Oracle® Spatial

GeoRaster

10*g* Release 2 (10.2)

B14254-01

June 2005

Provides usage and reference information for the GeoRaster feature of Oracle Spatial, which lets you store, index, query, analyze, and deliver GeoRaster data (raster image and gridded data and its associated metadata).

ORACLE®

Oracle Spatial GeoRaster, 10*g* Release 2 (10.2)

B14254-01

Primary Author: Chuck Murray

Contributors: Janet Blowney, Jeffrey Xie, Terry Xu, Sophia Yuditskaya, Zhihai Zhang

Contents

2 GeoRaster Data Types and Related Structures

3 GeoRaster Operations

4 SDO_GEOR Package Reference

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle Spatial GeoRaster, 10*g* Release 2 (10.2)

B14254-01

Oracle 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 title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: Spatial Documentation
- Postal service:

  Oracle Corporation
  Oracle Spatial Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  USA

If you would like a reply, please give your name and contact information.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle Spatial GeoRaster provides usage and reference information for the GeoRaster feature of Oracle Spatial, referred to in this guide as *GeoRaster*. GeoRaster lets you store, index, query, analyze, and deliver GeoRaster data, that is, raster image and gridded data and its associated metadata. GeoRaster provides Oracle Spatial data types and an object-relational schema. You can use these data types and schema objects to store multidimensional grid layers and digital images that can be referenced to positions on the Earth's surface or a local coordinate system.

GeoRaster is not a separate product. It is available when you install Oracle Spatial.

> **Note:** To use GeoRaster, you must understand the main concepts, data types, techniques, operators, procedures, and functions of Oracle Spatial, which are documented in *Oracle Spatial User's Guide and Reference*.

This preface contains these topics:

Audience

This guide is intended for anyone who needs to store GeoRaster data in an Oracle database.

You should be familiar with Oracle Spatial, PL/SQL programming, and Oracle object-relational technology.

You should also be familiar with raster concepts and terminology, techniques for capturing or creating raster data, and techniques for processing raster data. For example, this guide mentions that data can be georeferenced if it is georectified; however, it does not explain the process of georectification or the challenges and techniques involved.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Organization

This guide contains conceptual, usage, and reference information. It has the following elements.

Chapter 1, "GeoRaster Overview and Concepts"

Introduces GeoRaster concepts, including the data model and physical storage model.

Chapter 2, "GeoRaster Data Types and Related Structures"

Explains the object-relational schema associated with GeoRaster.

Chapter 3, "GeoRaster Operations"

Explains the main operations that you can perform using GeoRaster.

Chapter 4, "SDO_GEOR Package Reference"

Provides reference information about the functions and procedures in the SDO_GEOR package.

Chapter 5, "SDO_GEOR_UTL Package Reference"

Provides reference information about the functions and procedures in the SDO_GEOR_UTL (utility) package.

Appendix A, "GeoRaster Metadata XML Schema"
Provides the XML schema definition of the GeoRaster metadata.

Related Documentation

For more information, see the following document:

- *Oracle Spatial User's Guide and Reference*

Oracle error message documentation is only available in HTML. If you only have access to the Oracle Documentation CD, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, go to the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation`

Conventions

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 used in this guide:

Convention	Meaning
. . .	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 text indicates a term defined in the text.
`monospace text`	Monospace text is used for the names of parameters, files, and directory paths. It is also used for SQL and PL/SQL code examples.
italic text	Italic text is used for book titles, emphasis, and some special terms.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

What's New in GeoRaster?

This section describes new and changed features of Oracle Spatial GeoRaster for the current release.

JPEG and DEFLATE Compression and Decompression

You can use native JPEG and DEFLATE compression with GeoRaster objects, and you can decompress compressed GeoRaster objects. All GeoRaster operations in the SDO_GEOR package that can be performed on a decompressed (uncompressed) GeoRaster object can be performed on a compressed GeoRaster object. For more information, see Section 1.8.

Third-party Wavelet Compression and Decompression

GeoRaster supports some wavelet compression types through third-party plug-ins. For information about the availability and features of any such plug-ins, check the Oracle Spatial page on the Oracle Technology Network (OTN):

`http://www.oracle.com/technology/products/spatial/`

Requirements for GeoRaster Tables and Raster Data Tables

The following requirements apply to tables related to GeoRaster:

- A GeoRaster table and its associated raster data table or tables must have the same owner. However, cross-schema usage and operations (such as query, create, insert, update, delete, or copy operations) of GeoRaster objects and their GeoRaster tables and associated raster data tables are allowed, as explained in Section 1.4.4.
- Each raster data table (RDT) name must be unique within the database.

New Procedures for RDT Name Uniqueness

The SDO_GEOR_UTL package (documented in Chapter 5) contains the following new procedures that can help to ensure that raster data table names are unique within the database:

- The SDO_GEOR_UTL.makeRDTNamesUnique procedure renames some existing raster data tables that do not have unique names so that all raster data table names are unique within the database, and it updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.

- The SDO_GEOR_UTL.renameRDT procedure renames one or more existing raster data tables, and updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.

Migration Might Require RDT Name Changes

If you migrate from Oracle Database release 10.1.0.2 or 10.1.0.3 to release 10.1.0.4 or to the current release, some raster data table names might not be unique in the database. If any RDT names are not unique, you must use the SDO_GEOR_UTL.makeRDTNamesUnique or SDO_GEOR_UTL.renameRDT procedure (documented in Chapter 5).

GeoRaster Objects in Schemas Other Than the Current Schema

All SDO_GEOR subprograms (documented in Chapter 4) can now work on GeoRaster objects defined in schemas other than the current connection schema, as long as they have the appropriate privileges. For information about cross-schema considerations, see Section 1.4.4.

changeFormat and scale Procedures Deprecated

The SDO_GEOR.changeFormat and SDO_GEOR.scale procedures have been deprecated; you should instead use the SDO_GEOR.changeFormatCopy and SDO_GEOR.scaleCopy procedures, respectively. Thus, if you use the supported procedures, the format change or scaling is always done on a copy of the input GeoRaster object, and the result is a new GeoRaster object.

These procedures are documented in Chapter 4.

Enhanced GeoRaster Tools

The GeoRaster viewer, loader, and exporter tools have been enhanced as follows:

- The viewer interface provides more features yet is easier to use.
- In the viewer, you can connect to multiple databases simultaneously. The GeoRaster objects from each database are listed in the left pane. You can also display image files (such as exported GeoRaster objects).
- In the viewer, you can quickly switch among views at various resolutions, from the original image (pyramid level 0) to the overview (highest pyramid level).
- The viewer includes new image enhancement features, such as linear stretch (automatic, manual, or piecewise), normalization, equalization, and controls for brightness, contrast, and threshold. These image enhancement features enable you to visualize GeoRaster objects of any cell depth and any cell data range.
- The viewer supports window selection on the GeoRaster object in the image panel, enabling you to export the selected window to image files.
- In the viewer, you can call the GeoRaster loader and exporter tools, thus enabling you to use a single tool as an interface to the capabilities of all the GeoRaster tools.
- The loader can compress raster data and directly store the data into JPEG or DEFLATE compressed GeoRaster objects.

- The loader supports loading of an ESRI world file into an existing GeoRaster object, georeferencing it without reloading the raster data. You can also specify a SRID with a world file, and optionally generate its spatial extent.
- The exporter gives you the option of exporting just the world file instead of both the world file and its associated image file.

The new loader tool features are also available with the SDO_GEOR.importFrom procedure. The new exporter tool feature is also available with the SDO_GEOR.exportTo procedure

The GeoRaster tools are briefly described in Section 1.10.

1

GeoRaster Overview and Concepts

GeoRaster is a feature of Oracle Spatial that lets you store, index, query, analyze, and deliver **GeoRaster data**, that is, raster image and gridded data and its associated metadata. GeoRaster provides Oracle spatial data types and an object-relational schema. You can use these data types and schema objects to store multidimensional grid layers and digital images that can be referenced to positions on the Earth's surface or in a local coordinate system. If the data is georeferenced, you can find the location on Earth for a cell in an image; or given a location on Earth, you can find the cell in an image associated with that location.

GeoRaster can be used with data from any technology that captures or generates images, such as remote sensing, photogrammetry, and thematic mapping. It can be used in a wide variety of application areas, including general business applications, online imagery and photo archiving, environmental monitoring and assessment, geological engineering and exploration, natural resource management, defense, emergency response, telecommunications, transportation, urban planning, and even medical imagery.

Note: To use GeoRaster, you must understand the main concepts, data types, techniques, operators, procedures, and functions of Oracle Spatial, which are documented in *Oracle Spatial User's Guide and Reference*.

You should also be familiar with raster and image concepts and terminology, techniques for capturing or creating raster data, and techniques for processing raster data. For example, this guide mentions that data can be georeferenced if it is georectified; however, it does not explain the process of rectification and orthorectification or the challenges and techniques involved.

GeoRaster uses and depends upon several components that are included with Oracle Database, including the Java virtual machine (JVM), Oracle XML DB, and the ORDSYS schema (used for *inter*Media data types).

Note: If you are upgrading from a previous release of Oracle Database, you must ensure that Oracle XML DB Repository is installed and that the value of the `COMPATIBILITY` database initialization parameter is `10.0` or greater. For more information, see the appendix about installation, compatibility, and upgrade issues in *Oracle Spatial User's Guide and Reference*.

This chapter contains the following major sections:

1.1 Vector and Raster Data

Geographic features can be represented in vector or raster format, or both. With vector data, points are represented by their explicit x,y,z coordinates, lines are strings of points, and areas are represented as polygons whose borders are lines. This kind of vector format can be used to record precisely the location and shape of spatial objects. With raster data, you can represent spatial objects by assigning values to the cells that cover the objects, and you can represent the cells as arrays. This kind of raster format has less precision than vector format, but it is ideal for many types of spatial analysis.

In the raster geographic information systems (GIS) world, this kind of raster data is normally called gridded data. In image processing systems, the raster data representations are typically called *images* instead of grids. Despite any differences between grids and images, both forms of spatial information are usually represented as matrix structures (that is, arrays of cells), and each cell is usually regularly aligned in the space.

1.2 GeoRaster Data Sources

GeoRaster data is collected and used by a variety of geographic information technologies, including remote sensing, airborne photogrammetry, cartography, and global positioning systems. The collected data is then analyzed by digital image processing systems, computer graphics applications, and computer vision technologies. These technologies use several data formats and create a variety of products.

This section briefly describes some of the main data sources and uses for GeoRaster, focusing on concepts and techniques you need to be aware of in developing applications. It does not present detailed explanations of the technologies; you should consult standard textbooks and reference materials for that information.

1.2.1 Remote Sensing

Remote sensing obtains information about an area or object through a device that is not physically connected to the area or object. For example, the sensor might be in a satellite, balloon, airplane, boat, or ground station. The sensor device can be any of a

variety of devices, including a frame camera, pushbroom (swath) imager, synthetic aperture radar (SAR), hydrographic sonar, or paper or film scanner. Remote sensing applications include environmental assessment and monitoring, global change detection and monitoring, and natural resource surveying.

The data collected by remote sensing is often called **geoimagery**. The wavelength, number of bands, and other factors determine the radiometric characteristics of the geoimages. The geoimages can be single-band, multiband, or hyperspectral, all of which can be managed by GeoRaster. These geoimages can cover any area of the Earth (especially for images sensed by satellite). The temporal resolution can be high, such as with meteorological satellites, making it easier to detect changes. For remote sensing applications, various types of resolution (temporal, spatial, spectral, and radiometric) are often important.

1.2.2 Photogrammetry

Photogrammetry derives metric information from measurements made on photographs. Most photogrammetry applications use airborne photos or high-resolution images collected by satellite remote sensing. In traditional photogrammetry, the main data includes images such as black and white photographs, color photographs, and stereo photograph pairs.

Photogrammetry rigorously establishes the geometric relationship between the image and the object as it existed at the time of the imaging event, and enables you to derive information about the object from its imagery. The relationship between image and object can be established by several means, which can be grouped in two categories: analog (using optical, mechanical, and electronic components) or analytical (where the modeling is mathematical and the processing is digital). Analog solutions are increasingly being replaced by analytical/digital solutions, which are also referred to as *softcopy photogrammetry*.

The main product from a softcopy photogrammetry system may include digital elevation models (DEMs) and orthoimagery. GeoRaster can manage all this raster data, together with its georeferencing information.

1.2.3 Geographic Information Systems

A geographic information system (GIS) captures, stores, and processes geographically referenced information. GIS software has traditionally been either vector-based or raster-based; however, with the GeoRaster feature, Oracle Spatial handles both raster and vector data.

Raster-based GIS systems typically process georectified gridded data. Gridded data can be discrete or continuous. Discrete data, such as political subdivisions, land use and cover, bus routes, and oil wells, is usually stored as integer grids. Continuous data, such as elevation, aspect, pollution concentration, ambient noise level, and wind speed, is usually stored as floating-point grids. GeoRaster can store all this data.

The attributes of a discrete grid layer are stored in a relational table called a **value attribute table (VAT)**. A VAT contains columns specified by the GIS vendor, and may also contain user-defined columns. The VAT can be stored in the Oracle database as a plain table. The VAT name can be registered within the corresponding GeoRaster object so that raster GIS applications can use the table.

1.2.4 Cartography

Cartography is the science of creating maps, which are two-dimensional representations of the three-dimensional Earth (or of a non-Earth space using a local

coordinate system). Today, maps are digitized or scanned into digital forms, and map production is largely automated. Maps stored on a computer can be queried, analyzed, and updated quickly.

There are many types of maps, corresponding to a variety of uses or purposes. Examples of map types include base (background), thematic, relief (three-dimensional), aspect, cadastral (land use), and inset. Maps usually contain several annotation elements to help explain the map, such as scale bars, legends, symbols (such as the north arrow), and labels (names of cities, rivers, and so on).

Maps can be stored in raster format (and thus can be managed by GeoRaster), in vector format, or in a hybrid format.

1.2.5 Digital Image Processing

Digital image processing is used to process raster data in standard image formats, such as TIF, GIF, JFIF (JPEG), and Sun Raster, as well as in many geoimage formats, such as ERDAS, PCX, and HDF. Image processing techniques are widely used in remote sensing and photogrammetry applications. These techniques are used as needed to enhance, correct, and restore images to facilitate interpretation; to correct for any blurring, distortion, or other degradation that may have occurred; and to classify geo-objects automatically and identify targets. The source, intermediate, and result imagery can be loaded and managed by GeoRaster.

1.2.6 Geology, Geophysics, and Geochemistry

Geology, geophysics, and geochemistry all use digital data and produce some digital raster maps that can be managed by GeoRaster.

- In geology, the data includes regional geological maps, stratum maps, and rock slide pictures. In geological exploration and petroleum geology, computerized geostratum simulation, synthetic mineral prediction, and 3-D oil field characterization, all of which involve raster data, are widely used.
- In geophysics, data about gravity, the magnetic field, seismic wave transportation, and other subjects is saved, along with georeferencing information.
- In geochemistry, the contents of multiple chemical elements can be analyzed and measured. The triangulated irregular network (TIN) technique is often used to produce raster maps for further analysis, and image processing is widely used.

1.3 GeoRaster Data Model

Raster data can have some or all of the following elements:

- Cells or pixels
- Spatial domain (footprint)
- Spatial, temporal, and band reference information
- Cell attributes
- Metadata
- Processing data and map support data

GeoRaster uses a generic raster data model that is component-based, logically layered, and multidimensional. The core data in a raster is a multidimensional matrix of raster cells. Each cell is one element of the matrix, and its value is called the cell value. If the GeoRaster object represents an image, a cell can also be called a pixel, which has only

one value. (In GeoRaster, the terms *cell* and *pixel* are interchangeable.) The matrix has a number of dimensions, a cell depth, and a size for each dimension. The cell depth is the data size of the value of each cell. The cell depth defines the range of all cell values, and it applies to each single cell, not to an array of cells. This core raster data set can be blocked for optimal storage and retrieval.

The data model has a logically layered structure. The core data consists of one or more logical layers. For example, for multichannel remote sensing imagery, the layers are used to model the channels of the imagery. (Bands and layers are explained in Section 1.5.) In the current release, each layer is a two-dimensional matrix of cells that consists of the row dimension and the column dimension.

GeoRaster data has metadata and attributes, and each layer of the GeoRaster data can have its own metadata and attributes. In the GeoRaster data model, all data other than the core cell matrix is the GeoRaster metadata. The GeoRaster metadata is further divided into different components (and is thus called component-based), which contain the following kinds of information:

- Object information
- Raster information
- Spatial reference system information
- Date and time (temporal reference system) information
- Band reference system information
- Layer information for each layer

Based on this data model, GeoRaster objects are described by the GeoRaster metadata XML schema (described in Appendix A), which is used to organize the metadata. Some schema components and subcomponents are required and others are optional. You must understand this XML schema if you develop GeoRaster loaders, exporters, or other applications. This XML schema can also be extended to include any kind of raster metadata that is not already included in the schema. Some restrictions on the metadata exist for the current release, and these are described in the Usage Notes for the SDO_GEOR.validateGeoraster function (documented in Chapter 4), which checks the validity of the metadata for a GeoRaster object.

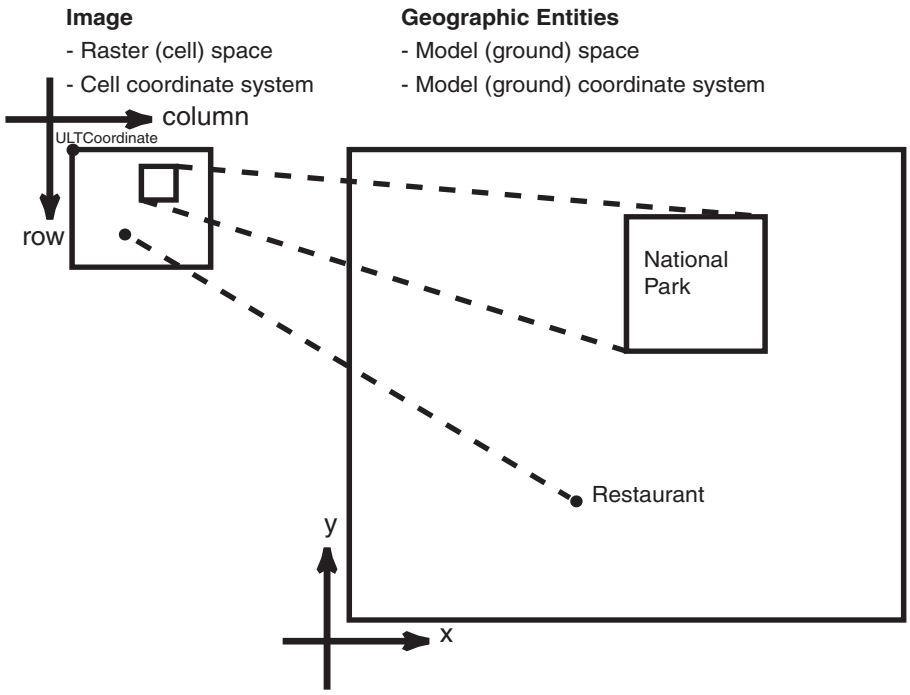
The GeoRaster object data types, described in Chapter 2, are based on the GeoRaster data model.

In this data model, two different types of coordinates need to be considered: the coordinates of each pixel (cell) in the raster matrix and the coordinates on the Earth that they represent. Consequently, two types of coordinate systems or spaces are defined: the cell coordinate system and the model coordinate system.

The **cell coordinate system** (also called the *raster space*) is used to describe pixels in the raster matrix, and its dimensions are (in this order) row, column, and band. The **model coordinate system** (also called the *ground coordinate system* or the *model space*) is used to describe points on the Earth or any other coordinate system associated with an Oracle SRID value. The spatial dimensions of the model coordinate system are (in this order) X and Y, corresponding to the column and row dimensions, respectively, in the cell coordinate system. The logical layers correspond to the band dimension in the cell space.

Figure 1–1 shows the relationship between a raster image and its associated geographical (spatial) extent, and between parts of the image and their associated geographical entities.

Figure 1–1 Raster Space and Model Space



In Figure 1–1:

- In the objects on the left, the medium-size rectangle represents a raster image, and within it are a rectangular area showing a national park and a point identifying the location of a specific restaurant. Each pixel in the image can be identified by its coordinates in a cell coordinate system (the coordinate system associated with the raster image). The upper-left corner of the medium-size rectangle has the coordinate values associated with the `ULTCoordinate` value of the cell space for the GeoRaster object.
- In the objects on the right, the large rectangle represents the geographical area (in the model, or ground, space) that is shown in the raster image, and within it are spatial geometries for the national park and the specific restaurant. Each entire geographical area and geometries within it can be identified using coordinates in its model (or, ground) coordinate system, such as WGS 84 for longitude/latitude data.

In GeoRaster, the upper-left corner of the raster data can have a different coordinate in its cell space from the coordinate of the origin of the cell space. In other words, the (row, column) coordinate of the upper-left corner is not necessarily (0,0). The upper-left corner is called the **ULTCoordinate**, and its value is registered in the metadata. If there is a band dimension, the ULTCoordinate value is always (row,column,0). The coordinate of each cell is relative to the origin of the cell space, not to the ULTCoordinate value. The origin of the cell coordinate system may not be exactly at the ULTCoordinate value.

For two-dimensional single-layer GeoRaster data, the cell coordinate system has a column dimension pointing to the right and a row dimension pointing downward, as shown in Figure 1–1. The unit is in cells, or pixels, and the cell coordinates are identified by integer column and row numbers. For a multiband image, the axis along bands is called the band dimension. For a time series multilayer image (where each layer has a different date or timestamp), the axis along layers is called the temporal

dimension. Three-dimensional GeoRaster data includes the vertical dimension, which is vertical to both the row and column dimensions.

> **Note:** Only row, column, and band dimensions in the cell coordinate system are currently supported. The row and column dimensions are used to model two-dimensional spatial coordinates. The band dimension can be used to model multichannel remote sensing imagery or photographs and any other types of layers, such as temporal layers and multiple-grid themes.

The model coordinate system consists of spatial dimensions, and other dimensions if there are any. The spatial dimensions are called the *x*, *y*, and *z* dimensions, and values in these dimensions can be associated with a geodetic, projected, or local coordinate system. Other dimensions include spectral and temporal dimensions (called the *s* dimension and *t* dimension, respectively). GeoRaster currently supports only two spatial dimensions (X,Y) in the model coordinate system. (For information about coordinate systems, including the different types of coordinate systems, see *Oracle Spatial User's Guide and Reference*.)

The relationships between cell coordinates and model coordinates are modeled by GeoRaster reference systems (mapping schemes). The following GeoRaster reference systems are defined:

- **Spatial reference system**, also called *GeoRaster SRS*, which maps cell coordinates (row,column,vertical) to model coordinates (X,Y,Z). Using the spatial reference system with GeoRaster data is referred to as *georeferencing* the data. (Georeferencing is discussed in Section 1.6.)
- **Temporal reference system**, also called *GeoRaster TRS*, which maps cell coordinates (temporal) to model coordinates (T).
- **Band reference system**, also called *GeoRaster BRS*, which maps cell coordinates (band) to model coordinates (S, for Spectral).

Each of these reference systems is currently defined, at least partially, in the GeoRaster XML schema. However, for the current release, only the two-dimensional spatial reference system is supported. This means that only the relationship between (row,column) and (X,Y) coordinates can be mapped. If the model coordinate system is geodetic, (X,Y) means (longitude,latitude). The temporal and band reference systems can be used, however, to store useful temporal and spectral information, such as the spectral resolution and when the raster data was collected.

Other metadata is stored in the `<layerInfo>` element in the GeoRaster XML metadata, as explained in Section 1.5.

1.4 GeoRaster Physical Storage

As mentioned in Section 1.3, GeoRaster data consists of a multidimensional matrix of cells and the GeoRaster metadata. Most metadata is stored as an XML document using the Oracle XMLType data type. The metadata is defined according to the GeoRaster metadata XML schema, which is described in Appendix A. The spatial extent (footprint) of a GeoRaster object is part of the metadata, but it is stored separately as an attribute of the GeoRaster object. This approach allows GeoRaster to take advantage of the spatial geometry type and related capabilities, such as using R-tree indexing on GeoRaster objects.

The multidimensional matrix of cells is blocked into small subsets for large-scale GeoRaster object storage and optimal retrieval and processing. Each block is stored in a table as a binary large object (BLOB), and a geometry object (of type SDO_GEOMETRY) is used to define the precise extent of the block. Each row of the table stores only one block and the blocking information related to that block. (This blocking scheme applies to any pyramids also.)

The dimension sizes (along row, column, and band dimensions) may not be evenly divided by their respective block sizes. GeoRaster adds **padding** to the boundary blocks that do not have enough original cells to be completely filled. The boundary blocks are the end blocks along the positive direction of each dimension. The padding cells have the same cell depth as other cells and have values equal to zero. Padding makes each block have the same BLOB size. Padding mainly applies to row and column blocks, but for multiband and hyperspectral imagery, padding can be applied to the band dimension also. For example, assume the following specification: band interleaved by line, blocking as (64,64,3), and 8 bands, each with 64 rows and 64 columns. In this case:

1. Bands 0, 1, and 2 are stored interleaved by line in the first block.
2. Bands 3, 4, and 5 are stored interleaved by line in the second block.
3. The third block holds the following in this order: line 1 of band 6, line 1 of band 7, 64 column values that are padding, line 2 of band 6, line 2 of band 7, 64 column values that are padding, and so on, until all 64 rows are stored.

However, the top-level pyramids are not padded if both the row and column dimension sizes of the pyramid level are less than or equal to one-half the row block size and column block size, respectively. See Section 1.7 for information about the physical storage of pyramids.

If the cell depth is greater than 8 bits, GeoRaster cell data is stored in big-endian format in raster blocks (BLOBs). If the cell depth is less than 8 bits, each byte in the raster blocks (BLOBs) contains two or more cells, so that the bits of a byte are fully filled with cell data. The cells are always filled into the byte from left to right. For example, if the cell depth is 4 bits, one byte contains two cells: the first four bits of the byte contain the value of a cell, and the second four bits contain the value of its following cell, which is determined by the interleaving type.

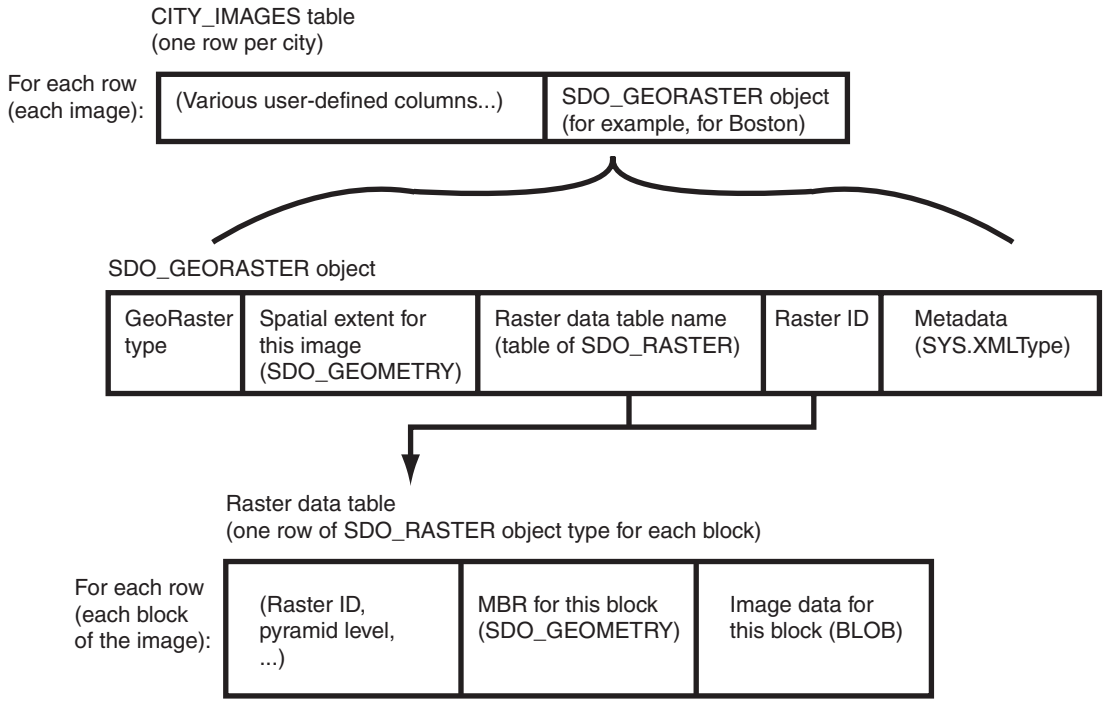
Based on this physical storage model, two object types are provided: SDO_GEORASTER for the raster data set and related metadata, and SDO_RASTER for each block in a raster image.

- The SDO_GEORASTER object contains a spatial extent geometry (footprint or coverage extent) and relevant metadata. A table containing one or more columns of this object type is called a **GeoRaster table**.
- The SDO_RASTER object contains information about a block (tile) of a GeoRaster object, and it uses a BLOB object to store the raster cell data for the block. An object table of this object type is called a **raster data table** (**RDT**).

Each SDO_GEORASTER object has a pair of attributes (`rasterDataTable`, `rasterID`) that uniquely identify the RDT and the rows within the RDT that are used to store the raster cell data for the GeoRaster object.

Figure 1–2 shows the storage of GeoRaster objects, using as an example an image of Boston, Massachusetts in a table that contains rows with images of various cities.

Figure 1–2 Physical Storage of GeoRaster Data



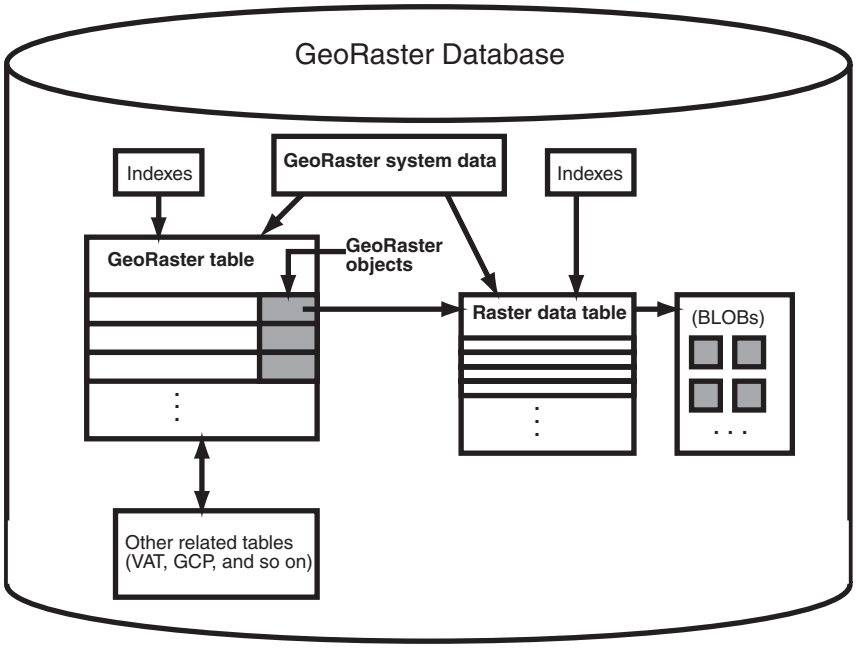
As shown in Figure 1–2:

- Each row in the table of city images contains information about the image for a specific city (such as Boston), including an SDO_GEORASTER object.
- The SDO_GEORASTER object includes the spatial extent geometry covering the entire area of the image, the metadata, the raster ID, and the name of the raster data table associated with this image.
- Each row in the raster data table contains information about a block (or tile) of the image, including the block's minimum bounding rectangle (MBR) and image data (stored as a BLOB). The raster data table is described in Section 1.4.2.

The SDO_GEORASTER and SDO_RASTER object types are described in detail in Chapter 2.

Figure 1–3 shows the physical storage of GeoRaster data and several related objects in a database.

Figure 1–3 GeoRaster Data in an Oracle Database



In Figure 1–3:

- Each GeoRaster object in the GeoRaster table has an associated raster data table, which has an entry for each block of the raster image.
- The BLOB with image data for each raster image block is stored separately from the raster table data. You can specify storage parameters (described in Section 1.4.1) for the BLOBs.
- Each GeoRaster object has a raster data table associated with it. However, a raster data table can store blocks of multiple GeoRaster objects, and GeoRaster objects in a GeoRaster table can be associated with one or multiple raster data tables.
- GeoRaster system data (described in Section 2.4) maintains the relationship between the GeoRaster tables and the raster data tables.
- Indexes (standard and Spatial) can be built on the GeoRaster table and raster data tables. For information about indexing GeoRaster data, see Section 3.8.
- Additional tables, such as ground control point (GCP) tables and value attribute tables (VATs), can be related to the GeoRaster objects.

You should maintain a one-to-many relationship between a GeoRaster table and its associated raster data tables. That is, you should let a raster data table only contain cell data of GeoRaster objects that belong to the same GeoRaster table.

The following considerations apply to schema, table, and column names that are stored in any Oracle Spatial metadata views. For example, these considerations apply to geometry tables, GeoRaster tables, raster data tables, and geometry and GeoRaster columns.

- The name must contain only letters, numbers, and underscores. For example, the name cannot contain a space ( ), an apostrophe (`'`), a quotation mark (`"`), or a comma (`,`).
- All letters in the names are converted to uppercase before the names are stored in geometry metadata views or GeoRaster system data (xxx_SDO_GEOR_SYSDATA)

views or before the tables are accessed. This conversion also applies to any schema name specified with the table name.

For more information about raster data tables, see Section 1.4.2.

For information about cross-schema support with GeoRaster tables and raster data tables, see Section 1.4.4.

1.4.1 Storage Parameters

Several GeoRaster operations let you specify or change aspects of the storage. The relevant subprograms contain a parameter named `storageParam`, which is a quoted string of keywords and their values. The `storageParam` parameter keywords apply to characteristics of the raster data (see Table 1–1).

Note: The keywords in this section either do not apply or only partially apply to the `storageParam` parameter of the SDO_GEOR.importFrom procedure and the `subsetParam` parameter of the SDO_GEOR.exportTo procedure. See the reference information about the relevant parameters for each of these procedures in Chapter 4.

Table 1–1 storageParam Keywords for Raster Data

Keyword	Explanation
blocking	Specifies whether or not raster data is blocked. `TRUE` causes raster data to be blocked using the blocks of the specified or default `blockSize` value; `FALSE` causes raster data not to be blocked (that is, only one block will be used for the entire image). The default value for `blocking` is `TRUE` if you specify the `blockSize` keyword. If you specify `blocking=TRUE` but do not specify the `blockSize` keyword, the default `blockSize` is (256,256,*B*), where *B* is the number of bands in the output GeoRaster object. If you specify neither `blocking` nor `blockSize`, default values are derived from the source GeoRaster object: that is, if the original data is not blocked, the data in the output GeoRaster object is by default not blocked; and if the original data is blocked, the data in the output GeoRaster object is blocked with the same blocking scheme.
blockSize	Specifies the block size, that is, the number of cells per block. You must specify a value for each dimension of the output GeoRaster object. For example, `blocksize=(256,64,3)` specifies 256 for the row dimension, 64 for the column dimension, and 3 for the band dimension; and `blocksize=(128,128)` specifies row and column block sizes of 128 for a GeoRaster object that has no band dimension. For the row and column dimensions, the value must be a positive integer equal to 1 or a positive power of 2. For the band dimension, the value can be any positive integer; although if it is greater than the total number of bands, padding is applied. (These requirements apply only to the use of the `blocksize` keyword to perform reblocking; the dimension sizes of the original GeoRaster object can be any values.) See also the explanation of the `blocking` keyword. Only regular blocking is supported; that is, all blocks must be the same size and be aligned with each other, except for some top-level pyramids.

Table 1–1 (Cont.) storageParam Keywords for Raster Data

Keyword	Explanation
cellDepth	Specifies the cell depth of the raster data set, which indicates the number of bits and the sign for the data type of all cells. Note, however, that changing the cell depth can cause loss of data and a reduction in precision and image quality. Must be one of the following values (_U indicating unsigned and _S indicating signed): 1BIT, 2BIT, 4BIT, 8BIT_U, 8BIT_S, 16BIT_U, 16BIT_S, 32BIT_U, 32BIT_S, 32BIT_REAL, or 64BIT_REAL. If cellDepth is not specified, the value from the source GeoRaster object is used by default. Example: celldepth=16BIT_U
compression	Specifies the compression type to be applied to the GeoRaster object. Must be one of the following values: JPEG-B, JPEG-F, DEFLATE, or NONE. (You can use NONE to decompress a compressed GeoRaster object.) If compression is not specified, the compression type of the source GeoRaster object is used. For more information about compression and decompression, see Section 1.8. Example: compression=JPEG-F If the source GeoRaster object is blank, the compression keyword is ignored, except for the SDO_GEOR.getRasterSubset and SDO_SDO_GEOR.getRasterData functions. (Blank GeoRaster objects are explained in Section 1.4.3.) The compression keyword is not supported in the storageParam parameter for the deprecated SDO_GEOR.changeFormat and SDO_GEOR.scale procedures; however, it is supported for the SDO_GEOR.changeFormatCopy and SDO_GEOR.scaleCopy procedures.
interleaving	Specifies the interleaving type. Must be one of the following values: BSQ (band sequential), BIL (band interleaved by line), or BIP (band interleaved by pixel). Example: interleaving=BSQ
pyramid	TRUE specifies to keep the original pyramid data; FALSE specifies not to keep the original pyramid data. The default value depends on the specific procedure: the default is TRUE for SDO_GEOR.copy, SDO_GEOR.changeFormat, and SDO_GEOR.changeFormatCopy; the default is FALSE for SDO_GEOR.scale, SDO_GEOR.scaleCopy, SDO_GEOR.mosaic, and SDO_GEOR.subset. (A value of TRUE is invalid and is ignored for SDO_GEOR.scale, SDO_GEOR.scaleCopy, SDO_GEOR.mosaic, or SDO_GEOR.subset.) You cannot generate pyramid data through the use of storage parameters; instead, you must use the SDO_GEOR.generatePyramid procedure after creating the GeoRaster object.
quality	Specifies the JPEG compression quality, which is the degree of lossiness caused by the compression. Must be an integer from 0 (lowest quality) through 100 (highest quality) to be applied to the GeoRaster object. The default value is 75. For more information about compression quality, see Section 1.8.1. Example: quality=80

Example 1–1 shows a GeoRaster object being copied, with its block size changed and any pyramid data from the original object not copied.

Example 1–1 Using storageParam Keywords

```
DECLARE
   gr1 sdo_georaster;
   gr2 sdo_georaster;
BEGIN
INSERT INTO georaster_table (georid, georaster) VALUES (2, sdo_geor.init('RDT_1'))
       RETURNING georaster INTO gr2;
SELECT georaster INTO gr1 FROM georaster_table WHERE georid=1;
sdo_geor.changeFormatCopy(gr1, 'blocksize=(128,128) pyramid=FALSE', gr2);
UPDATE georaster_table SET georaster=gr2 WHERE georid=2;
```

```
COMMIT;
END;
/
```

In Example 1–1, the raster data table for GeoRaster object `gr2` is RDT_1. If raster data is to be written into table RDT_1, that table must exist before the PL/SQL block is run; otherwise, an error is generated by the SDO_GEOR.changeFormatCopy procedure.

> **Note:** If you insert, update, or delete GeoRaster cell data or metadata, update the GeoRaster object before committing the transaction, as shown in Example 1–1 and as explained in Section 3.16.

Example 1–1 and many examples in Chapter 4 refer to a table named GEORASTER_TABLE, which has the following definition:

```
CREATE TABLE georaster_table
( georid    NUMBER,
  name      VARCHAR2(32),
  georaster SDO_GEORASTER );

CALL SDO_GEOR_UTL.createCMLTrigger('georaster_table', 'georaster');
```

1.4.2 Raster Data Table

A raster data table must be an object table of SDO_RASTER type, and it must have the primary key defined on the columns `(rasterID, pyramidLevel, bandBlockNumber, rowBlockNumber, columnBlockNumber)`.

You can control the placement and storage characteristics of the raster data table (for example, if the table should be partitioned for better performance). Specifying a large CHUNK size (8, 16, or 32 KB) for the rasterBlock LOB column improves performance. (The CHUNK value must be greater than the database block size.) For a large GeoRaster object, consider putting its raster data in a separate raster data table and partitioning the raster data table by pyramid level or block numbers, or both.

Each raster data table name must be unique in the database. To resolve any duplication in raster data table names, you can use the SDO_GEOR_UTL.makeRDTNamesUnique or SDO_GEOR_UTL.renameRDT procedure, both of which are documented in Chapter 5.

Example 1–2 creates a raster data table named RDT_1 (the same name specified in Example 1–1).

Example 1–2 Creating a Raster Data Table

```
CREATE TABLE rdt_1 OF SDO_RASTER
  (PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber,
    rowBlockNumber, columnBlockNumber))
  TABLESPACE geor_tbs NOLOGGING
  LOB(rasterBlock) STORE AS rdt_1_rbseg
    (TABLESPACE  geor_tbs_2
     CHUNK 8192
     CACHE READS
     NOLOGGING
     PCTVERSION 0
     STORAGE (PCTINCREASE 0)
    );
```

For reference information about creating tables, including specifying LOB storage, see the section about the CREATE TABLE statement in *Oracle Database SQL Reference*.

Do not use the SYSTEM tablespace for storing GeoRaster tables and raster data tables. Instead, create separate locally managed (the default) tablespaces for GeoRaster tables.

In choosing block sizes for raster data, consider the following:

- The maximum length of a raster block is 4 GB; therefore, do not specify a block size greater than 4 GB.
- Consider the `cellDepth` value of the GeoRaster object when you calculate the desired size for a raster block.
- Choosing an appropriate block size is a trade-off between the size of a raster block and the number of blocks needed for a GeoRaster object. A blocking size value that results in a raster block close to 4 KB is usually a bad choice, because 4 KB is the threshold for storing an Oracle BLOB out-of-line.

1.4.3 Blank and Empty GeoRaster Objects

A **blank GeoRaster object** is a special type of GeoRaster object in which all cells have the same value. There is no need to store its cells in any SDO_RASTER block; instead, the cell value is registered in the metadata in the `blankCellValue` element. Otherwise, blank GeoRaster objects are treated in the same way as other GeoRaster objects. Use the SDO_GEOR.createBlank function to create a blank GeoRaster object, the SDO_GEOR.isBlank function to check if a GeoRaster object is a blank GeoRaster object, and the SDO_GEOR.getBlankCellValue function to return the value of the cells in a blank GeoRaster object.

An **empty GeoRaster object** contains only a rasterDataTable name and a rasterID. To create an empty GeoRaster object, use the SDO_GEOR.init function. You must create an empty GeoRaster object before you perform an action that outputs a new GeoRaster object, so that the output can be stored in the previously initialized empty GeoRaster object.

1.4.4 Cross-Schema Support with GeoRaster

A GeoRaster table and its associated raster data table or tables must have the same owner. However, users with appropriate privileges can create GeoRaster tables and associated raster data tables owned by other schemas, and they can also create, query, update, and delete GeoRaster objects owned by other schemas. For cross-schema query of GeoRaster objects, you must have the SELECT privilege on the GeoRaster tables and their associated raster data tables. For cross-schema update of GeoRaster objects, you must have the SELECT, INSERT, UPDATE, and DELETE privileges on the GeoRaster tables and their associated raster data tables.

The ALL_SDO_GEOR_SYSDATA view (described in Section 2.4) contains information about all GeoRaster objects accessible to the current user. For each object listed, the GeoRaster table must be accessible by the current user. If the current user also needs to access the raster data, that user must also have the appropriate privileges on the associated raster data table.

All SDO_GEOR subprograms can work on GeoRaster objects defined in schemas other than the current connection schema.

1.5 Bands, Layers, and Interleaving

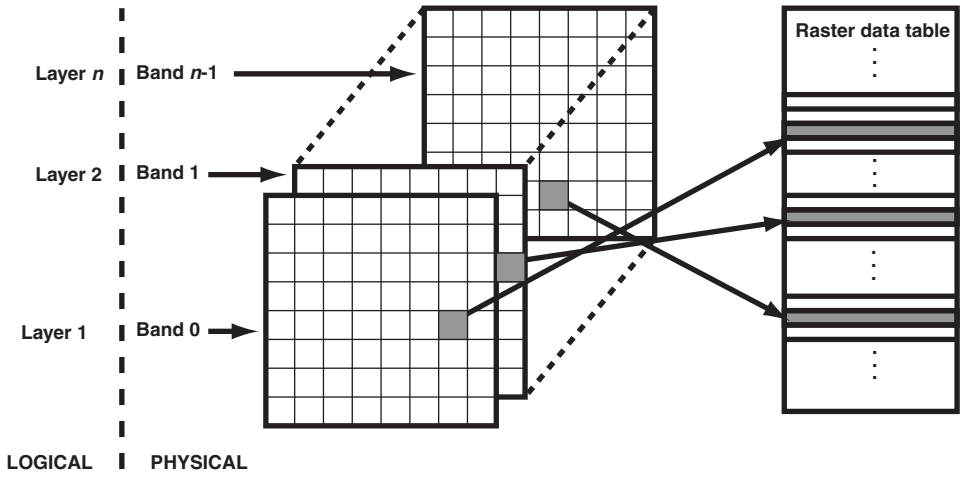
In GeoRaster, *band* and *layer* are different concepts. **Band** is a physical dimension of the multidimensional raster data set; that is, it is one ordinate in the cell space. For example, the cell space might have the ordinates row, column, and band. Bands are numbered from 0 to *n*-1, where *n* is the highest layer number. **Layer** is a logical concept in the GeoRaster data model. Layers are mapped to bands. Typically, one layer corresponds to one band, and it consists of a two-dimensional matrix of size `rowDimensionSize` and `columnDimensionSize`. Layers are numbered from 1 to *n*; that is, `layerNumber = bandNumber + 1`.

A GeoRaster object can contain multiple bands, which can also be called multiple layers. For example, electromagnetic wave data from remote sensing devices is grouped into a certain number of channels, where the number of possible channels depends on the capabilities of the sensing device. *Multispectral* images contain multiple channels, and *hyperspectral* images contain a very large number (say, 50 or more) of channels. The channels are all mapped into GeoRaster bands, which are associated with layers.

In raster GIS applications, a data set can contain multiple raster layers, and each layer is called a **theme**. For example, a raster may have a population density layer, where different cell values are used to depict neighborhoods or counties depending on their average number of inhabitants per square mile or kilometer. Other examples of themes might be average income levels, land use (agricultural, residential, industrial, and so on), and elevation above sea level. The raster GIS themes can be stored in different GeoRaster objects or in one GeoRaster object, and each theme is modeled as one layer. The raster themes and multispectral image channels can also be stored together in one GeoRaster object as different layers, as long as they have the same dimensions.

Figure 1–4 shows an image with multiple layers and a single raster data table. Each layer contains multiple blocks, each of which typically contains many cells. Each block has an entry in the raster data table. Note that GeoRaster starts layer numbering at 1 and band numbering at 0 (zero), as shown in Figure 1–4.

Figure 1–4 Layers, Bands, and the Raster Data Table



The GeoRaster XML metadata refers to the object layer and to layers. The **object layer** refers to the whole GeoRaster object, which may or may not contain multiple layers. If the GeoRaster object contains multiple layers, each layer is a sublayer of the object layer, and it refers to a single band.

Each layer can have an optional set of metadata associated with it. The metadata items for a layer include the layer ID, description, scaling function, bin function, statistical data set (including histogram), grayscale lookup table, and colormap (or, pseudocolor lookup table, also called a PCT). Applications can use the available metadata information. For example, to display a 64-bit GeoRaster object, a typical approach is to process the original cell values through scaling or binning so that the cell value range is less than the 64-bit range and thus can be displayed on a normal monitor. The metadata items are defined in the GeoRaster metadata XML schema, which is presented in Appendix A. See also the explanations of the SDO_GEOR_COLORMAP object type in Section 2.3.2 and SDO_GEOR_GRAYSCALE object type in Section 2.3.3.

The metadata associated with the object layer applies to the whole GeoRaster object. The metadata associated with a layer applies only to that layer. For example, the statistical data set for the object layer is calculated based on all cells of the GeoRaster object, regardless of how many layers the object has; but the statistical data for a layer is calculated based only on the cells in that layer.

The metadata for the object layer and other layers is stored using `<layerInfo>` elements in the GeoRaster XML metadata, which is described in Appendix A, and sometimes in separate tables, such as a colormap table or a histogram table. Metadata stored in the GeoRaster XML metadata is managed by GeoRaster, and you can use the GeoRaster API to retrieve and modify this metadata. For metadata stored in separate tables, the table name can be registered in the GeoRaster XML schema, in which case applications can retrieve the name of the table. However, GeoRaster does not check the existence or validity of that table or provide any operations on that table.

Three types of **interleaving** are supported: BSQ (band sequential), BIL (band interleaved by line), and BIP (band interleaved by pixel). Interleaving applies between bands or layers only. Interleaving is limited to the interleaving of cells inside each block of a GeoRaster object. This means GeoRaster always applies blocking on a GeoRaster object first, and then it applies interleaving inside each block independently. However, each block of the same GeoRaster object has the same interleaving type. You can change the interleaving type of a copy of a GeoRaster object by calling SDO_GEOR.changeFormatCopy procedure, so that the data can be more efficiently processed and used.

1.6 Georeferencing

The GeoRaster spatial reference system (SRS), a metadata component of the GeoRaster object, includes information related to georeferencing. **Georeferencing** establishes the relationship between cell coordinates of GeoRaster data and real-world ground coordinates (or some local coordinates). Georeferencing assigns ground coordinates to cell coordinates, and cell coordinates to ground coordinates.

In GeoRaster, georeferencing is different from geocorrection, rectification, or orthorectification. In these three latter processes, cell resampling is often performed on the raster data, and the resulting GeoRaster data might have a different model coordinate system and dimension sizes. Georeferencing establishes the relationship between cell coordinates and real-world coordinates or some local coordinates. Georeferencing can be accomplished by providing an appropriate mathematical formula, enough ground control point (GCP) coordinates, or rigorous model data from the remote sensing system. Georeferencing does not change the GeoRaster cell data or other metadata, except as needed to facilitate the transformation of coordinates between the cell coordinate system and the model coordinate system.

GeoRaster currently supports six-parameter affine transformation that georeferences two-dimensional raster data. The affine transformation is a special type of the

Functional Fitting polynomial model. If an affine transformation is provided and is valid in the metadata, the GeoRaster object is considered georeferenced, and the `isReferenced` value in the SRS metadata will be `TRUE`; otherwise, it should be `FALSE`.

Rectification can be done with horizontal coordinates, so that cells of a GeoRaster data set can be mapped to a projection map coordinate system. After rectification, each cell is regularly sized in the map units and is aligned with the model coordinate system, that is, with the East-West dimension and the North-South dimension. If elevation data (DEM) is used in rectification, it is called **orthorectification**, a special form of rectification that corrects terrain displacement. If a GeoRaster object is rectified, the `isRectified` value in its metadata will be `TRUE`; otherwise, it should be `FALSE`. If a GeoRaster object is orthorectified, the `isOrthoRectified` value in its metadata will be `TRUE`; otherwise, it should be `FALSE`.

To georeference a GeoRaster object, see Section 3.6.

1.6.1 GeoRaster Georeferencing Method

In the current release, GeoRaster uses low order polynomials when georeferencing image or raster data according to the following six-parameter affine transformation formulas:

```
row = a + b * x + c * y
col = d + e * x + f * y
```

In these formulas:

- row = Row index of the cell in raster space.
- col = Column index of the cell in raster space.
- x = East-West position of the point on the ground or in model space.
- y = North-South position of the point on the ground or in model space.
- a, b, c, d, e, and f are coefficients, and they are stored in the SRS metadata.
- b*f – c*e should not be equal to 0 (zero).

In the formulas, if b = 0, f = 0, c = -e, and both c and e are not 0 (zero), the raster data is rectified, and the formula becomes:

```
row = a + c * y
col = d - c * x
```

This is the simplest case for georeferencing.

In the current release, the `cellRepresentationType` value must be `UNDEFINED`. In other words, a cell is just a scalar value (or an element of an array) without any shape defined in its cell space. However, when the GeoRaster object is georeferenced, each cell covers a specific square or rectangular area or represents a point of this area in the model space. In the cell space, each cell has an integer coordinate. Through georeferencing, the cell's integer coordinate can be transformed into model coordinates, which identify an exact location of a point. This point or model coordinate may be either the upper-left corner or the center of the area represented by the cell in the model space.

The model coordinates have the same unit as that of the specified SRID and should be in the value range defined by the model coordinate system. For example, if the GeoRaster object is georeferenced to a geodetic coordinate system such as 8307, the unit of the model coordinates derived from the SRS must be decimal degrees, and

values should be in the ranges of -180.0 to +180.0 or 0.0 to 360.0 for longitude and -90.0 to +90.0 for latitude. In this case, you cannot use meters or other unit.

In the GeoRaster XML schema, the `modelCoordinateLocation` element specifies whether the base of the area in the model space represented by a cell is at the upper-left corner or the center of the area, as follows:

```
<xsd:element name="modelCoordinateLocation" minOccurs="0">
        <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="CENTER"/>
                        <xsd:enumeration value="UPPERLEFT"/>
                </xsd:restriction>
        </xsd:simpleType>
</xsd:element>
```

If the original georeferencing information in the source data is in the inverse direction, such as in an ESRI world file, the transformation formulas are the following:

```
x = A * col + B * row + C
y = D * col + E * row + F
```

In this case, the preceding A, B, C, D, E, and F coefficients that you specify to the SDO_GEOR.georeference procedure are automatically adjusted internally to produce the correct georeferencing result: a, b, c, d, e, and f coefficients.

1.6.2 Cell Coordinate and Model Coordinate Transformation

Cell coordinate and model coordinate transformations are based on the GeoRaster spatial reference system (SRS) and the ULTCoordinate (upper-left coordinate) of the raster data. (The SRS and the ULTCoordinate are described in Section 1.3.) The ULTCoordinate can have a different coordinate (row and column values) from the coordinate of the origin of the cell space. That is, the (row, column) coordinate of the upper-left corner is not necessarily (0,0). For any transformation between the cell space and the model space, the values of the ULTCoordinate must be taken into consideration.

Both before and after transformation using the GeoRaster SRS, the (row, column) coordinate values of a cell are in the GeoRaster cell space, and these values take the ULTCoordinate into consideration, as shown in the following formulas:

```
row = row0 + m
col = col0 + n
```

In these formulas:

- row = Row index (starting at 0 for the first row) of the cell relative to the origin of cell space.
- col = Column index (starting at 0 for the first column) of the cell relative to the origin of cell space.
- row0 = Row index (starting at 0 for the first row) of the ULTCoordinate relative to the origin of cell space.
- col0 = Column index (starting at 0 for the first column) of the ULTCoordinate relative to the origin of cell space.
- m = Row index (that is, the *m*th row, starting at 0 for the first row) of the cell relative to the ULTCoordinate.

- n = Column index (that is, the *n*th column, starting at 0 for the first column) of the cell relative to the ULTCoordinate.

In most applications, the ULTCoordinate and the origin of cell space are the same (that is, row0 = 0 and col0 = 0), in which case row = m and col = n.

1.7 Pyramids

Pyramids are subobjects of a GeoRaster object that represent the raster image or raster data at differing sizes and degrees of resolution. The size is usually related to the amount of time that an application needs to retrieve and display an image, particularly over the Web. That is, the smaller the image size, the faster it can be displayed; and as long as detailed resolution is not needed (for example, if the user has "zoomed out" considerably), the display quality for the smaller image is adequate.

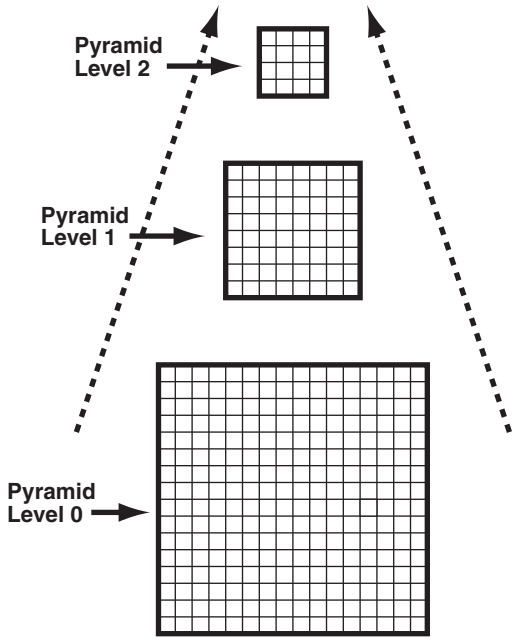
Pyramid levels represent reduced or increased resolution images that require less or more storage space, respectively. (GeoRaster currently supports only reduced resolution pyramids.) A pyramid level of 0 indicates the original raster data; that is, there is no reduction in the image resolution and no change in the storage space required. Values greater than 0 (zero) indicate increasingly reduced levels of image resolution and reduced storage space requirements.

Pyramid type indicates the type of pyramid, and can be one of the following values:

- `DECREASE` means that pyramids decrease in size as the pyramid level increases.
- `NONE` means that there are no pyramids associated with the GeoRaster object.

Figure 1–5 shows the concept of pyramid levels with a pyramid type of `DECREASE`. It conveys the idea that as the pyramid level number increases, the file size decreases, but the resolution also decreases because fewer pixels are used to represent the image.

Figure 1–5 Pyramid Levels



The size of the pyramid image at each level is determined by the original image size and the pyramid level, according to the following formulas:

```
r(n) = (int)(r(0) / 2^n)
c(n) = (int)(c(0) / 2^n)
```

In the preceding formulas:

- `r(0)` and `c(0)` are the original row and column dimension size.
- `r(n)` and `c(n)` are the row and column dimension size of pyramid level *n*.
- `int` rounds off a number to the integer value that is less than but closest to that number.
- `2^n` means 2 to the power of *n*.

The smaller of the row and column dimension sizes of the top-level overview (the smallest top-level pyramid) is 1. This determines the maximum reduced-resolution pyramid level, which is calculated as follows: `(int)(log2(a))`

In the preceding calculation:

- `log2` is a logarithmic function with 2 as its base.
- `a` is the smaller of the original row and column dimension size.

The pyramids are stored in the same raster data table as the GeoRaster object. The pyramidLevel attribute in the raster data table identifies all the blocks related to a specific pyramid level. In general, the blocking scheme for each pyramid level is the same as that for the original level (which is defined in the GeoRaster object metadata), except in the following cases:

- If the original GeoRaster object is not blocked, that is, if the original cell data is stored in one block (BLOB) of the exact size of the object, the cell data of each pyramid level is stored in one block, and its size is the same as that of the actual pyramid level image.
- If the original GeoRaster object is blocked (even if blocked as one block), the cell data of each pyramid level is blocked in the same way as for the original level data, and each block is stored in a different BLOB object as long as the maximum dimension size of the actual pyramid level image is larger than the block sizes. However, if lower-resolution pyramids are generated (that is, if both the row and column dimension sizes of the pyramid level are less than or equal to one-half the row block size and column block size, respectively), the cell data of each such pyramid level is stored in one BLOB object and its size is the same as that of the actual pyramid level image.

When pyramids are generated on a GeoRaster object or when a GeoRaster object is scaled, resampling of cell data is required. GeoRaster provides the following standard resampling methods:

- Nearest neighbor
- Bilinear interpolation using 4 neighboring cells
- Cubic convolution using 16 neighboring cells
- Average4 using 4 neighboring cells
- Average16 using 16 neighboring cells

The following subprograms (described in Chapter 4) are associated with GeoRaster support for pyramids:

- SDO_GEOR.generatePyramid generates pyramid data for a GeoRaster object.
- SDO_GEOR.deletePyramid deletes pyramid data for a GeoRaster object.

- SDO_GEOR.getPyramidMaxLevel returns the maximum pyramid level of a GeoRaster object.
- SDO_GEOR.getPyramidType returns the pyramid type for a GeoRaster object.

1.8 Compression and Decompression

GeoRaster provides two types of native compression to reduce storage space requirements for GeoRaster objects: JPEG (JPEG-B or JPEG-F) and DEFLATE. With both types, each block is compressed individually, as a distinct raster representation; and when a compressed GeoRaster object is decompressed, each block is decompressed individually.

Any GeoRaster operation that can be performed on a decompressed (uncompressed) GeoRaster object can also be performed on a compressed GeoRaster object. When GeoRaster performs an operation, if the source GeoRaster object is compressed, GeoRaster internally decompresses blocks of the source object as needed, performs the specified operation, and then compresses the resulting object in the format specified by the `compression` keyword or, if the `compression` keyword is not specified, in the source object's compression format. Therefore, you do not need to decompress compressed GeoRaster objects before performing certain operations, but you might gain some overall performance benefit if you decompress the objects before performing other operations.

Before a database user compresses or decompresses a GeoRaster object, ensure that the database has been created with a default temporary tablespace or that the user has been assigned a temporary tablespace or tablespace group. Otherwise, by default the `SYSTEM` tablespace is used for the temporary tablespace, and large temporary LOB data generated during GeoRaster operations are put in the `SYSTEM` tablespace, possibly affecting overall database performance. For information about managing temporary tablespaces, see *Oracle Database Administrator's Guide*.

To specify compression or decompression of a GeoRaster object, use the `compression` keyword in the `storageParam` parameter, which is described in Section 1.4.1. You can use the `compression` keyword in the `storageParam` parameter with several GeoRaster procedures, including SDO_GEOR.changeFormatCopy, SDO_GEOR.getRasterData, SDO_GEOR.getRasterSubset, SDO_GEOR.importFrom, SDO_GEOR.mosaic, SDO_GEOR.scaleCopy, and SDO_GEOR.subset. (There are no separate procedures for compressing and decompressing a GeoRaster object.)

If the source GeoRaster object is blank, the `compression` keyword is ignored, except for the SDO_GEOR.getRasterSubset and SDO_GEOR.getRasterData functions. That is, a blank GeoRaster object is never compressed, and the compression type in the metadata is always `NONE`. (Blank GeoRaster objects are explained in Section 1.4.3.)

This section covers the following topics:

- JPEG compression of GeoRaster objects (Section 1.8.1)
- DEFLATE compression of GeoRaster objects (Section 1.8.2)
- Decompression of GeoRaster objects (Section 1.8.3)

1.8.1 JPEG Compression of GeoRaster Objects

JPEG compression is supported only for GeoRaster objects with a `cellDepth` value of `8BIT_U` and no more than 4 bands per block, and each block must have 1 band, 3 bands, or 4 bands. (2 bands per block is not supported for JPEG compression.) You can

JPEG compress GeoRaster objects of more than 4 bands by reblocking the GeoRaster object with a band block size of 1, 3, or 4 bands.

Although JPEG compression is supported for GeoRaster objects of any size, the total size (`columnsPerBlock * rowsPerBlock * bandsPerBlock * cellDepth / 8`) of each block of the GeoRaster object must not exceed 50 megabytes (MB). For large GeoRaster objects, you can call the SDO_GEOR.changeFormatCopy procedure to block the GeoRaster object into blocks smaller than 50 MB, and then compress the GeoRaster object; or you can perform the blocking and compression in the same call to the SDO_GEOR.changeFormatCopy procedure.

GeoRaster supports the JPEG-B and JPEG-F compression modes:

- JPEG-B mode compresses objects in the abbreviated baseline JPEG format.
- JPEG-F mode compresses objects in the full-format baseline JPEG format.

Both modes are described in the CCITT Rec. T.81 JPEG specification (or ICO/IEC IS 10918-1). For both modes, GeoRaster uses the quantization table in Table K.2 of the CCITT Rec. T.81 JPEG specification and (for the Huffman tables) standard chrominance tables in Tables K.4 and K.6 of that specification. The quantization table is scaled by the compression quality before the table is applied to data during the compression process.

Because JPEG-B mode compression is an abbreviated format, the quantization and Huffman tables (which are required for decoding) are not included in the JPEG-B compressed data. These tables are included in JPEG-F compressed data.

The JPEG-B abbreviated format provides better compression than the JPEG-F format, and thus significantly reduces the storage space required if a large GeoRaster object is blocked into many smaller blocks. For example, for a remote sensing image compressed with a quality of 75, the average JPEG-B compression ratio might be about 20:1, while the average JPEG-F compression ratio might be about 2.5:1.

Both JPEG-B and JPEG-F are lossy compression formats. You can control the degree of loss with the `quality` keyword to the `storageParam` parameter. The `quality` keyword takes an integer value from 0 to 100. A value of 0 (zero) provides maximum compression, but causes substantial loss of data. A value of 75 (the GeoRaster default) provides an image that most people perceive as having no loss of quality, but that provides significant compression. A value of 100 provides the least compression, but the best quality.

1.8.2 DEFLATE Compression of GeoRaster Objects

DEFLATE compression compresses objects according to the Deflate Compressed Data Format Specification (Network Working Group RFC 1951), and it stores the compressed data in ZLIB format, as described in the ZLIB Compressed Data Format Specification (Network Working Group RFC 1950). The ZLIB header and checksum fields are included in the compressed GeoRaster object.

Although DEFLATE compression is supported for GeoRaster objects of any size, the total size (`columnsPerBlock * rowsPerBlock * bandsPerBlock * cellDepth / 8`) of each block of the GeoRaster object must not exceed 1 gigabyte (GB). For large GeoRaster objects, you can call the SDO_GEOR.changeFormatCopy procedure to block the GeoRaster object into blocks smaller than1 GB, and then compress the GeoRaster object; or you can perform the blocking and compression in the same call to the SDO_GEOR.changeFormatCopy procedure.

Because DEFLATE compression is lossless, compression quality does not apply, and is ignored if it is specified.

1.8.3 Decompression of GeoRaster Objects

You can decompress a compressed GeoRaster object in the database by specifying `compression=NONE` in the `storageParam` parameter. In this case, compression quality (if applicable) is derived from the metadata for JPEG-B compression or from the header data for JPEG-F compression; you should not specify compression quality as a storage parameter.

You can decompress a compressed GeoRaster object outside the database (that is, on the client side) by using an existing application programming interface (API), such as PL/SQL or the Oracle Call Interface (OCI), to retrieve the BLOB objects corresponding to the GeoRaster object's blocks, and decoding each compressed block individually according to the specifications of the relevant compression format. For example, if a GeoRaster object is compressed in JPEG-F mode, the decoding process should first parse the JPEG headers to retrieve the tables and block dimensions, and then apply Huffman decoding and dequantization to the image data.

Implementing JPEG decompression completely on your own is a complex, detail-oriented process. Depending on the application, it may be better to use an existing implementation. Libraries such as `jpeglib` in C and several imaging APIs in Java (for example, Sun Microsystems J2SE and JAI) already implement JPEG decompression, and you can adapt them to perform the decoding process on JPEG-compressed GeoRaster objects. You can apply essentially the same approach for DEFLATE compression using a ZLIB C library or Java API.

1.9 GeoRaster PL/SQL Subprogram Categories

GeoRaster provides the SDO_GEOR and SDO_GEOR_UTL PL/SQL packages, which contain subprograms (functions and procedures) to work with GeoRaster data and metadata. This section groups the subprograms into categories based on the types of actions they perform.

Chapter 4 and Chapter 5 contain detailed reference information about the subprograms in the SDO_GEOR and SDO_GEOR_UTL packages, respectively. The subprograms are presented in alphabetical order in those chapters.

1.9.1 Subprograms to Create, Load, and Export GeoRaster Data

Table 1–2 lists subprograms for creating, loading, and exporting GeoRaster data.

Table 1–2 Subprograms to Create, Load, and Export GeoRaster Data

Subprogram	Description
SDO_GEOR.init	Initializes an empty GeoRaster object, which will be registered by GeoRaster in the xxx_SDO_GEOR_SYSDATA views.
SDO_GEOR.createBlank	Creates a blank GeoRaster object, in which all cells have the same value.
SDO_GEOR.copy	Makes a copy of an existing GeoRaster object.
SDO_GEOR.importFrom	Imports an image file or BLOB object into a GeoRaster object stored in the database.
SDO_GEOR.exportTo	Exports a GeoRaster object or a subset of a GeoRaster object to a file or to a BLOB object.

1.9.2 Subprograms to Validate and Process GeoRaster Objects

Table 1–3 lists subprograms for validating and processing GeoRaster objects.

Table 1–3 Subprograms to Validate and Process GeoRaster Objects

Subprogram	Description
SDO_GEOR.validateGeoraster	Validates a GeoRaster object.
SDO_GEOR.schemaValidate	Validates a GeoRaster object's metadata against the GeoRaster XML schema.
SDO_GEOR.generateSpatialExtent	Generates a Spatial geometry that contains the spatial extent of the GeoRaster object.
SDO_GEOR.generatePyramid	Generates pyramid data for a GeoRaster object, which is stored together with the original data.
SDO_GEOR.deletePyramid	Deletes the pyramid data of a GeoRaster object.
SDO_GEOR.subset	Performs either or both of the following operations: (1) spatial crop, cut, or clip, or (2) layer or band subset.
SDO_GEOR.scale (deprecated)	Scales a GeoRaster object by enlarging or reducing the image along row and column dimensions.
SDO_GEOR.scaleCopy	Scales a GeoRaster object by enlarging or reducing the image along row and column dimensions, and puts the result into a new object that reflects the scaling.
SDO_GEOR.changeFormat (deprecated)	Changes the storage format of an existing GeoRaster object (for example, changing the blocking, cell depth, or interleaving).
SDO_GEOR.changeFormatCopy	Makes a copy of an existing GeoRaster object using a different storage format (for example, changing the blocking, cell depth, or interleaving).
SDO_GEOR.georeference	Georeferences a GeoRaster object using specified cell-to-model transformation coefficients.
SDO_GEOR.mosaic	Mosaics GeoRaster objects into one GeoRaster object.

1.9.3 Subprograms to Get and Set GeoRaster Metadata and Data

Many functions, most of whose names start with *get*, return metadata information about GeoRaster objects (and sometimes cell or pixel values). These functions can be further divided into subcategories according to the kind of metadata or data each retrieves related to each of the following:

- Whole object
- Cell coordinates and values
- Cell metadata
- Spatial reference system
- Date and time the data was collected
- Imagery band spectral information
- Layers
- Pyramids

For many of the *get* functions, there is a corresponding procedure, whose name starts with *set*, to set, modify, or delete the value of a metadata attribute. For most *set* procedures, to delete the value of the metadata attribute that the procedure is designed to modify, specify a null value for the attribute. For example, to delete the bin table for a layer of a GeoRaster object, call the SDO_GEOR.setBinTable procedure and specify a null `tableName` parameter. However, in most cases you cannot specify a null value

for other related attributes. For example, you cannot specify a null `layerNumber` parameter in a call to the SDO_GEOR.setBinTable procedure.

GeoRaster automatically validates the GeoRaster object after any *set* procedure completes.

Note: Most GeoRaster metadata can also be retrieved and changed using XMLType PL/SQL interfaces, such as `existsNode` and `extract`. However, if a GeoRaster *get* or *set* subprogram exists for the metadata attribute you want to retrieve or change, use the GeoRaster subprogram instead of an XMLType interface, because the GeoRaster subprograms validate any changes before they are made.

Table 1–4 lists subprograms for getting and setting information about the whole GeoRaster object.

Table 1–4 Subprograms to Get and Set Whole Object Metadata

Subprogram	Description
SDO_GEOR.getID	Returns the user-defined identifier value associated with a GeoRaster object.
SDO_GEOR.setID	Sets a user-defined identifier to be associated with a GeoRaster object, or deletes the existing value if you specify a null `id` parameter.
SDO_GEOR.getVersion	Returns the user-specified version of a GeoRaster object.
SDO_GEOR.setVersion	Sets the user-specified version of a GeoRaster object.
SDO_GEOR.setRasterType	Sets the raster type of a GeoRaster object.
SDO_GEOR.isSpatialReferenced	Returns `TRUE` if the GeoRaster object is spatially referenced, or `FALSE` if the GeoRaster object is not spatially referenced.
SDO_GEOR.setSpatialReferenced	Specifies whether or not a GeoRaster object is spatially referenced, or deletes the existing value if you specify a null `isReferenced` parameter.
SDO_GEOR.isBlank	Returns `TRUE` if the GeoRaster object is a blank GeoRaster object, or `FALSE` if the GeoRaster object is not a blank GeoRaster object.
SDO_GEOR.getBlankCellValue	Returns the cell value to be used for all cells if a specified GeoRaster image is a blank GeoRaster object.
SDO_GEOR.setBlankCellValue	Sets the cell value to be used for all cells if a specified GeoRaster object is a blank GeoRaster object.
SDO_GEOR.getDefaultColorLayer	Returns the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object.
SDO_GEOR.setDefaultColorLayer	Sets the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object, or deletes the existing values if you specify a null `defaultRGB` parameter.
SDO_GEOR.getDefaultRed	Returns the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object.

Table 1–4 (Cont.) Subprograms to Get and Set Whole Object Metadata

Subprogram	Description
SDO_GEOR.setDefaultRed	Sets the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultRed` parameter.
SDO_GEOR.getDefaultGreen	Returns the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object.
SDO_GEOR.setDefaultGreen	Sets the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultGreen` parameter.
SDO_GEOR.getDefaultBlue	Returns the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object.
SDO_GEOR.setDefaultBlue	Sets the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultBlue` parameter.
SDO_GEOR.getInterleavingType	Returns the interleaving type for a GeoRaster object.
SDO_GEOR.getSpatialDimNumber	Returns the number of spatial dimensions of a GeoRaster object.
SDO_GEOR.getSpatialDimSizes	Returns the number of cells in each spatial dimension of a GeoRaster object.
SDO_GEOR.getBandDimSize	Returns the number of bands in a GeoRaster object.
SDO_GEOR.getLayerDimension	Returns the dimension that is mapped as the logical layer dimension of a GeoRaster object.
SDO_GEOR.getTotalLayerNumber	Returns the total number of layers in a GeoRaster object.

Table 1–5 lists subprograms for getting and setting information about cell coordinates and values.

Table 1–5 Subprograms to Get and Set Cell Coordinates and Values

Subprogram	Description
SDO_GEOR.getModelCoordinate	Returns the coordinates in the model (ground) coordinate system associated with the point at the specified cell (raster) coordinates.
SDO_GEOR.getCellCoordinate	Returns the coordinates in the cell (raster) coordinate system associated with the point at the specified model (ground) coordinates.
SDO_GEOR.getCellValue	Returns the value of a single cell located anywhere in the GeoRaster object by specifying its row, column, and band number in its cell coordinate system, or by specifying a point geometry in its model coordinate system and its logical layer number.
SDO_GEOR.changeCellValue	Changes the value of raster data cells in a specified window of a GeoRaster object.
SDO_GEOR.getRasterSubset	Creates a single BLOB object containing all cells of a specified pyramid level that are inside or on the boundary of a specified window.

Table 1–5 (Cont.) Subprograms to Get and Set Cell Coordinates and Values

Subprogram	Description
SDO_GEOR.getRasterBlocks	Returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that have any spatial interaction with a specified window.
SDO_GEOR.getRasterData	Creates a single BLOB object that contains all raster data of the input GeoRaster object at the specified pyramid level.
SDO_GEOR.getCellDepth	Returns the cell depth in bits.
SDO_GEOR.getNODATA	Returns the value representing NODATA cells in a GeoRaster object.
SDO_GEOR.getULTCoordinate	Returns the cell coordinates of the upper-left corner of a GeoRaster object.
SDO_GEOR.setULTCoordinate	Sets the cell coordinate values of the upper-left corner of a GeoRaster object, or deletes the existing values if you specify a null `ultCoord` parameter.
SDO_GEOR.getCompressionType	Returns the compression type for a GeoRaster object.
SDO_GEOR.getBlockingType	Returns the blocking type for a GeoRaster object.
SDO_GEOR.getBlockSize	Returns the number of cells for each dimension in each block of a GeoRaster object in an array showing the number of cells for each row, column, and (if relevant) band.

Table 1–6 lists subprograms for getting and setting information about the GeoRaster spatial reference system.

Table 1–6 Subprograms to Get and Set Spatial Reference System Metadata

Subprogram	Description
SDO_GEOR.getSRS	Returns an object of type SDO_GEOR_SRS containing information related to the spatial referencing of a GeoRaster object.
SDO_GEOR.setSRS	Sets the spatial reference information of a GeoRaster object, or deletes the existing information if you specify a null `srs` parameter.
SDO_GEOR.isRectified	Returns `TRUE` if the GeoRaster object is identified as rectified, or `FALSE` if the GeoRaster object is not identified as rectified.
SDO_GEOR.setRectified	Specifies whether or not a GeoRaster object is rectified, or deletes the existing value if you specify a null `isRectified` parameter.
SDO_GEOR.isOrthoRectified	Returns `TRUE` if the GeoRaster object is identified as orthorectified, or `FALSE` if the GeoRaster object is not identified as orthorectified.
SDO_GEOR.setOrthoRectified	Specifies whether or not a GeoRaster object is orthorectified, or deletes the existing value if you specify a null `isOrthoRectified` parameter.
SDO_GEOR.getModelSRID	Returns the coordinate system (SDO_SRID value) associated with the model (ground) space for a GeoRaster object.

Table 1–6 (Cont.) Subprograms to Get and Set Spatial Reference System Metadata

Subprogram	Description
SDO_GEOR.setModelSRID	Sets the coordinate system (SDO_SRID value) for the model (ground) space for a GeoRaster object, or deletes the existing value if you specify a null `srid` parameter.
SDO_GEOR.getSpatialResolutions	Returns the spatial resolution value along each spatial dimension of a GeoRaster object.
SDO_GEOR.setSpatialResolutions	Sets the spatial resolution value along each spatial dimension of a GeoRaster object, or deletes the existing values if you specify a null `resolutions` parameter.

Table 1–7 lists subprograms for getting and setting information about the date and time that the GeoRaster data was collected.

Table 1–7 Subprograms to Get and Set Date and Time Metadata

Subprogram	Description
SDO_GEOR.getBeginDateTime	Returns the beginning date and time for raster data collection in the metadata for a GeoRaster object.
SDO_GEOR.setBeginDateTime	Sets the beginning date and time for raster data collection in the metadata for a GeoRaster object, or deletes the existing value if you specify a null `beginTime` parameter.
SDO_GEOR.getEndDateTime	Returns the ending date and time for raster data collection in the metadata for a GeoRaster object.
SDO_GEOR.setEndDateTime	Sets the ending date and time for raster data collection in the metadata for a GeoRaster object, or deletes the existing value if you specify a null `endTime` parameter.

Table 1–8 lists subprograms for getting and setting information about the imagery band reference system.

Table 1–8 Subprograms to Get and Set Imagery Band System Metadata

Subprogram	Description
SDO_GEOR.getSpectralResolution	Returns the spectral resolution of a GeoRaster object if it is a hyperspectral or multiband image.
SDO_GEOR.setSpectralResolution	Sets the spectral resolution of a GeoRaster object if it is a hyperspectral or multiband image, or deletes the existing value if you specify a null `resolution` parameter.
SDO_GEOR.getSpectralUnit	Returns the unit of measurement for identifying the wavelength interval for a band.
SDO_GEOR.setSpectralUnit	Sets the unit of measurement for identifying the wavelength interval for a band, or deletes the existing value if you specify a null `unit` parameter.

Table 1–9 lists subprograms for getting and setting information about layers (or bands).

Table 1–9 Subprograms to Get and Set Layer Metadata

Subprogram	Description
SDO_GEOR.getScaling	Returns the coefficients of the scaling function for a layer of a GeoRaster object.

Table 1–9 (Cont.) Subprograms to Get and Set Layer Metadata

Subprogram	Description
SDO_GEOR.setScaling	Sets the scaling function associated with a layer, or deletes the existing value if you specify a null `scalingFunc` parameter.
SDO_GEOR.getBinTable	Returns the name of the bin table associated with a layer.
SDO_GEOR.setBinTable	Sets the name of the bin table associated with a layer, or deletes the existing value if you specify a null `tableName` parameter.
SDO_GEOR.getBinType	Returns the bin type associated with a layer.
SDO_GEOR.getStatistics	Returns statistical data associated with a layer.
SDO_GEOR.setStatistics	Sets statistical data associated with a layer.
SDO_GEOR.getHistogram	Returns the histogram for a layer.
SDO_GEOR.getHistogramTable	Returns the histogram table for a layer.
SDO_GEOR.setHistogramTable	Sets the histogram table for a layer.
SDO_GEOR.hasGrayScale	Checks if a layer of a GeoRaster object has grayscale information.
SDO_GEOR.getGrayScale	Returns the grayscale mappings for a layer.
SDO_GEOR.getGrayScaleTable	Returns the grayscale mapping table for a layer.
SDO_GEOR.setGrayScale	Sets the grayscale mappings for a layer in a GeoRaster object, or deletes the existing values if you specify a null `grayScale` parameter.
SDO_GEOR.setGrayScaleTable	Sets the grayscale mapping table for a layer in a GeoRaster object, or deletes the existing value if you specify a null `tableName` parameter.
SDO_GEOR.hasPseudoColor	Checks if a layer has pseudocolor information.
SDO_GEOR.getColorMap	Returns the colormap for pseudocolor display of a layer.
SDO_GEOR.getColorMapTable	Returns the colormap table for pseudocolor display of a layer.
SDO_GEOR.setColorMap	Sets the colormap for a layer in a GeoRaster object, or deletes the existing value if you specify a null `colorMap` parameter.
SDO_GEOR.setColorMapTable	Sets the colormap table for a layer in a GeoRaster object, or deletes the existing value if you specify a null `tableName` parameter.
SDO_GEOR.getVAT	Returns the name of the value attribute table (VAT) associated with a layer.
SDO_GEOR.setVAT	Sets the name of the value attribute table (VAT) associated with a layer of a GeoRaster object, or deletes the existing value if you specify a null `vatName` parameter.
SDO_GEOR.getLayerOrdinate	Returns the band ordinate for a layer in a GeoRaster object.
SDO_GEOR.setLayerOrdinate	Sets the band ordinate value for a specified layer in a GeoRaster object, or deletes the existing value if you specify a null `ordinate` parameter.
SDO_GEOR.getLayerID	Returns the user-defined identifier value associated with a layer in a GeoRaster object.

Table 1–9 (Cont.) Subprograms to Get and Set Layer Metadata

Subprogram	Description
SDO_GEOR.setLayerID	Sets a user-defined identifier to be associated with a layer in a GeoRaster object, or deletes the existing value if you specify a null `id` parameter.

Table 1–10 lists subprograms for getting information about pyramids.

Table 1–10 Subprograms to Get Pyramid Metadata

Subprogram	Description
SDO_GEOR.getPyramidType	Returns the pyramid type for a GeoRaster object.
SDO_GEOR.getPyramidMaxLevel	Returns the level number of the top pyramid of a GeoRaster object.

1.9.4 Utility Subprograms

Table 1–11 lists the utility subprograms in the SDO_GEOR_UTL package.

Table 1–11 Subprogram for GeoRaster Triggers

Subprogram	Description
SDO_GEOR_UTL.createDMLTrigger	Creates the standard GeoRaster data manipulation language (DML) trigger on a GeoRaster column in a GeoRaster table, so that the appropriate operations are performed when its associated trigger is fired. (For information about using triggers with GeoRaster, see Section 3.1.1.)
SDO_GEOR_UTL.makeRDTNamesUnique	Renames all existing raster data tables that do not have unique names so that all names are unique, and updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.
SDO_GEOR_UTL.renameRDT	Renames one or more existing raster data tables, and updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.

1.10 GeoRaster Tools: Viewer, Loader, Exporter

GeoRaster includes several client-side tools. If you installed the demo files from the Companion CD, these tools are in the following directory (assuming the default Spatial installation directory of *$ORACLE_HOME*/md):

```
$ORACLE_HOME/md/demos/georaster/java
```

This directory contains a file named `README`, which includes helpful usage information, as well as instructions for using the following tools:

- GeoRaster viewer, which displays GeoRaster objects and metadata, as well as raster images, in a Java viewer application. You can connect to multiple databases simultaneously, and see the GeoRaster objects from each database listed in the left pane. You can quickly switch among views at various resolutions, from the original image (pyramid level 0) to the overview (highest pyramid level). You can perform image enhancement, such as linear stretch (automatic, manual, or piecewise), normalization, equalization, and controls for brightness, contrast, and

threshold. (For more information about viewing GeoRaster objects, see Section 3.14.)

In the viewer, you can call the GeoRaster loader and exporter tools, thus enabling you to use a single tool as an interface to the capabilities of all the GeoRaster tools. The loader and exporter tools are described in this section and in the `README` file.

- GeoRaster loader, which loads raster data into the GeoRaster objects. You can use this as an alternative to the SDO_GEOR.importFrom procedure, which is documented in Chapter 4. However, on non-Windows systems this loader tool does not support the BMP or GIF image formats, so you must use the SDO_GEOR.importFrom procedure with these formats on non-Windows systems. This tool does not support raster data that has a cell depth value of `2BIT`, or source multiband raster data with BIL or BSQ interleaving types. The imported GeoRaster object has the BIP interleaving type. The loading operation of this tool cannot be rolled back.

  When an image in JPEG file format is loaded, the amount of memory required for the operation depends on the size of the uncompressed image, and can be specified as a command line parameter using the `-Xmx` option (for example, `java -Xmx256M oracle.spatial.georaster.tools.GeoRasterLoader ...`).

- GeoRaster exporter, which exports GeoRaster objects to image files. You can use this as an alternative to the SDO_GEOR.exportTo procedure, which is documented in Chapter 4. The GeoRaster exporter tool does not support GIF as a destination file format; the SDO_GEOR.exportTo procedure does not support GIF or JPEG as a destination file format. The GeoRaster exporter tool does not support GeoRaster objects that have a `cellDepth` value of `2BIT`. GeoRaster objects with a cell depth of 8 bits or greater that have a BSQ or BIL interleaving are exported in BIP interleaved format. For information about limits on the amount of GeoRaster data that can be exported in a single operation, see the Usage Notes for the SDO_GEOR.exportTo procedure.

These tools are developed in Java, so you can run them anywhere through an intranet or the Internet, as long as you establish a network connection with the Oracle database.

GeoRaster can load and export the following image formats: TIF/GeoTIFF, (georeferencing information), JPEG, BMP, GIF, and PNG. Georeferencing information can be loaded from and exported to ESRI world files, but georeferencing information in GeoTIFF imagery is ignored during loading and exporting. Some restrictions regarding image size and types may apply; see the `README` file for the GeoRaster tools.

After raster or image files are loaded into GeoRaster objects, the data is completely stored in the native GeoRaster object data type and is independent from any specific file formats.

If you want to create your own GeoRaster loader and exporter tools, you can develop them using OCI, Oracle C++ Call Interface (OCCI), or Java, and you can implement them as client-side commands or server-side SQL procedures or functions.

1.11 GeoRaster PL/SQL Demo Files

GeoRaster includes several PL/SQL demo files that show common operations. If you installed the demo files from the Companion CD, these files are in the following directory under the Spatial installation directory (which by default is *`$ORACLE_HOME`*`/md`):

/demo/georaster/plsql

The README file in that directory introduces the PL/SQL files in the directory, which are listed in Table 1–12.

Table 1–12 GeoRaster PL/SQL Demo Files

File	Description
georaster_demo.sql	Master file that invokes each of the other files (in the sequence listed in this table).
create_georaster_table.sql	Creates a GeoRaster table, the DML trigger on the GeoRaster table, and some raster data tables.
import_georaster.sql	Initializes empty GeoRaster objects, and imports TIFF images into the GeoRaster objects.
generate_georaster.sql	Generates GeoRaster objects for use by the demo files.
validate_georaster.sql	Validates a column of GeoRaster objects, and validates a GeoRaster object against the GeoRaster metadata XML schema.
query_georaster.sql	Shows many query operations on GeoRaster metadata and data.
ccf_georaster.sql	Copies GeoRaster objects, with and without changes to the format.
subset_georaster.sql	Subsets (crops) GeoRaster objects.
scale_georaster.sql	Scales (enlarges or shrinks) GeoRaster objects.
pyramid_georaster.sql	Generates pyramids for a GeoRaster object and queries pyramid metadata.
drop_georaster_table.sql	Drops the GeoRaster table and the raster data tables.

1.12 README File for Spatial and Related Features

A README.txt file supplements the information in the following manuals: *Oracle Spatial User's Guide and Reference*, *Oracle Spatial GeoRaster* (this manual), and *Oracle Spatial Topology and Network Data Models*. This file is located at:

$ORACLE_HOME/md/doc/README.txt

2

GeoRaster Data Types and Related Structures

The object-relational implementation of GeoRaster consists of a set of object data types for storing data and system data. Each image or gridded raster data is stored in a column of type SDO_GEORASTER, and the blocks in that raster data are stored in a raster data table of type SDO_RASTER, as explained and illustrated in Section 1.4. This chapter contains the following major sections:

- Section 2.1, "SDO_GEORASTER Object Type"
- Section 2.2, "SDO_RASTER Object Type and the Raster Data Table"
- Section 2.3, "Other GeoRaster Types"
- Section 2.4, "GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)"
- Section 2.5, "GeoRaster XML Schema Table"

2.1 SDO_GEORASTER Object Type

In the GeoRaster object-relational model, a raster image or grid object is stored in a single row, in a single column of object type SDO_GEORASTER in a user-defined table. Tables with one or more columns of type SDO_GEORASTER are referred to as GeoRaster tables.

The SDO_GEORASTER object type is defined as:

```
CREATE TYPE sdo_georaster AS OBJECT (
 rasterType       NUMBER,
 spatialExtent    SDO_GEOMETRY,
 rasterDataTable  VARCHAR2(32),
 rasterID         NUMBER,
 metadata         XMLType);
```

The sections that follow describe the semantics of each SDO_GEORASTER attribute.

2.1.1 rasterType Attribute

The `rasterType` attribute must be a 5-digit number in the format `[d][b][t][gt]`, where:

- `[d]` identifies the number of spatial dimensions. Must be 2 for the current release.
- `[b]` indicates band or layer information: 0 means one band or layer; 1 means one or more than one band or layer. Note that you are *not* specifying the total number of bands or layers in this field. (For information about bands and layers, see Section 1.5.)

- `[t]` is reserved for future use and should be specified as 0 (zero).
- `[gt]` identifies the 2-digit GeoRaster type, and must be one of the following values:

[gt] Value	Meaning
00	Reserved for Oracle use.
01	Any GeoRaster type. This is the only value supported for the current release. This value causes GeoRaster not to apply any restrictions associated with specific types that might be implemented in future releases.
02-50	Reserved for Oracle use.
51-99	Reserved for customer use in future releases.

For example, a RasterType value of 20001 means:

- Two-dimensional data
- One band (layer)
- Any GeoRaster type

2.1.2 spatialExtent Attribute

The `spatialExtent` attribute identifies the **spatial extent**, or *footprint*, associated with the raster data. The spatial extent is an Oracle Spatial geometry of type SDO_GEOMETRY. The spatial extent geometry can be in any coordinate system; however, it is in the model (ground) space of the GeoRaster object if the GeoRaster object is georeferenced and if you generate the spatial extent geometry using any of the following methods: calling the SDO_GEOR.generateSpatialExtent function, or specifying `spatialExtent=TRUE` as a storage parameter to the SDO_GEOR.importFrom procedure or the GeoRaster client-side loader (described in Section 1.10). The spatial extent is set to null, rather than cell space, if its SRID value is null or 0 (zero). The SDO_GEOMETRY data type is described in *Oracle Spatial User's Guide and Reference*.

Because of the potential performance benefits of spatial indexing for GeoRaster applications, the geometry is associated with the `spatialExtent` attribute, rather than being included in the XML `metadata` attribute described in Section 2.1.5. For information about indexing GeoRaster data, see Section 3.8.

2.1.3 rasterDataTable Attribute

The `rasterDataTable` attribute identifies the name of the raster data table. The raster data table must be an object table of type SDO_RASTER. It contains a row of type SDO_RASTER for each raster block that is stored. You must create and (if necessary) drop the raster data table. You should never modify the rows in this table directly, but you can query this table to access the raster data.

For more information about the raster data table and the SDO_RASTER type, see Section 2.2.

2.1.4 rasterID Attribute

The `rasterID` attribute value is stored in the rows of the raster data table to identify which rows belong to the GeoRaster object. The `rasterDataTable` attribute and `rasterID` attribute together uniquely identify the GeoRaster object in the database.

That is, each GeoRaster object has a raster data table, although a raster data table can contain data from multiple GeoRaster objects.

You can specify the `rasterID` and `rasterDataTable` attributes for new GeoRaster objects, as long as each pair is unique in the database. If you do not specify these values, they are automatically generated by the SDO_GEOR.init and SDO_GEOR.createBlank functions.

2.1.5 metadata Attribute

The `metadata` attribute contains the GeoRaster metadata that is defined by Oracle. The metadata is described by the GeoRaster metadata XML schema, which is documented in Appendix A. The metadata of any GeoRaster object must be validated against this XML schema, and it must also be validated using the SDO_GEOR.validateGeoraster function, which imposes additional restrictions not defined by this XML schema.

2.2 SDO_RASTER Object Type and the Raster Data Table

In the GeoRaster object-relational model, a raster data table is used to store all cell data in a raster image. The cell data of a GeoRaster object is blocked, and each block is stored in the raster data table as one row. You specify this table in the `rasterDataTable` attribute of the SDO_GEORASTER object, as explained in Section 2.1.3. You must create the raster data table before you store any cell data in it.

The raster data table is an object table, defined as a table of SDO_RASTER object type. The SDO_RASTER object type is defined as:

```
CREATE TYPE sdo_raster AS OBJECT (
 rasterID           NUMBER,
 pyramidLevel       NUMBER,
 bandBlockNumber    NUMBER,
 rowBlockNumber     NUMBER,
 columnBlockNumber  NUMBER,
 blockMBR           SDO_GEOMETRY,
 rasterBlock        BLOB);
```

The sections that follow describe the semantics of each SDO_RASTER attribute.

2.2.1 rasterID Attribute

The `rasterID` attribute in the SDO_RASTER object must be a number that matches the `rasterID` value in its associated SDO_GEORASTER object. (The `rasterID` attribute of the SDO_GEORASTER object is described in Section 2.1.4.) The matching of these numbers identifies the raster block as belonging to a specific GeoRaster object.

2.2.2 pyramidLevel Attribute

The `pyramidLevel` attribute identifies the pyramid level for this block of cells. The pyramid level is 0 or any positive integer. Pyramid levels are used to create reduced resolution images that require less storage space. A pyramid level of 0 indicates the original raster data; that is, there is no reduction in the image resolution and no change in the storage space required. Values greater than 0 (zero) indicate increasingly reduced levels of image resolution and reduced storage space requirements. For more information about pyramids, see Section 1.7.

2.2.3 bandBlockNumber Attribute

The `bandBlockNumber` attribute identifies the block number along the band dimension. (For information about bands and layers, see Section 1.5.)

2.2.4 rowBlockNumber Attribute

The `rowBlockNumber` attribute identifies the block number along the row dimension.

2.2.5 columnBlockNumber Attribute

The `columnBlockNumber` attribute identifies the block number along the column dimension.

2.2.6 blockMBR Attribute

The `blockMBR` attribute is the geometry (of type SDO_GEOMETRY) for the minimum bounding rectangle (MBR) for this block. The geometry is in cell space (that is, its SRID value is null), and all ordinates are integers. The ordinates represent the minimum row and column and the maximum row and column stored in this block.

2.2.7 rasterBlock Attribute

The `rasterBlock` attribute contains all raster cell data for this block. The `rasterBlock` attribute is of type BLOB.

2.3 Other GeoRaster Types

In addition to SDO_GEORASTER (described in Section 2.1) and SDO_RASTER (described in Section 2.2), GeoRaster provides several other object and collection types, which are used for specific kinds of operations. Unlike the SDO_GEORASTER and SDO_RASTER types, which are used for storage in the database (for example, to define a column in a table), the types described in this section are used only with the GeoRaster PL/SQL API in the current release.

2.3.1 SDO_GEOR_HISTOGRAM Object Type

The SDO_GEOR_HISTOGRAM object type is used to contain the histogram data of a GeoRaster object or a layer. Each cell has a value, and for each cell value there may be any number of cells having that value. The histogram contains the cell values and the total number of cells related to each cell value.

The SDO_GEOR_HISTOGRAM object type is defined as:

```
CREATE TYPE sdo_geor_histogram AS OBJECT(
  cellValue  SDO_NUMBER_ARRAY,
  count      SDO_NUMBER_ARRAY);
```

Table 2–1 describes the attributes of the SDO_GEOR_HISTOGRAM object type. The `cellValue` array and the `count` array must have the same length.

Table 2–1 SDO_GEOR_HISTOGRAM Object Type Attributes

Attribute	Description
cellValue	Array of cell values.
count	Number of cells that correspond to each value in `cellValue`.

2.3.2 SDO_GEOR_COLORMAP Object Type

The SDO_GEOR_COLORMAP object type contains colormap information, that is, pseudocolor information for identifying the red, green, blue, and (optionally) alpha values of the color to be used to display cells that have a specific value. The colormap is also called the pseudocolor table or the palette table. The colormap in GeoRaster is in the default sRGB ColorSpace, which is a proposed standard RGB color space, as explained at

http://www.w3.org/pub/WWW/Graphics/Color/sRGB.html

The ranges for red, green, blue, and alpha values are all scaled to be 8-bit unsigned integers from 0 to 255.

Alpha is also called opacity. An alpha value of 255 means that the color is completely opaque, and an alpha value of 0 means that the color is completely transparent. The color component values are never premultiplied by the alpha value.

The SDO_GEOR_COLORMAP object type is defined as:

```
CREATE TYPE sdo_geor_colormap AS OBJECT(
  cellValue  SDO_NUMBER_ARRAY,
  red        SDO_NUMBER_ARRAY,
  green      SDO_NUMBER_ARRAY,
  blue       SDO_NUMBER_ARRAY,
  alpha      SDO_NUMBER_ARRAY);
```

Table 2–2 describes the attributes of the SDO_GEOR_COLORMAP object type. Each attribute is an array of numbers. The arrays must have the same length, and the values of the same index in each array must correspond to each other. Each `cellValue` value must be consistent with the `cellDepth` value of the GeoRaster object.

Table 2–2 SDO_GEOR_COLORMAP Object Type Attributes

Attribute	Description
cellValue	Array of cell values.
red	Array of red component values for pseudocolor display of cells that have the values in `cellValue`. Must be integer values from 0 to 255.
green	Array of green component values for pseudocolor display of cells that have the values in `cellValue`. Must be integer values from 0 to 255.
blue	Array of blue component values for pseudocolor display of cells that have the values in `cellValue`. Must be integer values from 0 to 255.
alpha	Array of alpha component values for pseudocolor display of cells that have the values in `cellValue`. Must be integer values from 0 to 255.

2.3.3 SDO_GEOR_GRAYSCALE Object Type

The SDO_GEOR_GRAYSCALE object type contains grayscale information for identifying the grayscale value to be used to display cells that have a specific value. The grayscale table cell values can be "stretched" in linear proportion using this grayscale table, so that the original raster data can be properly displayed. The grayscale table value range is 8-bit unsigned integer values from 0 to 255. The grayscale table is also called the contrast table or the lookup table.

The SDO_GEOR_GRAYSCALE object type is defined as:

```
CREATE TYPE sdo_geor_grayscale AS OBJECT(
  cellValue  SDO_NUMBER_ARRAY,
  gray       SDO_NUMBER_ARRAY);
```

Table 2–3 describes the attributes of the SDO_GEOR_GRAYSCALE object type. The `cellValue` array and the `gray` array must have the same length. Each `cellValue` value must be consistent with the `cellDepth` value of the GeoRaster object.

Table 2–3 SDO_GEOR_GRAYSCALE Object Type Attributes

Attribute	Description
cellValue	Array of cell values.
gray	Array of gray component values for grayscale display of cells that have the values in `cellValue`. Must be integer values from 0 to 255.

2.3.4 SDO_RASTERSET Collection Type

The SDO_RASTERSET collection type is used as the return type of table functions that query the raster data blocks (one or many blocks, the whole set or a subset).

The SDO_RASTERSET collection type is defined as:

```
CREATE TYPE sdo_rasterset AS TABLE of SDO_RASTER;
```

The SDO_RASTER type is described in Section 2.2.

2.3.5 SDO_GEOR_SRS Object Type

The SDO_GEOR_SRS object type is used to contain information related to the spatial referencing of a GeoRaster object.

The SDO_GEOR_SRS object type is defined as:

```
CREATE TYPE sdo_geor_srs AS OBJECT (
   isReferenced       VARCHAR2(5),
   isRectified        VARCHAR2(5),
   isOrthoRectified   VARCHAR2(5),
   srid               NUMBER,
   spatialResolution  SDO_NUMBER_ARRAY,
   spatialTolerance   NUMBER,
   coordLocation      NUMBER,
   rowOff             NUMBER,
   columnOff          NUMBER,
   xOff               NUMBER,
   yOff               NUMBER,
   zOff               NUMBER,
   rowScale           NUMBER,
   columnScale        NUMBER,
   xScale             NUMBER,
   yScale             NUMBER,
   zScale             NUMBER,
   rowRMS             NUMBER,
   columnRMS          NUMBER,
   totalRMS           NUMBER,
   rowNumerator       SDO_NUMBER_ARRAY,
   rowDenominator     SDO_NUMBER_ARRAY,
   columnNumerator    SDO_NUMBER_ARRAY,
   columnDenominator  SDO_NUMBER_ARRAY);
```

Table 2–4 describes the attributes of the SDO_GEOR_SRS object type.

Table 2–4 SDO_GEOR_SRS Object Type Attributes

Attribute	Description
isReferenced	`TRUE` if the GeoRaster object is georeferenced; `FALSE` if the GeoRaster object is not georeferenced.
isRectified	`TRUE` if the GeoRaster object is both georectified and georeferenced; `FALSE` if the GeoRaster object is not georectified.
isOrthoRectified	`TRUE` if the GeoRaster object is orthorectified, georectified, and georeferenced; `FALSE` if the GeoRaster object is not orthorectified.
srid	SRID value of the model (ground) coordinate system.
spatialResolution	Spatial resolution values: an array of numeric values, one for each spatial dimension. Each value indicates the number of units of measurement associated with the data area represented by that spatial dimension of a cell.
spatialTolerance	Tolerance value. (For an explanation of tolerance, see *Oracle Spatial User's Guide and Reference*.)
coordLocation	The model coordinate location representing either the upper-left corner or the center of each cell in the model space when cell coordinates (integer numbers) are converted to model coordinates (double numbers).
rowOff	Must be 0 (zero) for the current release.
columnOff	Must be 0 (zero) for the current release.
xOff	Must be 0 (zero) for the current release.
yOff	Must be 0 (zero) for the current release.
zOff	Must be 0 (zero) for the current release.
rowScale	Must be 1 for the current release.
columnScale	Must be 1 for the current release.
xScale	Must be 1 for the current release.
yScale	Must be 1 for the current release.
zScale	Must be 1 for the current release.
rowRMS	Must be NULL for the current release.
columnRMS	Must be NULL for the current release.
totalRMS	Must be NULL for the current release.
rowNumerator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the numerator of the row polynomial, where pType=1, nVars=2, order=1, and nCoefficients=3. The three coefficients are `a`, `b`, `c` in the formulas in Section 1.6.1.
rowDenominator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the denominator of the row polynomial, where pType=1, nVars=0, order=0, and nCoefficients=1. The value of the single coefficient must be 1 for the current release.
columnNumerator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the numerator of the column polynomial, where pType=1, nVars=2, order=1, and nCoefficients=3. The three coefficients are `d`, `e`, `f` in the formulas in Section 1.6.1.
columnDenominator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the denominator of the column polynomial, where pType=1, nVars=0, order=0, and nCoefficients=1. The value of the single coefficient must be 1 for the current release.

2.4 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)

GeoRaster uses a system data table to maintain the relationship between GeoRaster tables and their related raster data tables. Each GeoRaster object (if it is not null) has a related raster data table, and it might have other tables, such as ground control point (GCP) tables and value attribute tables (VATs).

For a given user, the raster data table name plus the `rasterID` uniquely identify a GeoRaster object. It is possible for many GeoRaster objects (each with a different `rasterID` value) in one GeoRaster table to share one raster data table.

Whenever a new GeoRaster object (including empty and blank GeoRaster objects) is created, a raster data table is assigned to it and a `rasterID` value is assigned. All SDO_GEORASTER objects (except atomic null objects) are automatically recorded in the system data table when they are created.

The GeoRaster system data table is under the MDSYS schema. Most of the information in the GeoRaster system data table is available for retrieval through system data views. Each GeoRaster user has the following system data views available in the schema associated with that user:

- USER_SDO_GEOR_SYSDATA contains system data for all GeoRaster objects owned by the current user.
- ALL_SDO_GEOR_SYSDATA contains system data for all GeoRaster objects accessible by the current user.

The GeoRaster system data table and the USER_SDO_GEOR_SYSDATA and ALL_SDO_GEOR_SYSDATA views should never be modified directly by users, although they are updated by the DML trigger that you must create on each SDO_GEORASTER column in each GeoRaster table. (For information about using GeoRaster triggers, see Section 3.1.1).

The USER_SDO_GEOR_SYSDATA view has the following definition:

```
(
   TABLE_NAME            VARCHAR2(32),
   COLUMN_NAME           VARCHAR2(1024),
   METADATA_COLUMN_NAME  VARCHAR2(1024),
   RDT_TABLE_NAME        VARCHAR2(32),
   RASTER_ID             NUMBER,
   OTHER_TABLE_NAMES     SDO_STRING_ARRAY
);
```

The ALL_SDO_GEOR_SYSDATA view has all columns in the USER_SDO_GEOR_SYSDATA view, but it also has an OWNER column identifying the schema that owns the table specified in the TABLE_NAME column.

This section describes each of the columns common to both views. Note that for VARCHAR2 data in any columns, names are stored in all uppercase characters.

2.4.1 TABLE_NAME Column

The TABLE_NAME column contains the name of a GeoRaster table that has at least one column of type SDO_GEORASTER.

2.4.2 COLUMN_NAME Column

The COLUMN_NAME column contains the name of a column of type SDO_GEORASTER in the GeoRaster table specified in the TABLE_NAME column.

2.4.3 METADATA_COLUMN_NAME Column

The METADATA_COLUMN_NAME column is ignored for the current release.

2.4.4 RDT_TABLE_NAME Column

The RDT_TABLE_NAME column contains the name of the raster data table associated with the table and column specified in the TABLE_NAME and COLUMN_NAME columns. (The raster data table is explained in Section 2.2.)

2.4.5 RASTER_ID Column

The RASTER_ID column contains a number that, together with the RDT_TABLE_NAME column value, uniquely identifies each GeoRaster object.

2.4.6 OTHER_TABLE_NAMES Column

The OTHER_TABLE_NAMES column is ignored for the current release.

2.5 GeoRaster XML Schema Table

GeoRaster uses a table named SDO_GEOR_XMLSCHEMA_TABLE to store the GeoRaster metadata XML schema and other information. This table is under the MDSYS schema, and you must include the schema name if you reference this table. For example:

```
DESCRIBE mdsys.sdo_geor_xmlschema_table
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 GEORASTERFORMAT                                    VARCHAR2(1024)
 XMLSCHEMA                                          CLOB
```

Table 2–5 describes the columns of the SDO_GEOR_XMLSCHEMA_TABLE table.

Table 2–5 SDO_GEOR_XMLSCHEMA_TABLE Table Columns

Column Name	Data Type	Description
id	NUMBER	ID number, assigned by Oracle.
georasterFormat	VARCHAR2(1024)	GeoRaster format identifier, assigned by Oracle.
xmlSchema	CLOB	GeoRaster metadata XML schema definition. This definition is included in Appendix A.

There are no GeoRaster views defined on this table. It is mainly of interest to advanced users who might want to query the table.

You are encouraged not to modify the contents of this table, unless you want to define your own XML schema for other metadata that is not included in the GeoRaster XML schema, and to store that metadata in a new row in this table. If you add a row for your own metadata, do not use an ID column value of `1` or a GEORASTERFORMAT column value of `GEORASTER`, because these column values are reserved for use by Oracle.

3

GeoRaster Operations

This chapter describes how to perform the main kinds of GeoRaster operations. A typical GeoRaster workflow consists of most or all of the following steps:

1. Create the GeoRaster table, GeoRaster DML trigger, and raster data table (see Section 3.1).
2. Initialize or create GeoRaster objects (see Section 3.2).
3. Adjust the Java pool size before importing GeoRaster data, if necessary (see Section 3.3).
4. Load raster imagery or grids (see Section 3.4).
5. Validate GeoRaster objects, if they have not already been validated (see Section 3.5).
6. Georeference the GeoRaster objects, if necessary (see Section 3.6).
7. Set the spatial extents of the GeoRaster objects (see Section 3.7).
8. Create spatial indexes or other indexes, or both (see Section 3.8).
9. Change the GeoRaster storage format, if necessary (see Section 3.9).
10. Query and update the GeoRaster metadata (see Section 3.10).
11. Query and update cell data (see Section 3.11).
12. Process GeoRaster objects (see Section 3.12).
13. Compress GeoRaster objects, if appropriate (see Section 3.13).
14. View GeoRaster objects (see Section 3.14).
15. Export GeoRaster objects (see Section 3.15).
16. Update GeoRaster objects before committing the transaction (see Section 3.16).
17. Transfer GeoRaster data between databases (see Section 3.17).
18. Ensure raster data table name uniqueness, if necessary (see Section 3.18).
19. Manually maintain the GeoRaster system, if necessary (see Section 3.19).
20. Deal with possible GeoRaster data problems, if necessary (see Section 3.20).

After you create the GeoRaster objects, load the data, and validate the GeoRaster objects, you can perform the remaining operations in any order, depending on your application needs. You may also be able to skip certain operations.

Some operations can be performed using SQL, and some operations must be performed using PL/SQL blocks. For examples of these operations, see the demo files described in Section 1.11 and the examples in Chapter 4.

This chapter contains the sections that explain the main kinds of GeoRaster operations.

Chapter 4 contains detailed reference information about the SDO_GEOR package, which contains subprograms (functions and procedures) to work with GeoRaster data and metadata.

3.1 Creating the GeoRaster Table, Trigger, and Raster Data Table

Before you can work with GeoRaster objects, you must create a GeoRaster table, the GeoRaster DML trigger for that table, and one or more raster data tables if they do not already exist. Follow these steps:

1. Create a GeoRaster table with a column of type SDO_GEORASTER. Example 3–1 creates a GeoRaster table named CITY_IMAGES, which contains a column named IMAGE for storing GeoRaster objects.

Example 3–1 Creating a GeoRaster Table for City Images

```
CREATE TABLE city_images (image_id NUMBER, image_description VARCHAR2(50), image
SDO_GEORASTER);
```

   For more information about GeoRaster tables, see Section 1.4.

2. Create the standard GeoRaster DML trigger for the GeoRaster table. Example 3–2 creates the standard GeoRaster DML trigger for the table named CITY_IMAGES that contains a GeoRaster column named IMAGE.

Example 3–2 Creating the GeoRaster DML Trigger for the City Images Table

```
EXECUTE sdo_geor_utl.createDMLTrigger('CITY_IMAGES', 'IMAGE');
```

   For more information about the standard GeoRaster DML trigger, see Section 3.1.1.

3. Create a raster data table (or several raster data tables) to be used with the objects in the GeoRaster table. Example 3–3 creates a raster data table named CITY_IMAGES_RDT, which will be used to store information about each block of each GeoRaster object in the CITY_IMAGES table. (The association between a GeoRaster table and a raster table is not made until you create a GeoRaster object, as explained in Section 3.2.)

Example 3–3 Creating a Raster Data Table for City Images

```
CREATE TABLE city_images_rdt OF SDO_RASTER
  (PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber,
    rowBlockNumber, columnBlockNumber))
  TABLESPACE tbs3 NOLOGGING
  LOB(rasterBlock) STORE AS lobseg
    (TABLESPACE  tbs3_2
     CHUNK 8192
     CACHE READS
     NOLOGGING
     PCTVERSION 0
     STORAGE (PCTINCREASE 0)
    );
```

   For more information about the keywords and options when creating a raster data table, see Section 1.4.2.

3.1.1 GeoRaster DML Trigger

To ensure the consistency and integrity of internal GeoRaster tables and data structures, GeoRaster supplies a trigger that performs necessary actions after each of the following data manipulation language (DML) operations affecting a GeoRaster object: insertion of a row, update of a GeoRaster object, and deletion of a row. You must ensure that the trigger is used properly by calling the SDO_GEOR_UTL.createDMLTrigger procedure (described in Chapter 5) to create a trigger on each GeoRaster column in each GeoRaster table. For example, if a table contains two GeoRaster columns, you must call the SDO_GEOR_UTL.createDMLTrigger procedure twice (once for each combination of table name and GeoRaster column) before you perform any DML operations on the table.

You should create the necessary DML trigger or triggers immediately after you create a GeoRaster table, and you must create the trigger or triggers before you perform any operations on the table.

Each time you call the SDO_GEOR_UTL.createDMLTrigger procedure successfully, GeoRaster creates a trigger with a unique name. (All GeoRaster DML trigger names start with the string `GRDMLTR_`.) When you drop a GeoRaster table, all GeoRaster triggers associated with the table are automatically dropped also.

If you have created the GeoRaster DML trigger on a column, GeoRaster automatically performs the following actions when the trigger is fired as a result of a DML operation affecting that column:

- After an insert operation, the trigger inserts a row with the GeoRaster table name, GeoRaster column name, raster data table name, and `rasterID` value into the USER_SDO_GEOR_SYSDATA view (described in Section 2.4). If an identical entry already exists, an exception is raised.
- After an update operation, if the new GeoRaster object is null or empty, the trigger deletes the old GeoRaster object. If there is no entry in the USER_SDO_GEOR_SYSDATA view for the old GeoRaster object (that is, if the old GeoRaster object is null), the trigger inserts a row into that view for the new GeoRaster object. If there is an entry in the USER_SDO_GEOR_SYSDATA view for the old GeoRaster object, the trigger updates the information to reflect the new GeoRaster object.
- After a delete operation, the trigger deletes raster data blocks for the GeoRaster object in its raster data table, and it deletes the row in the USER_SDO_GEOR_SYSDATA view for the GeoRaster object.

3.2 Creating New GeoRaster Objects

Before you can store a GeoRaster image in a GeoRaster table, you must create the GeoRaster object. To create a new GeoRaster data object, you have the following options:

- Initialize an empty GeoRaster object, using the SDO_GEOR.init function.
- Create a blank GeoRaster object, using the SDO_GEOR.createBlank function.

You cannot perform any GeoRaster operations if the object has not been properly created (that is, if the object is an atomic null). The SDO_GEOR.init and SDO_GEOR.createBlank functions initialize GeoRaster objects with their raster data table and raster ID values if these are not already specified, and ensure that the raster data table name and raster ID value pair is unique for the current user.

If the new GeoRaster object will hold raster cell data (resulting from another GeoRaster procedure, such as SDO_GEOR.importFrom, SDO_GEOR.subset, or SDO_

GEOR.copy), and if the raster data table for this new GeoRaster object does not exist, you must first create the raster data table. For information about creating a raster data table, see Section 1.4.2, especially Example 1–2.

To avoid potential GeoRaster data problems (some of which are described in Section 3.20), always register an initialized GeoRaster object in the GeoRaster system views by inserting the GeoRaster object into a GeoRaster table, and do this before you perform any other operations on the GeoRaster object.

3.3 Adjusting Java Pool Size Before Importing GeoRaster Objects

Loading GeoRaster objects with the SDO_GEOR.importFrom procedure may require you to increase the size of the Java pool. This section briefly explains how to configure the Java pool size.

There are four auto-tuned SGA initialization parameters: DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, and JAVA_POOL_SIZE. You can also set these parameters with the ALTER SYSTEM command. Depending on whether automatic shared memory management is enabled, these parameters behave differently, as explained in the following sections.

3.3.1 Using Automatic Shared Memory Management

You can enable the automatic shared memory management feature by setting the SGA_TARGET parameter to a value greater than zero. For example:

```
ALTER SYSTEM SET SGA_TARGET=248M;
```

The value specified for SGA_TARGET cannot exceed the value specified for SGA_MAX_SIZE. In addition, you must ensure that the STATISTICS_LEVEL initialization parameter is set to TYPICAL (the default) or ALL.

After the parameter SGA_TARGET is set at a value greater than zero, DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE and JAVA_POOL_SIZE parameters specify the *minimum* sizes for the four associated SGA components. To allow the Java pool size to grow during the SDO_GEOR.importFrom procedure, you might need to set some combination of the DB_CACHE_SIZE, SHARED_POOL_SIZE, and LARGE_POOL_SIZE parameters to smaller values (if they are not already zero). The following example sets three parameters to relatively small values and the JAVA_POOL_SIZE parameter to 80 megabytes:

```
ALTER SYSTEM SET DB_CACHE_SIZE=8M;
ALTER SYSTEM SET SHARED_POOL_SIZE=50M;
ALTER SYSTEM SET LARGE_POOL_SIZE=0;
ALTER SYSTEM SET JAVA_POOL_SIZE=80M;
```

3.3.2 Using Manual Memory Management

If the SGA_TARGET parameter is not set or is set to zero, automatic shared memory management is disabled, and the DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, and JAVA_POOL_SIZE parameters behave as they did in releases before Oracle10g. That is, in this case these parameters specify the *maximum* sizes for the associated SGA components. Because the sum of the sizes for the different SGA components is bounded by the SGA_MAX_SIZE value, before you can increase the JAVA_POOL_SIZE value with the ALTER SYSTEM command, you might have to decrease the values of the other SGA-related parameters. The following example shows the general format for statements to achieve this result:

```
ALTER SYSTEM SET DB_CACHE_SIZE=<current_db_cache_size> - m1;
```

```
ALTER SYSTEM SET SHARED_POOL_SIZE=<current_shared_pool_size> - m2;
ALTER SYSTEM SET LARGE_POOL_SIZE=<current_large_pool_size> - m3;
ALTER SYSTEM SET JAVA_POOL_SIZE=<current_java_pool_size> + m4;
```

You can also increase the SGA_MAX_SIZE value by shutting down the database and updating the initialization parameter file. After the database is restarted, you can also resize the SGA components as described in this section and in Section 3.3.1.

3.4 Loading GeoRaster Data

To load GeoRaster data, you have the following options:

- Call the SDO_GEOR.importFrom procedure to load images into GeoRaster objects.
- Use the GeoRaster loader tool or viewer tool, which are described in Section 1.10.

With both options, you can do the following:

- Compress raster data and store the data in JPEG-compressed or DEFLATE-compressed GeoRaster objects.
- Load an ESRI world file into an existing GeoRaster object, and georeference the raster data without reloading it. You can also specify an SRID with the world file and generate the spatial extent of the data.

Because an ESRI world file does not contain coordinate system information, you can specify the SRID value of a coordinate reference system for the load operation. If you do not specify an SRID, the model SRID of the GeoRaster objects is set to 0 (zero) by default, which means that the model space is the same as the raster (cell) space. However, if you do not want to set the model space SRID to zero, but you do not yet know the coordinate system of the model space, you can specify the SRID value as 999999, which means that the coordinate reference system is unknown. (Specifically, SRID 999999 is associated with a coordinate reference system named `unknown CRS`.) Later, when you know the actual coordinate reference system of the model space, you can set the SRID value accordingly.

For more information about the `unknown CRS` (SRID 999999) coordinate reference system, see *Oracle Spatial User's Guide and Reference*.

3.5 Validating GeoRaster Objects

Before you use a GeoRaster object, you should ensure that it is valid. Validation for a GeoRaster object includes checking the metadata and the raster cell data, and making sure that they are consistent. For example, validation checks the raster type, dimension information, and the actual sizes of cell blocks, and it performs other checks.

If you used the GeoRaster loader tool described in Section 1.10, the GeoRaster objects were validated during the load operation.

GeoRaster provides the following validation subprograms:

- SDO_GEOR.validateGeoraster validates the GeoRaster object, including cell data and metadata. It returns `TRUE` if the object is valid; otherwise, it returns one of the following: an Oracle error code indicating why the GeoRaster object is invalid, `FALSE` if validation fails for an unknown reason, or `NULL` if the GeoRaster object is null. You should always use this function after you create a GeoRaster object.
- SDO_GEOR.schemaValidate validates the metadata against the GeoRaster XML schema. You can use this function to locate errors if the SDO_GEOR.validateGeoraster function returned the error code 13454. The SDO_

GEOR.schemaValidate and SDO_GEOR.validateGeoraster functions do not validate the spatial extent geometry.

3.6 Georeferencing GeoRaster Objects

Georeferencing, as explained in Section 1.6, establishes the relationship between cell coordinates of GeoRaster data and real-world ground coordinates (or some local coordinates). If you need to georeference GeoRaster objects, the following approaches are available:

- If the original image is already georeferenced and if the georeferencing information is stored in an ESRI world file, you can use the SDO_GEOR.importFrom procedure to load an ESRI world file from a file or from a CLOB object, along with the image data itself (in either FILE or BLOB format). You can also use the GeoRaster client-side loader tool (described in Section 1.10) to load an ESRI world file from a file, along with the image file itself.

  Because an ESRI world file does not specify the model coordinate system, you can set the model space of the georeferenced GeoRaster object using an Oracle SRID in either of the following ways: specify the SRID along with the world file as a parameter to the SDO_GEOR.importFrom procedure or the GeoRaster client-side loader (described in Section 1.10); or, after loading the world file, call the SDO_GEOR.setModelSRID procedure. You can also call the SDO_GEOR.setModelSRID procedure to change the model space of a georeferenced GeoRaster object.

- You can use the SDO_GEOR.setSRS procedure to add, modify, and delete georeferencing information. For example, you can create an SDO_GEOR_SRS object and assign the coefficients and related georeferencing information, and then call the SDO_GEOR.setSRS procedure to add or update the spatial reference information of any GeoRaster object. If you know that one GeoRaster object has the same SRS information as another GeoRaster object, you can call the SDO_GEOR.getSRS function to retrieve an SDO_GEOR_SRS object from this GeoRaster object, and then call the SDO_GEOR.setSRS procedure to georeference the first GeoRaster object.

- You can call the SDO_GEOR.georeference procedure to georeference a GeoRaster object directly. This function takes the coefficients `A`, `B`, `C`, `D`, `E`, `F` (described in a formula in Section 1.6.1) and other information, converts them into the coefficients `a`, `b`, `c`, `d`, `e`, `f`, and stores them in the spatial reference information of a GeoRaster object. If the original raster data is rectified and if the model coordinate of its origin (upper-left corner) is (x0, y0) and its spatial resolution or scale is `s`, then the following are true: A = s, B = 0, C = x0, D = 0, E = -s, F = y0.

Based on the SRS information of a georeferenced GeoRaster object, transforming GeoRaster coordinate information means finding the model (ground) coordinate associated with a specific cell (raster) coordinate, and the reverse. That is, you can do the following:

- Given a specific cell coordinate, you can find the associated model space coordinate using the SDO_GEOR.getModelCoordinate function. For example, if you identify a point in an image, you can find the longitude and latitude coordinates associated with that point.

- Given a model space coordinate, you can find the associated cell coordinate using the SDO_GEOR.getCellCoordinate function. For example, if you identify longitude and latitude coordinates, you can find the cell in an image associated with those coordinates.

3.7 Generating and Setting Spatial Extents

When a GeoRaster object is created, its spatial extent (`spatialExtent` attribute, described in Section 2.1.2) is not necessarily the enclosing geometry in its model space coordinate system. The spatial extent (footprint) geometry might initially be null, or it might reflect the cell space coordinate system or some other coordinate system. The ability to generate and set spatial extents is useful for building large GeoRaster databases of a global or large regional scope, in which the spatial extents are in one global geodetic coordinate system while the GeoRaster objects (imagery, DEMs, and so on) are in different projected coordinate systems. In such a case, you can create a spatial (R-tree) index on the spatial extents, which requires that all spatial extent geometries have the same SRID value.

To ensure that the spatial extent geometry of each GeoRaster object in a table is correct for its model space coordinate system (or for any other coordinate system that you may want to use), you must set the spatial extent. Moreover, to use a spatial index on the spatial extent geometries (described in Section 3.8), all indexed geometries must be based on the same coordinate system (that is, have the same SRID value).

You can set the spatial extent in either of the following ways: specify `spatialExtent=TRUE` as a storage parameter to the SDO_GEOR.importFrom procedure or the GeoRaster client-side loader (described in Section 1.10), or use the SQL UPDATE statement. If you use the SDO_GEOR.importFrom procedure or the loader, the SRID cannot be null or 0 (zero), and if there is an R-tree index on the GeoRaster spatial extent, the SRID of the spatial extent must match the SRID of the existing spatial index; otherwise, the spatial extent is set to a null value.

In addition, if you do not already have the spatial extent geometry, you can generate it using the SDO_GEOR.generateSpatialExtent function, and use that geometry to update the GeoRaster object. The following example updates the spatial extent geometry of a specified GeoRaster object in the CITY_IMAGES table (created in Example 3–1 in Section 3.1) to the generated spatial extent (reflecting the model coordinate system) of that object:

```
UPDATE city_images c
  SET c.image.spatialExtent = sdo_geor.generateSpatialExtent(image)
    WHERE c.image_id = 100;
COMMIT;
```

If you already know the spatial extent geometry for a GeoRaster object, or if you want the spatial extent geometry to be based on a coordinate system other than the one for the model space, construct the SDO_GEOMETRY object or select it from a table, and then update the GeoRaster object to set its spatial extent attribute to that geometry, as shown in the following example:

```
DECLARE
 geom sdo_geometry;
BEGIN
-- Set geom to an SDO_GEOMETRY object that covers the spatial extent
-- of the desired GeoRaster object. If necessary, perform coordinate
-- system transformation before setting geom.
-- geom := sdo_geometry(...);
  UPDATE city_images c
    SET c.image.spatialExtent = geom WHERE c.image_id = 100;
  COMMIT;
END;
```

3.8 Indexing GeoRaster Data

GeoRaster data can be indexed in various ways. The most important index you can create on a GeoRaster object is a spatial index on the spatial extent (footprint) geometry of the GeoRaster object (`spatialExtent` attribute, described in Section 2.1.2). For information about creating spatial indexes, see *Oracle Spatial User's Guide and Reference*.

You can also create one or more other indexes, such as:

- Function-based indexes on metadata objects using the Oracle XMLType or Oracle Text document indexing functionality
- Standard indexes on other user-defined columns of the GeoRaster table, such as cloud coverage, water coverage, or vegetation

In addition to any indexes that you may create, a B-tree index for Oracle internal use is built on each raster data table.

3.9 Changing Raster Storage

You can change some aspects of the way raster image data is stored: the raster blocking size, cell depth, interleaving type, and other aspects. To make such changes, use the SDO_GEOR.changeFormatCopy procedure, and specify the desired storage parameter values with the `storageParam` parameter. You can also specify storage parameters with several other functions and procedures that load and process a GeoRaster object to create another GeoRaster object.

For information about the storage parameters that you can specify, see Section 1.4.1.

3.10 Querying and Updating GeoRaster Metadata

You can query metadata for a GeoRaster object, and you can update many attributes of the metadata.

You can use many functions, most of whose names start with *get*, to query the metadata and ancillary information (for example, SDO_GEOR.getTotalLayerNumber and SDO_GEOR.hasPseudoColor).

You can use several subprograms, most of whose names start with *set*, to update metadata and ancillary data (for example, SDO_GEOR.setSRS and SDO_GEOR.setColorMap).

See Section 1.9.3 for categories and lists of subprograms that get and set GeoRaster metadata and cell data.

3.11 Querying and Updating Cell Data

To display part or all of a raster image, you can query the data for a cell (pixel), a range of cells, or the entire image associated with a GeoRaster object:

- SDO_GEOR.getCellValue returns the value of a single cell of the GeoRaster object.
- SDO_GEOR.getRasterSubset creates a single BLOB object containing all cells of a precise subset of the GeoRaster object (as specified by a window, layer or band numbers, and pyramid level). This BLOB object contains only raster cells and no related metadata.

- SDO_GEOR.getRasterData creates a single BLOB object containing all cells of the GeoRaster object at a specified pyramid level. This BLOB object contains only raster cells and no related metadata.
- SDO_GEOR.getRasterBlocks returns an object that includes all image data inside or touching a specified window. Specifically, it returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that are inside or touch a specified window.

You can also use the SDO_GEOR.exportTo procedure to export all or part of a raster image to a BLOB object (binary image format) or to a file of a specified file format type.

To change the value of raster cells in a specified window to a single value, you can use the SDO_GEOR.changeCellValue procedure.

Note: If you use any procedure that adds or overwrites data in the input GeoRaster object, you should make a copy of the original GeoRaster object and use the procedure on the copied object. After you are satisfied with the result of the procedure, you can discard the original GeoRaster object if you wish.

See Section 1.9.3 for categories and lists of subprograms that get and set GeoRaster metadata and cell data.

3.12 Processing GeoRaster Objects

You can perform a variety of processing operations on GeoRaster data, including changing the format, subsetting (cropping), scaling, and generating pyramids. See the GeoRaster PL/SQL demo files, described in Section 1.11, for examples and explanatory comments.

3.13 Compressing and Decompressing GeoRaster Objects

You can reduce the storage space requirements for GeoRaster objects by compressing them using JPEG-B, JPEG-F, or DEFLATE compression. You can decompress any compressed GeoRaster object, although this is not required for any GeoRaster operations, because any GeoRaster operation that can be performed on an uncompressed (decompressed) GeoRaster object can be performed on a compressed GeoRaster object.

To compress or decompress a GeoRaster object, use the compression keyword in the `storageParam` parameter with the SDO_GEOR.changeFormatCopy procedure, or with several other procedures that load and process a GeoRaster object to create another GeoRaster object, including SDO_GEOR.importFrom, SDO_GEOR.mosaic, SDO_GEOR.scaleCopy, and SDO_GEOR.subset. (There are no separate procedures for compressing and decompressing a GeoRaster object.)

For more information about GeoRaster compression and decompression, see Section 1.8.

3.14 Viewing GeoRaster Objects

To view GeoRaster objects, you have the following options:

- Call the SDO_GEOR.exportTo procedure to export GeoRaster objects to image files, and then display the images using image tools or a Web browser.

- Use the standalone GeoRaster viewer tool (one of the tools described in Section 1.10).

With the GeoRaster viewer tool, you can select any GeoRaster object of a database schema (user), query and display the whole or a subset of a GeoRaster object, zoom in and zoom out, scroll, and perform other basic operations. The pyramid level, cell coordinates, and model coordinates (if the object is georeferenced) are displayed for the point at the mouse pointer location. You can display individual cell values and choose different layers of a multiband or hyperspectral image for RGB full color display. The blocking boundaries can be overlapped on the top of the display. Depending on the data and your requests, the viewer can display the raster data in grayscale, pseudocolor, and 24-bit true color over an intranet or the Internet. (For the current release, bitmap, two-dimensional grayscale, pseudocolor, and three-band full color are supported.) Some of the basic GeoRaster metadata is also displayed.

In the GeoRaster viewer tool, the data displayed by every operation is retrieved from the GeoRaster server; it is not generated in memory.

The GeoRaster viewer tool also includes menu commands to call the GeoRaster loader and exporter tools, thus enabling you to use a single tool as an interface to the capabilities of all the GeoRaster tools.

3.15 Exporting GeoRaster Objects

To export GeoRaster objects to image files, you have the following options:

- Call the SDO_GEOR.exportTo procedure (which can export either to a file or to a BLOB object).
- Use the GeoRaster exporter tool or viewer tool, which are described in Section 1.10.

3.16 Updating GeoRaster Objects Before Committing

Before you commit a database transaction that inserts, updates, or deletes GeoRaster cell data or metadata, you should update the GeoRaster object. If you do not update the GeoRaster object after changing cell data, one or more of the following can result: an invalid GeoRaster object, dangling raster data, and inconsistent metadata. If you do not update the GeoRaster object after changing GeoRaster metadata, the metadata changes will not take effect.

If you decide to roll back the transaction instead of committing it, an UPDATE statement is not needed.

In Example 3–4, the UPDATE statement is required after the call to the SDO_GEOR.changeFormatCopy procedure and before the COMMIT statement.

Example 3–4 Updating a GeoRaster Object Before Committing

```
DECLARE
    gr1 sdo_georaster;
    gr2 sdo_georaster;
BEGIN
    SELECT georaster INTO gr2 from georaster_table WHERE georid=11 FOR UPDATE;
    SELECT georaster INTO gr1 from georaster_table WHERE georid=1;
    sdo_geor.changeFormatCopy(gr1, 'blocksize=(2048,2048)', gr2);
    UPDATE georaster_table SET georaster=gr2 WHERE georid=11;
    COMMIT;
END;
```

/

3.17 Transferring GeoRaster Data Between Databases

You can use either the Data Pump Export and Import utilities or the original Export and Import utilities to transfer GeoRaster data between databases. You must export and import rows from both the GeoRaster table and its related raster data table or tables. After the transfer, you may also need to insert the GeoRaster system data for the imported GeoRaster objects into the USER_SDO_GEOR_SYSDATA view (described in Section 2.4) in the target schema, and you should use the SDO_GEOR.validateGeoraster function to check the validity of imported GeoRaster objects.

For information about the Data Pump Export and Import utilities and the original Export and Import utilities, see *Oracle Database Utilities*.

To transfer GeoRaster data between databases, follow these general steps:

1. Check for and resolve any conflicts, as explained in Section 3.17.1.
2. Perform the data transfer, as explained in Section 3.17.2.

3.17.1 Checking for and Resolving Conflicts

For a successful import of GeoRaster data into a target schema, there must be no conflicts in the target schema's GeoRaster system data. The following conditions can cause a conflict:

- A raster data table with the same name is already defined in another schema in the target database.
- Any pairs of raster data table name and raster ID to be inserted into the target schema's USER_SDO_GEOR_SYSDATA view are not unique.

To check for possible raster data table name conflicts, connect to the target database as a user with the DBA role and enter a query in the following form:

```
SELECT UNIQUE owner FROM all_sdo_geor_sysdata
  WHERE rdt_table_name=UPPER('<rdt_to_be_exported>')
    AND owner<>UPPER('<target_schema>');
```

In the preceding example, replace `<rdt_to_be_exported>` with the name of the raster data table to be exported, and replace `<target_schema>` with the target schema name.

If the query returns one or more rows, the export operation would cause a conflict between raster data table names in the target database; and if the query returns more than one row, there is already a conflict between raster data table names in the target database. To resolve each conflict, call the SDO_GEOR_UTL.renameRDT procedure (documented in Chapter 5) in either the source or target database before you perform the data transfer.

If any pairs of raster data table name and raster ID to be inserted into the target schema's USER_SDO_GEOR_SYSDATA view are not unique, you must modify one of the GeoRaster objects involved in the conflict in either the source or the target schema. To avoid moving data around, fixing a conflict usually means changing the raster ID of one GeoRaster object to another number. For example:

1. Find a raster ID that is not being used (in either the source or the target schema) in the raster data table involved in the conflict.
2. Modify one of the GeoRaster objects. For example:

```
UPDATE georaster_table t SET t.georaster_col.rasterid=new_raster_id
  WHERE t.georaster_col.rasterid=old_raster_id ;
```

After the UPDATE statement, if the required standard GeoRaster data manipulation language (DML) trigger has been created and enabled, the raster ID value shown in the USER_SDO_GEOR_SYSDATA view for the fixed GeoRaster object is updated correspondingly. You should validate the fixed GeoRaster object before performing a commit or any other operation.

3.17.2 Performing the GeoRaster Data Transfer

To export GeoRaster data from one database and import it into another, you must export and import rows from both the GeoRaster table and its associated raster data table or tables. You can export and import raster data tables just as you would other data tables; however, special considerations apply to importing a GeoRaster table. You have the following basic options for each GeoRaster table:

- Importing the GeoRaster table definition and table row data in separate steps

  This approach ensures that all system data related to the GeoRaster table is automatically maintained during the import operation. However, the GeoRaster table may need to be imported separately from the raster data table or tables associated with it (for example, to achieve acceptable import performance with very large raster data tables), although the tables can be exported together and several GeoRaster tables can be grouped together for an import operation.

- Importing the GeoRaster table definition and table row data in a single step

  This approach allows you to import the GeoRaster table with its associated raster data table or tables, and it usually provides better performance. However, you must take specific actions to update the GeoRaster system data in the target schema after the import operation.

You must re-create the required GeoRaster DML trigger on each GeoRaster table after the import operation. If you imported the GeoRaster table with its triggers, you must drop its GeoRaster DML trigger and then re-create that trigger. (Depending on whether other triggers are also defined on a GeoRaster table, you can choose to import the table either with or without its triggers.)

To import the GeoRaster table definition and table row data in separate steps, follow this procedure:

1. Import the table definition, with or without triggers. The following command (to be entered on one line) imports the table definition, without triggers, for a GeoRaster table named GEORASTER_TABLE:

   ```
   impdp gr/gr TABLES=georaster_table CONTENT=METADATA_ONLY EXCLUDE=TRIGGER
   DUMPFILE=dump_dir:exp.dmp
   ```

2. If you included triggers in the import operation, drop the GeoRaster DML trigger on the table.

3. Create the required GeoRaster DML trigger on the GeoRaster table by calling the SDO_GEOR_UTL.createDMLTrigger procedure.

4. Import the table row data of the GeoRaster table. The following command imports the table data of a GeoRaster table named GEORASTER_TABLE:

   ```
   impdp gr/gr TABLES=georaster_table CONTENT=DATA_ONLY DUMPFILE=dump_dir:exp.dmp
   ```

To import the GeoRaster table definition and table row data in a single step, follow this procedure:

1. For each target schema, create a table to hold all related GeoRaster system data. For example:

   ```
   CREATE TABLE tmp_sysdata_table
     AS SELECT * FROM all_sdo_geor_sysdata
       WHERE table_name=UPPER('<table_to_be_exported>')
         AND owner=(UPPER('<source_schema>');
   ```

   In the preceding example, replace `<table_to_be_exported>` with the name of the GeoRaster table to be exported, and replace `<source_schema>` with the source schema name.

   If you use the QUERY parameter to filter the GeoRaster objects to be exported, add corresponding conditions in the WHERE clause in the preceding example to filter the GeoRaster system data as well.

2. Export and import the table that you created in Step 1, together with GeoRaster tables and their related raster data tables. You can use any mode for the import operation.

3. If you included triggers in the import operation, drop the GeoRaster DML trigger on each GeoRaster table.

4. In the target schema, create the required GeoRaster DML trigger on each GeoRaster table-column pair by calling the SDO_GEOR_UTL.createDMLTrigger procedure.

5. Insert the rows in the table that you created in Step 1 back into the USER_SDO_GEOR_SYSDATA view in the target schema. For example:

   ```
   INSERT INTO user_sdo_geor_sysdata
     SELECT table_name, column_name, metadata_column_name, rdt_table_name,
         raster_id, other_table_names
       FROM tmp_sysdata_table;
   ```

6. Drop the table that you created in Step 1.

3.18 Ensuring Raster Data Table Name Uniqueness

Each raster data table name must be unique in the database. To check if any raster data table name conflicts exist in the database, connect to the database as a user with the DBA role, and enter the following query:

```
SELECT UNIQUE owner, rdt_table_name
  FROM (SELECT rdt, count(*) count
        FROM (SELECT UNIQUE owner, rdt_table_name rdt
              FROM all_sdo_geor_sysdata)
        GROUP BY rdt) a, all_sdo_geor_sysdata b
  WHERE a.rdt = b.rdt_table_name AND a.count > 1
  ORDER BY rdt_table_name, owner;
```

If this query returns any rows, one or more raster data table name conflicts exist in the database. To resolve all conflicts, use the SDO_GEOR_UTL.makeRDTNamesUnique procedure; or use the SDO_GEOR_UTL.renameRDT procedure as needed to resolve specific conflicts. Both procedures are documented in Chapter 5.

3.19 Manually Maintaining GeoRaster System Data

In general, you should not change the name of a GeoRaster table or its GeoRaster column. If you want to change the name of a GeoRaster table or GeoRaster column, the recommended approach is to create a new GeoRaster table with the desired table and column names, create the required GeoRaster DML trigger on the table, and copy each GeoRaster object from the old table to the new table by using the SDO_GEOR.copy procedure.

However, if you instead decide to use SQL statements to rename a GeoRaster table or Georaster column, you must manually maintain the GeoRaster system data to reflect the change. For example, use an UPDATE statement of the following general form:

```
UPDATE USER_SDO_GEOR_SYSDATA
   SET TABLE_NAME=UPPER('new_table_table'), COLUMN_NAME=UPPER('new_column_name')
 WHERE TABLE_NAME=UPPER('old_table_name')
   AND COLUMN_NAME=UPPER('old_column_name');
```

You must also drop and re-create the required GeoRaster DML trigger specifying the new GeoRaster table and GeoRaster column names.

If you insert into or update the USER_SDO_GEOR_SYSDATA view, you must ensure the following for each new or changed row:

- The TABLE_NAME column specifies a valid table owned by the current user.
- The COLUMN_NAME column specifies a valid column of type SDO_GEORASTER in the specified table.
- The RDT_TABLE_NAME column specifies a valid raster data table owned by the current user.
- The name of the table specified in the RDT_TABLE_NAME column is unique in the database. (That is, no other user owns a table having this name.)
- The RDT_TABLE_NAME and RASTER_ID columns specify a unique pair of values.

If you use the FLASHBACK TABLE statement to restore an earlier state of a GeoRaster table, you must manually restore the GeoRaster system data. You should also drop and re-create any required GeoRaster DML triggers defined on the table after a FLASHBACK TABLE TO BEFORE DROP statement, because the original trigger names cannot be restored.

3.20 Dealing with Possible GeoRaster Data Problems

If you do not perform GeoRaster operations in the required sequence, or if you perform an incorrect or inappropriate operation, some data problems can occur. For example:

- A nonblank GeoRaster object might have been created, but no rows or an incorrect number of rows exist in the raster data table for that object, or the raster data blocks associated with the object have an incorrect length.
- Raster data table rows might exist for a nonexistent GeoRaster object. The raster blocks associated with such rows are referred to as *dangling blocks*.

If a GeoRaster object is invalid because of a raster data error, delete the GeoRaster object and create it again.

If dangling raster blocks exist, they cause wasted disk space in the raster data table, although otherwise they do not present a problem as long as the necessary primary

key is defined on the raster data table. If you want to remove the raster data table rows associated with dangling raster blocks, you can try to find rows associated with problems and to fix the problems. To find rows associated with problems, follow these steps:

1. To determine whether any dangling raster block data exists in a raster data table, issue a query of the following general form:

```
SELECT unique rasterid FROM rdt_tab
 WHERE rasterid NOT IN
       (SELECT RASTER_ID FROM USER_SDO_GEOR_SYSDATA
         WHERE RDT_TABLE_NAME=UPPER('rdt_tab'));
```

2. To determine whether the dangling raster block data belongs to some GeoRaster object in a GeoRaster table, issue a query of the following general form:

```
SELECT t.georaster_col FROM georaster_tab t
 WHERE UPPER(t.georaster_col.rasterDataTable) = UPPER('rdt_name') AND
       t.georaster_col.rasterId = dangling_raster_id;
```

If multiple rows are returned from the preceding query, the GeoRaster-related DML trigger associated with this specific GeoRaster column is either missing or disabled and the returned GeoRaster objects are corrupted. In this case, the data is beyond repair. Delete the corrupted GeoRaster objects and clean up the dangling raster block data.

To remove the dangling raster block data from a raster data table, delete the rows associated with problems.

If no data is missing, you can manually repair a GeoRaster object by establishing the relationship between a GeoRaster object and some dangling raster block data. To do so, execute a statement of the following general form:

```
INSERT INTO USER_SDO_GEOR_SYSDATA
   VALUES (UPPER('georaster_table'), UPPER('georaster_column'), NULL,
           UPPER('rdt_name'), dangling_raster_id, NULL);
```

Always use the SDO_GEOR.validateGeoraster function to check the validity of a GeoRaster object after attempting any repair operation.

4

SDO_GEOR Package Reference

The MDSYS.SDO_GEOR package contains subprograms (functions and procedures) for creating, modifying, and retrieving information about GeoRaster objects. This chapter presents reference information, with one or more examples, for each subprogram.

The subprograms are presented in alphabetical order in this chapter. They can be grouped into several logical categories, as explained in Section 1.9. Many of the subprograms are also discussed in Chapter 3, "GeoRaster Operations".

Many examples in this chapter refer to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.

All SDO_GEOR subprograms can work on GeoRaster objects defined in schemas other than the current connection schema.

SDO_GEOR.calcCompressionRatio

Format

```
SDO_GEOR.calcCompressionRatio(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the compression ratio of a specified GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The compression ratio is the sum of the sizes of all raster blocks (including pyramids) in the uncompressed GeoRaster object divided by the sum of these sizes in the compressed form of the object. If the input GeoRaster object is not compressed, the compression ratio is 1 (that is, 1:1).

For a given original GeoRaster object, the greater the compression ratio, the smaller is the compressed GeoRaster object. For example, a compression ratio of 20 (that is, 20:1), the data is compressed to 5 percent of its original size.

For more information about GeoRaster compression, see Section 1.8.

Examples

The following example returns the compression type and compression ratio for the GeoRaster object in the GEORASTER column of table GEORASTER_TABLE where the GEORID column value is 1.

```
SELECT sdo_geor.getCompressionType(georaster) compType,
    sdo_geor.calcCompressionRatio(georaster) compRatio
  FROM georaster_table WHERE georid=1;
```

SDO_GEOR.changeCellValue

Format

```
SDO_GEOR.changeCellValue(
   georaster       IN OUT SDO_GEORASTER,
   window          IN SDO_NUMBER_ARRAY,
   bandNumbers  IN    VARCHAR2,
   newCellValue   IN    NUMBER);
```

or

```
SDO_GEOR.changeCellValue(
   georaster       IN OUT SDO_GEORASTER,
   window          IN SDO_GEOMETRY,
   layerNumbers  IN    VARCHAR2,
   newCellValue   IN    NUMBER);
```

Description

Changes the value of raster cells in a specified window of a GeoRaster object to a single new value.

Parameters

georaster
GeoRaster object.

window
Window in which to change the values of all cells to `newCellValue`. The data type can be SDO_NUMBER_ARRAY or SDO_GEOMETRY. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_GEOMETRY, see the Usage Notes for SDO_SRID requirements and other information.

bandNumbers
A string identifying the physical band numbers on which the operation is to be performed. Use commas to delimit the values, and a hyphen to indicate a range (for example, `1-3` for bands 1, 2, and 3).

layerNumbers
A string identifying the logical layer numbers on which the operation is to be performed. Use commas to delimit the values, and a hyphen to indicate a range (for example, `2-4` for layers 2, 3, and 4).

newCellValue
The new cell value for each cell inside the window in the specified bands or layers. The value must be in the range designated by the `cellDepth` value for the GeoRaster object.

Usage Notes

Because this procedure overwrites data in the input GeoRaster object, you should make a copy of the original GeoRaster object and use this procedure on the copied object. After you are satisfied with the result of this procedure, you can discard the original GeoRaster object if you wish.

This procedure can be used to mask, or conceal, parts of an image. For example, you can change irrelevant parts of an image to a dull color before displaying the image, to help people to focus on the relevant parts.

If the `window` parameter data type is SDO_GEOMETRY, the SDO_SRID value must be one of the following:

- Null, to specify raster space
- A value from the SRID column of the MDSYS.CS_SRS table

  If the SDO_SRID values for the `window` parameter geometry and the model space are different, the `window` parameter geometry is automatically transformed to the coordinate system of the model space before the operation is performed. (Raster space and model space are explained in Section 1.3.)

If the `window` parameter specifies a nonrectangular SDO_GEOMETRY object, this function calculates the MBR of the geometry and update the cells inside that MBR, including the cells on the boundary of the MBR.

If `georaster` is a blank GeoRaster object and the whole area is updated, the result is a blank GeoRaster object with the `blankCellValue` value set to `newCellValue`.

If `georaster` is a blank GeoRaster object and it is only partially updated, the result is a nonblank GeoRaster object with the original `blankCellValue` and `newCellValue` values set according to the `window` parameter and the `bandNumbers` or `layerNumbers` parameter.

If `georaster` is a nonblank GeoRaster object, the result is a nonblank GeoRaster object, even if all cells are set to the `newCellValue` value.

If `georaster` is null, this procedure performs no operation. If `georaster` is invalid, an exception is raised.

If any pyramids are defined on the GeoRaster object, the corresponding cell values for the pyramids are updated.

To return the value of a single cell located anywhere in the GeoRaster object, use the SDO_GEOR.getCellValue function.

Examples

The following example changes the value of all cells to 151 in a specified window in band number 1. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=110 FOR UPDATE;
  sdo_geor.changeCellValue(gr, sdo_number_array(100,67,134,113), '1', 151);
  UPDATE georaster_table SET georaster=gr WHERE georid=110;
  COMMIT;
END;
/
```

SDO_GEOR.changeFormat

Format

```
SDO_GEOR.changeFormat(
   georaster       IN OUT SDO_GEORASTER,
   storageParam  IN VARCHAR2);
```

Description

Changes the storage format of an existing GeoRaster object (for example, changing the blocking, cell depth, or interleaving).

> **Note:** This procedure is deprecated. Instead, use the SDO_GEOR.changeFormatCopy procedure, which makes a copy of an existing GeoRaster object using a different storage format.

Parameters

georaster
The SDO_GEORASTER object whose format is to be changed.

storageParam
A string specifying storage parameters, as explained in Section 1.4.1. However, the `compression` keyword is not supported in the `storageParam` parameter for this procedure, while it is supported for the SDO_GEOR.changeFormatCopy procedure.

Usage Notes

This procedure, which changes the input GeoRaster object, is deprecated and will not be supported in a future release of Spatial. Instead, use the SDO_GEOR.changeFormatCopy procedure. After you use the SDO_GEOR.changeFormatCopy procedure, you can check to ensure that the desired changes were made in the copy of the original GeoRaster object, and then discard the original GeoRaster object if you wish.

If `georaster` is null, this procedure performs no operation.

An exception is raised if `georaster` is invalid.

Examples

The following example changes the interleaving type of a GeoRaster object to `BIL`. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
    gr1 sdo_georaster;
BEGIN
    SELECT georaster INTO gr1 from georaster_table WHERE georid=11 FOR UPDATE;
    sdo_geor.changeFormat(gr1, 'interleaving=BIL');
    UPDATE georaster_table SET georaster=gr1 WHERE georid=11;
    COMMIT;
END;
/
```

SDO_GEOR.changeFormatCopy

Format

```
SDO_GEOR.changeFormatCopy(
   inGeoraster     IN SDO_GEORASTER,
   storageParam    IN VARCHAR2,
   outGeoraster    IN OUT SDO_GEORASTER);
```

Description

Makes a copy of an existing GeoRaster object using a different storage format (for example, changing the blocking, cell depth, or interleaving).

Parameters

inGeoraster
The SDO_GEORASTER object whose format is to be copied.

storageParam
A string specifying storage parameters, as explained in Section 1.4.1.

outGeoraster
The SDO_GEORASTER object to hold the copy.

Usage Notes

This procedure does not change the input GeoRaster object, but creates a new GeoRaster object that has the specified changes. After you use this procedure, you can check to ensure that the desired changes were made in the copy of the original GeoRaster object, and then discard the original GeoRaster object if you wish.

To compress or decompress a GeoRaster object, use the `compression` keyword in the `storageParam` parameter. (There is no separate GeoRaster function or procedure for compressing or decompressing a GeoRaster object.)

If `inGeoraster` is null, this procedure performs no operation.

If `storageParam` is null, `inGeoraster` is copied to `outGeoraster`.

If `outGeoraster` has any raster data, it is deleted before the copy operation.

If pyramid data exists for `inGeoraster`, the pyramid data is copied to `outGeoraster` unless the `storageParam` string contains `pyramid=FALSE`.

An exception is raised if one or more of the following are true:

- `inGeoraster` is invalid.
- `outGeoraster` has not been initialized.
- A raster data table for `outGeoraster` does not exist and `outGeoraster` is not a blank GeoRaster object.

Examples

The following example creates a GeoRaster object that is the same as the input object except that the block size is set to 2048 for both dimensions. (It refers to a table named

GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
    gr1 sdo_georaster;
    gr2 sdo_georaster;
BEGIN
    SELECT georaster INTO gr2 from georaster_table WHERE georid=11 FOR UPDATE;
    SELECT georaster INTO gr1 from georaster_table WHERE georid=1;

    sdo_geor.changeFormatCopy(gr1, 'blocksize=(2048,2048)', gr2);
    UPDATE georaster_table SET georaster=gr2 WHERE georid=11;
    COMMIT;
END;
/
```

SDO_GEOR.copy

Format

```
SDO_GEOR.copy(
   inGeoraster    IN SDO_GEORASTER,
   outGeoraster  IN OUT SDO_GEORASTER);
```

Description

Makes a copy of an existing GeoRaster object.

Parameters

inGeoraster
GeoRaster object to be copied.

outGeoraster
GeoRaster object to hold the result of the copy operation.

Usage Notes

The `outGeoraster` object is an exact copy of the `inGeoraster` object. To make any changes to the output GeoRaster object during a copy operation, use the SDO_GEOR.changeFormatCopy procedure.

If `inGeoraster` is null, this procedure performs no operation.

If `outGeoraster` has any raster data, it is deleted before the copy operation.

If pyramid data exists for `inGeoraster`, the pyramid data is copied to `outGeoraster`.

An exception is raised if one or more of the following are true:

- `inGeoraster` is invalid.
- `outGeoraster` has not been initialized.
- A raster data table for `outGeoraster` does not exist and `outGeoraster` is not a blank GeoRaster object.

Examples

The following example inserts an initialized GeoRaster object (`gr2`) into the GEORASTER column of table GEORASTER_TABLE, makes `gr2` an exact copy of another GeoRaster object (`gr1`), and updates the row that had been inserted using `gr2` for the GEORASTER column value. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO georaster_table VALUES (11, sdo_geor.init('RDT_11', 1))
    RETURNING georaster INTO gr2;
  SELECT georaster INTO gr1 from georaster_table WHERE georid=1;

  sdo_geor.copy(gr1, gr2);
```

```
  UPDATE georaster_table SET georaster=gr2 WHERE georid=11;
  COMMIT;
END;
/
```

SDO_GEOR.createBlank

Format

```
SDO_GEOR.createBlank(
   rasterType        IN INTEGER,
   ultCoord          IN SDO_NUMBER_ARRAY,
   dimSizes          IN SDO_NUMBER_ARRAY,
   cellValue         IN NUMBER,
   rasterDataTable   IN VARCHAR2 DEFAULT NULL,
   rasterID          IN NUMBER DEFAULT NULL
   ) RETURN SDO_GEORASTER;
```

Description

Creates a blank GeoRaster object, in which all cells have the same value.

Parameters

rasterType
The 5-digit rasterType attribute value, as specified in Section 2.1.1.

ultCoord
An array of the upper-left coordinate integer values for the GeoRaster object. The default value is `(0,0)` for a GeoRaster object without a band dimension, and `(0,0,0)` for a GeoRaster object with a band dimension. If this parameter is null, the default value of 0 is used for each dimension. If a value in the specified array is null, the default value of 0 is used for the corresponding dimension. The value for the band dimension must be 0, and you do not need to specify it. (If you specify an array of values, the number of values must not be less than the number of the spatial dimensions or more than the number of total dimensions.)

dimSizes
The number of cells along each dimension. The number of values in the array must be equal to the total number of dimensions, and the size of each dimension must be explicitly specified. The row and column dimension sizes must be greater than 1.

cellValue
The cell value for all raster cells in the created GeoRaster object. Must be from 0 to 255, because the cell depth of the created GeoRaster object is `8BIT_UNSIGNED`.

rasterDataTable
Name of the object table of type SDO_RASTER that stores the cell data blocks. If you do not specify this parameter, GeoRaster generates a unique table name to be used for the raster data table. If you specify this parameter and the table already exists but is not an object table of type SDO_RASTER, an exception is raised.

rasterID
Number that uniquely identifies the cell blocks of this GeoRaster object in the raster data table. If you do not specify this parameter, a unique sequence number is generated for the ID.

Usage Notes

The created GeoRaster object has no spatial reference information; therefore, its spatial extent geometry has a null SRID (coordinate system) value. The spatial extent geometry reflects the `ultCoord` and `dimSizes` values.

This function does not require that the specified raster data table exist. However, the table must exist before any raster data can be inserted into it.

Although the cell depth of the created GeoRaster object is `8BIT_UNSIGNED`, you can change the cell depth after you create the blank GeoRaster object by calling the SDO_GEOR.changeFormatCopy procedure. You can then call the SDO_GEOR.setBlankCellValue procedure to reset the cell value in a different range.

For guidelines that apply to the SDO_GEOR.createBlank and SDO_GEOR.init functions when a table has multiple GeoRaster object columns, see the Usage Notes for the SDO_GEOR.init function.

An exception is raised if any value for an input parameter is invalid.

Examples

The following example inserts a row containing a blank GeoRaster object into the table. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
INSERT INTO georaster_table (georid, georaster) VALUES (
  1,
  sdo_geor.createBlank(20001, SDO_NUMBER_ARRAY(0,0),
                       SDO_NUMBER_ARRAY(1024,1024), 255, 'RDT_1')
);
```

SDO_GEOR.deletePyramid

Format

```
SDO_GEOR.deletePyramid(
   georaster          IN OUT SDO_GEORASTER);
```

Description

Deletes the pyramid data of a GeoRaster object.

Parameters

georaster
GeoRaster object for which pyramid data is to be deleted.

Usage Notes

For information about pyramid data, see Section 1.7.

If `georaster` is null or has no pyramid data, this procedure performs no operation.

An exception is raised if `georaster` is invalid.

Examples

The following example deletes the pyramid data for a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr1 sdo_georaster;
BEGIN
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=21;

  sdo_geor.deletePyramid(gr1);
  UPDATE georaster_table SET georaster=gr1 WHERE georid=21;
  COMMIT;
END;
/
```

SDO_GEOR.exportTo

Format

```
SDO_GEOR.exportTo(
    georaster       IN SDO_GEORASTER,
    subsetParam     IN VARCHAR2,
    r_destFormat    IN VARCHAR2,
    r_destType      IN VARCHAR2,
    r_destName      IN VARCHAR2,
    h_destFormat    IN VARCHAR2 DEFAULT NULL,
    h_destType      IN VARCHAR2 DEFAULT NULL,
    h_destName      IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.exportTo(
    georaster       IN SDO_GEORASTER,
    subsetParam     IN VARCHAR2,
    r_destFormat    IN VARCHAR2,
    r_destBLOB      IN BLOB);
```

or

```
SDO_GEOR.exportTo(
    georaster       IN SDO_GEORASTER,
    subsetParam     IN VARCHAR2,
    r_destFormat    IN VARCHAR2,
    r_destBLOB      IN BLOB,
    h_destFormat    IN VARCHAR2 DEFAULT NULL,
    h_destCLOB      IN CLOB DEFAULT NULL);
```

Description

Exports a GeoRaster object or a subset of a GeoRaster object to a file or to a BLOB object.

Parameters

georaster
GeoRaster object that will be exported.

subsetParam
String containing subset parameters, for exporting a subset of the GeoRaster object. The format and usage are as explained in Section 1.4.1, although some keywords described in that section do not apply to this procedure. The following keywords are supported:

- pLevel: Pyramid level to be exported. The default is 0.
- cropArea: Specify the area to be exported in the format cropArea = (startCol, startRow, endCol, endRow). startCol is the index of the leftmost pixel to be exported relative to the original image; startRow is the index of the top pixel to be exported; endCol is the index of the rightmost pixel to be exported; and endRow is the index of the bottom pixel to be exported. If cropArea is not specified, the entire image is exported.
- layerNumbers: Layer numbers of the layers to be exported. For example, layerNumbers=(3-5) exports layers 3, 4, and 5; and layerNumbers=(1,3,5) exports layers 1, 3, and 5.

r_destFormat
Raster destination format. Must be one of the following: TIFF, BMP, or PNG. (JPEG and GIF are not supported for this procedure.)

r_destType
Type of destination for the export operation. Must be FILE.

r_destName
Destination file name (with full path specification) if destType is FILE. Do not specify the file extension. If you are using this procedure only to export the world file, specify a null value for this parameter.

r_destBLOB
BLOB object to hold the image file resulting from the export operation.

h_destFormat
Geoheader destination format. Must be WORLDFILE.

h_destType
Geoheader type of destination for the export operation. Must be FILE.

h_destName
Geoheader destination file name (with full path specification) if h_destType is FILE. Do not specify the file extension.

h_destCLOB
CLOB object to hold the geoheader file resulting from the export operation.

Usage Notes

Use a format with both *r_xxx* and *h_xxx* parameters only if the raster image and geoheader are in separate files.

This procedure does not support JPEG or GIF as a destination file format. You can use the client-side GeoRaster exporter tool, described in Section 1.10, to export to a JPEG file.

This procedure does not support GeoRaster objects that have a cellDepth value of 2BIT.

GeoRaster objects with a cell depth of 8 bits or greater that have a BSQ or BIL interleaving are exported in BIP interleaved format.

Before you call this procedure, you must have write permission on the output file or the directory to contain the files. The following example (run as user SYSTEM) grants write permission on a specified file to user HERMAN:

```
call dbms_java.grant_permission('HERMAN','SYS:java.io.FilePermission',
```

```
   'sdo/demos/georaster/data/img1.tif', 'write' );
```

The maximum amount of GeoRaster data that can be exported in a single operation is 67 megabytes (MB). Thus, the maximum dimensions of a GeoRaster object that can be exported at one time must be such that `width*height*bands*cellDepth/8 <= 67 MB` and `rowBlockSize*columnBlockSize*bands*cellDepth/8 <= 67 MB`. For example, for a 3-band, 8-bit GeoRaster object in which the width and height are equal:

- The largest exportable width and height are 4728x4728.
- The largest exportable block dimensions are 4096x4096.

Examples

The following example shows two export operations. The first operation exports an entire GeoRaster object (except for any georeferencing information) into a BMP format file. The second operation exports a subset of the GeoRaster object to a file with an ESRI world file.

```
DECLARE
  geor SDO_GEORASTER;
  fileName VARCHAR2(1024);
  tfwName VARCHAR2(1024);

BEGIN

SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;

-- Export the whole GeoRaster object into a BMP file, excluding any
-- georeferencing information.
sdo_geor.exportTo(geor, NULL, 'BMP', 'file',
  'sdo/demos/georaster/data/img1_export');

-- Export a subset to a file with a world file.
fileName := '/mydir/parrotExported';
tfwName := '/mydir/parrotWorldFile';
SELECT georaster INTO geor FROM georaster_table WHERE georid = 8;
sdo_geor.exportTo(geor, 'cropArea=(0,0,500,500)',
  'TIFF', 'file', fileName, 'WORLDFILE', 'FILE', tfwName);

END;
/
```

The following example exports GeoRaster objects into BLOB and CLOB objects.

```
CREATE TABLE blob_table (blob_col BLOB, blobid NUMBER unique, clob_col CLOB);
INSERT INTO blob_table values (empty_blob(), 3, null);
INSERT INTO blob_table VALUES (empty_blob(), 4, empty_clob());

DECLARE
  lobd1 BLOB;
  lobd2 BLOB;
  lobd3 CLOB;
  geor1 SDO_GEORASTER;
  geor2 SDO_GEORASTER;

BEGIN

-- Example 1: Export to BLOB.
SELECT blob_col INTO lobd1 FROM blob_table WHERE blobid=3 for update;
```

```
SELECT georaster INTO geor1 FROM georaster_table WHERE georid = 13;
sdo_geor.exportTo(geor1, '', 'TIFF', lobd1);
UPDATE blob_table set blob_col = lobd1 WHERE blobid=3;
COMMIT;

-- Example 2: Export GeoRaster to BLOB with world file exported to CLOB.
SELECT blob_col INTO lobd2 FROM blob_table WHERE blobid=4 for update;
SELECT clob_col INTO lobd3 FROM blob_table WHERE blobid=4 for update;
SELECT georaster INTO geor2 FROM georaster_table WHERE georid = 8;
sdo_geor.exportTo(geor2, 'cropArea=(0,0,500,500)', 'TIFF', lobd2,
  'WORLDFILE', lobd3);
UPDATE blob_table set blob_col = lobd2, clob_col = lobd3 WHERE blobid = 4;
COMMIT;

END;
/
```

SDO_GEOR.generatePyramid

Format

```
SDO_GEOR.generatePyramid(
   georaster          IN OUT SDO_GEORASTER,
   pyramidParams  IN VARCHAR2);
```

Description

Generates pyramid data, which is stored together with the original data.

Parameters

georaster
GeoRaster object for which pyramid data is to be generated and stored.

pyramidParams
A string containing the pyramid parameters. See the Usage Notes for information about the available keywords and values.

Usage Notes

For information about pyramid data, see Section 1.7.

`pyramidParams` must be a quoted string that contains one or more of the following keywords, each with an appropriate value:

- `rLevel` (for example, `rLevel=2`): Specifies the number of pyramid levels to create at a smaller (reduced) size than the original object. If you do not specify this keyword, pyramid levels are generated until the smaller of the number of rows or columns is between 64 and 128. The dimension sizes at each lower resolution level are equal to the truncated integer values of the dimension sizes at the next higher resolution level, divided by 2.
- `resampling` (for example, `resampling=NN`): Specifies the resampling method. Must be one of the following: `NN` (value of the nearest neighbor cell in the original GeoRaster object), `BILINEAR` (distance-weighted average of the 4 nearest cells in the original GeoRaster object), `AVERAGE4` (simple average of the 4 nearest cells in the original GeoRaster object), `AVERAGE16` (simple average of the 16 nearest cells in the original GeoRaster object), `CUBIC` (cubic convolution of the 16 nearest cells in the original GeoRaster object).

If `georaster` is null or is a blank GeoRaster object, or if pyramid data exists for `georaster` but it was created with the same pyramid parameters specified in `pyramidParams`, this procedure performs no operation.

If pyramid data exists for `georaster` and it was created using different pyramid parameters from those specified in `pyramidParams`, the old pyramid data is deleted and new pyramid data is generated.

If you do not specify an `rLevel` value, the `rLevel` value is set to the default, which is calculated as follows:

```
(int)(log2(a / 64))
```

In the preceding calculation:

- log2 is a logarithmic function with 2 as its base.
- a is the smaller of the original row or column dimension size.

In the default case, the smaller of the row and column dimension sizes of the top-level overview (the smallest top-level pyramid) is between 64 and 128. If you specify an rLevel value greater than the maximum reduced-resolution level, the rLevel value is set to the maximum reduced-resolution level, which is calculated as follows:

```
(int)(log2(a))
```

In this case, the smaller of the row and column dimension sizes of the top-level overview is 1.

An exception is raised if georaster is invalid.

Examples

The following example creates pyramid data for a GeoRaster object.

```
DECLARE
  gr sdo_georaster;
BEGIN

  SELECT georaster INTO gr
    FROM georaster_table WHERE georid = 6 FOR UPDATE;

  -- Generate pyramids.
  sdo_geor.generatePyramid(gr, 'rLevel=5, resampling=NN');

  -- Update the original GeoRaster object.
  UPDATE georaster_table SET georaster = gr WHERE georid = 6;

  COMMIT;
END;
/
```

SDO_GEOR.generateSpatialExtent

Format

```
SDO_GEOR.generateSpatialExtent(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_GEOMETRY;
```

Description

Generates a Spatial geometry that contains the spatial extent (footprint) of the GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The returned SDO_GEOMETRY object is based on the model coordinate system (SDO_SRID value) of the GeoRaster object. If the GeoRaster object is not georeferenced, the SDO_GEOMETRY object has a null SDO_SRID value, which means the footprint geometry is in cell space.

If the GeoRaster object is georeferenced, the footprint is automatically adjusted, based on its model coordinate location (`CENTER` or `UPPERLEFT`), to cover the whole area in the model space.

If `georaster` is null, this function returns a null SDO_GEOMETRY object.

This function does not set the spatial extent of the GeoRaster object (`spatialExtent` attribute, described in Section 2.1.2). For information about setting the spatial extent, see Section 3.7.

An exception is raised if `georaster` is not valid.

Examples

The following examples return the spatial extent geometry of GeoRaster objects in the GEORASTER column of the GEORASTER_TABLE table. (They refer to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.generateSpatialExtent(georaster) spatialExtent
  FROM georaster_table WHERE georid=2;

SPATIALEXTENT(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINA
--------------------------------------------------------------------------------

SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(0, 0, 256, 0, 511, 0, 511, 256, 511, 511, 256, 511, 0, 511, 0, 256, 0, 0))


SET NUMWIDTH 20
SELECT sdo_geor.generateSpatialExtent(georaster) spatialExtent
  FROM georaster_table WHERE georid=4;
```

```
SPATIALEXTENT(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO,
SDO_ORDINA
--------------------------------------------------------------------------------

SDO_GEOMETRY(2003, 82263, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_AR
RAY(1828466.0909315, 646447.1932945, 1828466.0909315, 644479.85524, 1828466.0909
315, 642512.5171855, 1830433.428986, 642512.5171855, 1832400.7670405, 642512.517
1855, 1832400.7670405, 644479.85524, 1832400.7670405, 646447.1932945, 1830433.42
8986, 646447.1932945, 1828466.0909315, 646447.1932945))
```

SDO_GEOR.georeference

Format

```
SDO_GEOR.georeference(
   georaster                IN OUT SDO_GEORASTER,
   srid                     IN NUMBER,
   modelCoordinateLocation  IN NUMBER,
   xCoefficients            IN SDO_NUMBER_ARRAY,
   yCoefficients            IN SDO_NUMBER_ARRAY);
```

Description

Georeferences a GeoRaster object using specified cell-to-model transformation coefficients.

Parameters

georaster
The SDO_GEORASTER object to be georeferenced.

srid
Model coordinate system. Must not be null or 0 (zero). It can be a value from the SRID column of the MDSYS.CS_SRS table. If it is not a value from the SRID column of the MDSYS.CS_SRS table, the SRID is not supported by Oracle Spatial, and some SRID-related operations may not be supported.

modelCoordinateLocation
A value specifying the model location of the base of the area represented by a cell: `0` for `CENTER` or `1` for `UPPERLEFT`.

xCoefficients
An array specifying the A, B, and C coefficient values in the calculation, as explained in Section 1.6.

yCoefficients
An array specifying the D, E, and F coefficient values in the calculation, as explained in Section 1.6.

Usage Notes

Use this procedure to georeference a GeoRaster object. Georeferencing is explained in Section 1.6 and Section 3.6.

This procedure sets the spatial resolutions of the GeoRaster object.

The following also perform operations related to georeferencing:

- The SDO_GEOR.setSRS procedure sets or deletes georeferencing information.
- The SDO_GEOR.importFrom procedure can load an ESRI world file from a file or from a CLOB object.
- The GeoRaster loader tool (described in Section 1.10) can load an ESRI world file from a file.

Examples

The following example georeferences a GeoRaster object, and it calls the SDO_GEOR.getSRS function to retrieve information related to the spatial referencing of the object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid = 1 FOR UPDATE;
  sdo_geor.georeference(gr, 82394, 1,
                        sdo_number_array(-28.5, 0, 1232804.04),
                        sdo_number_array(0, 28.5, 13678.09));
  UPDATE georaster_table SET georaster = gr WHERE georid = 1;
  COMMIT;
END;
/

PL/SQL procedure successfully completed.

SET NUMWIDTH 20
SELECT georid, sdo_geor.getSRS(georaster) SRS FROM georaster_table
  WHERE georid = 1;

             GEORID
--------------------
SRS(ISREFERENCED, ISRECTIFIED, ISORTHORECTIFIED, SRID,
SPATIALRESOLUTION, SPATIA
--------------------------------------------------------------------------------

                   1
SDO_GEOR_SRS('TRUE', 'TRUE', NULL, 82394, SDO_NUMBER_ARRAY(28.5, 28.5),NULL, NU
LL, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, NULL, NULL, NULL, SDO_NUMBER_ARRAY(1,2, 1, 3,
 -479.93298245614, 0, .0350877192982456), SDO_NUMBER_ARRAY(1, 0, 0, 1,1), SDO_N
UMBER_ARRAY(1, 2, 1, 3, 43256.2821052632, -.0350877192982456, 0), SDO_NUMBER_ARR
AY(1, 0, 0, 1, 1))
```

SDO_GEOR.getBandDimSize

Format

```
SDO_GEOR.getBandDimSize(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the number of bands in a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For an explanation of bands, see Section 1.5.

If `georaster` or its metadata is null, this function returns a null value.

Examples

The following example returns the spatial dimension sizes and the number of bands (one in this case) for a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. The output is reformatted for readability.)

```
SELECT sdo_geor.getSpatialDimSizes(georaster) spatialDimSizes,
       sdo_geor.getBandDimSize(georaster) bandDimSize
  FROM georaster_table WHERE georid=21;

SPATIALDIMSIZES                BANDDIMSIZE
-------------------------      -----------
SDO_NUMBER_ARRAY(512, 512)           1
```

SDO_GEOR.getBeginDateTime

Format

```
SDO_GEOR.getBeginDateTime(
   georaster  IN SDO_GEORASTER
   ) RETURN TIMESTAMP WITH TIME ZONE;
```

Description

Returns the beginning date and time for raster data collection in the metadata for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

To set the beginning date and time for raster data collection in the metadata for a GeoRaster object, use the SDO_GEOR.setBeginDateTime procedure.

If `georaster` or its metadata is null, this function returns a null value.

Examples

The following example returns the beginning and ending dates and times for raster data collection in the metadata for the GeoRaster object in a table named GEORASTER_TABLE where the GEORID column contains the value 4. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.getBeginDateTime(georaster) beginDateTime,
  sdo_geor.getEndDateTime(georaster) endDateTime
  FROM georaster_table WHERE georid=4;

BEGINDATETIME
---------------------------------------------------------------------------
ENDDATETIME
---------------------------------------------------------------------------
01-JAN-00 05.00.00.000000000 AM +00:00
15-NOV-02 08.00.00.000000000 PM +00:00
```

SDO_GEOR.getBinTable

Format

```
SDO_GEOR.getBinTable(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the name of the bin table associated with a layer.

> **Note:** GeoRaster does not perform operations using the BIN function or the bin table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the bin table name. A value of 0 (zero) indicates the object layer.

Usage Notes

This function is relevant only if the bin type is `EXPLICIT`. To retrieve the bin type, use the SDO_GEOR.getBinType function.

To specify a bin table for a layer, use the SDO_GEOR.setBinTable procedure.

See also the information in the Usage Notes for the SDO_GEOR.getBinType function.

If `georaster` or its metadata is null, this function returns a null value.

An exception is raised if `layerNumber` is null, negative, or greater than the maximum layer number.

Examples

The following example returns the name of the bin table for layer number 4 of a specified GeoRaster object in a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getBinTable(georaster, 4) FROM georaster_table WHERE georid=4;
```

SDO_GEOR.getBinType

Format

```
SDO_GEOR.getBinType(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the bin type associated with a layer.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the bin type. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns one of the following bin type values: `LINEAR`, `LOGARITHM`, or `EXPLICIT`.

The `LINEAR` bin type is defined as follows:

```
binNumber = numbins * (cellValue - min) / (max - min) + firstBinNumber
if (binNumber less than 0) binNumber =  firstBinNumber
if (binNumber greater than or equal to numbins) binNumber = numbins +
firstBinNumber - 1
```

The `LOGARITHM` bin type is defined as follows:

```
binNumber = numbins * (ln (1.0 + ((cellValue - min)/(max - min)))/ ln (2.0)) +
firstBinNumber
if (binNumber less than 0) binNumber =  firstBinNumber
if (binNumber greater than or equal to numbins) binNumber = numbins +
firstBinNumber - 1
```

The `EXPLICIT` bin type means that the value (or value range) for each bin is stored in a bin table (which you can set using the SDO_GEOR.setBinTable procedure and retrieve using the SDO_GEOR.getBinTable function).

A bin function maps values or value ranges of the GeoRaster cells to specific bin numbers, which are all integers. GeoRaster does not provide interfaces to manipulate and process bin functions.

If `georaster` or its metadata is null, this function returns a null value.

An exception is raised if `layerNumber` is null, negative, or greater than the maximum layer number.

Examples

The following example returns the bin types for layers 0 and 1 of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getBinType(georaster, 0),1,20) binType0,
       substr(sdo_geor.getBinType(georaster, 1),1,20) binType1
  FROM georaster_table WHERE georid=4;

BINTYPE0        BINTYPE1
--------------- ---------------
EXPLICIT        LINEAR
```

SDO_GEOR.getBlankCellValue

Format

```
SDO_GEOR.getBlankCellValue(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the cell value for all cells if a specified GeoRaster object is a blank GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

In a blank GeoRaster object, all cells have the same cell value. This function returns the cell value for all cells if the specified GeoRaster object is a blank GeoRaster object.

To set the cell value to be used if a specified GeoRaster object is a blank GeoRaster object, use the SDO_GEOR.setBlankCellValue procedure. To determine if a specified GeoRaster object is a blank GeoRaster object, use the SDO_GEOR.isBlank function.

If `georaster` is null, invalid, or is not a blank GeoRaster object, the SDO_GEOR.getBlankCellValue function returns a null value.

Examples

The following example returns the blank cell values for all blank GeoRaster objects in the GEORASTER column of table GEORASTER_TABLE.

```
SELECT georid, sdo_geor.getBlankCellValue(georaster) blankValue
  FROM georaster_table WHERE sdo_geor.isBlank(georaster)='TRUE';

    GEORID BLANKVALUE
---------- ----------
         1        255
         2        155
```

SDO_GEOR.getBlockingType

Format

```
SDO_GEOR.getBlockingType(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the blocking type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns one of the following values: `NONE` or `REGULAR`:

- `NONE` means that the GeoRaster object is not blocked, but is a single BLOB object.
- `REGULAR` means that the GeoRaster object uses regular blocking, that is, each block has the same dimension sizes.

If `georaster` or its metadata is null, this function returns a null value.

Examples

The following example returns the cell depth, interleaving type, and blocking type of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getCellDepth(georaster) CellDepth,
       substr(sdo_geor.getInterleavingType(georaster),1,8) interleavingType,
       substr(sdo_geor.getBlockingType(georaster),1,8) blocking
  FROM georaster_table WHERE georid=21;

 CELLDEPTH INTERLEA BLOCKING
---------- -------- --------
         8 BSQ      REGULAR
```

SDO_GEOR.getBlockSize

Format

```
SDO_GEOR.getBlockSize(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the number of cells for each dimension in each block of a GeoRaster object in an array showing the number of cells for each row, column, and (if relevant) band.

Parameters

georaster
GeoRaster object.

Usage Notes

If `georaster` or its metadata is null, or if `georaster` is not blocked, this function returns a null value.

Examples

The following example returns the number of cells (512 in each dimension) in each block of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getBlockSize(georaster) blockSize
   FROM georaster_table WHERE georid=21;

BLOCKSIZE
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(512, 512)
```

SDO_GEOR.getCellCoordinate

Format

```
SDO_GEOR.getCellCoordinate(
   georaster          IN SDO_GEORASTER,
   pyramidLevel       IN NUMBER,
   modelCoordinate  IN SDO_GEOMETRY
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the coordinates in the cell (raster) coordinate system associated with the point at the specified model (ground) coordinates.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level containing the cell specified in `modelCoordinate`.

modelCoordinate
A point geometry that contains two coordinates, such as (longitude,latitude) or (x,y) values, identifying the point in the model (ground) coordinate system.

Usage Notes

Use this function to transform a point in the ground coordinate system (a longitude, latitude pair) to the location of a point on the GeoRaster image.

Contrast this function with the SDO_GEOR.getModelCoordinate function, which returns a point geometry containing the coordinates in the model (ground) coordinate system associated with the point at the specified cell coordinates.

Examples

The following example returns the cell coordinates in the raster image associated with model coordinate values (32343.64,7489527.23) in a specified GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.getCellCoordinate(georaster, 0, sdo_geometry(2001,82394,
  sdo_point_type(32343.64,7489527.23,null), null,null)) coord
FROM georaster_table WHERE georid=4;

COORD
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(100, 100)
```

SDO_GEOR.getCellDepth

Format

```
SDO_GEOR.getCellDepth(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the cell depth in bits.

Parameters

georaster
GeoRaster object.

Usage Notes

The cell depth determines the precision and the data size of an image. As the cell depth value decreases, less disk space is needed to store the image; as the cell depth value increases, more disk space is needed to store the image.

To return the cell depth as a string (such as `32BIT_S`) instead of a number, you can use the XMLType PL/SQL interface `extract`. The possible string values are listed in the `cellDepthType` definition in the GeoRaster metadata XML schema, which is described in Appendix A. The following example returns a string value for the cell depth of the GeoRaster object with the GEORID column value of 21 in the GEORASTER_TABLE table:

```
SELECT t.georaster.metadata.extract(
         '/georasterMetadata/rasterInfo/cellDepth/text()',
         'xmlns=http://xmlns.oracle.com/spatial/georaster')
  FROM georaster_table t WHERE t.georid=21;
```

Examples

The following example returns the cell depth, interleaving type, and blocking type of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getCellDepth(georaster) CellDepth,
       substr(sdo_geor.getInterleavingType(georaster),1,8) interleavingType,
       substr(sdo_geor.getBlockingType(georaster),1,8) blocking
  FROM georaster_table WHERE georid=21;

 CELLDEPTH INTERLEA BLOCKING
---------- -------- --------
         8 BSQ      REGULAR
```

SDO_GEOR.getCellValue

Format

```
SDO_GEOR.getCellValue(
   georaster      IN SDO_GEORASTER,
   pyramidLevel IN NUMBER,
   rowNumber   IN NUMBER,
   colNumber     IN NUMBER,
   bandNumber  IN NUMBER
   ) RETURN NUMBER;
```

or

```
SDO_GEOR.getCellValue(
   georaster      IN SDO_GEORASTER,
   pyramidLevel IN NUMBER,
   ptGeom         IN SDO_GEOMETRY,
   layerNumber  IN NUMBER
   ) RETURN NUMBER;
```

Description

Returns the value of a single cell located anywhere in the GeoRaster object by specifying its row, column, and band number in its cell coordinate system, or by specifying a point geometry in its model coordinate system and its logical layer number.

To change the value of raster data cells in a specified window of a GeoRaster object, use the SDO_GEOR.changeCellValue procedure.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level containing the cell whose value is to be returned.

rowNumber
Number of the row that contains the cell whose value is to be returned.

colNumber
Number of the column that contains the cell whose value is to be returned.

bandNumber
Number of the physical band that contains the cell whose value is to be returned.

ptGeom
Point geometry that identifies the cell whose value is to be returned.

layerNumber
Number of the logical layer that contains the cell whose value is to be returned. (As mentioned in Section 1.5, the logical layer number is the physical band number plus 1.)

Usage Notes

This function returns the original cell value stored in the raster object. It does not apply the scaling function defined in the metadata (which is typically used to scale the original cell data to a desired value or range of values), and it does not apply the BIN function. To get the scaled cell value, follow these steps:

1. Call the SDO_GEOR.getCellValue function to return the original cell value.
2. Call the SDO_GEOR.getScaling function to return the coefficients of the scaling function (a_0, a_1, b_0, b_1).
3. Using PL/SQL or another programming language, calculate the result using the following formula:

   $value = (a_0 + a_1 * cellvalue) / (b_0 + b_1 * cellvalue)$

Examples

The following example returns the values of four cells of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getCellValue(georaster,0,383,47,0) V383_47,
       sdo_geor.getCellValue(georaster,0,47,383,0) V47_383,
       sdo_geor.getCellValue(georaster,0,128,192,0) V128_192,
       sdo_geor.getCellValue(georaster,0,320,256,0) V320_256
  FROM georaster_table WHERE georid=21;

   V383_47    V47_383   V128_192   V320_256
---------- ---------- ---------- ----------
        48         55         52         53
```

SDO_GEOR.getColorMap

Format

```
SDO_GEOR.getColorMap(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_GEOR_COLORMAP;
```

Description

Returns the colormap for pseudocolor display of a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the colormap. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns an object of type SDO_GEOR_COLORMAP. Section 2.3.2 describes colormaps and this object type.

To set the colormap for a layer in a GeoRaster object, use the SDO_GEOR.setColorMap procedure.

If `georaster` or its metadata is null, this function returns a null value.

An exception is raised if `layerNumber` is null, negative, or greater than the maximum layer number.

Examples

The following example returns the colormap for layer 1 of a GeoRaster object. (Part of the output is omitted.)

```
SELECT sdo_geor.getColorMap(georaster, 1) FROM georaster_table
   WHERE georid = 4;

SDO_GEOR.GETCOLORMAP(GEORASTER,1)(CELLVALUE, RED, GREEN, BLUE, ALPHA)
--------------------------------------------------------------------------------
SDO_GEOR_COLORMAP(SDO_NUMBER_ARRAY(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126,
127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158,
159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190,
191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206,
```

```
207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,
223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238,
239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254,
255), SDO_NUMBER_ARRAY(180, 180, 180, 180, 180, 180, 180, 180, 180, 180, 180, 18
0, 127, 127, 100, 50, 50, 127, 159, 191, 223, 255, 255, 255, 255, 218, 182, 145,
 109, 72, 36, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 36, 72, 109, 145, 182, 218, 255, 200, 206, 212, 218, 224, 230, 236, 242,
248, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255), SDO_NUMBER_ARRAY(127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 180, 127, 50, 100, 50, 127,
159, 191, 223, 255, 200, 150, 100, 122, 144, 166, 188, 210, 232, 255, 255, 255,
248, 241, 234, 227, 220, 213, 206, 200, 150, 100, 87, 75, 62, 50, 37, 25, 12, 0,
200, 127, 0, 0, 0, 0, 0, 0, 0, 0, 0, 28, 56, 85, 113, 141, 170, 198, 226, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255), SDO_NUMBER_ARRAY(127, 127, 127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 180, 50, 50, 100, 127, 95, 63, 31,
0, 0, 0, 0, 18, 36, 54, 72, 90, 108, 127, 100, 50, 43, 37, 31, 25, 18, 12, 6, 0,
0, 0, 31, 63, 95, 127, 159, 191, 223, 255, 255, 255, 127, 108, 90, 72, 54, 36,
18, 0, 0, 28, 56, 85, 113, 141, 170, 198, 226, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255), SDO_NUMBER_ARRAY(255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
```

```
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255))
```

SDO_GEOR.getColorMapTable

Format

```
SDO_GEOR.getColorMapTable(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the colormap table for pseudocolor display of a layer in a GeoRaster object.

Note: GeoRaster does not perform operations using the colormap table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the colormap table. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns the name of a user-defined colormap table. For information about colormaps, see Section 2.3.2.

To set the colormap table for a layer in a GeoRaster object, use the SDO_GEOR.setColorMapTable procedure.

If `georaster` or its metadata is null, this function returns a null value.

An exception is raised if `layerNumber` is null, negative, or greater than the maximum layer number.

Examples

The following example returns the colormap table for layer 2 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getColorMapTable(georaster, 2) FROM georaster_table WHERE
georid=4;

SDO_GEOR.GETCOLORMAPTABLE(GEORASTER,2)
--------------------------------------------------------------------------------
CMT1

1 row selected.
```

SDO_GEOR.getCompressionType

Format

```
SDO_GEOR.getCompressionType(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the compression type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function can return `DEFLATE`, `JPEG-B`, `JPEG-F`, or `NONE` (the latter value meaning that the GeoRaster object is not compressed). For information about GeoRaster compression, see Section 1.8.

Examples

The following example returns the compression type for the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, substr(sdo_geor.getCompressionType(georaster),1,20) compressionType
  FROM georaster_table;

    GEORID COMPRESSIONTYPE
---------- --------------------
         2 DEFLATE
         4 JPEG-B
```

SDO_GEOR.getDefaultBlue

Format

```
SDO_GEOR.getDefaultBlue(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The default red, green, and blue values are used for true-color displays, not for pseudocolor or grayscale displays. These values are optional, and they are intended for use only when visualizing multilayer or hyperspectral GeoRaster objects.

You can return the layer numbers for all three color components (RGB) by using the SDO_GEOR.getDefaultColorLayer function.

Examples

The following example returns the layer numbers for the red, blue, and green color components for displaying the GeoRaster objects in the table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getDefaultRed(georaster) red,
  sdo_geor.getDefaultGreen(georaster) green,
  sdo_geor.getDefaultBlue(georaster) blue
FROM georaster_table;

    GEORID        RED      GREEN       BLUE
---------- ---------- ---------- ----------
         1          1          2          3
         2
         3         31         20         13
```

SDO_GEOR.getDefaultColorLayer

Format

```
SDO_GEOR.getDefaultColorLayer(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The RGB layer numbers returned are used for true-color displays, not for pseudocolor or grayscale displays.

You can return the layer number for each color component (RGB) by using the SDO_GEOR.getDefaultRed, SDO_GEOR.getDefaultGreen, and SDO_GEOR.getDefaultBlue functions.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GEORASTER column) in table GEORASTER_TABLE, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 2);
  sdo_geor.setDefaultGreen(grobj, 3);
  sdo_geor.setDefaultBlue(grobj, 1);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(georaster) FROM georaster_table WHERE
georid=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(GEORASTER)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(2, 3, 1)

1 row selected.
```

SDO_GEOR.getDefaultGreen

Format

```
SDO_GEOR.getDefaultGreen(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The default red, green, and blue values are used for true-color displays, not for pseudocolor or grayscale displays. These values are optional, and they are intended for use only when visualizing multilayer or hyperspectral GeoRaster objects.

You can return the layer numbers for all three color components (RGB) by using the SDO_GEOR.getDefaultColorLayer function.

Examples

The following example returns the layer numbers for the red, blue, and green color components for displaying the GeoRaster objects in the table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getDefaultRed(georaster) red,
  sdo_geor.getDefaultGreen(georaster) green,
  sdo_geor.getDefaultBlue(georaster) blue
FROM georaster_table;

    GEORID        RED      GREEN       BLUE
---------- ---------- ---------- ----------
         1          1          2          3
         2
         3         31         20         13
```

SDO_GEOR.getDefaultRed

Format

```
SDO_GEOR.getDefaultRed(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The default red, green, and blue values are used for true-color displays, not for pseudocolor or grayscale displays. These values are optional, and they are intended for use only when visualizing multilayer or hyperspectral GeoRaster objects.

You can return the layer numbers for all three color components (RGB) by using the SDO_GEOR.getDefaultColorLayer function.

Examples

The following example returns the layer numbers for the red, blue, and green color components for displaying the GeoRaster objects in the table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getDefaultRed(georaster) red,
  sdo_geor.getDefaultGreen(georaster) green,
  sdo_geor.getDefaultBlue(georaster) blue
FROM georaster_table;

    GEORID        RED      GREEN       BLUE
---------- ---------- ---------- ----------
         1          1          2          3
         2
         3         31         20         13
```

SDO_GEOR.getEndDateTime

Format

```
SDO_GEOR.getEndDateTime(
   georaster  IN SDO_GEORASTER
   ) RETURN TIMESTAMP WITH TIME ZONE;
```

Description

Returns the ending date and time for raster data collection in the metadata for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

To set the ending date and time for raster data collection in the metadata for a GeoRaster object, use the SDO_GEOR.setEndDateTime procedure.

If `georaster` or its metadata is null, this function returns a null value.

Examples

The following example returns the beginning and ending dates and times for raster data collection in the metadata for the GeoRaster object in a table named GEORASTER_TABLE where the GEORID column contains the value 4. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.getBeginDateTime(georaster) beginDateTime,
  sdo_geor.getEndDateTime(georaster) endDateTime
  FROM georaster_table WHERE georid=4;

BEGINDATETIME
---------------------------------------------------------------------------
ENDDATETIME
---------------------------------------------------------------------------
01-JAN-00 05.00.00.000000000 AM +00:00
15-NOV-02 08.00.00.000000000 PM +00:00
```

SDO_GEOR.getGrayScale

Format

```
SDO_GEOR.getGrayScale(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_GEOR_GRAYSCALE;
```

Description

Returns the grayscale mappings for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the grayscale mappings. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns an object of type SDO_GEOR_GRAYSCALE. Section 2.3.3 describes grayscale display and this object type.

To set the grayscale mappings for a layer in a GeoRaster object, use the SDO_GEOR.setGrayScale procedure.

Examples

The following example returns the grayscale mappings for layer 0 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 0 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getGrayScale(georaster, 0) FROM georaster_table WHERE georid=0;

SDO_GEOR.GETGRAYSCALE(GEORASTER,0)(CELLVALUE, GRAY)
--------------------------------------------------------------------------------
SDO_GEOR_GRAYSCALE(SDO_NUMBER_ARRAY(10, 20, 30, 255), SDO_NUMBER_ARRAY(180, 210,
230, 250))
```

SDO_GEOR.getGrayScaleTable

Format

```
SDO_GEOR.getGrayScaleTable(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the grayscale mapping table for a layer in a GeoRaster object.

> **Note:** GeoRaster does not perform operations using the grayscale mapping table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the grayscale mapping table. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns the name of a user-defined grayscale table. Section 2.3.3 describes grayscale display.

To set the grayscale mapping table for a layer in a GeoRaster object, use the SDO_GEOR.setGrayScaleTable procedure.

Examples

The following example returns the grayscale mapping tables for layers 0, 1, 2, and 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getGrayScaleTable(georaster, 0),1,20) grayScaleTable0,
       substr(sdo_geor.getGrayScaleTable(georaster, 1),1,20) grayScaleTable1,
       substr(sdo_geor.getGrayScaleTable(georaster, 2),1,20) grayScaleTable2,
       substr(sdo_geor.getGrayScaleTable(georaster, 3),1,20) grayScaleTable3
  FROM georaster_table WHERE georid=4;

GRAYSCALETABLE0      GRAYSCALETABLE1      GRAYSCALETABLE2      GRAYSCALETABLE3
-------------------- -------------------- -------------------- -----------------
SCL0                 SCL1                 SCL2                 SCL3
```

SDO_GEOR.getHistogram

Format

```
SDO_GEOR.getHistogram(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_GEOR_HISTOGRAM;
```

Description

Returns the histogram for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the histogram. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns an object of type SDO_GEOR_HISTOGRAM. Section 2.3.1 describes this object type and briefly discusses histograms.

Examples

The following example returns the histogram for layer 1 of a 4-bit GeoRaster object in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getHistogram(georaster, 1) layer1
 FROM georaster_table WHERE georid=17;

LAYER1(CELLVALUE, COUNT)
--------------------------------------------------------------------------------
SDO_GEOR_HISTOGRAM(SDO_NUMBER_ARRAY(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12, 13,
 14, 15), SDO_NUMBER_ARRAY(10, 18, 10, 110, 200, 120, 130, 150, 160, 103, 106,
 190, 12, 17, 10, 5))
```

SDO_GEOR.getHistogramTable

Format

```
SDO_GEOR.getHistogramTable(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the histogram table for a layer in a GeoRaster object.

> **Note:** GeoRaster does not perform operations using the histogram table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the name of the histogram table. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns a user-defined histogram table. Section 2.3.1 briefly discusses histograms.

To set the name of the histogram table for a layer, use the SDO_GEOR.setHistogramTable procedure.

Examples

The following example returns the histogram tables for layers 0 (the whole object), 1, 2, and 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getHistogramTable(georaster, 0),1,20) histogramTable0,
       substr(sdo_geor.getHistogramTable(georaster, 1),1,20) histogramTable1,
       substr(sdo_geor.getHistogramTable(georaster, 2),1,20) histogramTable2,
       substr(sdo_geor.getHistogramTable(georaster, 3),1,20) histogramTable3
  FROM georaster_table WHERE georid=4;

HISTOGRAMTABLE0      HISTOGRAMTABLE1      HISTOGRAMTABLE2      HISTOGRAMTABLE3
-------------------- -------------------- -------------------- ----------------
HIST0                HIST1                HIST2                HIST3
```

SDO_GEOR.getID

Format

```
SDO_GEOR.getID(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the user-defined identifier value associated with a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

To set a user-defined identifier value for a GeoRaster object, use the SDO_GEOR.setID procedure.

Examples

The following example returns the user-defined identifier values of the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, substr(sdo_geor.getID(georaster),1,50) GEOR_ID
  FROM georaster_table;

    GEORID GEOR_ID
---------- --------------------------------------------------
         2 TM_102
         4 TM_104
```

SDO_GEOR.getInterleavingType

Format

```
SDO_GEOR.getInterleavingType(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the interleaving type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns one of the following values: `BSQ` (band sequential), `BIL` (band interleaved by line), or `BIP` (band interleaved by pixel).

To change the interleaving type for a GeoRaster object, use the SDO_GEOR.changeFormatCopy procedure, and use the `interleaving` keyword in the `storageParam` parameter string.

Examples

The following example returns the cell depth, interleaving type, and blocking type of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getCellDepth(georaster) CellDepth,
       substr(sdo_geor.getInterleavingType(georaster),1,8) interleavingType,
       substr(sdo_geor.getBlockingType(georaster),1,8) blocking
  FROM georaster_table WHERE georid=21;

 CELLDEPTH INTERLEA BLOCKING
---------- -------- --------
         8 BSQ      REGULAR
```

SDO_GEOR.getLayerDimension

Format

```
SDO_GEOR.getLayerDimension(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_STRING_ARRAY;
```

Description

Returns the dimension that is mapped as the logical layer dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The **layer dimension** refers to the physical entity associated with the logical term *layer*. For the current release, the only supported layer dimension is `BAND`: that is, the logical concept *layer* is associated with the physical term *band*, as shown in Figure 1–4 in Section 1.5. In this case, layers will be mapped to the `BAND` dimension, so that the first layer is band 0, the second layer is band 1, and so on.

Examples

The following example returns the layer dimension of each GeoRaster object (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1. (The output is reformatted for readability.)

```
SELECT georid, sdo_geor.getLayerDimension(georaster) FROM georaster_table;

    GEORID SDO_GEOR.GETLAYERDIMENSION(GEORASTER)
---------- --------------------------------------------------------------------
         2 SDO_STRING_ARRAY('BAND')
         4 SDO_STRING_ARRAY('BAND')
```

SDO_GEOR.getLayerID

Format

```
SDO_GEOR.getLayerID(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the user-defined identifier value associated with a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the user-defined identifier value. A value of 0 (zero) indicates the object layer.

Usage Notes

To set a user-defined identifier value for a layer in a GeoRaster object, use the SDO_GEOR.setLayerID procedure.

Examples

The following example returns the user-defined identifier values of layers 0, 1, 2, and 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT substr(sdo_geor.getLayerID(georaster, 0),1,12) layerID0,
       substr(sdo_geor.getLayerID(georaster, 1),1,12) layerID1,
       substr(sdo_geor.getLayerID(georaster, 2),1,12) layerID2,
       substr(sdo_geor.getLayerID(georaster, 3),1,12) layerID3
  FROM georaster_table WHERE georid=4;

LAYERID0     LAYERID1     LAYERID2     LAYERID3
------------ ------------ ------------ ------------
TM543        TM3          TM4          TM5
```

SDO_GEOR.getLayerOrdinate

Format

```
SDO_GEOR.getLayerOrdinate(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN NUMBER;
```

Description

Returns the band ordinate for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the physical band ordinate. A value of 0 (zero) indicates the object layer.

Usage Notes

The returned number refers to the physical band that a layer (`layerNumber` parameter value) is associated with. For the current release, by default the associations are as shown in Figure 1–4 in Section 1.5: layer 1 is band 0, layer 2 is band 1, and so on.

To set the band ordinate value for a layer, use the SDO_GEOR.setLayerOrdinate procedure.

Examples

The following example returns the band numbers associated with layers 0, 1, 2, and 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getLayerOrdinate(georaster, 0) layerOrdinate0,
       sdo_geor.getLayerOrdinate(georaster, 1) layerOrdinate1,
       sdo_geor.getLayerOrdinate(georaster, 2) layerOrdinate2,
       sdo_geor.getLayerOrdinate(georaster, 3) layerOrdinate3
  FROM georaster_table WHERE georid=4;

LAYERORDINATE0 LAYERORDINATE1 LAYERORDINATE2 LAYERORDINATE3
-------------- -------------- -------------- --------------
             0              1              2
```

SDO_GEOR.getModelCoordinate

Format

```
SDO_GEOR.getModelCoordinate(
   georaster       IN SDO_GEORASTER,
   pyramidLevel    IN NUMBER,
   cellCoordinate  IN SDO_NUMBER_ARRAY
   ) RETURN SDO_GEOMETRY;
```

Description

Returns a point geometry object that contains the coordinates in the model (ground) coordinate system associated with the point at the specified cell (raster) coordinates.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level containing the cell specified in `cellCoordinate`.

cellCoordinate
Array of two coordinates identifying the point in the cell coordinate system. The two coordinates are the row number and column number of the point.

Usage Notes

Use this function to transform the location of a point on the GeoRaster object to the longitude and latitude coordinates of its associated point in the ground coordinate system.

The input GeoRaster data must be georeferenced. The resulting geometry has the same SDO_SRID value as the input GeoRaster object.

Contrast this function with the SDO_GEOR.getCellCoordinate function, which returns the coordinates in the cell (raster) coordinate system associated with the point at the specified model (ground) coordinates.

Examples

The following example returns a point geometry object containing the model coordinates associated with cell coordinates (100,100) in a specified GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SET NUMWIDTH 20
SELECT sdo_geor.getModelCoordinate(georaster, 0,
sdo_number_array(100,100)) mcoord
  FROM georaster_table WHERE georid=4;

MCOORD(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------

SDO_GEOMETRY(2001, 82394, SDO_POINT_TYPE(347.666315789474, 43274.9052631579, NUL
```

```
L), NULL, NULL)
```

SDO_GEOR.getModelSRID

Format

```
SDO_GEOR.getModelSRID(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the coordinate system (SDO_SRID value) associated with the model (ground) space for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns a null value if no coordinate system is associated with the model space.

To set the coordinate system (SDO_SRID value) associated with the model space, use the SDO_GEOR.setModelSRID procedure.

Examples

The following example returns the SDO_SRID values associated with the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getModelSRID(georaster) SRID FROM georaster_table;

    GEORID       SRID
---------- ----------
         2       82394
         4       82394
```

SDO_GEOR.getNODATA

Format

```
SDO_GEOR.getNODATA(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the value representing NODATA cells in a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

Some cells of a GeoRaster object may have no meaningful value assigned or collected. The NODATA value is the cell value for those cells, and it means that those cells are not semantically defined. The application is responsible for defining the meaning or significance of cells identified as NODATA cells.

If this function returns a null value, it means that all cells of the GeoRaster object are defined and have a meaningful cell value.

Examples

The following example returns the value to be used for NODATA cells in the GeoRaster objects (GEORASTER column) in table GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getNODATA(georaster) NODATA from georaster_table;

    GEORID     NODATA
---------- ----------
         1
         2    -9999.99
```

SDO_GEOR.getPyramidMaxLevel

Format

```
SDO_GEOR.getPyramidMaxLevel(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the level number of the top pyramid of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For information about pyramids, see Section 1.7.

Examples

The following example returns the pyramid type and level number of the top pyramid for the GeoRaster object (GEORASTER column) in the row with an GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT substr(sdo_geor.getPyramidType(georaster),1,10) pyramidType,
       sdo_geor.getPyramidMaxLevel(georaster) maxLevel
  FROM georaster_table WHERE georid=21;

PYRAMIDTYP   MAXLEVEL
---------- ----------
DECREASE            3
```

SDO_GEOR.getPyramidType

Format

```
SDO_GEOR.getPyramidType(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the pyramid type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The pyramid type can be `NONE` (no pyramids) or `DECREASE`.

For information about pyramids, see Section 1.7.

Examples

The following example returns the pyramid type and level number of the top pyramid for the GeoRaster object (GEORASTER column) in the row with an GEORID column value of 21 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT substr(sdo_geor.getPyramidType(georaster),1,10) pyramidType,
       sdo_geor.getPyramidMaxLevel(georaster) maxLevel
  FROM georaster_table WHERE georid=21;

PYRAMIDTYP   MAXLEVEL
---------- ----------
DECREASE            3
```

SDO_GEOR.getRasterBlocks

Format

```
SDO_GEOR.getRasterBlocks(
   georaster      IN SDO_GEORASTER,
   pyramidLevel   IN NUMBER,
   window         IN SDO_NUMBER_ARRAY
   ) RETURN SDO_RASTERSET;
```

or

```
SDO_GEOR.getRasterBlocks(
   georaster      IN SDO_GEORASTER,
   pyramidLevel   IN NUMBER,
   window         IN SDO_GEOMETRY
   ) RETURN SDO_RASTERSET;
```

Description

Returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that have any spatial interaction with a specified window.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level from which to return the blocks that have any spatial interaction with the specified window.

window
Window from which to return the blocks that are in `pyramidLevel`. The data type can be SDO_NUMBER_ARRAY or SDO_GEOMETRY. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_GEOMETRY, see the Usage Notes for SDO_SRID requirements.

Usage Notes

The SDO_RASTERSET collection type is described in Section 2.3.4.

If the `window` parameter data type is SDO_GEOMETRY, the SDO_SRID value must be one of the following:

- Null, to specify raster space
- A value from the SRID column of the MDSYS.CS_SRS table

  If the SDO_SRID values for the `window` parameter geometry and the model space are different, the `window` parameter geometry is automatically transformed to the

coordinate system of the model space before the operation is performed. (Raster space and model space are explained in Section 1.3.)

Examples

The following example returns a collection set that identifies all raster blocks that have any spatial interaction with the specified window. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr sdo_georaster;
  ds sdo_rasterset;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=2;
  ds := sdo_geor.getRasterBlocks(gr, 0, sdo_number_array(11,65,192,244));
  COMMIT;
END;
/
```

SDO_GEOR.getRasterData

Format

```
SDO_GEOR.getRasterData(
   georaster       IN SDO_GEORASTER,
   pyramidLevel    IN NUMBER,
   rasterBlob      IN OUT BLOB,
   storageParam    IN VARCHAR2 DEFAULT NULL);
```

Description

Creates a single BLOB object that contains all raster data of the input GeoRaster object at the specified pyramid level.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level for which to perform the operation.

rasterBlob
BLOB object to hold the result.

storageParam
A string specifying storage parameters to be applied in creating `rasterBlob`. The only `storageParam` keywords supported for this procedure are `celldepth`, `compression`, `interleaving`, and `quality`; all other keywords are ignored. Storage parameters are explained in Section 1.4.1.

If `storageParam` is null or not specified, the cell depth, interleaving, and compression type (and compression quality, if applicable) are the same as for the input GeoRaster object.

Usage Notes

If the GeoRaster object is blocked, the mosaic of all blocks of the specified pyramid level is returned.

After the procedure completes, the `rasterBlob` object contains the cell (pixel) data without tiling.

You can specify compression even if the input GeoRaster object is not compressed or is compressed in a different format from what you specify in the `storageParam` parameter. To have decompressed output for a compressed input GeoRaster object, specify `compression=NONE` in the `storageParam` parameter. For information about GeoRaster compression and decompression, see Section 1.8.

Examples

The following example creates a BLOB object, using full-format baseline JPEG (JPEG-F) compression, with all raster data from the GeoRaster object whose ID value is

2 in the GEORASTER_TABLE table. The definition of this table is presented after Example 1–1 in Section 1.4.1.

```
DECLARE
  gr sdo_georaster;
  lb blob;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=2;
  dbms_lob.createTemporary(lb, FALSE);
  sdo_geor.getRasterData(gr, 0, lb, 'compress=JPEG-F');
  dbms_lob.freeTemporary(lb);
END;
/
```

SDO_GEOR.getRasterSubset

Format

```
SDO_GEOR.getRasterSubset(
   georaster       IN SDO_GEORASTER,
   pyramidLevel    IN NUMBER,
   window          IN SDO_NUMBER_ARRAY,
   bandNumbers     IN VARCHAR2,
   rasterBlob      IN OUT BLOB,
   storageParam    IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.getRasterSubset(
   georaster       IN SDO_GEORASTER,
   pyramidLevel    IN NUMBER,
   window          IN SDO_GEOMETRY,
   layerNumbers    IN VARCHAR2,
   rasterBlob      IN OUT BLOB,
   storageParam    IN VARCHAR2 DEFAULT NULL);
```

Description

Creates a single BLOB object containing all cells of a specified pyramid level that are inside or on the boundary of a specified window.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level on which to perform the operation.

window
A rectangular window from which to crop the cells. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_GEOMETRY, the MBR of the geometry object is used as the window; see also the Usage Notes for SDO_SRID requirements.

If `window` is of type SDO_NUMBER_ARRAY, use the `bandNumbers` parameter to specify one or more band numbers; if `window` is of type SDO_GEOMETRY, use the `layerNumbers` parameter to specify one or more layer numbers.

layerNumbers
A string identifying the logical layer numbers on which the operation or operations are to be performed. Use commas to delimit the values, and a hyphen to indicate a

range (for example, `2-4` for layers 2, 3, and 4). If you specify a null value for this parameter, the operation or operations are performed on all layers.

bandNumbers
A string identifying the physical band numbers on which the operation or operations are to be performed. Use commas to delimit the values, and a hyphen to indicate a range (for example, `1-3` for bands 1, 2, and 3). If you specify a null value for this parameter, the operation or operations are performed on all bands.

rasterBlob
BLOB object to hold the result (the mosaicked raster subset) of the operation. It must exist or have been initialized before the operation.

storageParam
A string specifying storage parameters to be applied in creating `rasterBlob`. The only supported `storageParam` keywords supported for this procedure are `celldepth`, `compression`, `interleaving`, and `quality`; all other keywords are ignored. Storage parameters are explained in Section 1.4.1.

If storageParam is null or not specified, the cell depth, interleaving, and compression type (and compression quality, if applicable) are the same as for the input GeoRaster object.

Usage Notes

If the `window` parameter data type is SDO_GEOMETRY, the SDO_SRID value must be one of the following:

- Null, to specify raster space
- A value from the SRID column of the MDSYS.CS_SRS table

  If the SDO_SRID values for the `window` parameter geometry and the model space are different, the `window` parameter geometry is automatically transformed to the coordinate system of the model space before the operation is performed. (Raster space and model space are explained in Section 1.3.)

After the procedure completes, the `rasterBLOB` parameter contains the cell (pixel) data in the cropped window without tiling. The BLOB has no padding, except when the cell depth is less than 8 bits and the total number of bits needed for the output cannot be divided by 8; in these cases, unlike normal padding, only the last byte of the result is padded with 0 (zeros) for the trailing bits.

You can specify compression even if the input GeoRaster object is not compressed or is compressed in a different format from what you specify in the `storageParam` parameter. To have decompressed output for a compressed input GeoRaster object, specify `compression=NONE` in the `storageParam` parameter. For information about GeoRaster compression and decompression, see Section 1.8.

Examples

The following example creates a BLOB object with all raster data from a specified window in pyramid level 0 of the GeoRaster object whose ID value is 4 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
DECLARE
  gr sdo_georaster;
  lb blob;
BEGIN
```

```
  SELECT georaster INTO gr FROM georaster_table WHERE georid=4;
  dbms_lob.createTemporary(lb, FALSE);
  sdo_geor.getRasterSubset(gr, 0, sdo_number_array(-21,100,100,200), null, lb);
  dbms_lob.freeTemporary(lb);
END;
/
```

SDO_GEOR.getScaling

Format

```
SDO_GEOR.getScaling(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the coefficients of the scaling function for a layer of a GeoRaster object.

Note: GeoRaster does not perform operations using the scaling function in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the coefficients. A value of 0 (zero) indicates the object layer.

Usage Notes

The scaling function is as follows:

$value = (a_0 + a_1 * cellvalue) / (b_0 + b_1 * cellvalue)$

The order of the coefficients is: a_0, a_1, b_0, b_1.

Examples

The following example returns the scaling coefficients for layer number 0 (the whole object) of a specified GeoRaster object in a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. It scales original value range 0.0 to 1000.0 to be in the range 0.0 to 250.0.

```
SELECT sdo_geor.getScaling(georaster, 0) FROM georaster_table WHERE georid=0;

SDO_GEOR.GETSCALING(GEORASTER,0)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0.0, 0.25, 1, 0.0)
```

SDO_GEOR.getSpatialDimNumber

Format

```
SDO_GEOR.getSpatialDimNumber(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the number of spatial dimensions of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For the current release, this function always returns 2.

To return the number of cells in each spatial dimension of a GeoRaster object, use the SDO_GEOR.getSpatialDimSizes function.

Examples

The following example returns the GEORID column value, the number of spatial dimensions, and the number of cells in each spatial dimension for the GeoRaster objects in the table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. (The output is reformatted for readability.)

```
SELECT georid, sdo_geor.getSpatialDimNumber(georaster) spatialDim,
          sdo_geor.getSpatialDimSizes(georaster) spatialDimSizes
   FROM georaster_table;

    GEORID SPATIALDIM SPATIALDIMSIZES
---------- ---------- ------------------------------------------------------
         0          2 SDO_NUMBER_ARRAY(1024, 1024)

         1          2 SDO_NUMBER_ARRAY(384, 251)

         2          2 SDO_NUMBER_ARRAY(512, 512)

         4          2 SDO_NUMBER_ARRAY(512, 512)

        11          2 SDO_NUMBER_ARRAY(7957, 5828)
```

SDO_GEOR.getSpatialDimSizes

Format

```
SDO_GEOR.getSpatialDimSizes(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the number of cells in each spatial dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

To return the number of spatial dimensions for a GeoRaster object, use the SDO_GEOR.getSpatialDimNumber function.

Examples

The following example returns the spatial dimension sizes and the number of bands for a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. The output is reformatted for readability.)

```
SELECT sdo_geor.getSpatialDimSizes(georaster) spatialDimSizes,
       sdo_geor.getBandDimSize(georaster) bandDimSize
  FROM georaster_table WHERE georid=21;

SPATIALDIMSIZES                BANDDIMSIZE
-------------------------      -----------
SDO_NUMBER_ARRAY(512, 512)           1
```

SDO_GEOR.getSpatialResolutions

Format

```
SDO_GEOR.getSpatialResolutions(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the spatial resolution value along each spatial dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

Each value indicates the number of units of measurement associated with the data area represented by that spatial dimension of a pixel. For example, if the spatial resolution values are (10,10) and the unit of measurement for the ground data is meters, each pixel represents an area of 10 meters by 10 meters.

The spatial resolutions may be inconsistent with the georeferencing information, especially when the GeoRaster object is not georectified. You can use the SDO_GEOR.setSpatialResolutions procedure to set the spatial resolutions to be the average resolutions for an image or the resolutions when the data was collected. In this case, georeferencing information should be used for precise measurement.

Examples

The following example returns the spatial resolution values along the column and row (X and Y) dimensions of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.getSpatialResolutions(georaster) spatialResolution
  FROM georaster_table WHERE georid=42;

SPATIALRESOLUTION
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(28.5, 28.5)
```

SDO_GEOR.getSpectralResolution

Format

```
SDO_GEOR.getSpectralResolution(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the spectral resolution of a GeoRaster object if it is a hyperspectral or multiband image.

Parameters

georaster
GeoRaster object.

Usage Notes

Taken together, the spectral unit and spectral resolution identify the wavelength interval for a band. For example, if the spectral resolution value is 2 and the spectral unit value is `MILLIMETER`, the wavelength interval for a band is 2 millimeters.

To set the spectral resolution for a GeoRaster object, use the SDO_GEOR.setSpectralResolution procedure.

Examples

The following example returns the spectral unit and spectral resolution for all spatially referenced GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, substr(sdo_geor.getSpectralUnit(georaster),1,20) spectralUnit,
               sdo_geor.getSpectralResolution(georaster) spectralResolution
  FROM georaster_table
 WHERE sdo_geor.isSpatialReferenced(georaster)='TRUE';

    GEORID SPECTRALUNIT         SPECTRALRESOLUTION
---------- -------------------- ------------------
         4 MILLIMETER                        0.075
```

SDO_GEOR.getSpectralUnit

Format

```
SDO_GEOR.getSpectralUnit(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the unit of measurement for identifying the wavelength interval for a band.

Parameters

georaster
GeoRaster object.

Usage Notes

This function can return one of the following values: `METER`, `MILLIMETER`, `MICROMETER`, `NANOMETER`.

Taken together, the spectral unit and spectral resolution identify the wavelength interval for a band. For example, if the spectral resolution value is 2 and the spectral unit value is `MILLIMETER`, the wavelength interval for a band is 2 millimeters.

To set the spectral unit for a GeoRaster object, use the SDO_GEOR.setSpectralUnit procedure.

Examples

The following example returns the spectral unit and spectral resolution for all spatially referenced GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, substr(sdo_geor.getSpectralUnit(georaster),1,20) spectralUnit,
               sdo_geor.getSpectralResolution(georaster) spectralResolution
  FROM georaster_table
 WHERE sdo_geor.isSpatialReferenced(georaster)='TRUE';

    GEORID SPECTRALUNIT         SPECTRALRESOLUTION
---------- -------------------- ------------------
         4 MILLIMETER                        0.075
```

SDO_GEOR.getSRS

Format

```
SDO_GEOR.getSRS(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_GEOR_SRS;
```

Description

Returns an object of type SDO_GEOR_SRS containing information related to the spatial referencing of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The SDO_GEOR_SRS object type is described in Section 2.3.5.

Examples

The following example returns information related to the spatial referencing of all spatially referenced GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getSRS(georaster) SRS
  FROM georaster_table
  WHERE sdo_geor.isSpatialReferenced(georaster)='TRUE';

    GEORID
----------
SRS(ISREFERENCED, ISRECTIFIED, ISORTHORECTIFIED, SRID, SPATIALRESOLUTION, SPATIA
--------------------------------------------------------------------------------
         4
SDO_GEOR_SRS('TRUE', 'TRUE', NULL, 82262, SDO_NUMBER_ARRAY(28.5, 28.5), NULL, NU
LL, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, NULL, NULL, NULL, SDO_NUMBER_ARRAY(1, 2, 1, 3,
 32631.5614, 0, -.03508772), SDO_NUMBER_ARRAY(1, 0, 0, 1, 1), SDO_NUMBER_ARRAY(1
, 2, 1, 3, -7894.7544, .035087719, 0), SDO_NUMBER_ARRAY(1, 0, 0, 1, 1))
```

SDO_GEOR.getStatistics

Format

```
SDO_GEOR.getStatistics(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns statistical data associated with a layer.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the statistics. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns statistical data described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in Appendix A. The function returns an array with the following values: `MIN`, `MAX`, `MEAN`, `MEDIAN`, `MODEVALUE`, and `STD`.

To set the statistical data associated with a layer, use the SDO_GEOR.setStatistics procedure.

Examples

The following example returns statistical data for layer 1 of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.getStatistics(georaster, 1) layer1
FROM georaster_table WHERE georid=4;

LAYER1
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0, 255, 100, 127, 95, 25)
```

SDO_GEOR.getTotalLayerNumber

Format

```
SDO_GEOR.getTotalLayerNumber(
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the total number of layers in a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For information about layers, see Section 1.5.

Examples

The following example returns the total number of layers in each GeoRaster object (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT georid, sdo_geor.getTotalLayerNumber(georaster) totalLayerNumber
  FROM georaster_table;

    GEORID TOTALLAYERNUMBER
---------- ----------------
         2                1
         4                3
```

SDO_GEOR.getULTCoordinate

Format

```
SDO_GEOR.getULTCoordinate(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_NUMBER_ARRAY ;
```

Description

Returns the cell coordinates of the upper-left corner of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns two or three numbers. If it returns two numbers, they are row and column ordinates. If it returns three numbers, they are row, column, and band ordinates.

Examples

The following example returns the row, column, and band ordinates for the upper-left corner of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT sdo_geor.getULTCoordinate(georaster) FROM georaster_table WHERE georid=23;

SDO_GEOR.GETULTCOORDINATE(GEORASTER)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(256, 0, 0)
```

SDO_GEOR.getVAT

Format

```
SDO_GEOR.getVAT(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Returns the name of the value attribute table (VAT) associated with a layer of a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the VAT. A value of 0 (zero) indicates the object layer.

Usage Notes

For more information about value attribute tables, see Section 1.2.3.

To set the name of the value attribute table to be associated with a layer of a GeoRaster object, use the SDO_GEOR.setVAT procedure.

Examples

The following example returns the value attribute tables for layers 0, 1, 2, and 3 of the GeoRaster objects (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1. The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getVAT(georaster, 0),1,20) vatTable0,
       substr(sdo_geor.getVAT(georaster, 1),1,20) vatTable1,
       substr(sdo_geor.getVAT(georaster, 2),1,20) vatTable2,
       substr(sdo_geor.getVAT(georaster, 3),1,20) vatTable3
  FROM georaster_table WHERE georid=4;

VATTABLE0            VATTABLE1            VATTABLE2            VATTABLE3
-------------------- -------------------- -------------------- ----------------
VAT0                 VAT1                 VAT2                 VAT1
```

SDO_GEOR.getVersion

Format

```
SDO_GEOR.getVersion(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the user-specified version of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The version returned is in the format *major-version.minor-version*.

To set the user-specified version of a GeoRaster object, use the SDO_GEOR.setVersion procedure.

Examples

The following example returns the user-specified version of the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1. (The output is reformatted for readability.)

```
SELECT georid, sdo_geor.getVersion(georaster) version FROM georaster_table;

    GEORID VERSION
---------- --------------------------------------------------------------------
         2  10.1
         4  9i.2
```

SDO_GEOR.hasGrayScale

Format

```
SDO_GEOR.hasGrayScale(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Checks if a layer of a GeoRaster object has grayscale information.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer to check. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns `TRUE` if the layer has grayscale information, or `FALSE` if the layer does not use grayscale representation. Section 2.3.3 describes grayscale display.

If the layer has grayscale information, you can get and set the grayscale mappings and the grayscale mapping table name. See the following: SDO_GEOR.getGrayScale and SDO_GEOR.getGrayScaleTable functions, and SDO_GEOR.setGrayScale and SDO_GEOR.setGrayScaleTable procedures.

Examples

The following example checks if layers 0 and 1 of a specified GeoRaster object (GEORASTER column) have grayscale information. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT substr(sdo_geor.hasGrayScale(georaster, 0),1,15) hasGrayScale0,
       substr(sdo_geor.hasGrayScale(georaster, 1),1,15) hasGrayScale1
  FROM georaster_table WHERE georid=4;

HASGRAYSCALE0   HASGRAYSCALE1
--------------- ---------------
TRUE            FALSE
```

SDO_GEOR.hasPseudoColor

Format

```
SDO_GEOR.hasPseudoColor(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN VARCHAR2;
```

Description

Checks if a layer of a GeoRaster object has pseudocolor information.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer to check. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns `TRUE` if the layer has pseudocolor information, or `FALSE` if the layer does not have pseudocolor information (that is, does not use pseudocolor representation). Section 2.3.2 describes colormaps and pseudocolor display.

If the layer has pseudocolor information, you can get and set the colormap and colormap table name. See the following: SDO_GEOR.getColorMap and SDO_GEOR.getColorMapTable functions, and SDO_GEOR.setColorMap and SDO_GEOR.setColorMapTable procedures.

Examples

The following example checks if layers 0 and 1 of a specified GeoRaster object (GEORASTER column) have pseudocolor information. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT substr(sdo_geor.hasPseudoColor(georaster, 0),1,15) hasPseudoColor0,
       substr(sdo_geor.hasPseudoColor(georaster, 1),1,15) hasPseudoColor1
  FROM georaster_table WHERE georid=4;

HASPSEUDOCOLOR0 HASPSEUDOCOLOR1
--------------- ---------------
FALSE           TRUE
```

SDO_GEOR.importFrom

Format

```
SDO_GEOR.importFrom(
    georaster          IN OUT SDO_GEORASTER,
    storageParam       IN VARCHAR2,
    r_sourceFormat     IN VARCHAR2,
    r_sourceType       IN VARCHAR2,
    r_sourceName       IN VARCHAR2,
    h_sourceFormat     IN VARCHAR2 DEFAULT NULL,
    h_sourceType       IN VARCHAR2 DEFAULT NULL,
    h_sourceName       IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.importFrom(
    georaster          IN OUT SDO_GEORASTER,
    storageParam       IN VARCHAR2,
    r_sourceFormat     IN VARCHAR2,
    r_sourceBLOB       IN BLOB,
    h_sourceFormat     IN VARCHAR2 DEFAULT NULL,
    h_sourceCLOB       IN CLOB DEFAULT NULL);
```

Description

Imports an image file or BLOB object into a GeoRaster object stored in the database.

Parameters

georaster
GeoRaster object to hold the result of the operation.

storageParam
String containing storage parameters. The format and usage are as explained in Section 1.4.1. Currently, the keywords supported for this operation are:

- `blocking`: `FALSE` causes the image to be stored as a single block. If the blocksize parameter is not specified, `TRUE` causes the image to be reblocked using the default reblocking parameter values: (256,256,*B*), where *B* is the total number of bands that the image contains. If the `blocksize` parameter is specified, blocking is automatically interpreted as `TRUE`.
- `blocksize`: (See the explanation in Table 1–1 in Section 1.4.1.)
- `compression`: (See the explanation in Table 1–1 in Section 1.4.1.) The default value is `NONE`, which causes the raw data to be loaded without any compression.
- `quality`: (See the explanation in Table 1–1 in Section 1.4.1.)

- `spatialExtent`: `FALSE` (the default) causes a spatial extent not to be generated; `TRUE` causes a spatial extent to be generated if the SRID is nonzero and matches the SRID of any existing spatial extent index.

r_sourceFormat
Raster source format. Must be one of the following: `TIFF`, `GIF`, `BMP`, or `PNG`. (`JPEG` is not supported for this procedure; however, you can use the client-side GeoRaster loader tool, described in Section 1.10, to import a JPEG file.)

r_sourceType
Type of source for the import operation. Must be `FILE`.

r_sourceName
Source file name (with full path specification) if `r_sourceType` is `FILE`. If you are using this procedure only to load the world file into an existing GeoRaster object, specify a null value for this parameter.

r_sourceBLOB
Raster source object of type BLOB.

h_sourceFormat
Geoheader source format. Must be `WORLDFILE`.

h_sourceType
Geoheader type of source for the import operation. Must be `FILE`.

h_sourceName
Geoheader source file name (with full path specification) if `h_sourceType` is `FILE`., and optionally an SRID value. To specify the SRID value, add it after the file name, separated by a comma. Example: `'/mypath/mydir/worldfile.tfw,82934'` (UNIX or Linux) or `'C:\mypath\mydir\worldfile.tfw,82934'` (Windows)

h_sourceCLOB
Geoheader source as an object of type CLOB.

Usage Notes

For information about using this procedure or the GeoRaster loader tool to load GeoRaster data, see Section 3.4.

Before you load GeoRaster data, you may need to increase the Java pool size. See Section 3.3 for more information.

Specify values for the parameters with names that start with *r_* and *h_* only if the raster image and the geoheader are in separate files or objects.

This procedure can load an ESRI world file from a file or from a CLOB object.

This procedure does not support JPEG as a source file format. You can use the client-side GeoRaster loader tool, described in Section 1.10, to import a JPEG file.

This procedure does not support raster data that has a cell depth value of `2BIT` or source multiband raster data with BIL and BSQ interleaving types.

The imported GeoRaster object has the BIP interleaving type.

Before you call this procedure, you must have read permission on the files to be imported or the directory that contains the files. The following example (run as user `SYSTEM`) grants read permission on a file to user `HERMAN`:

```
call dbms_java.grant_permission('HERMAN','SYS:java.io.FilePermission',
```

```
  'sdo/demos/georaster/data/img1.tif', 'read' );
```

Examples

The following example initializes an empty GeoRaster object into which an external image in TIFF format is to be imported, and then imports the image.

```
DECLARE
   geor SDO_GEORASTER;
BEGIN
-- Initialize an empty GeoRaster object into which the external image
-- is to be imported.
INSERT INTO georaster_table
   values( 1, 'TIFF', sdo_geor.init('rdt_1') );

-- Import the TIFF image.
SELECT georaster INTO geor FROM georaster_table
   WHERE georid = 1 FOR UPDATE;
sdo_geor.importFrom(geor, NULL, 'TIFF', 'file',
   'sdo/demos/georaster/data/img1.tif');
UPDATE georaster_table SET georaster = geor WHERE georid = 1;
COMMIT;
END;/
```

The following example imports images from a BLOB and an ESRI world file from a CLOB.

```
CREATE TABLE blob_table (blob_col BLOB, blobid NUMBER unique, clob_col CLOB);
INSERT INTO blob_table VALUES (empty_blob(), 1, null);
INSERT INTO blob_table VALUES (empty_blob(), 2, empty_clob());
COMMIT;

DECLARE
  geor1 SDO_GEORASTER;
  lobd1 BLOB;
  lobd2 CLOB;
  fileName VARCHAR2(1024);
  file BFILE;
  wfile BFILE;
  wfname VARCHAR2(1024);
  amt INTEGER;
  amt1 INTEGER;

BEGIN
-- Import BLOB into georaster object.
-- First, if appropriate, load an existing image file into a BLOB object.
EXECUTE IMMEDIATE 'CREATE DIRECTORY  blob_test_one AS ''/xyz''';
fileName := '/parrot.tif';
file := BFILENAME('BLOB_TEST_ONE', fileName);
wfname := '/parrot.tfw';
wfile := BFILENAME('BLOB_TEST_ONE', wfname);
SELECT clob_col into lobd2 from blob_table WHERE blobid = 2 for update;
SELECT blob_col into lobd1 from blob_table WHERE blobid = 2 for update;
dbms_lob.fileopen(file, dbms_lob.file_readonly);
dbms_lob.fileopen(wfile, dbms_lob.file_readonly);
amt1 := dbms_lob.getLength(wfile);
dbms_lob.loadfromfile(lobd1, file, amt);
dbms_lob.loadfromfile(lobd2, wfile, amt1);
COMMIT;
dbms_lob.fileclose(file);
dbms_lob.fileclose(wfile);
```

```
-- Then, import this BLOB into a georaster object.
SELECT georaster INTO geor1 from georaster_table WHERE georid = 14 for update;
sdo_geor.importFrom(geor1,'', 'TIFF', lobd1, 'WORLDFILE', lobd2);
sdo_geor.setModelSRID(geor1, 82394);
UPDATE georaster_table SET georaster = geor1 WHERE georid = 14;
COMMIT;
END;
/
```

SDO_GEOR.init

Format

```
SDO_GEOR.init(
   rasterDataTable  IN VARCHAR2 DEFAULT NULL,
   rasterID         IN NUMBER DEFAULT NULL
   ) RETURN SDO_GEORASTER;
```

Description

Initializes an empty GeoRaster object, which will be registered by GeoRaster in the xxx_SDO_GEOR_SYSDATA views (described in Section 2.4).

Parameters

rasterDataTable
Name of the object table of type SDO_RASTER that stores the cell data blocks. If you do not specify this parameter, GeoRaster generates a unique table name to be used for the raster data table. If you specify this parameter and the table already exists but is not an object table of type SDO_RASTER, an exception is raised.

rasterID
Number that uniquely identifies the blocks of this GeoRaster object in its raster data table. If you do not specify this parameter, a unique sequence number is generated for the ID.

Usage Notes

This function returns an empty SDO_GEORASTER object with its `rasterDataTable` and `rasterID` attributes set. All other attributes of the SDO_GEORASTER object are null.

This function does not require that the specified raster data table exist. However, the table must exist before any data can be inserted into it, and you must create the table.

If a table has multiple GeoRaster object columns, and if for each column you plan to call the SDO_GEOR.init or SDO_GEOR.createBlank function with identical parameter values that contain a null `rasterDataTable` or `rasterID` parameter value, do not try to use the SDO_GEOR.init or SDO_GEOR.createBlank function on all such columns with a single INSERT or UPDATE statement. For example, assuming a table named LSAT_TABLE containing the columns (`georid NUMBER, type VARCHAR2(32), image_date VARCHAR2(32), image_15m SDO_GEORASTER, image_30m SDO_GEORASTER, image_60m SDO_GEORASTER`), do *not* use a statement like the following:

```
INSERT INTO lsat_table VALUES(1, 'L1G', '2004-02-25',
  sdo_geor.init('RDT_1'), sdo_geor.init('RDT_1'),
  sdo_geor.init('RDT_1'));
```

Instead, in cases such as this, do either of the following:

- Always specify a rasterID parameter value when calling the function. The following example specifies raster ID values of 1, 2, and 3 for the GeoRaster objects being inserted into the last three columns:

```
INSERT INTO lsat_table VALUES(1, 'L1G', '2004-02-25',
  sdo_geor.init('RDT_1', 1), sdo_geor.init('RDT_1', 2),
  sdo_geor.init('RDT_1', 3));
```

- Use the function with only one GeoRaster object with each INSERT or UPDATE statement. The following example inserts a row initializing one GeoRaster object column and specifying the other two as null, and then updates the row twice to initialize the second and third GeoRaster object columns:

```
INSERT INTO lsat_table VALUES(1, 'L1G', '2004-02-25',
  sdo_geor.init('RDT_1'), null, null);
UPDATE lsat_table SET image_30m = sdo_geor.init('RDT_1')
  WHERE georid = 1;
UPDATE lsat_table SET image_60m = sdo_geor.init('RDT_1')
  WHERE georid = 1;
```

Examples

The following example inserts an initialized GeoRaster object into the GEORASTER_TABLE table. The raster data table associated with the GeoRaster object is RDT_1. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
INSERT INTO georaster_table (georid, georaster)
  VALUES (1, sdo_geor.init('RDT_1'));
```

SDO_GEOR.isBlank

Format

```
SDO_GEOR.isBlank(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the string `TRUE` if the GeoRaster object is a blank GeoRaster object, or `FALSE` if the GeoRaster object is not a blank GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

In a blank GeoRaster object, all cells have the same cell value.

To change the cell value of an existing blank GeoRaster object, use the SDO_GEOR.setBlankCellValue procedure. To return the cell value of a specified GeoRaster object, use the SDO_GEOR.getBlankCellValue function.

Examples

The following example determines whether or not each GeoRaster object in the GEORASTER column of the GEORASTER_TABLE table is a blank GeoRaster object. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT georid, substr(sdo_geor.isBlank(georaster),1,7) isBlank
  FROM georaster_table;

    GEORID ISBLANK
---------- -------
         2 FALSE
         4 FALSE
```

SDO_GEOR.isOrthoRectified

Format

```
SDO_GEOR.isOrthoRectified(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the string `TRUE` if the GeoRaster object is identified as orthorectified, or `FALSE` if the GeoRaster object is not identified as orthorectified.

Parameters

georaster
GeoRaster object.

Usage Notes

This function checks the GeoRaster metadata for the object to see if it is specified as orthorectified. It does not check if the object is actually orthorectified. Users are responsible for validating the GeoRaster object and ensuring that orthorectification is performed.

To specify that a GeoRaster object is orthorectified, use the SDO_GEOR.setOrthoRectified procedure.

Examples

The following example checks if the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table are specified as spatially referenced, rectified, and orthorectified. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT georid, substr(sdo_geor.isSpatialReferenced(georaster),1,20)
             isSpatialReferenced,
           substr(sdo_geor.isRectified(georaster),1,20) isRectified,
           substr(sdo_geor.isOrthoRectified(georaster),1,20) isOrthoRectified
  FROM georaster_table;

    GEORID ISSPATIALREFERENCED  ISRECTIFIED          ISORTHORECTIFIED
---------- -------------------- -------------------- --------------------
         2 TRUE                 TRUE                 TRUE
         4 TRUE                 TRUE                 FALSE
```

SDO_GEOR.isRectified

Format

```
SDO_GEOR.isRectified(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the string `TRUE` if the GeoRaster object is identified as rectified, or `FALSE` if the GeoRaster object is not identified as rectified.

Parameters

georaster
GeoRaster object.

Usage Notes

This function checks the GeoRaster metadata for the object to see if it is specified as rectified. Users are responsible for validating the GeoRaster object and ensuring that rectification is performed.

To specify that a GeoRaster object is rectified, use the SDO_GEOR.setRectified procedure.

Examples

The following example checks if the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table are specified as spatially referenced, rectified, and orthorectified. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT georid, substr(sdo_geor.isSpatialReferenced(georaster),1,20)
             isSpatialReferenced,
           substr(sdo_geor.isRectified(georaster),1,20) isRectified,
           substr(sdo_geor.isOrthoRectified(georaster),1,20) isOrthoRectified
  FROM georaster_table;

    GEORID ISSPATIALREFERENCED  ISRECTIFIED          ISORTHORECTIFIED
---------- -------------------- -------------------- --------------------
         2 TRUE                 TRUE                 TRUE
         4 TRUE                 TRUE                 FALSE
```

SDO_GEOR.isSpatialReferenced

Format

```
SDO_GEOR.isSpatialReferenced(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the string `TRUE` if the GeoRaster object is spatially referenced, or `FALSE` if the GeoRaster object is not spatially referenced.

Parameters

georaster
GeoRaster object.

Usage Notes

The GeoRaster object must have been validated.

Examples

The following example checks if the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table are specified as spatially referenced, rectified, and orthorectified. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT georid, substr(sdo_geor.isSpatialReferenced(georaster),1,20)
             isSpatialReferenced,
           substr(sdo_geor.isRectified(georaster),1,20) isRectified,
           substr(sdo_geor.isOrthoRectified(georaster),1,20) isOrthoRectified
  FROM georaster_table;

    GEORID ISSPATIALREFERENCED  ISRECTIFIED          ISORTHORECTIFIED
---------- -------------------- -------------------- --------------------
         2 TRUE                 TRUE                 TRUE
         4 TRUE                 TRUE                 FALSE
```

SDO_GEOR.mosaic

Format

```
SDO_GEOR.mosaic(
   georasterTableName     IN VARCHAR2,
   georasterColumnName  IN VARCHAR2,
   georaster                  IN OUT SDO_GEORASTER,
   storageParam             IN VARCHAR2);
```

Description

Mosaics GeoRaster objects into one GeoRaster object.

Parameters

georasterTableName
Name of the table containing all source GeoRaster objects.

georasterColumnName
Column of type SDO_GEORASTER in `georasterTableName`.

georaster
GeoRaster object to hold the result of the mosaic operation.

storageParam
A string specifying storage parameters, as explained in Section 1.4.1. If this parameter is null, the resulting GeoRaster object has the same storage parameters (`blockSize`, `cellDepth`, `interleaving`, and `compression`) as the upper-left corner source GeoRaster object in the model space (if applicable) or cell space.

Usage Notes

The source GeoRaster objects must be prepared tiles or blocks so that they can be mosaicked directly and seamlessly. The GeoRaster objects to be mosaicked must:

- Be rectified, and have the same SRID value and spatial resolutions
- Completely cover the mosaic area, and not overlap or have any gaps
- Have the same number of layers or bands, and be spatially aligned along row and column dimensions
- Have the same mapping between band number and layers

If applicable, the resulting GeoRaster object takes the spatial reference metadata information from the upper-left corner source GeoRaster object in the model space.

If all source GeoRaster objects are blank and have the same `blankCellValue` value, the resulting GeoRaster object is blank and has that `blankCellValue` value; otherwise, the resulting GeoRaster object is not blank.

Any pyramid data for the source GeoRaster objects is not considered, and the `pyramid` parameter is ignored if it is specified in the `storageParam` string.

The mosaic operation performs internal commit operations at regular intervals, and thus it cannot be rolled back. If the operation is interrupted, dangling raster blocks

may exist in the raster data table. You can handle dangling raster blocks, as explained in Section 3.20.

Examples

The following example inserts an initialized GeoRaster object into the GEORASTER_TABLE table, returns the GeoRaster object into a variable named `gr`, mosaics all the GeoRaster objects in the GROBJ column of a table named GRTAB, and stores the resulting mosaicked GeoRaster object in the same variable. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1. The GRTAB table definition is not important to the example and is not presented here.)

```
DECLARE
  gr sdo_georaster;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
      VALUES (12, sdo_geor.init('rdt_1'))
      RETURNING georaster INTO gr;
  sdo_geor.mosaic('grtab', 'grobj', gr, 'blocksize=(512,512,1)');
  UPDATE georaster_table SET georaster=gr WHERE id=12;
END;
/
```

SDO_GEOR.scale

Format

```
SDO_GEOR.scale(
   inGeoraster       IN OUT SDO_GEORASTER,
   scaleParam        IN VARCHAR2,
   resampleParam  IN VARCHAR2,
   storageParam      IN VARCHAR2);
```

Description

Scales a GeoRaster object by enlarging or reducing the image along row and column dimensions.

> **Note:** This procedure is deprecated. Instead, use the SDO_GEOR.scaleCopy procedure, which makes a copy of an existing GeoRaster object using the specified scaling information.

Parameters

inGeoraster
The SDO_GEORASTER object on which the scaling operation is to be performed.

scaleParam
A string specifying a scaling parameter keyword and its associated value. The keyword must be one of the following:

- `scaleFactor`, to reduce or enlarge as a multiple of the original size. This keyword must have a numeric value greater than 0 (zero) (for example, `'scaleFactor=0.75'`). A value of 1.0 will not change the current size; a value less than 1 will reduce the image; a value greater than 1 will enlarge the image. The number of cells along each dimension is the original number multiplied by `scaleFactor`. For example, if the `scaleFactor` value is 2 and the GeoRaster object has X and Y dimensions, the number of cells along each dimension is doubled.
- `maxDimSize`, to specify a size in terms of the maximum number of cells for each dimension. This keyword must have a numeric value for each dimension (for example, `'maxDimSize=(512,512)'`). The aspect ratio is not changed.

resampleParam
A string specifying a resampling parameter (for example, `'resampling=NN'`). For the current release, the only supported keyword is `resampling`, and its value must be one of the following:

- `NN`: value of the nearest neighbor cell in the original GeoRaster object
- `BILINEAR`: distance-weighted average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE4`: simple average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE16`: simple average of the 16 nearest cells in the original GeoRaster object

- CUBIC: cubic convolution of the 16 nearest cells in the original GeoRaster object

storageParam
A string specifying storage parameters, as explained in Section 1.4.1. However, the compression keyword is not supported in the storageParam parameter for this procedure, while it is supported for the SDO_GEOR.scaleCopy procedure.

Usage Notes

This procedure, which changes the input GeoRaster object to reflect the specified scaling, is deprecated and will not be supported in a future release of Spatial. Instead, use the SDO_GEOR.scaleCopy procedure. After you use the SDO_GEOR.scaleCopy procedure, you can check to ensure that the desired changes were made in the copy of the original GeoRaster object, and then discard the original GeoRaster object if you wish.

This procedure does not scale along the band dimension.

If you need to get the scaled cell values, use the procedure described in the Usage Notes for the SDO_GEOR.getCellValue function.

Any pyramid data in the GeoRaster object is deleted as a result of the scaling operation.

After the operation, the row and column ULT coordinates are always set to 0 (zero), even if no scaling is performed (that is, even if scaleFactor=1).

If the metadata contains spatial reference information and if the GeoRaster object is georeferenced with a valid affine transformation, the spatial reference information is updated accordingly; otherwise, the spatial reference information is removed.

Examples

The following example reduces an image to three-fourths (0.75) size, specifies bilinear resampling, and specifies a block size of 64 for each dimension in the storage parameters. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr1 sdo_georaster;
BEGIN
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=21 FOR UPDATE;
  sdo_geor.scale(gr1, 'scaleFactor=0.75', 'resampling=BILINEAR',
                    'blocksize=(64,64)');
  UPDATE georaster_table SET georaster=gr1 WHERE georid=21;
  COMMIT;
END;
/
```

SDO_GEOR.scaleCopy

Format

```
SDO_GEOR.scaleCopy(
   inGeoraster       IN SDO_GEORASTER,
   scaleParam        IN VARCHAR2,
   resampleParam  IN VARCHAR2,
   storageParam     IN VARCHAR2,
   outGeoraster      IN OUT SDO_GEORASTER);
```

Description

Scales a GeoRaster object by enlarging or reducing the image along row and column dimensions, and puts the result into a new object that reflects the scaling.

Parameters

inGeoraster
The SDO_GEORASTER object on which the scaling operation is to be performed to create the new object (`outGeoraster`).

scaleParam
A string specifying a scaling parameter keyword and its associated value. The keyword must be one of the following:

- `scaleFactor`, to reduce or enlarge as a multiple of the original size. This keyword must have a numeric value greater than 0 (zero) (for example, `'scaleFactor=0.75'`). A value of 1.0 will not change the current size; a value less than 1 will reduce the image; a value greater than 1 will enlarge the image. The number of cells along each dimension is the original number multiplied by `scaleFactor`. For example, if the `scaleFactor` value is 2 and the GeoRaster object has X and Y dimensions, the number of cells along each dimension is doubled.
- `maxDimSize`, to specify a size in terms of the maximum number of cells for each dimension. This keyword must have a numeric value for each dimension (for example, `'maxDimSize=(512,512)'`). The aspect ratio is not changed.

resampleParam
A string specifying a resampling parameter (for example, 'resampling=NN'). For the current release, the only supported keyword is `resampling`, and its value must be one of the following:

- `NN`: value of the nearest neighbor cell in the original GeoRaster object
- `BILINEAR`: distance-weighted average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE4`: simple average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE16`: simple average of the 16 nearest cells in the original GeoRaster object
- `CUBIC`: cubic convolution of the 16 nearest cells in the original GeoRaster object

storageParam
A string specifying storage parameters, as explained in Section 1.4.1.

outGeoraster
The new SDO_GEORASTER object that reflects the results of the scaling operation.

Usage Notes

Use this procedure to create a new GeoRaster object reflecting the specified scaling, without changing the original object. After you use this procedure, you can check to ensure that the desired changes were made in the copy of the original GeoRaster object, and then discard the original GeoRaster object if you wish.

This procedure does not scale along the band dimension.

If you need to get the scaled cell values, use the procedure described in the Usage Notes for the SDO_GEOR.getCellValue function.

Any pyramid data in the input GeoRaster object is not included in the output GeoRaster object.

After the operation, the row and column ULT coordinates are always set to 0 (zero), even if no scaling is performed (that is, even if `scaleFactor=1`).

If the metadata contains spatial reference information and if the GeoRaster object is georeferenced with a valid affine transformation, the spatial reference and georeferencing information is updated accordingly; otherwise, the spatial reference information is removed.

An exception is raised if one or more of the following are true:

- `inGeoraster` is invalid.
- `outGeoraster` has not been initialized.
- A raster data table for `outGeoraster` does not exist and `outGeoraster` is not a blank GeoRaster object.

Examples

The following example reduces an image to three-fourths (0.75) size, specifies `AVERAGE4` resampling, and specifies a block size of 32 for each dimension in the storage parameters. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
    VALUES (21, sdo_geor.init('RDT_1'))
    RETURNING georaster INTO gr2;

  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=2;

  sdo_geor.scaleCopy(gr1, 'scaleFactor=0.75', 'resampling=AVERAGE4',
                     'blocksize=(32,32)', gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=21;
  COMMIT;
END;
/
```

SDO_GEOR.schemaValidate

Format

```
SDO_GEOR.schemaValidate(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Validates a GeoRaster object's metadata against the GeoRaster XML schema.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns `TRUE` if the metadata is valid, `NULL` if the GeoRaster object or its metadata is null, or one or more Oracle error codes indicating why the metadata is not valid and the exact location of the errors.

Use this function with the SDO_GEOR.validateGeoraster function. If the SDO_GEOR.validateGeoraster function identifies a GeoRaster object as invalid with an error code of 13454, the object's metadata is not valid according to the GeoRaster XML schema. If this happens, call the SDO_GEOR.schemaValidate function to get specific information, including the location in the metadata, about the errors.

Examples

The following example validates a GeoRaster object's metadata.

```
SELECT t.georid,
       sdo_geor.schemavalidate(t.georaster)
  FROM georaster_table t
  WHERE t.georid = 1;
```

SDO_GEOR.setBeginDateTime

Format

```
SDO_GEOR.setBeginDateTime(
   georaster   IN OUT SDO_GEORASTER,
   beginTime  TIMESTAMP WITH TIME ZONE);
```

Description

Sets the beginning date and time for raster data collection in the metadata for a GeoRaster object, or deletes the existing value if you specify a null `beginTime` parameter.

Parameters

georaster
GeoRaster object.

beginTime
Time specification.

Usage Notes

To see the current beginning date and time (if any) in the metadata for the GeoRaster object, use the SDO_GEOR.getBeginDateTime function.

An exception is raised if `beginTime` is later than the ending date and time specified in the metadata for the GeoRaster object (see the SDO_GEOR.setEndDateTime procedure).

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the beginning and ending dates and times for raster data collection in the metadata for a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setBeginDateTime(grobj, timestamp '2002-11-15 15:00:00');
  sdo_geor.setEndDateTime(grobj, timestamp '2002-11-15 15:00:10');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setBinTable

Format

```
SDO_GEOR.setBinTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   tableName      IN VARCHAR2);
```

Description

Sets the name of the bin table associated with a layer, or deletes the existing value if you specify a null `tableName` parameter.

> **Note:** GeoRaster does not perform operations using the BIN function or the bin table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the bin table name. A value of 0 (zero) indicates the object layer.

tableName
Name of the bin table associated with a layer.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

This procedure is relevant only if the bin type is `EXPLICIT`. To retrieve the bin type, use the SDO_GEOR.getBinType function.

To return the bin table for a layer, use the SDO_GEOR.getBinTable function.

See also the information in the Usage Notes for the SDO_GEOR.getBinType function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if `tableName` is an empty string (`''`).

Examples

The following example sets `BINT1` as the name of the bin table for layer number 3 of a specified GeoRaster object in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setBinTable(grobj, 3, 'BINT1');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
```

```
END;
/
```

SDO_GEOR.setBlankCellValue

Format

```
SDO_GEOR.setBlankCellValue(
   georaster  IN OUT SDO_GEORASTER,
   value      IN NUMBER);
```

Description

Sets (modifies) the cell value to be used for all cells if a specified GeoRaster object is a blank GeoRaster object.

Parameters

georaster
GeoRaster object.

value
Cell value to be used for the blank GeoRaster object. Cannot be a null value.

Usage Notes

In a blank GeoRaster object, all cells have the same cell value.

The GeoRaster object is automatically validated after the operation completes.

To return the blank cell value of a blank GeoRaster object, use the SDO_GEOR.getBlankCellValue function. To determine if a specified GeoRaster object is a blank GeoRaster object, use the SDO_GEOR.isBlank function.

An exception is raised if `value` is null or inconsistent with the `cellDepth` specification, or if the GeoRaster object is not blank.

Examples

The following example specifies a value of 255 to be used for all cells in the GeoRaster object column (GEORASTER) in the GEORASTER_TABLE table for the row with an GEORID column value of 1. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=1 FOR UPDATE;
  sdo_geor.setBlankCellValue(grobj, 255);
  UPDATE georaster_table SET georaster = grobj WHERE georid=1;
  COMMIT;
END;
/
```

SDO_GEOR.setColorMap

Format

```
SDO_GEOR.setColorMap(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   colorMap       IN SDO_GEOR_COLORMAP);
```

Description

Sets the colormap for a layer in a GeoRaster object, or deletes the existing value if you specify a null `colorMap` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

colorMap
Colormap object of type SDO_GEOR_COLORMAP, which is described in Section 2.3.2.

Usage Notes

The following must be true of the specified colormap object:

- The `cellValue` values are consistent with and in the value range for the `cellDepth` value of the GeoRaster object.
- The red, green, blue, and alpha values are integers from 0 to 255.
- The `cellValue` array contains no duplicate entries.
- The entries in the `cellValue` array are in ascending order.

The GeoRaster object is automatically validated after the operation completes.

You can create a colormap or retrieve a colormap from an existing GeoRaster object for use. To return the colormap for a layer in a GeoRaster object, use the SDO_GEOR.getColorMap function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if any of the following exist in `colorMap`: the red, green, blue, or alpha value is null or out of scope; duplicate values exist in the `cellValue` array, or any `cellValue` values are null, out of scope, or out of order.

Examples

The following example sets the colormap for layer 2 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. It assumes that the GeoRaster object is a bitmap. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
```

```
  cmobj sdo_geor_colormap;
BEGIN
  cmobj := sdo_geor_colormap(sdo_number_array(0, 1),
                             sdo_number_array(0, 255),
                             sdo_number_array(0, 0),
                             sdo_number_array(0, 0),
                             sdo_number_array(255, 255));

  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setColorMap(grobj, 2, cmobj);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setColorMapTable

Format

```
SDO_GEOR.setColorMapTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   tableName      IN VARCHAR2);
```

Description

Sets the colormap table for a layer in a GeoRaster object, or deletes the existing value if you specify a null `tableName` parameter.

> **Note:** This procedure registers the colormap table name with GeoRaster; however, GeoRaster does not perform operations using the colormap table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

tableName
Name of the user-defined colormap table. Section 2.3.2 describes colormaps.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To return the colormap table for a layer in a GeoRaster object, use the SDO_GEOR.getColorMapTable function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if `tableName` is an empty string (`''`).

Examples

The following example sets the colormap table to be null for layer 2 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setColorMapTable(grobj, 2, null);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setDefaultBlue

Format

```
SDO_GEOR.setDefaultBlue(
   georaster    IN OUT SDO_GEORASTER,
   defaultBlue  NUMBER);
```

Description

Sets the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultBlue` parameter.

Parameters

georaster
GeoRaster object.

defaultBlue
Number of the layer to be used for the blue color component (in the RGB color space) for displaying the specified GeoRaster object. Must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

The default red, green, and blue values are used for true-color displays, not for pseudocolor or grayscale displays. These values are optional, and they are intended for use only when visualizing multilayer or hyperspectral GeoRaster objects.

The GeoRaster object is automatically validated after the operation completes.

An exception is raised if you are trying to set or remove the number of the layer to be used for the blue color component only, or if `defaultBlue` is not a valid layer number for the GeoRaster object.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 5);
  sdo_geor.setDefaultGreen(grobj, 4);
  sdo_geor.setDefaultBlue(grobj, 3);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(georaster) FROM georaster_table
```

```
  WHERE georid=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(GEORASTER)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(5, 4, 3)

1 row selected.
```

SDO_GEOR.setDefaultColorLayer

Format

```
SDO_GEOR.setDefaultColorLayer(
   georaster     IN OUT SDO_GEORASTER,
   defaultRGB  SDO_NUMBER_ARRAY);
```

Description

Sets the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object, or deletes the existing values if you specify a null `defaultRGB` parameter.

Parameters

georaster
GeoRaster object.

defaultRGB
Array of three numbers identifying the red, green, and blue color components, respectively, for displaying the specified GeoRaster object. Each number must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

The RGB layer numbers specified are used for true-color displays, not for pseudocolor or grayscale displays.

The GeoRaster object is automatically validated after the operation completes.

You can set the layer number for each color component (RGB) by using the SDO_GEOR.setDefaultRed, SDO_GEOR.setDefaultGreen, and SDO_GEOR.setDefaultBlue procedures.

An exception is raised if `defaultRGB` is of the wrong size or if any elements in it are null or are invalid layer numbers for the GeoRaster object. All elements in the `defaultRGB` array must be either null or not null; you cannot mix null and non-null array elements, because the three layer numbers must be set or removed at the same time.

Examples

The following example specifies that layer number 1 is to be used for the red, green, and blue color components for displaying the GeoRaster object (GEORASTER column) in the row with an GEORID column value of 2 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=2 FOR UPDATE;
  sdo_geor.setDefaultColorLayer(grobj, sdo_number_array(1,1,1));
  UPDATE georaster_table SET georaster = grobj WHERE georid=2;
  COMMIT;
END;
```

/

SDO_GEOR.setDefaultGreen

Format

```
SDO_GEOR.setDefaultGreen(
   georaster      IN OUT SDO_GEORASTER,
   defaultGreen  NUMBER);
```

Description

Sets the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultGreen` parameter.

Parameters

georaster
GeoRaster object.

defaultGreen
Number of the layer to be used for the green color component (in the RGB color space) for displaying the specified GeoRaster object. Must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

The default red, green, and blue values are used for true-color displays, not for pseudocolor or grayscale displays. These values are optional, and they are intended for use only when visualizing multilayer or hyperspectral GeoRaster objects.

The GeoRaster object is automatically validated after the operation completes.

An exception is raised if you are trying to set or remove the number of the layer to be used for the green color component only, or if `defaultGreen` is not a valid layer number for the GeoRaster object.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 5);
  sdo_geor.setDefaultGreen(grobj, 4);
  sdo_geor.setDefaultBlue(grobj, 3);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(georaster) FROM georaster_table
```

```
  WHERE georid=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(GEORASTER)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(5, 4, 3)

1 row selected.
```

SDO_GEOR.setDefaultRed

Format

```
SDO_GEOR.setDefaultRed(
   georaster    IN OUT SDO_GEORASTER,
   defaultRed  IN NUMBER);
```

Description

Sets the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultRed` parameter.

Parameters

georaster
GeoRaster object.

defaultRed
Number of the layer to be used for the red color component (in the RGB color space) for displaying the specified GeoRaster object. Must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

The default red, green, and blue values are used for true-color displays, not for pseudocolor or grayscale displays. These values are optional, and they are intended for use only when visualizing multilayer or hyperspectral GeoRaster objects.

The GeoRaster object is automatically validated after the operation completes.

An exception is raised if you are trying to set or remove the number of the layer to be used for the red color component only, or if `defaultRed` is not a valid layer number for the GeoRaster object.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GEORASTER column) in the GEORASTER_TABLE table, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 5);
  sdo_geor.setDefaultGreen(grobj, 4);
  sdo_geor.setDefaultBlue(grobj, 3);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(georaster) FROM georaster_table
```

```
  WHERE georid=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(GEORASTER)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(5, 4, 3)

1 row selected.
```

SDO_GEOR.setEndDateTime

Format

```
SDO_GEOR.setEndDateTime(
   georaster  IN OUT SDO_GEORASTER,
   endTime    TIMESTAMP WITH TIME ZONE);
```

Description

Sets the ending date and time for raster data collection in the metadata for a GeoRaster object, or deletes the existing value if you specify a null `endTime` parameter.

Parameters

georaster
GeoRaster object.

endTime
Time specification.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To see the current ending date and time (if any) in the metadata for the GeoRaster object, use the SDO_GEOR.getEndDateTime function.

An exception is raised if `endTime` is earlier than the beginning date and time specified in the metadata for the GeoRaster object (see the SDO_GEOR.setBeginDateTime procedure).

Examples

The following example sets the beginning and ending dates and times for raster data collection in the metadata for a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setBeginDateTime(grobj, timestamp '2002-11-15 15:00:00');
  sdo_geor.setEndDateTime(grobj, timestamp '2002-11-15 15:00:10');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setGrayScale

Format

```
SDO_GEOR.setGrayScale(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   grayScale      IN SDO_GEOR_GRAYSCALE);
```

Description

Sets the grayscale mappings for a layer in a GeoRaster object, or deletes the existing values if you specify a null `grayScale` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the grayscale mappings. A value of 0 (zero) indicates the object layer.

grayScale
An object of type SDO_GEOR_GRAYSCALE, which is described in Section 2.3.3.

Usage Notes

The following must be true of the specified SDO_GEOR_GRAYSCALE object:

- The `cellValue` values are consistent with and in the value range for the `cellDepth` value of the GeoRaster object.
- The `gray` value is an integer from 0 to 255.
- The `cellValue` array contains no duplicate entries.
- The entries in the `cellValue` array are in ascending order.

The GeoRaster object is automatically validated after the operation completes.

To return the grayscale mappings for a layer in a GeoRaster object, use the SDO_GEOR.getGrayScale function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, any gray values are null or out of scope, the `cellValue` array contains any duplicate values, or any `cellValue` values are null, out of scope, or out of order.

Examples

The following example sets the grayscale mappings for layer 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
  gsobj sdo_geor_grayscale;
```

```
BEGIN
  gsobj := sdo_geor_grayscale(sdo_number_array(1, 10, 20, 30, 255),
                        sdo_number_array(0, 180, 210, 230, 250));

  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setGrayScale(grobj, 3, gsobj);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setGrayScaleTable

Format

```
SDO_GEOR.setGrayScaleTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   tableName      IN VARCHAR2);
```

Description

Sets the grayscale mapping table for a layer in a GeoRaster object, or deletes the existing value if you specify a null `tableName` parameter.

> **Note:** This procedure registers the grayscale mapping table name with GeoRaster; however, GeoRaster does not perform operations using the grayscale mapping table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the grayscale mapping table. A value of 0 (zero) indicates the object layer.

tableName
Name of the grayscale mapping table for a layer in the specified GeoRaster object.

Usage Notes

Section 2.3.3 describes grayscale display.

The GeoRaster object is automatically validated after the operation completes.

To return the grayscale mapping table for a layer in a GeoRaster object, use the SDO_GEOR.getGrayScaleTable function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if `tableName` is an empty string (`''`).

Examples

The following example sets `GST1` as the grayscale mapping table for layer 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setGrayScaleTable(grobj, 3, 'GST1');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
```

```
  COMMIT;
END;
/
```

SDO_GEOR.setHistogramTable

Format

```
SDO_GEOR.setHistogramTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER
   tableName      IN VARCHAR2);
```

Description

Sets the histogram table for a layer in a GeoRaster object.

> **Note:** This procedure registers the histogram table name with GeoRaster; however, GeoRaster does not perform operations using the histogram table in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the name of the histogram table. A value of 0 (zero) indicates the object layer.

tableName
Name of the histogram table. If this parameter is null, the metadata information for any existing histogram table (but not the actual table) is deleted. If there is no statistics information for the layer, this parameter must be null. The parameter value cannot be an empty string (that is, it cannot be `''`).

Usage Notes

This procedure specifies a user-defined histogram table. Section 2.3.1 briefly discusses histograms.

To return the name of the histogram table for a layer, use the SDO_GEOR.getHistogramTable function.

An exception is raised if one or more of the following are true:

- `layerNumber` is null or invalid for the GeoRaster object,.
- `tableName` is an empty string (`''`).
- The statistical data associated with the specified layer is not set.

  To set the statistical data for a layer, call the SDO_GEOR.setStatistics procedure.

Examples

The following example sets `HIST1` as the histogram table for layer 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setHistogramTable(grobj, 3, 'HIST1');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setID

Format

```
SDO_GEOR.setID(
   georaster  IN OUT SDO_GEORASTER,
   id         IN VARCHAR2);
```

Description

Sets a user-defined identifier to be associated with a GeoRaster object, or deletes the existing value if you specify a null `id` parameter.

Parameters

georaster
GeoRaster object.

id
ID value to be associated with the GeoRaster object.

Usage Notes

This procedure is useful for assigning unique meaningful alphanumeric identifiers to GeoRaster objects, so that users and applications can easily identify the objects.

The GeoRaster object is automatically validated after the operation completes.

To return the user-defined identifier value for a GeoRaster object, use the SDO_GEOR.getID function.

Examples

The following example sets `newid` as the user-defined identifier value of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 2 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=2 FOR UPDATE;
  sdo_geor.setID(grobj, 'newid');
  UPDATE georaster_table SET georaster = grobj WHERE georid=2;
  COMMIT;
END;
/
```

SDO_GEOR.setLayerID

Format

```
SDO_GEOR.setLayerID(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   id                IN VARCHAR2);
```

Description

Sets a user-defined identifier to be associated with a layer in a GeoRaster object, or deletes the existing value if you specify a null `id` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

id
ID value to be associated with the specified layer in the GeoRaster object.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To return the user-defined identifier value for a layer in a GeoRaster object, use the SDO_GEOR.getLayerID function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if `id` is null yet the corresponding layer information does exist.

Examples

The following example sets `TM_Band_2` as the user-defined identifier value of layer 2 in the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setLayerID(grobj, 2, 'TM_Band_2');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setLayerOrdinate

Format

```
SDO_GEOR.setLayerOrdinate(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   ordinate        IN NUMBER);
```

Description

Sets the band ordinate value for a specified layer in a GeoRaster object, or deletes the existing value if you specify a null `ordinate` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

ordinate
Band ordinate value of the layer along the band dimension.

Usage Notes

The band ordinate of the layer refers to the physical band that a layer (`layerNumber` parameter value) is associated with. For the current release, the associations must be as shown in Figure 1–4 in Section 1.5: layer 1 is band 0, layer 2 is band 1, and so on.

The band ordinate for the object layer is ignored by GeoRaster.

The GeoRaster object is automatically validated after the operation completes.

To return the band ordinate value for a layer, use the SDO_GEOR.getLayerOrdinate function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, if `ordinate` is null, or if ordinate does not equal `layerNumber`-1 when `layerNumber` does not specify the object layer.

Examples

The following example sets the band ordinate value for layer 1 to be 0 (zero) in the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setLayerOrdinate(grobj, 1, 0);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
```

/

SDO_GEOR.setModelSRID

Format

```
SDO_GEOR.setModelSRID(
   georaster  IN OUT SDO_GEORASTER,
   srid       IN NUMBER);
```

Description

Sets the coordinate system (SDO_SRID value) for the model (ground) space for a GeoRaster object, or deletes the existing value if you specify a null `srid` parameter.

Parameters

georaster
GeoRaster object.

srid
Coordinate system. Must be a value from the SRID column of the MDSYS.CS_SRS table if the GeoRaster metadata contains spatial reference information; or must be null (causing no coordinate system associated with the model space) if the GeoRaster metadata does not contain spatial reference information.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

If the original GeoRaster object had a different model space SRID value, this procedure does not change the raster data itself. In other words, this procedure does not cause any reprojection or resampling on the cell data of the GeoRaster object.

To return the coordinate system (SDO_SRID value) associated with the model space for a GeoRaster object, use the SDO_GEOR.getModelSRID function.

Examples

The following example changes the coordinate system for a GeoRaster object to *Longitude / Latitude (WGS 66)*, which is the coordinate system associated with SRID value 82394 in the MDSYS.CS_SRS system table. (The example refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setModelSRID(grobj, 82394);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setOrthoRectified

Format

```
SDO_GEOR.setOrthoRectified(
   georaster   IN OUT SDO_GEORASTER,
   isOrthoRectified  IN VARCHAR2);
```

Description

Specifies whether or not a GeoRaster object is orthorectified, or deletes the existing value if you specify a null `isOrthoRectified` parameter.

Parameters

georaster
GeoRaster object.

isOrthoRectified
Specify `TRUE` to specify that the GeoRaster object is orthorectified, `FALSE` to specify that the GeoRaster object is not orthorectified, or null if the GeoRaster metadata does not contain spatial reference information. Must be `TRUE` or `FALSE` (case-insensitive) if the GeoRaster metadata contains spatial reference information.

Usage Notes

This procedure modifies the GeoRaster metadata for the object. It does not actually orthorectify the object. Users are responsible for ensuring that orthorectification is performed.

The GeoRaster object is automatically validated after the operation completes.

To be set as orthorectified, a GeoRaster object must be spatially referenced and rectified.

Examples

The following example identifies the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table as orthorectified. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setOrthoRectified(grobj, 'TRUE');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setRasterType

Format

```
SDO_GEOR.setRasterType(
   georaster    IN OUT SDO_GEORASTER,
   rasterType  IN NUMBER);
```

Description

Sets the raster type of a GeoRaster object.

Parameters

georaster
GeoRaster object.

rasterType
Numeric value to be set as the rasterType attribute of the GeoRaster object. Must be a valid 5-digit numeric value, in the format described in Section 2.1.1.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

An exception is raised if `rasterType` is null or if the first three digits of the existing `rasterType` value are changed.

Examples

The following example sets the `rasterType` attribute value of a GeoRaster object to `20001`. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=2 FOR UPDATE;
  sdo_geor.setRasterType(grobj, 20001);
  UPDATE georaster_table SET georaster = grobj WHERE georid=2;
  COMMIT;
END;
/
```

SDO_GEOR.setRectified

Format

```
SDO_GEOR.setRectified(
   georaster   IN OUT SDO_GEORASTER,
   isRectified  IN VARCHAR2);
```

Description

Specifies whether or not a GeoRaster object is rectified, or deletes the existing value if you specify a null `isRectified` parameter.

Parameters

georaster
GeoRaster object.

isRectified
Specify `TRUE` to specify that the GeoRaster object is rectified, `FALSE` to specify that the GeoRaster object is not rectified, or null if the GeoRaster metadata does not contain spatial reference information. Must be `TRUE` or `FALSE` (case-insensitive) if the GeoRaster metadata contains spatial reference information.

Usage Notes

This procedure modifies the GeoRaster metadata for the object. It does not actually rectify the object. Users are responsible for ensuring that rectification is performed.

The GeoRaster object is automatically validated after the operation completes.

A GeoRaster object must be spatially referenced if you want to set `isRectified` to `TRUE` (see the SDO_GEOR.setSpatialReferenced procedure).

Examples

The following example identifies the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table as not rectified. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setRectified(grobj, 'false');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setScaling

Format

```
SDO_GEOR.setScaling(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   scalingFunc    IN SDO_NUMBER_ARRAY);
```

Description

Sets the scaling function associated with a layer, or deletes the existing value if you specify a null `scalingFunc` parameter.

> **Note:** GeoRaster does not perform operations using the scaling function in the current release.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

scalingFunc
An array of numeric values, with one value for each coefficient in the scaling function. The scaling function is as follows:

```
value = (a0 + a1 * cellvalue) / (b0 + b1 * cellvalue)
```

The order of the coefficients is: a_0, a_1, b_0, b_1.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object; if `scalingFunc` is of the wrong array size; if one of a_0, a_1, b_0, and b_1 is null; or if both b_0 and b_1 are 0 (zero).

Examples

The following example sets the coefficients of the scaling function for layer 2 of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setScaling(grobj, 2, sdo_number_array(1, 0.5, 1, 0));
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
```

/

SDO_GEOR.setSpatialReferenced

Format

```
SDO_GEOR.setSpatialReferenced(
   georaster      IN OUT SDO_GEORASTER,
   isReferenced   IN VARCHAR2);
```

Description

Specifies whether or not a GeoRaster object is spatially referenced, or deletes the existing value if you specify a null `isReferenced` parameter.

Parameters

georaster
GeoRaster object.

isReferenced
Specify `TRUE` to specify that the GeoRaster object is spatially referenced, `FALSE` to specify that the GeoRaster object is not spatially referenced, or null if the GeoRaster metadata does not contain spatial reference information. Must be `TRUE` or `FALSE` (case-insensitive) if the GeoRaster metadata contains spatial reference information.

Usage Notes

This procedure sets the GeoRaster object to be spatially referenced or not spatially referenced.

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table as not spatially referenced. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setSpatialReferenced(grobj, 'FALSE');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSpatialResolutions

Format

```
SDO_GEOR.setSpatialResolutions(
   georaster    IN OUT SDO_GEORASTER,
   resolutions  IN SDO_NUMBER_ARRAY);
```

Description

Sets the spatial resolution value along each spatial dimension of a GeoRaster object, or deletes the existing values if you specify a null `resolutions` parameter.

Parameters

georaster
GeoRaster object.

resolutions
An array of numeric values, one for each spatial dimension. Each value indicates the number of units of measurement associated with the data area represented by that spatial dimension of a pixel. For example, if the spatial resolution values are (10,10) and the unit of measurement for the ground data is meters, each pixel represents an area of 10 meters by 10 meters.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

If resolutions is not null and if the GeoRaster metadata currently does not contain spatial reference information, this procedure adds spatial reference information with minimum default values.

See also the Usage Notes for the SDO_GEOR.getSpatialResolutions function.

Examples

The following example sets the spatial resolution values along the column and row (X and Y) dimensions of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setSpatialResolutions(grobj, sdo_number_array(28.5,28.5));
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSpectralResolution

Format

```
SDO_GEOR.setSpectralResolution(
   georaster  IN OUT SDO_GEORASTER,
   resolution  IN NUMBER);
```

Description

Sets the spectral resolution of a GeoRaster object if it is a hyperspectral or multiband image, or deletes the existing value if you specify a null `resolution` parameter.

Parameters

georaster
GeoRaster object.

resolution
Spectral resolution value. Must be null if the GeoRaster metadata does not contain band reference information.

Usage Notes

Taken together, the spectral unit and spectral resolution identify the wavelength interval for a band. For example, if the spectral resolution value is 2 and the spectral unit value is `MILLIMETER`, the wavelength interval for a band is 2 millimeters.

The GeoRaster object is automatically validated after the operation completes.

To return the spectral resolution for a GeoRaster object, use the SDO_GEOR.getSpectralResolution function.

Examples

The following example sets 0.5 as the spectral resolution value for the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setSpectralResolution(grobj, 0.5);
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSpectralUnit

Format

```
SDO_GEOR.setSpectralUnit(
   georaster  IN OUT SDO_GEORASTER,
   unit       IN VARCHAR2);
```

Description

Sets the unit of measurement for identifying the wavelength interval for a band, or deletes the existing value if you specify a null `unit` parameter.

Parameters

georaster
GeoRaster object.

unit
Spectral unit. Must be one of the following values if the GeoRaster metadata contains band reference information: `METER`, `MILLIMETER`, `MICROMETER`, `NANOMETER`. Must be null if the GeoRaster metadata does not contain band reference information.

Usage Notes

Taken together, the spectral unit and spectral resolution identify the wavelength interval for a band. For example, if the spectral resolution value is 2 and the spectral unit value is `MILLIMETER`, the wavelength interval for a band is 2 millimeters.

The GeoRaster object is automatically validated after the operation completes.

To return the spectral unit for a GeoRaster object, use the SDO_GEOR.getSpectralUnit function.

Examples

The following example sets `MICROMETER` as the spectral unit for the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setSpectralUnit(grobj, 'micrometer');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSRS

Format

```
SDO_GEOR.setSRS(
   georaster  IN OUT SDO_GEORASTER,
   srs        IN SDO_GEOR_SRS);
```

Description

Sets the spatial reference information of a GeoRaster object, or deletes the existing information if you specify a null `srs` parameter.

Parameters

georaster
GeoRaster object.

srs
An object of type SDO_GEOR_SRS, which is described in Section 2.3.5.

In this object, `isReferenced`, `isRectified`, and `isOrthoRectified` must be `TRUE` or `FALSE` (case-insensitive); `spatialResolution` must be an array of the correct size; the spatial tolerance cannot be negative; `CoordLocation` must be 0 or 1; and the polynomial parameters cannot be null.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To return the SDO_GEOR_SRS information for a GeoRaster object, use the SDO_GEOR.getSRS function.

Examples

The following example specifies changes to several spatial reference attributes of a GeoRaster object, calls the setSRS procedure to update the spatial reference information, and then updates the GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
  srs   sdo_geor_srs;

BEGIN

SELECT georaster INTO grobj FROM georaster_table WHERE georid=4;
srs := sdo_geor.getSRS(grobj);
srs.isReferenced := 'TRUE';
srs.isRectified := 'TRUE';
srs.isOrthoRectified := null;
srs.srid := 82262;
srs.spatialResolution := sdo_number_array(28.5, 28.5);
srs.rowOff := 0;
srs.columnOff := 0;
srs.xOff := 0;
```

```
srs.yOff := 0;
srs.zOff := 0;
srs.rowScale := 1;
srs.columnScale := 1;
srs.xScale := 1;
srs.yScale := 1;
srs.zScale := 1;
srs.rowNumerator := SDO_NUMBER_ARRAY(1, 2, 1, 3, 32631.5614, 0, -.03508772);
srs.rowDenominator := SDO_NUMBER_ARRAY(1, 0, 0, 1, 1);
srs.columnNumerator := SDO_NUMBER_ARRAY(1, 2, 1, 3, -7894.7544, .035087719, 0);
srs.columnDenominator := SDO_NUMBER_ARRAY(1, 0, 0, 1, 1);

sdo_geor.setSRS(grobj, srs);

UPDATE georaster_table SET georaster = grobj WHERE georid=4;
COMMIT;
END;
/
```

SDO_GEOR.setStatistics

Format

```
SDO_GEOR.setStatistics(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   statistics       IN SDO_NUMBER_ARRAY);
```

Description

Sets statistical data associated with a layer.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the statistics. A value of 0 (zero) indicates the object layer.

statistics
An array with the following numeric values: `MIN`, `MAX`, `MEAN`, `MEDIAN`, `MODEVALUE`, `STD`. You must specify non-null values for all values in the array.

If this parameter is null, all statistical information associated with the layer is deleted.

Usage Notes

This procedure sets statistical data described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in Appendix A.

To retrieve the statistical data associated with a layer, use the SDO_GEOR.getStatistics function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if `statistics` is of the wrong array size or has any null array elements.

Examples

The following example sets the statistical data for layer 0 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setStatistics(grobj, 0, SDO_NUMBER_ARRAY(0, 255, 100, 127, 95, 25));
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setULTCoordinate

Format

```
SDO_GEOR.setULTCoordinate(
   georaster  IN OUT SDO_GEORASTER,
   ultCoord    IN SDO_NUMBER_ARRAY);
```

Description

Sets the cell coordinate values of the upper-left corner of a GeoRaster object, or deletes the existing values if you specify a null `ultCoord` parameter.

Parameters

georaster
GeoRaster object.

ultCoord
An array of two numbers (row and column ordinates) or three numbers (row, column, and band ordinates).

Usage Notes

If the metadata contains spatial reference information and the GeoRaster object is georeferenced, the spatial reference information is checked for validity. If it is valid, the spatial reference information including the georeferencing information is updated and adjusted according to the new ULT coordinates; otherwise, an exception is raised.

To return the upper-left coordinate values for a GeoRaster object, use the SDO_GEOR.getULTCoordinate function.

An exception is raised if `ultCoord` is null or of the wrong array size or has any null array elements.

Examples

The following example sets the row and column ordinates of the upper-left corner of a GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=1 FOR UPDATE;
  sdo_geor.setULTCoordinate(grobj, sdo_number_array(100, 100));
  UPDATE georaster_table SET georaster = grobj WHERE georid=1;
  COMMIT;
END;
/
```

SDO_GEOR.setVAT

Format

```
SDO_GEOR.setVAT(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   vatName        IN VARCHAR2);
```

Description

Sets the name of the value attribute table (VAT) associated with a layer of a GeoRaster object, or deletes the existing value if you specify a null `vatName` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

vatName
Name of the value attribute table.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

For more information about value attribute tables, see Section 1.2.3.

To return the name of the value attribute table associated with a layer of a GeoRaster object, use the SDO_GEOR.getVAT function.

An exception is raised if `layerNumber` is null or invalid for the GeoRaster object, or if `vatName` is an empty string (`''`).

Examples

The following example specifies `VATT1` as the value attribute table to be associated with layer 3 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setVAT(grobj, 3, 'VATT1');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.setVersion

Format

```
SDO_GEOR.setVersion(
   georaster      IN OUT SDO_GEORASTER,
   majorVersion   IN VARCHAR2,
   minorVersion   IN VARCHAR2);
```

Description

Sets the user-specified version of a GeoRaster object.

Parameters

georaster
GeoRaster object.

majorVersion
String representing the major version of the GeoRaster object. For example, if the complete version string is `15a.beta1`, specify the `majorVersion` value as `15a`.

If the parameter value is null, any existing `majorVersion` value in the GeoRaster object is deleted.

minorVersion
String representing the minor version of the GeoRaster object. For example, if the complete version string is `15a.beta1`, specify the `minorVersion` value as `beta1`.

If the parameter value is null, any existing `minorVersion` value in the GeoRaster object is deleted.

Usage Notes

The major and minor version strings can reflect any versioning scheme that you choose. The `majorVersion` and `minorVersion` values can be any string, except that neither can be an empty string (that is, neither can be `''`).

To retrieve the version string for a GeoRaster object, use the SDO_GEOR.getVersion function, which returns the version in the format *major-version.minor-version*.

Examples

The following example sets `15a.beta1` as the version for the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 4 in the GEORASTER_TABLE table. (The GEORASTER_TABLE table definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setVersion(grobj, '15a', 'beta1');
  UPDATE georaster_table SET georaster = grobj WHERE georid=4;
  COMMIT;
END;
/
```

SDO_GEOR.subset

Format

```
SDO_GEOR.subset(
    inGeoraster      IN SDO_GEORASTER,
    cropArea         IN SDO_GEOMETRY,
    layerNumbers     IN VARCHAR2,
    storageParam     IN VARCHAR2,
    outGeoraster     IN OUT SDO_GEORASTER);
```

or

```
SDO_GEOR.subset(
    inGeoraster      IN SDO_GEORASTER,
    cropArea         IN SDO_NUMBER_ARRAY,
    bandNumbers      IN VARCHAR2,
    storageParam     IN VARCHAR2,
    outGeoraster     IN OUT SDO_GEORASTER);
```

Description

Performs either or both of the following operations: (1) spatial crop, cut, or clip, or (2) layer or band subset or duplicate.

Parameters

inGeoraster
The SDO_GEORASTER object on which the operation or operations are to be performed.

cropArea
Crop area definition. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as the crop area; see also the Usage Notes for SDO_SRID requirements.

If `cropArea` is of type SDO_GEOMETRY, use the `layerNumbers` parameter to specify one or more layer numbers; if `cropArea` is of type SDO_NUMBER_ARRAY, use the `bandNumbers` parameter to specify one or more band numbers.

layerNumbers
A string identifying the logical layer numbers on which the operation or operations are to be performed. Use commas to delimit the values, and a hyphen to indicate a range (for example, `2-4` for layers 2, 3, and 4).

bandNumbers
A string identifying the physical band numbers on which the operation or operations are to be performed. Use commas to delimit the values, and a hyphen to indicate a range (for example, `1-3` for bands 1, 2, and 3).

storageParam
A string specifying storage parameters, as explained in Section 1.4.1.

outGeoraster
The new SDO_GEORASTER object.

Usage Notes

This procedure has a variety of possible uses. For example, you can call it to crop a small area or obtain a subset of a few layers of a GeoRaster object, you can duplicate layers, and you can specify storage parameters such as blocking and interleaving for the resulting object.

If the `cropArea` parameter data type is SDO_GEOMETRY, the SDO_SRID value must be one of the following:

- Null, to specify raster space
- A value from the SRID column of the MDSYS.CS_SRS table

  If the SDO_SRID values for the `cropArea` parameter geometry and the model space are different, the `window` parameter geometry is automatically transformed to the coordinate system of the model space before the operation is performed. (Raster space and model space are explained in Section 1.3.)

Any pyramid data in the input GeoRaster object is not copied to the output GeoRaster object.

An exception is raised if one or more of the following are true:

- `inGeoraster` is invalid.
- `outGeoraster` has not been initialized.
- A raster data table for `outGeoraster` does not exist and `outGeoraster` is not a blank GeoRaster object.

Examples

The following example creates a GeoRaster object that contains only specified bands from a specified window from the original object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
    VALUES (41, sdo_geor.init('RDT_1'))
    RETURNING georaster INTO gr2;

  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=4;

  sdo_geor.subset(gr1, sdo_geometry(2003, NULL, NULL,
                                sdo_elem_info_array(1, 1003, 3),
                                sdo_ordinate_array(0,256,255,511)),
                 '3,1-2', null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

SDO_GEOR.validateGeoraster

Format

```
SDO_GEOR.validateGeoraster(
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Validates a GeoRaster object, checking its raster data and metadata.

Parameters

georaster
GeoRaster object to be checked for validity.

Usage Notes

This function returns `TRUE` if the GeoRaster object is valid, `NULL` if the GeoRaster object is null, an Oracle error code if the error is known, or `FALSE` for an unknown error.

You should use this function after you create, load, or modify a GeoRaster object, to ensure that it is valid before you process it further.

If this function identifies a GeoRaster object as invalid with an error code of 13454, the object's metadata is not valid according to the GeoRaster XML schema. If this happens, call the SDO_GEOR.schemaValidate function to find specific locations and other information about the errors.

This function not only validates GeoRaster metadata against the GeoRaster XML schema, but it also enforces restrictions and requirements in the current release that are not described in the XML schema. The following are some of the restrictions and requirements enforced by this function:

- Layer numbers must be from 1 to *n* where *n* is the total number of layers.
- The `cellRepresentationType` value must be `UNDEFINED`.
- If `totalBandBlocks` or `bandBlockSize` is specified in the metadata, both must be specified. If there is only one band, no band blocking is allowed.
- The total number of blocks times the blocking size along a dimension must match the dimension size plus padding size, and the size of each cell data BLOB object must match the metadata description in terms of blocking or nonblocking.
- The size and number of GeoRaster data blocks stored in the raster data table must be consistent with the metadata description. For cell data, the number and size of the blocks are checked; the content of the blocks is not checked.
- The only pyramid types supported are `NONE` (no pyramids) and `DECREASE`. (For more information about pyramids, see Section 1.7.)
- For an uncompressed GeoRaster object, the size of the BLOB object in each raster block is checked based on the blocking size and cell depth. However, for a compressed GeoRaster object, the size of the BLOB object in each raster block is not checked. Thus, when a compressed GeoRaster object is decompressed, the data might not be valid with respect to size.

- Only one type of polynomial model is supported, as described in Section 1.6.1. The offsets, scales, and RMS values for the supported polynomial model are fixed. The pType, nVars, and number of coefficients are fixed for each polynomial, except for the values of the coefficients a, b, c, d, e, and f. The fixed values are described in Table 2–4 in Section 2.3.5. The SDO_GEOR_SRS data type is a precise map of the GeoRaster SRS polynomial model in the XML metadata document.
- The `RigorousModel` and `StoredFunction` georeferencing models are not supported, although you can store GCP (ground control point) data in an Oracle table and register the table name in the GeoRaster SRS metadata.
- Spatial resolutions can be inconsistent with the affine transformation scales if the GeoRaster object is georeferenced.
- GeoRaster temporal referencing and band referencing are not supported, although in the temporal reference system (TRS) and band reference system (BRS) you can store the beginning and ending date and time, the spectral resolution, the spectral unit, and related descriptive information.
- Only one `layerInfo` element is supported. A layer can be defined only along one dimension, and this dimension must be `BAND`. However, within the `layerInfo` element, the number of `subLayer` elements is limited only by the total number of layers. The layer number for the `objectLayer` elements is 0, and the layer numbers for `subLayer` elements are 1 to *n* where *n* is the total number of layers.
- The scaling function, BIN function, and statistical data or histogram can be stored in the GeoRaster metadata and must be valid against the XML schema, but the value ranges for these items are not restricted. GeoRaster interfaces that use this metadata are limited. Applications should validate this optional metadata before using it.
- The numbers of colormap values and grayscale mapping values are not restricted, but there must be no duplicate colormap or grayscale values, and the values in each array must be consistent with the `cellDepth` value of the GeoRaster object and must be in ascending order. The value range of the red, green, blue, alpha, and gray components must be integers from 0 to 255.
- This function does not check any external tables (such as a GCP table, bin table, histogram table, grayscale table, or colormap table) whose names are registered in the XML metadata.
- This function does not validate the spatial extent geometry, or whether or not the spatial relationship between the geometry and the raster data is correct. To validate the spatial extent geometry, use the SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT or SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT procedure, which are documented in *Oracle Spatial User's Guide and Reference*.
- This function does not validate the geometry specified in the `blockMBR` attribute in raster data tables, or whether or not the geometry precisely encloses the raster blocks. (The `blockMBR` attribute is described in Section 2.2.6.) To validate this geometry, use the SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT or SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT procedure, which are documented in *Oracle Spatial User's Guide and Reference*.

If there is no entry for the GeoRaster object in the ALL_SDO_GEOR_SYSDATA view (described in Section 2.4), this procedure returns an error stating that the GeoRaster object is not registered. To prevent this error, be sure that the GeoRaster object is inserted into a GeoRaster table and that this table has the required GeoRaster DML trigger created on it. To enable cross-schema access, you must also ensure that users

calling this procedure have an appropriate privilege on both the GeoRaster table and the associated raster data table.

Examples

The following example validates the GeoRaster objects in a table.

```
SELECT t.georid,
       sdo_geor.validategeoraster(t.georaster) isvalid
  from georaster_table t order by georid;

    GEORID ISVALID
---------- ----------
         3 TRUE
         4 TRUE
```

5

SDO_GEOR_UTL Package Reference

The MDSYS.SDO_GEOR_UTL package contains subprograms (functions and procedures) for utility operations related to GeoRaster. This chapter presents reference information, with one or more examples, for each subprogram.

SDO_GEOR_UTL.createDMLTrigger

Format

```
SDO_GEOR_UTL.createDMLTrigger(
   tableName     VARCHAR2,
   columnName  VARCHAR2);
```

Description

Creates the required standard GeoRaster data manipulation language (DML) trigger on a GeoRaster column in a GeoRaster table, so that the appropriate operations are performed when its associated trigger is fired.

Parameters

tableName
Name of a GeoRaster table (the table containing rows with at least one GeoRaster object column).

columnName
Name of a column of type SDO_GEORASTER in the GeoRaster table.

Usage Notes

To maintain the referential integrity of GeoRaster data structures, you should always use this procedure to create the standard GeoRaster DML trigger on each GeoRaster table and GeoRaster column combination before you perform any DML operations on the table. For example, if the GeoRaster table contains two GeoRaster columns, call this procedure twice, specifying the appropriate table name and column name for each call.

For more information about the standard GeoRaster DML trigger, including the operations that it performs, see Section 3.1.1.

Examples

The following example creates the standard GeoRaster DML trigger for a table named XYZ_GEOR_TAB containing a GeoRaster column named GEOR_COL.

```
EXECUTE sdo_geor_utl.createDMLTrigger('XYZ_GEOR_TAB', 'GEOR_COL');
```

SDO_GEOR_UTL.makeRDTNamesUnique

Format

SDO_GEOR_UTL.makeRDTNamesUnique;

Description

Renames some existing raster data tables that do not have unique names so that all raster data table names are unique within the database, and updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.

Parameters

None.

Usage Notes

If one or more raster data tables have the same name (under different schemas), you can use this procedure or the SDO_GEOR_UTL.renameRDT procedure, or both, to eliminate the duplication.

Run this procedure when you are connected to the database with the DBA role.

Examples

The following example automatically renames some existing raster data tables that do not have unique names so that all raster data table names are unique within the database, and it updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.

```
EXECUTE sdo_geor_utl.makeRDTNamesUnique;
```

SDO_GEOR_UTL.renameRDT

Format

```
SDO_GEOR_UTL.renameRDT(
   oldRDTs   VARCHAR2,
   newRDTs  VARCHAR2 DEFAULT NULL);
```

Description

Renames one or more existing raster data tables owned by the current user, and updates the GeoRaster system data and all affected GeoRaster objects to reflect the new names.

Parameters

oldRDTs
Name of the raster data table or tables to be renamed. For multiple tables, use a comma-delimited list.

newRDTs
New names to be assigned to the raster data table or tables that are specified in `oldRDTs`. For multiple tables, use a comma-delimited list with an order exactly reflecting the names in `oldRDTs`. If this parameter is null, GeoRaster assigns a unique new name to each input raster data table.

Usage Notes

If one or more raster data tables owned by different users have the same name, you can use this procedure or the SDO_GEOR_UTL.makeRDTNamesUnique procedure, or both, to eliminate the duplication.

Before using this procedure, you must connect to the database as the owner of the raster data table or tables. You cannot use this procedure to rename a raster data table owned by another user.

If any table in `oldRDTs` is not included in the GeoRaster system data, it is ignored.

If any table in `newRDTs` conflicts with a name in the GeoRaster system data or with the name of another object owned by the current user, an exception is raised.

Examples

The following example renames the raster data tables RDT_1 and RDT_2 to ST_RDT_1 and ST_RDT_2, respectively.

```
CONNECT scott/tiger;
. . .
EXECUTE sdo_geor_utl.renameRDT('RDT_1,RDT_2','ST_RDT_1,ST_RDT_2');
```

A

GeoRaster Metadata XML Schema

This appendix provides the XML schema definition that is used for GeoRaster metadata. The following is the definition of the GeoRaster metadata XML schema. (You can also see this definition by querying the SDO_GEOR_XMLSCHEMA_TABLE table, which is described in Section 2.5.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Oracle GeoRaster Metadata Schema -->
<xsd:schema targetNamespace="http://xmlns.oracle.com/spatial/georaster"
   xmlns="http://xmlns.oracle.com/spatial/georaster"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" version="0.0">
<xsd:annotation>
<xsd:documentation>=======================================
   This is the XML schema defining the metadata of Oracle GeoRaster object type.
   It consists of two parts: data type definitions and its element content.
    Part 1: Data Types
       Part 1.1: Data Types for Object Info
       Part 1.2: Data Types for Raster Info
       Part 1.3: Data Types for Spatial-Temporal-Band Reference Systems
       Part 1.3.1: Data Types for GeoRaster Spatial Reference System
       Part 1.3.2: Data Types for GeoRaster Temporal Reference System
       Part 1.3.3: Data Types for GeoRaster Band Reference System
       Part 1.4: Data Types for Layer Metadata
    Part 2: Metadata Elements / Content Structure of Oracle GeoRaster Object
                     ====================================
</xsd:documentation>
</xsd:annotation>
<xsd:annotation>
<xsd:documentation> ==========================
                         Part 1:      Data Types
         ====================================
 </xsd:documentation>
 </xsd:annotation>
 <xsd:annotation>
 <xsd:documentation> ==============================
          Part 1.1: Data Types for Object Info
          ==============================
 </xsd:documentation>
 </xsd:annotation>
 <xsd:complexType name="objectDescriptionType">
   <xsd:sequence>
      <xsd:element name="rasterType" type="xsd:integer"/>
      <xsd:element name="ID" type="xsd:string" minOccurs="0"/>
      <xsd:element name="description" type="xsd:string" minOccurs="0"
            maxOccurs="unbounded"/>
      <xsd:element name="majorVersion" type="xsd:string" minOccurs="0"/>
```

```
   <xsd:element name="minorVersion" type="xsd:string" minOccurs="0"/>
   <xsd:element name="isBlank" type="xsd:boolean" default="false"/>
   <xsd:element name="blankCellValue" type="xsd:double" minOccurs="0"/>
   <xsd:element name="defaultRed" type="xsd:positiveInteger" minOccurs="0"/>
   <xsd:element name="defaultGreen" type="xsd:positiveInteger"
           minOccurs="0"/>
   <xsd:element name="defaultBlue" type="xsd:positiveInteger" minOccurs="0"/>
   <xsd:any minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:annotation>
 <xsd:documentation> ==============================
         Part 1.2: Data Types for Raster Info
         ============================
 </xsd:documentation>
</xsd:annotation>
<xsd:simpleType name="cellRepresentationType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="POINT"/>
      <xsd:enumeration value="SEGMENT"/>
      <xsd:enumeration value="TRIANGLE"/>
      <xsd:enumeration value="SQUARE"/>
      <xsd:enumeration value="RECTANGLE"/>
      <xsd:enumeration value="CUBE"/>
      <xsd:enumeration value="TETRAHEDRON"/>
      <xsd:enumeration value="HEXAHEDRON"/>
      <xsd:enumeration value="UNDEFINED"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="cellDepthType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1BIT"/>
      <xsd:enumeration value="2BIT"/>
      <xsd:enumeration value="4BIT"/>
      <xsd:enumeration value="8BIT_U"/>
      <xsd:enumeration value="8BIT_S"/>
      <xsd:enumeration value="16BIT_U"/>
      <xsd:enumeration value="16BIT_S"/>
      <xsd:enumeration value="32BIT_U"/>
      <xsd:enumeration value="32BIT_S"/>
      <xsd:enumeration value="32BIT_REAL"/>
      <xsd:enumeration value="64BIT_REAL"/>
      <xsd:enumeration value="64BIT_COMPLEX"/>
      <xsd:enumeration value="128BIT_COMPLEX"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="supportedDimensionNumber">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="3"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="cellDimensionType">
   <xsd:annotation>
      <xsd:documentation>
      The "Band" dimension can be treated as any other semantic dimension
      or any "Layer" if not remote sensing imagery or photographs.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
```

```
      <xsd:enumeration value="ROW"/>
      <xsd:enumeration value="COLUMN"/>
      <xsd:enumeration value="VERTICAL"/>
      <xsd:enumeration value="BAND"/>
      <xsd:enumeration value="TEMPORAL"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="cellDimensionSizeType">
   <xsd:sequence>
      <xsd:element name="size" type="xsd:positiveInteger"
          default="1"/>
   </xsd:sequence>
   <xsd:attribute name="type" type="cellDimensionType"
          use="required"/>
</xsd:complexType>
<xsd:complexType name="cellCoordinateType">
   <xsd:sequence>
      <xsd:element name="row" type="xsd:integer" default="0"/>
      <xsd:element name="column" type="xsd:integer" default="0"/>
      <xsd:element name="vertical" type="xsd:integer"
               minOccurs="0"/>
      <xsd:element name="band" type="xsd:integer"
                minOccurs="0"/>
      <xsd:element name="temporal" type="xsd:integer"
               minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="compressionType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NONE"/>
        <xsd:enumeration value="RLE"/>
        <xsd:enumeration value="JPEG-B"/>
        <xsd:enumeration value="JPEG-F"/>
        <xsd:enumeration value="DEFLATE"/>
        <xsd:enumeration value="LT-MG2"/>
        <xsd:enumeration value="LT-MG3"/>
        <xsd:enumeration value="LT-JP2"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="compressionQuality">
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="compressionDescriptionType">
    <xsd:sequence>
        <xsd:element name="type" type="compressionType"
          default="NONE"/>
        <xsd:element name="quality" type="compressionQuality"
           minOccurs="0"/>
        <xsd:any minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="blockingType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NONE"/>
      <xsd:enumeration value="REGULAR"/>
   </xsd:restriction>
```

```
</xsd:simpleType>
<xsd:complexType name="blockingDescriptionType">
   <xsd:sequence>
      <xsd:element name="type" type="blockingType"
               default="NONE"/>
      <xsd:element name="totalRowBlocks" type="xsd:positiveInteger"
               default="1"/>
      <xsd:element name="totalColumnBlocks" type="xsd:positiveInteger"
               default="1"/>
      <xsd:element name="totalBandBlocks" type="xsd:positiveInteger"
               default="1" minOccurs="0"/>
      <xsd:element name="rowBlockSize" type="xsd:positiveInteger"/>
      <xsd:element name="columnBlockSize" type="xsd:positiveInteger"/>
      <xsd:element name="bandBlockSize" type="xsd:positiveInteger"
                minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="cellInterleavingType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="BSQ"/>
      <xsd:enumeration value="BIL"/>
      <xsd:enumeration value="BIP"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="pyramidType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NONE"/>
      <xsd:enumeration value="DECREASE"/>
      <xsd:enumeration value="INCREASE"/>
      <xsd:enumeration value="BIDIRECTION"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="resamplingType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NN"/>
      <xsd:enumeration value="BILINEAR"/>
      <xsd:enumeration value="CUBIC"/>
      <xsd:enumeration value="AVERAGE4"/>
      <xsd:enumeration value="AVERAGE16"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="pyramidDescriptionType">
   <xsd:sequence>
      <xsd:element name="type" type="pyramidType"
                default="NONE"/>
      <xsd:element name="resampling" type="resamplingType"
               default="NN" minOccurs="0"/>
      <xsd:element name="maxLevel" type="xsd:nonNegativeInteger"
               default="0" minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="rasterDescriptionType">
   <xsd:sequence>
      <xsd:element name="cellRepresentation"
               type="cellRepresentationType" default="UNDEFINED"/>
      <xsd:element name="cellDepth" type="cellDepthType"
               default="8BIT_U"/>
      <xsd:element name="NODATA" type="xsd:double"
```

```
                minOccurs="0"/>
      <xsd:element name="totalDimensions"
               type="supportedDimensionNumber" default="2"/>
      <xsd:element name="dimensionSize"
               type="cellDimensionSizeType" maxOccurs="5"/>
      <xsd:element name="ULTCoordinate" type="cellCoordinateType"/>
      <xsd:element name="blocking" type="blockingDescriptionType"/>
      <xsd:element name="interleaving" type="cellInterleavingType"
               default="BSQ"/>
      <xsd:element name="pyramid" type="pyramidDescriptionType"/>
      <xsd:element name="compression" type="compressionDescriptionType"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:annotation>
   <xsd:documentation>====================
  Part 1.3: Data Types for Spatial-Temporal-Band Reference Systems
    ==============================
         </xsd:documentation>
</xsd:annotation>
<xsd:annotation>
   <xsd:documentation>====================
  Part 1.3.1: Data Types for GeoRaster Spatial Reference System

  Spatial extent (footprint) is recorded as an attribute of the GeoRaster
  object. Its type is SDO_GEOMETRY, so it is not included in the metadata.
  The cell space coordinates are named as (row, column, vertical).
  The model space coordinates are named as (x, y, z).
  Spatial unit information is stored in the WKT of the specified SRID.
    ==============================
         </xsd:documentation>
</xsd:annotation>
<xsd:simpleType name="modelDimensionType">
   <xsd:annotation>
      <xsd:documentation>
The following "S" means "Spectral" for remote sensing imagery.
Any of X, Y and Z can be horizontal or vertical or any other spatial direction
depending on the user interpretation in "modelDimensionDescription".
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="X"/>
      <xsd:enumeration value="Y"/>
      <xsd:enumeration value="Z"/>
      <xsd:enumeration value="T"/>
      <xsd:enumeration value="S"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="resolutionType">
   <xsd:sequence>
      <xsd:element name="resolution" type="xsd:double"
              default="1"/>
   </xsd:sequence>
   <xsd:attribute name="dimensionType"
            type="modelDimensionType" use="required"/>
</xsd:complexType>
<xsd:simpleType name="doubleNumberListType">
   <xsd:list itemType="xsd:double"/>
</xsd:simpleType>
<xsd:complexType name="polynomialType">
```

```
      <xsd:sequence>
         <xsd:element name="polynomialCoefficients"
                  type="doubleNumberListType"/>
      </xsd:sequence>
      <xsd:attribute name="pType" type="xsd:nonNegativeInteger"
               use="optional" default="1"/>
      <xsd:attribute name="nVars" type="xsd:nonNegativeInteger"
               use="required"/>
      <xsd:attribute name="order" type="xsd:nonNegativeInteger"
               use="required"/>
      <xsd:attribute name="nCoefficients" type="xsd:nonNegativeInteger"
               use="required"/>
      <xsd:anyAttribute/>
   </xsd:complexType>
   <xsd:complexType name="rationalPolynomialType">
      <xsd:annotation>
         <xsd:documentation>
         row       =   pPolynomial(x, y, z) / qPolynomial(x, y, z)
         column =   rPolynomial(x, y, z) / sPolynomial(x, y, z)
               </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="pPolynomial" type="polynomialType"/>
         <xsd:element name="qPolynomial" type="polynomialType"/>
         <xsd:element name="rPolynomial" type="polynomialType"/>
         <xsd:element name="sPolynomial" type="polynomialType"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="rowOff" type="xsd:double" use="required"/>
      <xsd:attribute name="columnOff" type="xsd:double" use="required"/>
      <xsd:attribute name="xOff" type="xsd:double" use="required"/>
      <xsd:attribute name="yOff" type="xsd:double" use="required"/>
      <xsd:attribute name="zOff" type="xsd:double" use="required"/>
      <xsd:attribute name="rowScale" type="xsd:double" use="required"/>
      <xsd:attribute name="columnScale" type="xsd:double" use="required"/>
      <xsd:attribute name="xScale" type="xsd:double" use="required"/>
      <xsd:attribute name="yScale" type="xsd:double" use="required"/>
      <xsd:attribute name="zScale" type="xsd:double" use="required"/>
      <xsd:attribute name="rowRMS" type="xsd:double" use="optional"/>
      <xsd:attribute name="columnRMS" type="xsd:double" use="optional"/>
      <xsd:attribute name="totalRMS" type="xsd:double" use="optional"/>
      <xsd:anyAttribute/>
   </xsd:complexType>
   <xsd:simpleType name="rasterSpatialReferenceModelType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="RigorousModel"/>
         <xsd:enumeration value="StoredFunction"/>
         <xsd:enumeration value="FunctionalFitting"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="rasterSpatialReferenceSystemType">
      <xsd:sequence>
         <xsd:element name="isReferenced" type="xsd:boolean" default="false"/>
         <xsd:element name="isRectified" type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="isOrthoRectified" type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="description" type="xsd:string" minOccurs="0"/>
         <xsd:element name="SRID" type="xsd:nonNegativeInteger" default="0"/>
         <xsd:element name="verticalSRID" type="xsd:integer" minOccurs="0"/>
         <xsd:element name="modelDimensionDescription" type="xsd:string"
minOccurs="0"/>
```

```
         <xsd:element name="spatialResolution" type="resolutionType" minOccurs="0"
maxOccurs="3"/>
         <xsd:element name="spatialTolerance" type="xsd:double" minOccurs="0"/>
         <xsd:element name="modelCoordinateLocation" minOccurs="0">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="CENTER"/>
                  <xsd:enumeration value="UPPERLEFT"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="modelType" type="rasterSpatialReferenceModelType"
minOccurs="0" maxOccurs="3"/>
         <xsd:element name="polynomialModel" type="rationalPolynomialType"
minOccurs="0"/>
         <xsd:element name="gcpTableName" type="xsd:string" minOccurs="0"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:annotation>
      <xsd:documentation> ==========
      Part 1.3.2: Data Types for GeoRaster Temporal Reference System

      The TRS will be modeled by formulas in the future.
                                          ======================
      </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType name="rasterTemporalReferenceSystemType">
      <xsd:sequence>
         <xsd:element name="isReferenced" type="xsd:boolean" default="false"/>
         <xsd:element name="description" type="xsd:string" minOccurs="0"/>
         <xsd:element name="beginDateTime" type="xsd:dateTime" minOccurs="0"/>
         <xsd:element name="endDateTime" type="xsd:dateTime" minOccurs="0"/>
         <xsd:element name="temporalResolutionDescription" type="xsd:string"
                 minOccurs="0"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:annotation>
      <xsd:documentation> ==================
       Part 1.3.3: Data Types for GeoRaster Band Reference System

       For multispectral remote sensing images, each band is optionally
       described in the layerDescriptionType.
       The BRS is modeled by formulas for hyperspectral imagery
       (based on number of spectral segments, min and max wavelength,
       and number of bands for each segment).
       Detailed radiometric information will be added in the future.
       ========================
       </xsd:documentation>
   </xsd:annotation>
   <xsd:simpleType name="wavelengthUnit">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="METER"/>
         <xsd:enumeration value="MILLIMETER"/>
         <xsd:enumeration value="MICROMETER"/>
         <xsd:enumeration value="NANOMETER"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="extentType">
```

```
      <xsd:sequence>
         <xsd:element name="min" type="xsd:double"/>
         <xsd:element name="max" type="xsd:double"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="segmentationDataType">
      <xsd:sequence>
         <xsd:element name="totalSegNumber" type="xsd:positiveInteger"
                 default="1"/>
         <xsd:element name="firstSegNumber" type="xsd:integer"
                 default="1"/>
         <xsd:element name="extent" type="extentType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="bandReferenceType">
      <xsd:sequence>
         <xsd:element name="bands" type="segmentationDataType"
                  minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="rasterBandReferenceSystemType">
      <xsd:sequence>
         <xsd:element name="isReferenced" type="xsd:boolean"
                  default="false"/>
         <xsd:element name="description" type="xsd:string"
                  minOccurs="0"/>
         <xsd:element name="radiometricResolutionDescription"
                   type="xsd:string" minOccurs="0"/>
         <xsd:element name="spectralUnit" type="wavelengthUnit"
                  default="MICROMETER"/>
         <xsd:element name="spectralTolerance" type="xsd:double"
                  minOccurs="0"/>
         <xsd:element name="spectralResolutionDescription"
                  type="xsd:string" minOccurs="0"/>
         <xsd:element name="minSpectralResolution"
                  type="resolutionType" minOccurs="0"/>
         <xsd:element name="spectralExtent" type="extentType"/>
         <xsd:element name="bandReference"
                  type="bandReferenceType" minOccurs="0"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:annotation>
      <xsd:documentation> ===================
       Part 1.4: Data Types for Layer Metadata

For each sublayer the layerNumber is a positive integer; that is, layers are
logically numbered from 1 to n if the size of the specified layerDimension is n.
The layerDimensionOrdinate of each sublayer must be in the range of the
dimension and must be in the order of band ordinates.
For objectLayer, the layerNumber should be 0 but its layerDimensionOrdinate
is not used.
=======================
    </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType name="scalingFunctionType">
      <xsd:annotation>
         <xsd:documentation>
          value = (a0 + a1 * cellValue)  /  (b0 + b1 * cellValue)
```

```
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="a0" type="xsd:double" default="1"/>
         <xsd:element name="a1" type="xsd:double" default="0"/>
         <xsd:element name="b0" type="xsd:double" default="1"/>
         <xsd:element name="b1" type="xsd:double" default="0"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="binType">
      <xsd:annotation>
      <xsd:documentation>
LINEAR bin function:
binNumber = numbins * (cellValue - min) / (max - min) + firstBinNumber
if (binNumber less than 0) binNumber =  firstBinNumber
if (binNumber greater than or equal to numbins) binNumber = numbins +
     firstBinNumber - 1
LOGARITHM bin function:
binNumber = numbins * (ln (1.0 + ((cellValue - min)/(max - min)))/ ln (2.0))
     + firstBinNumber
if (binNumber less than 0) binNumber =  firstBinNumber
if (binNumber greater than or equal to numbins) binNumber = numbins +
      firstBinNumber - 1
EXPLICIT bin function means explicit (or direct) value (or value range)
for each bin, and it will be stored in a table.
      </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="LINEAR"/>
         <xsd:enumeration value="LOGARITHM"/>
         <xsd:enumeration value="EXPLICIT"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="binFunctionType">
      <xsd:annotation>
         <xsd:documentation>
The MAX and MIN in the statistics data set will be used if they are not provided
here. binTableName is used by EXPLICIT type only.
          </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:choice>
            <xsd:element name="binFunctionData" type="segmentationDataType"/>
            <xsd:element name="binTableName" type="xsd:string"/>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
         </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="type" type="binType" use="required"/>
   </xsd:complexType>
   <xsd:complexType name="rectangularWindowType">
      <xsd:sequence>
         <xsd:element name="origin" type="cellCoordinateType"/>
         <xsd:element name="rowHeight" type="xsd:positiveInteger"/>
         <xsd:element name="columnWidth" type="xsd:positiveInteger"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="cellCountType">
      <xsd:attribute name="value" type="xsd:double" use="required"/>
```

```
   <xsd:attribute name="count" type="xsd:nonNegativeInteger"
            use="required"/>
   <xsd:anyAttribute/>
</xsd:complexType>
<xsd:complexType name="rasterCountType">
   <xsd:sequence>
      <xsd:element name="cell" type="cellCountType"
               maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="histogramType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="counts" type="rasterCountType"/>
         <xsd:element name="tableName" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="statisticDatasetType">
   <xsd:sequence>
      <xsd:element name="samplingFactor" type="xsd:positiveInteger"
              default="1"/>
      <xsd:element name="samplingWindow"
               type="rectangularWindowType" minOccurs="0"/>
      <xsd:element name="MIN" type="xsd:double"/>
      <xsd:element name="MAX" type="xsd:double"/>
      <xsd:element name="MEAN" type="xsd:double"/>
      <xsd:element name="MEDIAN" type="xsd:double"/>
      <xsd:element name="MODEVALUE" type="xsd:double"/>
      <xsd:element name="STD" type="xsd:double"/>
      <xsd:element name="histogram" type="histogramType"
               minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="cellGrayType">
   <xsd:attribute name="value" type="xsd:double" use="required"/>
   <xsd:attribute name="gray" type="xsd:integer" use="required"/>
   <xsd:anyAttribute/>
</xsd:complexType>
<xsd:complexType name="rasterGrayType">
   <xsd:sequence>
      <xsd:element name="cell" type="cellGrayType"
               maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="grayScaleType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="grays" type="rasterGrayType"/>
         <xsd:element name="tableName" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="cellPseudoColorType">
   <xsd:attribute name="value" type="xsd:double" use="required"/>
   <xsd:attribute name="red" type="xsd:integer" use="required"/>
   <xsd:attribute name="green" type="xsd:integer" use="required"/>
   <xsd:attribute name="blue" type="xsd:integer" use="required"/>
   <xsd:attribute name="alpha" type="xsd:double" use="optional"/>
```

```
      <xsd:anyAttribute/>
   </xsd:complexType>
   <xsd:complexType name="rasterPseudoColorType">
      <xsd:sequence>
         <xsd:element name="cell" type="cellPseudoColorType"
                  maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="colorMapType">
      <xsd:sequence>
         <xsd:choice>
            <xsd:element name="colors" type="rasterPseudoColorType"/>
            <xsd:element name="tableName" type="xsd:string"/>
         </xsd:choice>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="layerType">
      <xsd:sequence>
         <xsd:element name="layerNumber" type="xsd:nonNegativeInteger"/>
         <xsd:element name="layerDimensionOrdinate" type="xsd:integer"/>
         <xsd:element name="layerID" type="xsd:string"/>
         <xsd:element name="description" type="xsd:string" minOccurs="0"
                 maxOccurs="unbounded"/>
         <xsd:element name="scalingFunction" type="scalingFunctionType"
                 minOccurs="0"/>
         <xsd:element name="binFunction" type="binFunctionType"
                  minOccurs="0"/>
         <xsd:element name="statisticDataset"
                  type="statisticDatasetType" minOccurs="0"/>
         <xsd:element name="grayScale" type="grayScaleType" minOccurs="0"/>
         <xsd:element name="colorMap" type="colorMapType" minOccurs="0"/>
         <xsd:element name="vatTableName" type="xsd:string" minOccurs="0"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="layerDescriptionType">
      <xsd:sequence>
         <xsd:element name="layerDimension"
                  type="cellDimensionType" default="BAND"/>
         <xsd:element name="objectLayer" type="layerType" minOccurs="0"/>
         <xsd:element name="subLayer" type="layerType"
                  minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:annotation>
      <xsd:documentation> =================
Part 2: Metadata Elements / Content Structure of Oracle GeoRaster Object
==============================
    </xsd:documentation>
   </xsd:annotation>
   <xsd:element name="georasterMetadata">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="objectInfo" type="objectDescriptionType"/>
            <xsd:element name="rasterInfo" type="rasterDescriptionType"/>
            <xsd:element name="spatialReferenceInfo"
                  type="rasterSpatialReferenceSystemType" minOccurs="0"/>
            <xsd:element name="temporalReferenceInfo"
                  type="rasterTemporalReferenceSystemType" minOccurs="0"/>
```

```
            <xsd:element name="bandReferenceInfo"
                  type="rasterBandReferenceSystemType" minOccurs="0"/>
            <xsd:element name="layerInfo" type="layerDescriptionType"
                  maxOccurs="unbounded"/>
            <xsd:element name="sourceInfo" type="xsd:string"
                  minOccurs="0" maxOccurs="unbounded"/>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

T

U

V

W

X

Z