
  char iname[2][FNAMELEN];
  char g_col[ARGLEN], l_col[ARGLEN], texture[ARGLEN], strct[ARGLEN];

  if (argc == 7)
    {

      strcpy(iname[0], argv[1]);
      strcpy(iname[1], argv[2]);
      strcpy(g_col,argv[3]);
      strcpy(l_col,argv[4]);
      strcpy(texture,argv[5]);
      strcpy(strct,argv[6]);
    }
  else if (argc == 1)
    {
      strcpy(iname[0], "virdemo1.dat");
```

```
    strcpy(iname[1], "virdemo2.dat");
    strcpy(g_col,"1.0");
    strcpy(l_col,"1.0");
    strcpy(texture,"1.0");
    strcpy(strct,"1.0");
  }
else
  {
    (void) fprintf(stdout,"Incorrect number of parameters\n");
    logout();
    return OCI_ERROR;

  }


if (init_handles())
  {
    (void) fprintf(stdout,"FAILED: init_handles()\n");
    return OCI_ERROR;
  }

if (attach_server())
  {
    (void)fprintf (stdout, "FAILED: attach_server()\n");
    logout();
    return OCI_ERROR;
  }

if (log_on(user,upwd))
  {
    (void) fprintf(stdout,"FAILED: log_on()\n");
    logout();
    return OCI_ERROR;
  }

if (create_table(iname))
  {
    (void) fprintf(stdout,"FAILED: create_table()\n");
```

```
      logout();
      return OCI_ERROR;
    }

  (void)fprintf (stdout, "\nLoading data into cartridge...\n");
  if (load_cart())
    {
      (void) fprintf(stdout,"FAILED: load_cart()\n");
      logout();
      return OCI_ERROR;
    }

  (void)fprintf (stdout, "\nTesting...\n");
  if (test_1_vir(g_col,l_col,texture,strct))
    {
      (void)fprintf (stdout, "\nFAILED: test_1_vir()\n");
      logout();
      return OCI_ERROR;
    }


  (void)fprintf (stdout, "\nDisconnecting from database...\n");
  logout();

  (void)fprintf (stdout, "\nDemo completed successfully.\n");
  return OCI_SUCCESS;

}  /* end main */


/*
 * ---------------------------------------------------------------------
 * init_handles - initialize environment, allocate handles, etc.
 * ---------------------------------------------------------------------
 */

sb4 init_handles()
{
```

```
sword status;

if (OCIInitialize((ub4) OCI_DEFAULT,
                  (dvoid *)0,
                  (dvoid * (*)(dvoid *, size_t)) 0,
                  (dvoid * (*)(dvoid *, dvoid *, size_t))0,
                  (void (*)(dvoid *, dvoid *)) 0 ))
  {
    (void) fprintf(stdout,"FAILED: OCIInitialize()\n");
    return OCI_ERROR;
  }

/* initialize environment handle */
if (OCIEnvInit((OCIEnv **) &envhp,
               (ub4) OCI_DEFAULT,
               (size_t) 0,
               (dvoid **) 0 ))
  {
    (void) fprintf(stdout,"FAILED: OCIEnvInit()\n");
    return OCI_ERROR;
  }

if (OCIHandleAlloc((dvoid *) envhp,
                   (dvoid **) &svchp,
                   (ub4) OCI_HTYPE_SVCCTX,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }

if (OCIHandleAlloc((dvoid *) envhp,
                   (dvoid **) &errhp,
                   (ub4) OCI_HTYPE_ERROR,
                   (size_t) 0,
                   (dvoid **) 0))
  {
```

```
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }

if (OCIHandleAlloc((dvoid *) envhp,
                   (dvoid **) &stmthp,
                   (ub4) OCI_HTYPE_STMT,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }

if (OCIHandleAlloc((dvoid *) envhp,
                   (dvoid **) &srvhp,
                   (ub4) OCI_HTYPE_SERVER,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }

if (OCIHandleAlloc((dvoid *) envhp,
                   (dvoid **) &authp,
                   (ub4) OCI_HTYPE_SESSION,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }

/* allocate the lob locator variables */
if (OCIDescriptorAlloc((dvoid *) envhp,
                       (dvoid **) &blob,
                       (ub4)OCI_DTYPE_LOB,
```

```
                        (size_t) 0,
                        (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIDescriptorAlloc(blob)\n");
    return OCI_ERROR;
  }

/* allocate the lob locator variables - will change to OCI_DTYPE_FILE */
if (OCIDescriptorAlloc((dvoid *) envhp,
                        (dvoid **) &bfile,
                        (ub4)OCI_DTYPE_LOB,
                        (size_t) 0,
                        (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIDescriptorAlloc(bfile)\n");
    return OCI_ERROR;
  }

/* allocate the define handles */
if (OCIHandleAlloc((dvoid *) stmthp,
                   (dvoid **) &defnp1,
                   (ub4) OCI_HTYPE_DEFINE,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }

if (OCIHandleAlloc((dvoid *) stmthp,
                   (dvoid **) &defnp2,
                   (ub4) OCI_HTYPE_DEFINE,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }
```

```
/* allocate the bind handles */
if (OCIHandleAlloc((dvoid *) stmthp,
                   (dvoid **) &bndhp,
                   (ub4) OCI_HTYPE_BIND,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }
if (OCIHandleAlloc((dvoid *) stmthp,
                   (dvoid **) &bndhp1,
                   (ub4) OCI_HTYPE_BIND,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }
if (OCIHandleAlloc((dvoid *) stmthp,
                   (dvoid **) &bndhp2,
                   (ub4) OCI_HTYPE_BIND,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }
if (OCIHandleAlloc((dvoid *) stmthp,
                   (dvoid **) &bndhp3,
                   (ub4) OCI_HTYPE_BIND,
                   (size_t) 0,
                   (dvoid **) 0))
  {
    (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
    return OCI_ERROR;
  }
```

```
  if (OCIHandleAlloc((dvoid *) stmthp,
                     (dvoid **) &bndhp4,
                     (ub4) OCI_HTYPE_BIND,
                     (size_t) 0,
                     (dvoid **) 0))
    {
      (void) fprintf(stdout,"FAILED: OCIHandleAlloc()\n");
      return OCI_ERROR;
    }

  return OCI_SUCCESS;

}  /* end init_handles */


/*
 * ---------------------------------------------------------------
 * attach_server - attach to default server
 * ---------------------------------------------------------------
 */
sb4 attach_server()
{
  /* attach to the server - use default host? */
  if (OCIServerAttach(srvhp,
                      errhp,
                      (text *) NULL,
                      0,
                      (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCIServerAttach()\n");
      return OCI_ERROR;
    }

  /* set the server attribute in the service context */
  if (OCIAttrSet((dvoid *) svchp,
                 (ub4) OCI_HTYPE_SVCCTX,
                 (dvoid *) srvhp,
                 (ub4) 0,
```

```
                (ub4) OCI_ATTR_SERVER,
                errhp))
    {
      (void) fprintf(stdout,"FAILED: OCIAttrSet()\n");
      return OCI_ERROR;
    }

  return (OCI_SUCCESS);

}  /* end attach_server */


/*
 * ---------------------------------------------------------------
 * log_on - log on to server
 * ---------------------------------------------------------------
 */
sb4 log_on(
          text *uid,
          text *pwd
          )
{
  if (OCIAttrSet((dvoid *) authp,
                (ub4) OCI_HTYPE_SESSION,
                (dvoid *) uid,
                (ub4) strlen((char *)uid),
                (ub4) OCI_ATTR_USERNAME,
                errhp))
    {
      (void) fprintf(stdout,"FAILED: OCIAttrSet()\n");
      return OCI_ERROR;
    }

  if (OCIAttrSet((dvoid *) authp,
                (ub4) OCI_HTYPE_SESSION,
                (dvoid *) pwd,
                (ub4) strlen((char *)pwd),
                (ub4) OCI_ATTR_PASSWORD,
```

```
                errhp))
    {
      (void) fprintf(stdout,"FAILED: OCIAttrSet()\n");
      return OCI_ERROR;
    }

  /* log on */
  if (OCISessionBegin(svchp,
                      errhp,
                      authp,
                      (ub4) OCI_CRED_RDBMS,
                      (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCISessionBegin()\n");
      return OCI_ERROR;
    }

  /* set the session attribute in the service context */
  if (OCIAttrSet((dvoid *) svchp,
                 (ub4) OCI_HTYPE_SVCCTX,
                 (dvoid *) authp,
                 (ub4) 0,
                 (ub4) OCI_ATTR_SESSION, errhp))
    {
      (void) fprintf(stdout,"FAILED: OCIAttrSet()\n");
      return OCI_ERROR;
    }

  return OCI_SUCCESS;

}  /* end log_on */


/*
 * ---------------------------------------------------------------------
 * create_table - Create table VIRDEMOTAB and insert NUMIMGS rows.
 * ---------------------------------------------------------------------
 */
```

```
sb4 create_table(char iname[NUMIMGS][FNAMELEN])
{
  text *crtstmt = (text *)
    "CREATE TABLE VIRDEMOTAB (C1 INT, C2 ORDSYS.ORDVirF, C3 ORDSYS.ORDVirB)";
  text insstmt[STMTLEN];
  int counter;

  /*
   * drop table first if it exists and then re-create.
   */
  drop_table();

  /*
   * create table
   */
  if (OCIStmtPrepare(stmthp,
                     errhp,
                     crtstmt,
                     (ub4) strlen((char *) crtstmt),
                     (ub4) OCI_NTV_SYNTAX,
                     (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCIStmtPrepare() crtstmt\n");
      report_error();
      return OCI_ERROR;
    }

  (void)fprintf (stdout, "\nCreating and populating table VIRDEMOTAB...\n");
  if (OCIStmtExecute(svchp,
                     stmthp,
                     errhp,
                     (ub4) 1,
                     (ub4) 0,
                     (CONST OCISnapshot *) 0,
                     (OCISnapshot *) 0,
                     (ub4) OCI_DEFAULT))
    {
```

```
      (void) fprintf(stdout,"FAILED: creating table\n");
      report_error();
      return OCI_ERROR;
    }

  /*
   * populate table with simple inserts
   */
  for (counter=0;counter<NUMIMGS;counter++) {
    sprintf ((char*)insstmt,"INSERT INTO VIRDEMOTAB VALUES (%d,ORDSYS.ORD-
VirF(bfilename('VIRDEMODIR','%s'), NULL,NULL,NULL,NULL,NULL,NULL,NULL),ORD-
SYS.ORDVirB(empty_blob(),NULL,NULL,NULL,NULL,NULL,NULL,NULL))",counter+1,
iname[counter]);

    if (OCIStmtPrepare(stmthp,
       errhp,
       insstmt,
       (ub4) strlen((char *) insstmt),
       (ub4) OCI_NTV_SYNTAX,
       (ub4) OCI_DEFAULT))
      {
(void) fprintf(stdout,"FAILED: OCIStmtPrepare() insstmt\n");
report_error();
return OCI_ERROR;
      }

    if (OCIStmtExecute(svchp,
       stmthp,
       errhp,
       (ub4) 1,
       (ub4) 0,
       (CONST OCISnapshot *) 0,
       (OCISnapshot *) 0,
       (ub4) OCI_DEFAULT))
      {
(void) fprintf(stdout,"FAILED: OCIStmtExecute() insstmt\n");
report_error();
return OCI_ERROR;
      }
```

```
  } /* end for stmt */
  (void) OCITransCommit(svchp, errhp, (ub4)0);

  return OCI_SUCCESS;

}  /* end create_table */



/*
 *---------------------------------------------------------------------
 * drop_table - Drop table VIRDEMOTAB
 *---------------------------------------------------------------------
 */

void drop_table()
{
  text  *sqlstmt = (text *) "DROP TABLE VIRDEMOTAB";
  ub1   ebuf[256];
  text  *sqlstate=0;
  ub4   errcodep;

  (void)fprintf(stdout,"\nDropping table VIRDEMOTAB...\n");
  if (OCIStmtPrepare(stmthp,
                     errhp,
                     sqlstmt,
                     (ub4) strlen((char *) sqlstmt),
                     (ub4) OCI_NTV_SYNTAX,
                     (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: drop table\n");
      return;
    }

  (void)OCIStmtExecute(svchp,
                       stmthp,
                       errhp,
                       (ub4) 1,
```

```
                        (ub4) 0,
                        (CONST OCISnapshot *) 0,
                        (OCISnapshot *) 0,
                        (ub4) OCI_DEFAULT);
  return;

}  /* end drop_table */

/*
 * ---------------------------------------------------------------------
 * logout - Logoff and disconnect from the server.  Free handles.
 * ---------------------------------------------------------------------
 */

void logout()
{
  (void) OCISessionEnd(svchp, errhp, authp, (ub4) 0);
  (void) OCIServerDetach(srvhp, errhp, (ub4) OCI_DEFAULT);

  (void) fprintf(stdout,"\nLogged off and detached from server.\n");

  (void) OCIHandleFree((dvoid *) srvhp, (ub4) OCI_HTYPE_SERVER);
  (void) OCIHandleFree((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX);
  (void) OCIHandleFree((dvoid *) errhp, (ub4) OCI_HTYPE_ERROR);
  (void) OCIHandleFree((dvoid *) authp, (ub4) OCI_HTYPE_SESSION);
  (void) OCIDescriptorFree((dvoid *) blob, (ub4) OCI_DTYPE_LOB);
  (void) OCIDescriptorFree((dvoid *) bfile, (ub4) OCI_DTYPE_LOB);
  (void) OCIHandleFree((dvoid *) stmthp, (ub4) OCI_HTYPE_STMT);

  return;

}  /* end logout */


/*
 * ---------------------------------------------------------------------
 * report_error - retrieve error message and print it out.
 * ---------------------------------------------------------------------
```

```
 */
void report_error()
{
  text  msgbuf[1024];
  sb4   errcode;

  (void) OCIErrorGet((dvoid *) errhp,
                     (ub4) 1,
                     (text *) NULL,
                     &errcode,
                     msgbuf,
                     (ub4) sizeof(msgbuf),
                     (ub4) OCI_HTYPE_ERROR);
  (void) fprintf(stdout,"ERROR CODE = %d\n", errcode);
  (void) fprintf(stdout,"%s\n", msgbuf);
  return;

}  /* end report_error */


/*
 * ---------------------------------------------------------------
 * load_cart - load data into Visual Information Retrieval Cartridge
 *
 *   Populate the cartidge columns based on the contents of the BFILE.
 *   The sequence of steps is to select the types and then use the
 *   contents to set the properties and perform the copy from the
 *   BFILE to the BLOB.
 * ---------------------------------------------------------------
 */
sb4 load_cart()
{
  int num_img = NUMIMGS;
  text *sqlstmt = (text *)
    "DECLARE \
       A ORDSYS.ORDVirF;\
       B ORDSYS.ORDVirB;\
     BEGIN\
```

```
    FOR i IN 1..:NUMIMGS LOOP\
    SELECT C2, C3 INTO A,B FROM VIRDEMOTAB  WHERE C1 = i FOR UPDATE;\
    \
    A.setProperties;\
    A.copyContent(B.content);\
    B.setProperties;\
    UPDATE VIRDEMOTAB SET C2 = A WHERE C1 = i;\
    UPDATE VIRDEMOTAB SET C3 = B WHERE C1 = i;\
   END LOOP;\
   COMMIT;\
   END;";

if (OCIStmtPrepare(stmthp,
                   errhp,
                   sqlstmt,
                   (ub4) strlen((char *)sqlstmt),
                   (ub4) OCI_NTV_SYNTAX,
                   (ub4) OCI_DEFAULT))
  {
    (void) fprintf(stdout,"FAILED: OCIStmtPrepare() sqlstmt\n");
    report_error();
    return OCI_ERROR;
  }
/* bind the placeholder :NUMIMGS to a program variable */
if (OCIBindByName (stmthp,
   &bndhp,
   errhp,
   (text *) ":NUMIMGS",
   -1,
   (ub1 *) &num_img,
   (sword) sizeof(num_img),
   SQLT_INT,
   (dvoid *) 0,
   (ub2 *) 0, (ub2) 0, (ub4) 0, (ub4 *) 0,
   OCI_DEFAULT))
  {
    (void) fprintf(stdout,"FAILED: OCIBindByName() sqlstmt\n");
    report_error();
```

```
      return OCI_ERROR;
    }

  /* execute the select and fetch one row */
  if (OCIStmtExecute(svchp,
                     stmthp,
                     errhp,
                     (ub4) 1,
                     (ub4) 0,
                     (CONST OCISnapshot*) 0,
                     (OCISnapshot*) 0,
                     (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCIStmtExecute() sqlstmt\n");
      report_error();
      return OCI_ERROR;
    }

  return OCI_SUCCESS;

}  /* end load_cart */

/*
 * ------------------------------------------------------------------------
 * test_1_vir = Analyze the contents of the BLOB and BFILE and return the
 * score based on the weights assigned to the different features
 * ------------------------------------------------------------------------
 */
sb4 test_1_vir(char *g_c, char * l_c, char * txtr, char * stre)
{
  float w_sum = -1.999; /* dummy value */
  text *sqlstmt = (text *)
    "DECLARE \
A            ORDSYS.ORDVirF; \
B            ORDSYS.ORDVirB; \
a_sig        RAW(2000);\
b_sig        RAW(2000);\
BEGIN \
```

```
SELECT C2 INTO A FROM VIRDEMOTAB WHERE C1=1 FOR UPDATE;\
SELECT C3 INTO B FROM VIRDEMOTAB WHERE C1=2 FOR UPDATE;\
ORDSYS.VIR.Analyze(A.content, a_sig);\
ORDSYS.VIR.Analyze(B.content, b_sig);\
:weighted_sum := ORDSYS.VIR.score(a_sig, b_sig, 'globalcolor='|| :g_color || ',
localcolor= '|| :l_color || ', texture=' || :texture || ', structure=' ||
:strct);\
END;";

  if (OCIStmtPrepare(stmthp,
                     errhp,
                     sqlstmt,
                     (ub4) strlen((char *)sqlstmt),
                     (ub4) OCI_NTV_SYNTAX,
                     (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCIStmtPrepare() sqlstmt\n");
      report_error();
      return OCI_ERROR;
    }
  /* bind the placeholder :weighted_sum to a program variable */

  if (OCIBindByName (stmthp,
     &bndhp,
     errhp,
     (text *) ":weighted_sum",
     -1,
     (ub1 *) &w_sum,
     (sword) sizeof(w_sum),
     SQLT_FLT,
     (dvoid *) 0,
     (ub2 *) 0, (ub2) 0, (ub4) 0, (ub4 *) 0,
     OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCIBindByName() sqlstmt : weighted_sum\n");
      report_error();
      return OCI_ERROR;
    }
```

```
/* bind the placeholder :g_color to a program variable */
if (OCIBindByName (stmthp,
   &bndhp1,
   errhp,
   (text *) ":g_color",
   strlen(":g_color"),
   (ub1 *) g_c,
   (sword) strlen(g_c)*sizeof(g_c),
   SQLT_STR,
   (dvoid *) 0,
   (ub2 *) 0, (ub2) 0, (ub4) 0, (ub4 *) 0,
   OCI_DEFAULT))
  {
    (void) fprintf(stdout,"FAILED: OCIBindByName() sqlstmt : g_color\n");
    report_error();
    return OCI_ERROR;
  }
/* bind the placeholder :l_color to a program variable */
if (OCIBindByName (stmthp,
   &bndhp2,
   errhp,
   (text *) ":l_color",
   strlen(":l_color"),
   (ub1 *) l_c,
   (sword) strlen(l_c)*sizeof(l_c),
   SQLT_STR,
   (dvoid *) 0,
   (ub2 *) 0, (ub2) 0, (ub4) 0, (ub4 *) 0,
   OCI_DEFAULT))
  {
    (void) fprintf(stdout,"FAILED: OCIBindByName() sqlstmt :l_color\n");
    report_error();
    return OCI_ERROR;
  }
/* bind the placeholder :texture to a program variable */
if (OCIBindByName (stmthp,
   &bndhp3,
   errhp,
```

```
   (text *) ":texture",
   strlen(":texture"),
   (ub1 *) txtr,
   (sword) strlen(txtr)*sizeof(txtr),
   SQLT_STR,
   (dvoid *) 0,
   (ub2 *) 0, (ub2) 0, (ub4) 0, (ub4 *) 0,
   OCI_DEFAULT))
  {
    (void) fprintf(stdout,"FAILED: OCIBindByName() sqlstmt\n : texture");
    report_error();
    return OCI_ERROR;
  }
/* bind the placeholder :strct to a program variable */
if (OCIBindByName (stmthp,
   &bndhp4,
   errhp,
   (text *) ":strct",
   strlen(":strct"),
   (ub1 *) stre,
   (sword) strlen(stre)*sizeof(stre),
   SQLT_STR,
   (void *) 0,
   (ub2 *) 0, (ub2) 0, (ub4) 0, (ub4 *) 0,
   OCI_DEFAULT))
  {
    (void) fprintf(stdout,"FAILED: OCIBindByName() sqlstmt : strct\n");
    report_error();
    return OCI_ERROR;
  }

/* execute the select and fetch one row */
if (OCIStmtExecute(svchp,
                   stmthp,
                   errhp,
                   (ub4) 1,
                   (ub4) 0,
                   (CONST OCISnapshot*) 0,
```

```
                    (OCISnapshot*) 0,
                    (ub4) OCI_DEFAULT))
    {
      (void) fprintf(stdout,"FAILED: OCIStmtExecute() sqlstmt\n");
      report_error();
      return OCI_ERROR;
    }

  (void) fprintf(stdout, "Image 1 and 2 have similarity score of %f\n", w_sum);

  return OCI_SUCCESS;
}



/* end of file virdemo.c */
```

C
Frequently Asked Questions

A text file containing a list of frequently asked questions is available on line after installing Visual Information Retrieval Cartridge. Depending on your operating system, the file may be located in:

```
<ORACLE_HOME>/ord/vir/doc/ordrfaq.txt
```

A few of questions from the FAQ have been included in this appendix for emphasis.

C.1 Errors During Installation

Symptom:

```
SQL> grant execute on ordvirb to public;
ORA-04042: procedure, function, package, or package body does not exist
```

Solution:

Check that you created the type in the ORDSYS user.

Symptom:

```
MGR-00072: Warning: PACKAGE BODY VIR created with compilation errors.
```

Symptom:

```
PLS-00201: identifier 'ORDVIRLIBS' must be declared
```

Solution:

The shared library doesn't exist. Recreate the library.

For UNIX systems:

```
SQL> create library ordvirlibS as 'oraclehome/lib/libordvir.so';
```

where oraclehome is the value of the ORACLE_HOME environment variable.

For Windows NT systems:

```
SQL> create library ordvirlibS as 'C:\ORANT\BIN\ORDVIR80.DLL';
```

where C:\ORANT is the value of the Oracle home directory.

C.2 Errors Using the Image Types

Symptom:

```
ORA-06520: PL/SQL: Error loading external library
ORA-06522: ld.so.1: extprocextproc: fatal: /path/libordvir.so: can't open file:
errno=2
ORA-06512: at "ORDSYS.VIR", line 207
ORA-06512: at "ORDSYS.ORDVIRF", line 8
```

Solution:

The path for the shared library is invalid. Recreate the library with the full path name of the library.

For UNIX systems:

```
SQL> create library ordvirlibS as '/oraclehome/lib/libordvir.so';
```

where oraclehome is the value of the ORACLE_HOME environment variable.

For Windows NT systems:

```
SQL> create library ordvirlibS as 'C:\ORANT\BIN\ORDVIR80.DLL';
```

where C:\ORANT is the value of the Oracle home directory.

Symptom:

```
ORA-04068: existing state of packages has been discarded
ORA-04063: package body "ORDSYS.VIR" has errors
ORA-06508: PL/SQL: could not find program unit being called
ORA-06512: at "ORDSYS.ORDVIRF", line 8
```

Solution:

The library ORDVIRLIBS may not exist or the package needs to be recompiled. Recompile the package.

```
SQL> connect ordsys/ordsys;
SQL> alter package VIR compile
```

C.3 RPC Errors After Installation

Symptom:

```
ORA-28576: lost RPC connection to external procedure agent
```

Answer:

The connection information in tnsnames.ora and listener.ora is correct, but the program failed. This is due to conflicting SID_NAME/SID entries in the listener.ora

and tnsnames.ora files. In other words, the extproc SID_NAME in listener.ora is also being used by another entry.

Solution:

Change the SID_NAME for the extproc entry to something unique, such as extproc or extproc2. Be sure to update the SID entry for extproc_connection_data in the tnsnames.ora file to also refer to extproc or extproc2.

Answer:

It is also possible, although unlikely, that there is a bug in the external callout that caused the program to crash. This, however, shouldn't happen with the imgdemo.dat image shipped with the product.

Solution:

Test using the imgdemo.dat image. If there are errors, then use the first solution above to track down a problem with the external callout mechanism configuration.

Solution:

If there are no errors with the imgdemo.dat image, but you get this error when you perform the same operations on your custom images that we support, then you've found a bug. Please notify your Oracle Worldwide Customer Support Services representative.

Symptom:

```
ORA-28575: unable to open RPC connection to external procedure agent
ORA-06512: at "ORDSYS.VIR", line 207
ORA-06512: at "ORDSYS.ORDVIRF", line 8
```

Answer:

The external callout program 'extproc' in your tnsnames.ora or listener.ora file is either invalid or not present.

Solution:

Add/edit the extproc entry in tnsnames.ora and listener.ora. Check the network configuration guide for your platform for details on creating an entry for extproc. After you edit the listener.ora file, you must stop and restart the listener.

For UNIX: Add this line in

/oraclehome/network/admin/tnsnames.ora

```
extproc_connection_data =
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=yourkey))
```

```
(CONNECT_DATA=(SID=extproc)))
```

Add / edit this line in

/ oraclehome / network / admin / listener.ora

```
LISTENER = (ADDRESS_LIST=
 (ADDRESS=(PROTOCOL=ipc)(KEY=yourkey))
)
SID_LIST_LISTENER = (SID_LIST=
 (SID_DESC=(SID_NAME=yourdb)(ORACLE_HOME=/vobs/oracle))
 (SID_DESC=(SID_NAME=extproc)(ORACLE_HOME=/vobs/oracle)
(PROGRAM=extproc))
)
```

To stop and restart the listener: lsnrctl

```
set password oracle
stop
start
quit
```

For Windows NT systems: Add / edit this line in
C:\ORANT\NET80\ADMIN\TNSNAMES.ora

```
extproc_connection_data.world = (
   (DESCRIPTION =
   (ADDRESS =
   PROTOCOL=tcp) (HOST=yourhost) (PORT=1521)
   CONNECT_DATA = (SID=extproc))
```

Add or edit the following line in
C:\ORANT\NET80\ADMIN\LISTENER.ora

```
SID_LIST_LISTENER =
  (SID_LIST=
     (SID_DESC=
     (SIDNAME=ORCL)
  )
   (SID_DESC=
      (SID_NAME=extproc) (PROGRAM=extproc)
   )
 )
```

To stop and restart the listener: lsnrctl80

```
set password oracle
```

```
stop
start
quit
```

Or, open the control panel and open services. Stop and restart the OracleTNSListenerXX service.

C.4 SQL Select Statement Does Not Work

Symptom:

```
SQL> create table image (width integer);
SQL> insert into image values (123);
SQL> select width, image.width from image,images;
WIDTH      WIDTH
---------- ----------
       123        123
```

Solution:

You should always use a table alias when using types to prevent naming conflicts with other objects. Rerun the statement using a table alias.

```
SQL> select width,t.image.width from image,images t;
WIDTH      IMAGE.WIDTH
---------- ----------
       123        600
```

C.5 Extract the File Name from a BFILE Image

You can get the file name and directory name using the dbms_lob package.

```
connect scott/tiger;
set serveroutput on
declare
  virf ordsys.ordvirf;
  filename varchar2(256);
  filepath varchar2(256);
begin
  select imagef into virf from files where id=1;
  dbms_lob.filegetname(virf.content,filepath,filename);
   -- filename is the name of the file
   -- filepath is the name of the directory
   dbms_output.put_line('The file ' || filename ||
      ' is in a directory known as ' || filepath || '.');
```

```
end;
```

This example would produce the following output:

```
The file imgdemo.dat is in a directory known as IMAGES.
```

The full path name may be found in the view named ALL_DIRECTORIES. The following example selects the full path for the images directory:

```
SQL> select directory_path from all_directories where directory_name = 'IMAGES';
```

D
Error Messages

VIR-01001, "analyze failed to generate the signature"

Cause: The Analyze( ) function could not generate the signature.

Action: Verify the image being analyzed is valid.

VIR-01002, "score failed to compare the signatures"

Cause: The Score( ) function could not compare the two signatures.

Action: Verify that the signatures were generated correctly.

VIR-01003, "signature buffer too small"

Cause: The signature being generated is larger than the storage allocated to receive it.

Action: Allocate more space for the signature, reduce the complexity of the image being analyzed, or crop the image to remove extraneous features.

VIR-01005, "empty or null attribute string"

Cause: An empty or null weight attributes string was passed to the Score( ) or Similar( ) function.

Action: Refer to the Visual Information Retrieval Cartridge documentation for a description of the correct usage and syntax for the attribute weights string.

VIR-01006, "invalid attribute value"

Cause: An invalid value was found while parsing the attribute weights string for the Score( ) or Similar( ) functions.

Action: Correct the statement by using a valid attribute value. Refer to the Visual Information Retrieval Cartridge documentation for a description of the correct usage and syntax for the Score( ) and Similar( ) attribute weights string.

VIR-01007, "internal error"

Cause: An internal error has occurred.

Action: Contact Oracle Worldwide Customer Support Services.

VIR-01008, application specific message

Cause: A syntax error was found while parsing the attribute weights string for the Score( ) or Similar( ) functions.

Action: Correct the statement by using valid parameter values. Refer to the Visual Information Retrieval Cartridge documentation for a description of the correct usage and syntax of the attribute weights string.

VIR-01009, "unable to read image data"

Cause: There is no image data in the CONTENT attribute.

Action: Refer to the Visual Information Retrieval Cartridge documentation for information on how to populate image data into the CONTENT attribute of the ORDVirB or ORDVirF type.

VIR-01010, "signature data has been corrupted or is invalid"

Cause: The data in the signature is not a valid Virage signature.

Action: Re-create the signature using the Analyze( ) method.

VIR-01011, "signature is in incorrect byte order"

Cause: The data in the signature may be a valid Virage signature, but is in the incorrect byte order.

Action: Use the Convert() method to change the byte order.

VIR-01012, "signature conversion failed"

Cause: The data in the signature may not be a valid Viisage signature.

Action: Re-write the signature with a valid Viisage signature.

VIR-01013, "invalid conversion operation"

Cause: The specified conversion operation is not valid.

Action: Refer to the Visual Information Retrieval Cartridge documentation for a description of the correct usage and syntax for the Convert operation string.

VIR-01014, "specified weights are not valid"

Cause: The weights specified are not valid for Score( ). Both standard and facial weights were specified.

Action: Do not specify both standard and facial attribute weights.

VIR-01015, "no weights specified"

Cause: All weight passed were zero. At least one attribute must be weighted.

Action: Specify a weight for at least one attribute.

VIR-01016, "internal error during initialization"

Cause: An internal error has occurred while trying to initialize the VIR image engine.

Action: Contact Oracle Worldwide Customer Support Services.

VIR-01017, "out of memory while analyzing image"

Cause: The external procedure agent has exhausted operating system memory while analyzing the image.

Action: See the database administrator or operating system administrator to increase the process memory quota for the external process agent.

VIR-01018, "unable to convert signature to native byte order"

Cause: The signature data might have been corrupted.

Action: Re-create the signature using the Analyze method.

VIR-01019, "signature is not a Viisage signature"

Cause: The incoming signature is not a Viisage signature.

Action: Re-write the signature with a valid Viisage signature.

IMG-00001, "unable to initialize Image Data Cartridge environment"

Cause: The image processing external procedure initialization process failed.

Action: Contact Oracle Worldwide Support.

IMG-00502, "invalid scale value"

Cause: An invalid scale value was found while parsing the parameters for the image process function.

Action: Correct the statement by using a valid scale value. Refer to the Image Cartridge documentation for a description of the correct usage and syntax for the image processing command string.

IMG-00505, "missing value in CUT rectangle"

Cause: An incorrect number of values was used to specify a rectangle.

Action: Use exactly four integer values for the lower left and upper right vertices.

IMG-00506, "extra value in CUT rectangle"

Cause: An incorrect number of values were used to specify a rectangle.

Action: Use exactly four integer values for the lower left and upper right vertices.

IMG-00510, application-specific-message

Cause: A syntax error was found while parsing the parameters for the image process function.

Action: Correct the statement by using valid parameter values. Refer to the Image Cartridge documentation for a description of the correct usage and syntax for the image processing command string.

IMG-00511, application-specific-message

Cause: An error was found while accessing image data.

Action: Contact Oracle Worldwide Support.

IMG-00531, "empty or null image processing command"

Cause: An empty or null image processing command was passed to the image process function.

Action: Refer to the Image Cartridge documentation for a description of the correct usage and syntax for the image processing command string.

IMG-00599, "internal error"

Cause: An internal error has occurred.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00601, "out of memory while copying image"

Cause: Operating system process memory has been exhausted while copying the image.

Action: See the database administrator or operating system administrator to increase process memory quota.

IMG-00602, "unable to access image data"

Cause: An error occurred while reading or writing image data.

Action: Contact your system administrator.

IMG-00603, "unable to access source image data"

Cause: The source image CONTENT attribute is invalid.

Action: Ensure that the CONTENT attribute of the source image is populated with image data.

IMG-00604, "unable to access destination image data"

Cause: The destination image CONTENT attribute is invalid.

Action: Ensure that the CONTENT attribute of the destination image is populated with a valid LOB locator.

IMG-00606, "unable to access image data"

Cause: An attempt was made to access an invalid image.

Action: Ensure that the CONTENT attribute of the image is populated with image data.

IMG-00607, "unable to write to destination image"

Cause: The destination image CONTENT attribute is invalid.

Action: Ensure that the CONTENT attribute of the destination image is populated with an initialized BLOB locator and that you have sufficient tablespace.

IMG-00609, "unable to read image stored in a BFILE"

Cause: The image stored in a BFILE cannot be opened for reading.

Action: Ensure that the access privileges of the image file and the image file's directory allow read access.

IMG-00701, "unable to set the properties of an empty image"

Cause: There is no data in the CONTENT attribute.

Action: Refer to the Image Cartridge documentation for information on how to populate image data into the CONTENT attribute of the ORDImgB or ORDImgF type.

IMG-00702, "unable to initialize image processing environment"

Cause: The image processing external procedure initialization process failed.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00703, "unable to read image data"

Cause: There is no image data in the CONTENT attribute.

Action: Refer to the Image Cartridge documentation for information on how to populate image data into the CONTENT attribute of the ORDImgB or ORDImgF type.

IMG-00704, "unable to read image data"

Cause: There is no image data in the CONTENT attribute.

Action: Refer to the Image Cartridge documentation for information on how to populate image data into the CONTENT attribute of the ORDImgB or ORDImgF type.

IMG-00705, "unsupported or corrupted input format"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00706, "unsupported or corrupted output format"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00707, "unable to access image data"

Cause: An error occurred while reading or writing image data.

Action: Contact your system administrator.

IMG-00710, "unable write to destination image"

Cause: The destination image is invalid.

Action: Ensure that the CONTENT attribute of the destination image is populated with an initialized BLOB locator and that you have sufficient tablespace.

IMG-00711, "unable to set properties of destination image"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00712, "unable to write to destination image"

Cause: The destination image is invalid.

Action: Ensure that the CONTENT attribute of the destination image is populated with an initialized BLOB locator and that you have sufficient tablespace.

IMG-00713, "unsupported destination image format"

Cause: A request was made to convert an image to a format that is not supported.

Action: Refer to the Oracle Image Cartridge Documentation for supported formats.

IMG-00714, "internal error"

Cause: This is an internal error.

Action: Contact Oracle Worldwide Customer Support Services.

IMG-00715, "Unable to open image stored in a BFILE"

Cause: The image stored in a BFILE could not be opened for reading.

Action: Ensure that the access privileges of the image file and the image file's directory allow read access.

ORA-06502, "PL/SQL: numeric or value error"

Cause: The valid range for the threshold argument to the Similar( ) function is from 0.0 to 100.0.

Action: Correct the statement and try again.

Glossary

ADT

See ODT (object data type).

BFILE

A large object whose value is composed of binary data, and is stored outside of the database in an operating system file. The file itself is not stored in the database, but a pointer to the file is stored in the database. Because they are outside of the database, BFILEs are read-only.

binary large object

See BLOB.

biometrics

The science of using personal characteristics to verify identity. This method uses a non-intrusive test, such as face or fingerprint recognition, rather than an intrusive test such as DNA or blood type analysis.

BLOB

Binary large object. Large objects are stored in the database tablespace in a way that optimizes space and provides efficient access. Only a pointer to the object is actually stored in the row.

content format

A description of the image data, such as the pixel or color format.

cropping

Selecting the portion of an image within a specified rectangle and removing everything outside of that rectangle. *See also* cutting.

cutting

Selecting the portion of an image within a specified rectangle to create a subimage. If the subimage is copied to a new image, the effect is a cut. If the subimage replaces the original, the effect is a crop. Use the Process( ) or ProcessCopy( ) procedures to cut or crop an image.

data cartridge

The mechanism by which clients, application-specific servers, and database servers can be easily and reliably extended.

default constructor

The type of constructor in an Oracle Call Interface (OCI) call that creates an empty instance of the constructor.

distance

A measure of how similar two images are. When two images are compared, a distance value for visual attribute and an overall distance value (weighted sum of the attribute distances) are calculated. The distance for each visual attribute can range from 0 (no difference) to 100 (maximum possible difference). Thus, the more similar two images are with respect to a visual attribute, the smaller the distance will be between their scores for that attribute.

file format

The file format of the image data, such as BMP or GIF.

global color

The visual attribute that represents the distribution of colors within the entire image. This distribution includes the amounts of each color, but not the locations of color.

IDL

Interface Definition Language.

image

A graphic picture. The source could be a photograph, drawing, or generated image.

image processing

A change to the properties of an image, such as through scaling, rotation, or compression.

local color

The visual attribute that represents color distributions *and where they occur* in an image, such as the fact that an RGB vector for sky blue occurs in the upper half of an image.

lossless compression

A means for reducing the storage space required for an image. The decompressed image is bit-for-bit identical to the original.

lossy compression

A means for reducing the storage space required for an image. The decompressed image has less resolution than the original, although this might not be noticeable to the naked eye.

Network Computing Architecture

A fully integrated design integrating text, spatial, and image data. The architecture supports the design, development, installation, and integration of manageable components across entire organizations.

object relational database

A database having both object-oriented and relational characteristics. Objects can be defined and stored, and then retrieved using standard relational methods.

OCI

Oracle Call Interface.

ODT (object data type)

A data type that encapsulates attributes of the data and methods (functions and procedures) for operating on the data.

An object data type is sometimes referred to as an abstract data type (ADT).

scaling

A change to the proportions of an image in one or both dimensions. To enlarge an image, scale by a factor greater than one. To shrink an image, scale by a factor between zero and one. Use the Process( ) or ProcessCopy( ) procedures to scale an image.

structure

The visual attribute that represents the shapes that appear in the image, as determined by shape-characterization techniques, rather than by local shape-segmentation methods (which are affected by problems due to lighting effects and occlusion).

texture

The visual attribute that represents the low-level patterns and textures within the image, such as graininess or smoothness. Unlike structure, texture is very sensitive to high-frequency features in the image.

threshold

The numeric value used in a comparison to determine whether or not two images match. If the weighted sum of the distances for the visual attributes is less than or equal to the threshold, the images match; if the weighted sum is greater than the threshold, the images do not match.

weight

A numeric value assigned to a visual attribute to determine how important that attribute is in determining whether or not two images being compared match. Each weight value can range from 0.0 (no importance) to 1.0 (highest importance), and reflects how sensitive the matching process should be to the degree of similarity or dissimilarity between two images.

Index

A

adding
  image column to table, 3-2
Analyze()
  reference information, 4-13

B

binary large object, Gloss-1
BMP Data Format, A-1

C

CALS Raster Data Format, A-2
color
  global, 2-3
  local, 2-3
compression
  formats, A-1
  lossless, 1-4
  lossy, 1-4
content format, Gloss-1
content-based retrieval
  benefits, 2-1
  definition, 1-2
  example, 3-4
  overview, 2-1
converting
  image to different format, 3-6
copyContent( ) method, 4-6
cropping images, Gloss-2
cutting images, Gloss-2

D

data cartridge, 1-3, Gloss-2
  definition, 1-3
distance, 2-7

E

error messages, C-1
examples
  adding image column to table, 3-2
  converting image to different format, 3-6
  loading images into table, 3-2
  retrieving an image (simple read), 3-4
  retrieving images similar to an image (content-based), 3-4
  x-ray screening, 2-10
extending a data cartridge, 1-3

F

face recognition, 2-6
facial signature, 2-6
file format, A-1, Gloss-2
format conversion, 3-6
formats
  compression, A-1
  file, A-1

G

GIF Data Format, A-3
global color, 2-3
Glossary, Gloss-1

I

IDL, Gloss-2
image, Gloss-2
image interchange formats
  overview, 1-5
image processing, Gloss-3
interchange formats
  overview, 1-5

J

JFIF Data Format, A-3

L

loading
  images into table, 3-2
local color, 2-3
lossless compression, 1-4, Gloss-3
lossy compression, 1-4, Gloss-3

M

matching
  preparing or selecting images for, 2-12
medical example (x-ray screening), 2-10
messages, error, C-1
methods, 4-5

N

Network Computing Architecture, Gloss-3

O

object relational database, Gloss-3
object relational technology, 1-5
Object Views, 3-8
OCI, Gloss-3
ORDVIRB
  reference information, 4-3
ORDVIRF
  reference information, 4-4

P

PCX Data Format, A-4
PICT Data Format, A-4
preparing
  images for matching, 2-12
Process()
  reference information, 4-7
ProcessCopy()
  reference information, 4-9
properties
  setting, 4-10

R

reference information, 4-1
retrieval, content-based
  benefits, 2-1
  definition, 1-2
  overview, 2-1
retrieving
  image from table, 3-4
  images similar to an image (content-based), 3-4

S

scaling, Gloss-4
Score()
  reference information, 4-17
selecting
  images for matching, 2-12
SetProperties()
  reference information, 4-10
setting
  properties, 4-10
signature, 2-2
Similar()
  reference information, 4-20
similarity calculation, 2-8
structure (visual attribute), 2-3
Sun Raster Data Format, A-5

T

Targa Data Format, A-6
texture, 2-3

threshold, 2-10
TIFF Data Format, A-7

W

weight, 2-6

X

x-ray screening example, 2-10