Oracle® Spatial

GeoRaster Developer's Guide

11*g* Release 2 (11.2)

E11827-05

August 2011

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

ORACLE®

Oracle Spatial GeoRaster Developer's Guide, 11*g* Release 2 (11.2)

E11827-05

Primary Author: Chuck Murray

Contributors: Fengting Chen, Qingyun (Jeffrey) Xie, Weisheng (Terry) Xu, Zhihai Zhang, Hongwei Zhu

Contents

4 SDO_GEOR Package Reference

5 SDO_GEOR_ADMIN Package Reference

List of Examples

List of Figures

List of Tables

Preface

Oracle Spatial GeoRaster Developer's Guide 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 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 Developer's Guide*.

Audience

This guide is intended for anyone who needs to store raster 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following document:

- *Oracle Spatial Developer's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in GeoRaster?

This section describes new and changed features of Oracle Spatial GeoRaster for Oracle Database Release 11.

Release 11.2 Changes

The following are new and changed features for Oracle Database 11*g* Release 2 (11.2).

Java API

GeoRaster now includes a Java API, which consists of interfaces and classes that support features available with the GeoRaster feature of Oracle Spatial. For more information, see Section 1.13.

Ground Control Point (GCP) Support

You can use ground control points (GCPs) to georeference GeoRaster objects. The support for this feature includes several new data types and PL/SQL subprograms, as well as enhancements to some existing types and subprograms. For information, see the following:

- Section 1.6.2, "Georeferencing Using GCPs"
- Section 2.3.5, "SDO_GEOR_SRS Object Type" (includes all GCP-related information)
- Section 2.3.6, "SDO_GEOR_GCP Object Type"
- Section 2.3.7, "SDO_GEOR_GCP_ COLLECTION Collection Type"
- Section 2.3.8, "SDO_GEOR_GCPGEOREFTYPE Object Type"
- SDO_GEOR.deleteControlPoint procedure
- SDO_GEOR.georeference (new function formats for using GCPs)
- SDO_GEOR.getControlPoint function
- SDO_GEOR.getGCPGeorefMethod function
- SDO_GEOR.getGCPGeorefModel function
- SDO_GEOR.getSRS function (includes GCP-related information)
- SDO_GEOR.setControlPoint procedure
- SDO_GEOR.setGCPGeorefMethod procedure
- SDO_GEOR.setGCPGeorefModel procedure
- SDO_GEOR.setSRS procedure (includes GCP-related information)

- Appendix A, "GeoRaster Metadata XML Schema" (includes new GCP-related elements)

Reprojection of GeoRaster Objects

You can reproject a GeoRaster object to a different Oracle Spatial coordinate system by using the new SDO_GEOR.reproject, which is documented in Chapter 4.

Optimized Blocking to Reduce Padding Space

For raster data that is blocked, you can specify the new OPTIMALPADDING keyword to cause any user-specified `blockSize` value for storage parameters to be adjusted automatically to an optimal value to reduce padding space. For more information, see the descriptions on the `blocking` and `blockSize` keywords in Table 1–1, " storageParam Keywords for Raster Data" in Section 1.4.1.

You can also use the new SDO_GEOR_UTL.calcOptimizedBlockSize procedure (documented in Chapter 6) to calculate an optimal `blockSize` value that will use less padding space in the GeoRaster object storage, and you can apply the result in any subprogram that has the `storageParam` parameter.

Grid Interpolations

The new SDO_GEOR.evaluateDouble function (documented in Chapter 4) evaluates a direct location based on neighboring cell values by using a specified interpolation method, and returns the raster values (double precision numbers) for the specified bands or layers for that location.

Polygon-Based Clipping in Queries

The SDO_GEOR.getRasterSubset and SDO_GEOR.subset procedures (documented in Chapter 4) are enhanced. In previous releases, only the MBR (rectangle) of the query polygon was used. Now, you can also clip the query result along the (irregular) polygon boundary.

setModelCoordLocation Procedure

The new SDO_GEOR.setModelCoordLocation procedure (documented in Chapter 4) enables you to change the cell coordinate system from CENTER to UPPERLEFT or from UPPERLEFT to CENTER. It applies only to georeferenced GeoRaster objects, and it automatically adjusts the functional fitting coefficients of the GeoRaster SRS accordingly to reflect the change (to ensure that the relationship between cell coordinates and model coordinates does not change).

getCellValue Function Allows Multiple Bands or Layers

The SDO_GEOR.getCellValue function (documented in Chapter 4) is enhanced to return cell values of multiple layers or bands for a specified location. In previous releases, it returned only a single cell value of a specified layer or band.

getGeoreferenceType Function Return Values

The SDO_GEOR.getGeoreferenceType function (documented in Chapter 4) can return the following values for georeference type in addition to those documented in the previous release of this manual: 5 for cubic polynomial, 6 for quadratic rational polynomial, or 7 for quadratic polynomial.

Release 11.1 Changes

The following are new and changed features for Oracle Database 11*g* Release 1 (11.1).

Enhanced Ease of Use, Reliability, and Manageability

For the current release, GeoRaster automates certain tasks that previously needed to be performed manually, and it provides new administrative tools for users who need to perform specialized tasks.

- When you create a GeoRaster table, you no longer need to create the GeoRaster DML triggers for the table. These DML triggers are created automatically, and their automatic creation and operation provides greater reliability.
- When you import GeoRaster data using the Data Pump Import utility, you no longer need to create GeoRaster DML triggers for the imported tables or GeoRaster system data entries for the imported GeoRaster objects. These triggers and entries are now created automatically.
- Internal enhancements that monitor DDL events on raster tables and activities on GeoRaster system data improve the manageability, reliability, robustness, and usability of GeoRaster.
- The new SDO_GEOR_ADMIN PL/SQL package contains subprograms to retrieve information that may be useful and to help you manage and maintain GeoRaster databases, including performing migration and upgrade operations. The SDO_GEOR_ADMIN subprograms are documented in Chapter 5.
- Raster data versioning with Oracle Workspace Manager is supported, as explained in Section 3.18.
- Raster data row-level security with Oracle Label Security is supported, as explained in Section 3.18.

For more information about GeoRaster and database management, see Section 1.11.

New Metadata and Raster Support

The current release provides support for new GeoRaster metadata and raster types:

- A generic and complex functional fitting georeferencing model is supported for georeferencing rectified and unrectified airborne photos and satellite images. The affine transformation, DLT, RPC, and other models are supported as special cases of this generic model. For more information, see Section 1.6.
- Bitmap masks (special one-bit deep rectangular raster grids with each pixel having either the value of 0 or 1) can be stored for GeoRaster objects and individual bands or layers. These masks are stored inside the GeoRaster objects. Pyramids of masks can also be created and stored inside the GeoRaster objects. For more information about bitmap masks, see Section 1.8.
- Multiple NODATA values and multiple NODATA value ranges are supported for GeoRaster objects and their individual bands or layers. For more information, see Section 1.9.
- GeoRaster objects can have empty raster blocks to save storage space and improve processing speed. For more information, see Section 1.4.4.
- Random blocking size is supported. Although each block must still have the same size, the raster blocking sizes can be randomly different numbers along row and column dimensions, not necessarily a power of 2.

New Functions, Procedures, and Other Features

The current release provides many new subprograms and other enhancements related to the GeoRaster PL/SQL API and other features, including the following:

- Update, query, and other DML operations on the new georeferencing models, bitmap masks, and multiple NODATA values and value ranges are supported.
- Existing subprograms are enhanced to support empty raster blocks, random blocking sizes, and the new metadata.
- Mosaic support allows for gaps, overlaps, and missing source GeoRaster objects. For information about mosaic support, see the description of the SDO_GEOR.mosaic procedure in Chapter 4.
- The union or merging of GeoRaster objects and layers is supported. For information, see the description of the SDO_GEOR.mergeLayers procedure in Chapter 4.
- Partial edit and update on a window of raster data and its pyramids using another image or gridded data is supported. For information, see the description of the SDO_GEOR.updateRaster procedure in Chapter 4.
- New GeoRaster template functions are provided to facilitate third-party software integration. For more information, see Section 3.17.
- Statistical analysis and histogram generation are supported. For information, see the description of the SDO_GEOR.generateStatistics procedure in Chapter 4.
- Sub-cell or sub-pixel addressing (floating row and column numbers) is supported in the GeoRaster cell spaces, as explained in Section 1.3.
- A new constructor is added to the SDO_GEOR_SRS object type, as explained in Section 2.3.5.
- Calculation of the actual and nominal storage sizes of a GeoRaster object is supported. (The nominal size does not consider compression and sparse data.) For more information, see the sections about the SDO_GEOR_UTL.calcRasterStorageSize and SDO_GEOR_UTL.calcRasterNominalSize functions in Chapter 6.
- GeoRaster validation is enhanced to require that each valid GeoRaster object must be registered (with an entry in the ALL_SDO_GEOR_SYSDATA view), the raster data table name attribute of the GeoRaster object must not contain spaces, period separators, or mixed-case letters in a quoted string, and all the alphanumeric characters must be uppercase. For information about validating GeoRaster objects, see Section 3.4, which also refers to specific functions for performing validation.
- GeoRaster supports the use of SecureFile LOBs (SecureFiles) which were introduced in Release 11.1 to supplement the original BasicFile LOBs. For more information, see Section 3.1.2.
- GeoTiff, JPEG 2000, and Digital Globe RPC file formats are supported for loading and exporting GeoRaster objects. JPEG files can be loaded without decompression. For more information about the GeoRaster loader and exporter tools, see Section 1.14.
- The GeoRaster viewer is enhanced to display masks, generic georeferencing models, empty raster blocks, and other features. For more information about the GeoRaster viewer tool, see Section 1.14.

The following table lists the new PL/SQL subprograms for this release. (It does not include existing subprograms that were significantly enhanced for this release.)

PL/SQL Package	New Subprograms
SDO_GEOR (documented in Chapter 4)	SDO_GEOR.addNODATA SDO_GEOR.addSourceInfo SDO_GEOR.createTemplate SDO_GEOR.deleteNODATA SDO_GEOR.generateBlockMBR SDO_GEOR.generateStatistics SDO_GEOR.getBinFunction SDO_GEOR.getBitmapMask SDO_GEOR.getBitmapMaskSubset SDO_GEOR.getBitmapMaskValue SDO_GEOR.getGeoreferenceType SDO_GEOR.getModelCoordLocation SDO_GEOR.getRasterBlockLocator SDO_GEOR.getSourceInfo SDO_GEOR.hasBitmapMask SDO_GEOR.hasNODATAMask SDO_GEOR.mergeLayers SDO_GEOR.setBinFunction SDO_GEOR.setBitmapMask SDO_GEOR.setSourceInfo SDO_GEOR.updateRaster SDO_GEOR.validateBlockMBR
SDO_GEOR_ADMIN (documented in Chapter 5)	(All subprograms are new because the SDO_GEOR_ADMIN package is new for this release.)
SDO_GEOR_UTL (documented in Chapter 6)	SDO_GEOR_UTL.calcRasterNominalSize SDO_GEOR_UTL.calcRasterStorageSize

1

GeoRaster Overview and Concepts

GeoRaster is a feature of Oracle Spatial that lets you store, index, query, analyze, and deliver 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 location based services, geoimagery archiving, environmental monitoring and assessment, geological engineering and exploration, natural resource management, defense, emergency response, telecommunications, transportation, urban planning, and homeland security.

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 Developer's Guide*.

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.

GeoRaster uses and depends upon several components that are included with Oracle Database, including the Java virtual machine (JVM) and Oracle XML DB.

Installation and Upgrade Notes: You must ensure that Oracle XML DB Repository is properly 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 Developer's Guide*.

After a database upgrade, you should call the SDO_GEOR_ADMIN.isUpgradeNeeded function to check for any invalid GeoRaster objects and invalid system data for the current version. For more information, see Section 3.20.

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 Raster Data Sources

Raster 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 TIFF, GIF, JFIF (JPEG), and Sun Raster, as well as in many geoimage formats, such as NITF, GeoTIFF, ERDAS IMG, and PCI PIX. 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, which is sampled at the center of the cell. 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. 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.

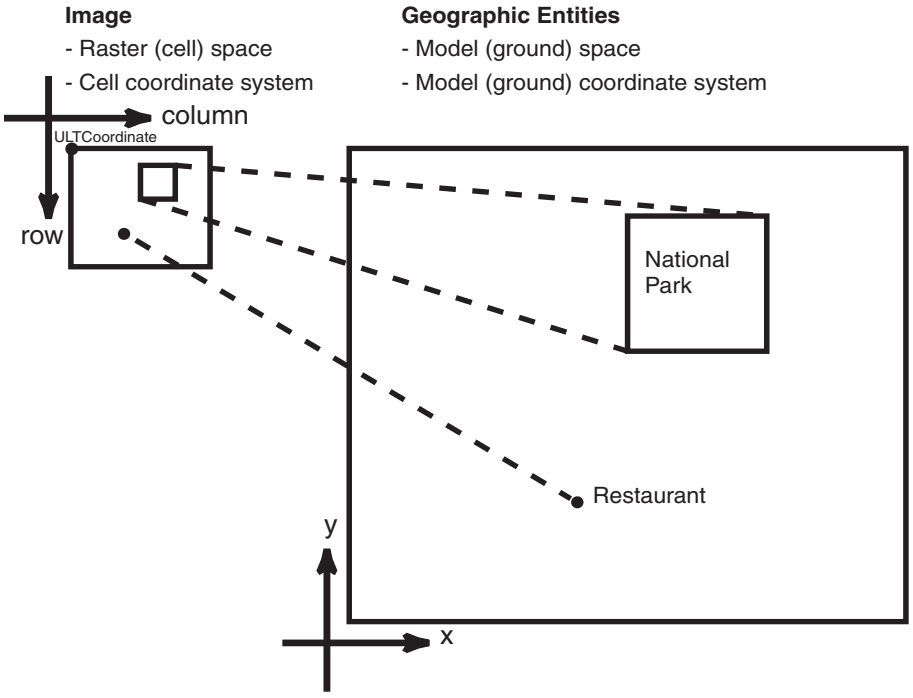
The GeoRaster object data types, described in Chapter 2, are based on the GeoRaster data model.

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 cells in the raster matrix and their spacing, 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.

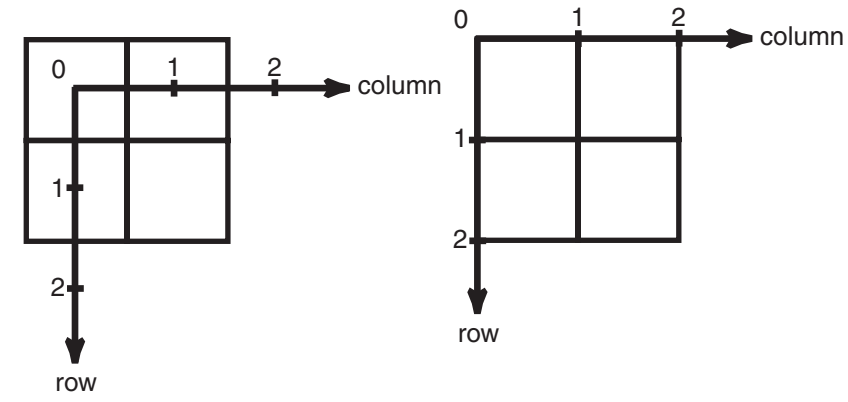
For two-dimensional single-layer GeoRaster data, the cell coordinate system has a column dimension pointing to the right and a row dimension pointing downward, as shown in Figure 1–1. The origin of the cell space is always (0,0). The spacing is 1 cell or 1 pixel, and in most cases the cell coordinates 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.

When the raster data is treated and processed as an array of numbers, integer addressing using column and row numbers is sufficient in most applications. However, the raster data array is generally a discretized representation of a continuous space, and so a one-to-one mapping of coordinates between the cell space and the model space is required, regardless of whether the value of a cell represents a collective value of an area or a single value of a point.

In other words, sub-cell (sub-pixel) addressing in the cell space is necessary. To support sub-cell addressing, GeoRaster defines two types of cell coordinate systems, depending on where the origin (0,0) of cells is defined. Figure 1–2, where each square represents one cell, shows the two types of cell coordinate systems: center-based and upperleft-based.

Figure 1–2 Two Types of Cell Coordinate Systems



The default cell coordinate system has its origin at the *center* of a cell, and is called the center-based cell coordinate system. The other cell coordinate system has its origin at the upper-left corner of a cell, and is called the upperleft-based cell coordinate system. In both systems, the cells are squares with equal size and the unit is 1 cell. Assuming that I and J are integers, and x and y are floating numbers:

- In center-based cell space, coordinate (x, y) is mapped to (I,J) as long as I-0.5 <= x < I+0.5 and J-0.5 <= y < J+0.5.
- In upperleft-based cell space, coordinate (x, y) is mapped to cell (I,J) as long as I <= x < I+1.0 and J <= y < J+1.0.

For example, sub-cell coordinate (0.3, 0.3) has the same integer cell coordinate (0,0) in both coordinate systems, while (0.3,0.6) means (0,1) in center-based cell space but means (0,0) in upperleft-based cell space. This two types of cell coordinate systems are defined by the `modelCoordinateLocation` element in the `spatialReferenceInfo` metadata; otherwise, the default type is center-based. GeoRaster supports both cell coordinate systems, and effective with Oracle Database

11*g*, sub-cell addresses are supported in the GeoRaster PL/SQL API. (Sub-cell addresses were internally supported in previous releases.)

In GeoRaster, while the origin of the cell space is always at (0,0), the upper-left corner cell of the raster data itself can have a different coordinate in its cell space from the coordinate of the origin of the cell space. In other words, the integer (row, column) coordinate of the upper-left corner cell is not necessarily (0,0). The upper-left corner is called the **ULTCoordinate**, and its value is registered in the metadata. It basically defines the relative location of the data in the cell space. 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 might not be exactly at the ULTCoordinate value.

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 SRS currently supports two spatial dimensions (X,Y) and three spatial dimensions (X, Y, Z) in the model coordinate system. (For information about coordinate systems, including the different types of coordinate systems, see *Oracle Spatial Developer's Guide*.)

The GeoRaster model coordinate system is defined by an Oracle Spatial SRID. 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 spatial reference system (SRS) must be decimal degrees, and values should be in the ranges of -180.0 to +180.0 for longitude and -90.0 to +90.0 for latitude.

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 spatial reference system is supported. This means that only the relationship between (row,column) and (X,Y) or (X, Y, Z) 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 spatial extent is described in Section 2.1.2.

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 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.

Each GeoRaster block has the same size. The dimension sizes of the blocks do not need to be a power of 2. They can be random integer values. The block sizes can be optimized automatically based on the dimension sizes of the GeoRaster object, so that each GeoRaster object uses only minimum padding space. See Table 1–1 in Section 1.4.1 for more information.

The raster blocks (BLOBs) contain the binary representation of the raster cell values. Specifically, floating-point cell values are represented in the IEEE 754 standard formats on supported platforms. If the cell depth is greater than 8 bits, GeoRaster cell data is stored in big-endian format in raster blocks. If the cell depth is less than 8 bits, each byte in the raster blocks 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.

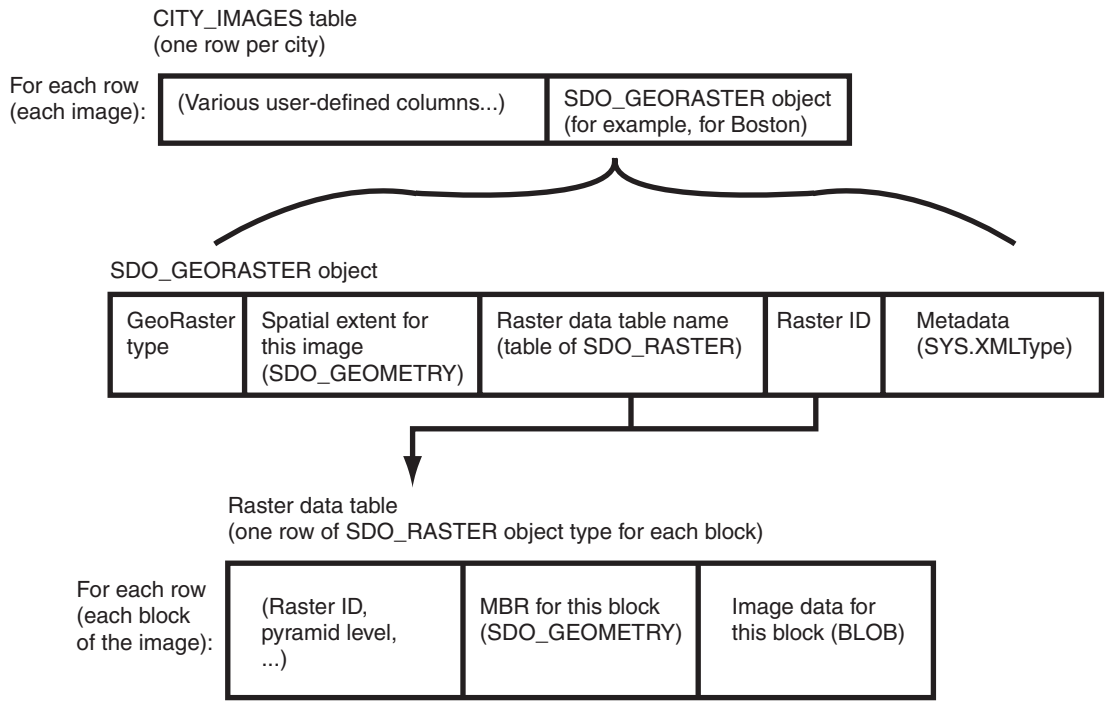
Based on this physical storage model, two object types are provided: SDO_GEORASTER for the raster data set and related metadata, and SDO_RASTER for each block in a raster image.

- 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 1–3 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 1–3 Physical Storage of GeoRaster Data



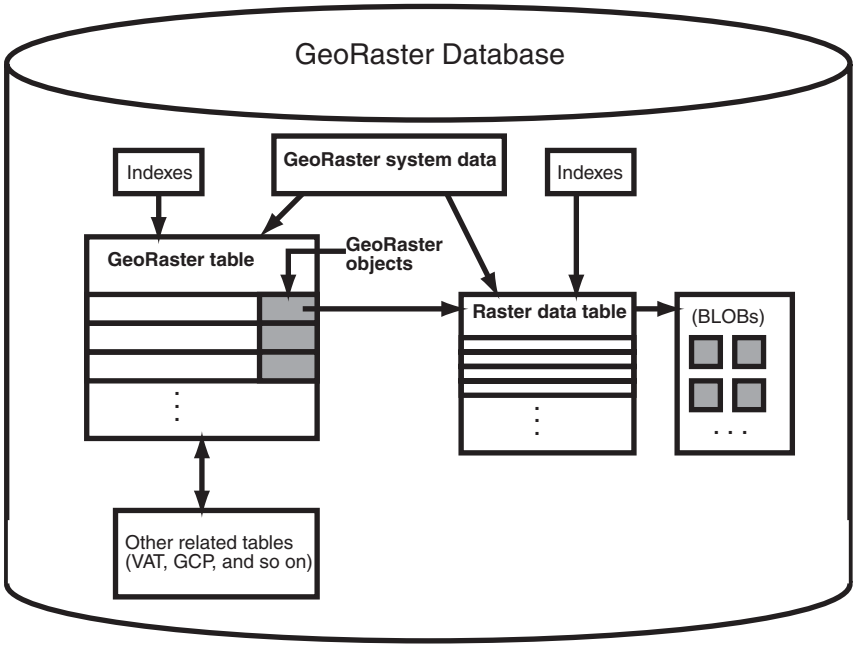
As shown in Figure 1–3:

- 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 1–4 shows the physical storage of GeoRaster data and several related objects in a database.

Figure 1–4 GeoRaster Data in an Oracle Database



In Figure 1–4:

- 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 3.7.
- Additional information, such as ground control points (GCPs) 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 1.4.5.

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
bitmapmask	Specifies whether or not bitmap masks are considered. `TRUE` specifies to consider any associated bitmap masks; `FALSE` specifies not to consider the bitmap masks. The default is `TRUE` for SDO_GEOR.copy, SDO_GEOR.changeFormatCopy, SDO_GEOR.mergeLayers, SDO_GEOR.scaleCopy, and SDO_GEOR.subset; the default is `FALSE` for SDO_GEOR.mosaic (A value of `TRUE` is invalid and is ignored for SDO_GEOR.mosaic.)
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; `OPTIMALPADDING` is the same as TRUE except that the specified `blockSize` value will be adjusted to an optimal value to reduce padding space; `FALSE` causes raster data not to be blocked (that is, only one block will be used for the entire image). Specifying `OPTIMALPADDING` causes GeoRaster to call the SDO_GEOR_UTL.calcOptimizedBlockSize procedure internally. 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.

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

Keyword	Explanation
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=(512,512,3)` specifies 512 for the row dimension, 512 for the column dimension, and 3 for the band dimension; and `blocksize=(256,256)` specifies row and column block sizes of 256 for a GeoRaster object that has no band dimension. The values must be non-negative integers. If a value is 0, it means the block size is the corresponding dimension size. If a value is greater than the corresponding dimension size, padding is applied. See also the explanation of the `blocking` keyword in this table and of the SDO_GEOR_UTL.calcOptimizedBlockSize procedure in Chapter 6. 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. However, the dimension sizes of the blocks do not need to be a power of 2. They can be random integer values. For example, the `blockSize` value can be `(589,1236,7)`. The physical storage size of a raster block must be less than or equal to 4GB.
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`. (Complex `cellDepth` types are not supported.) 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 1.10. Example: `compression=JPEG-F` If the source GeoRaster object is blank, the `compression` keyword is ignored, except for the SDO_GEOR.getRasterSubset and SDO_GEOR.getRasterData functions. (Blank GeoRaster objects are explained in Section 1.4.3.)
interleaving	Specifies the interleaving type. (Interleaving is explained in Section 1.5.) 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 and SDO_GEOR.changeFormatCopy; the default is `FALSE` for SDO_GEOR.scaleCopy, SDO_GEOR.mosaic, and SDO_GEOR.subset. (A value of `TRUE` is invalid and is ignored for 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 1.10.1. 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 PRIMARY KEY,
  name      VARCHAR2(32),
  georaster SDO_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)`.

Each raster data table name must be or equivalent to a valid nonquoted identifier, and it will be stored in the GeoRaster metadata views and in the SDO_GEORASTER objects in all uppercase characters, without any schema prefix. (Each GeoRaster column name must be or equivalent to a valid nonquoted identifier, and it is stored in the GeoRaster metadata views in all uppercase characters.) Each raster data table name must also be unique in the database. To resolve any duplication in raster data table names, you can use SDO_GEOR_ADMIN.maintainSysdataEntries function, which is documented in Chapter 5.

You can control the placement and storage characteristics of the raster data table (for example, if the table should be partitioned for better performance). 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. 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. For raster data of a large size, Oracle recommends at least 512 by 512 for the row and column dimension sizes. 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.

For information about creating raster data tables, including examples using BasicFile LOBs (BasicFiles) and SecureFile LOBs (SecureFiles), see Section 3.1.2.

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 Empty Raster Blocks

GeoRaster supports empty raster blocks to save storage space with large mosaic objects and to improve raster processing speed. Empty raster blocks are used when there is no raster data available for a specific raster block of a large GeoRaster object. Such GeoRaster data is of a special *sparse* data type. There is still an entry in the raster data table for each empty raster block, but the length of the BLOB is zero (indicating empty).

When a GeoRaster operation (for example, SDO_GEOR.changeCellValue, SDO_GEOR.changeFormatCopy, SDO_GEOR.generatePyramid, SDO_GEOR.getRasterData, SDO_GEOR.getRasterSubset, SDO_GEOR.mergeLayers, SDO_GEOR.mosaic, SDO_GEOR.scaleCopy, SDO_GEOR.subset, or SDO_GEOR.updateRaster) is applied to a source GeoRaster object with empty raster blocks, it may lead to empty or partially empty result raster blocks.

A resulting raster block is empty if all the cells in it are derived from empty source raster blocks. A resulting raster block is partially empty if only some of the cells in it are derived from empty source raster blocks. Any cells in a partially empty result raster block that are derived from an empty source raster block are either set to certain background values (as specified in the `bgValues` parameter) or set to 0 (if the `bgValues` parameter is not specified). Once this is done, a partially empty raster block becomes just like a normal non-empty raster block; and after the operation is finished, each raster block in the resulting GeoRaster object is either empty or non-empty.

Because the filling of partially empty raster blocks changes the raster data permanently, you should carefully choose consistent background values when

manipulating a GeoRaster object. The NODATA values stored in the GeoRaster metadata, if present, are good choices for background values, although you can also select other background values as long as they are used consistently.

If a GeoRaster object has empty raster blocks, its pyramid data may not contain any empty raster blocks at all because partially empty raster blocks are filled with background values or 0 during the SDO_GEOR.generatePyramid operation. When you call this function to generate the pyramid, be careful in choosing a consistent background value, as explained in this section.

A bitmap mask (see Section 1.8) can also have empty raster blocks, with the missing cell values indicating 0. If filling is required, the missing cells are always filled with the value 0.

1.4.5 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 Metadata

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`.

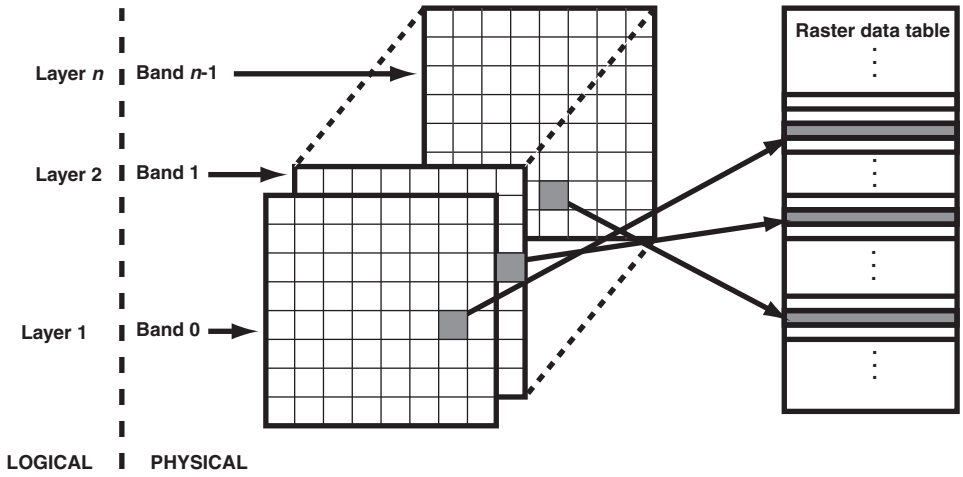
A GeoRaster object can contain multiple bands, which can also be called multiple layers. For example, electromagnetic wave data from remote sensing devices is grouped into a certain number of channels, where the number of possible channels depends on the capabilities of the sensing device. *Multispectral* 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 1–5 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 1–5.

Figure 1–5 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 user-defined layer ID, description, bitmap mask, NODATA values and value ranges, scaling function, bin function, statistical data set (including histogram), grayscale lookup table, and colormap (or, pseudocolor lookup table, also called a PCT). The metadata items are defined in the GeoRaster metadata XML schema, which is presented in Appendix A. the SDO_GEOR_HISTOGRAM object type in Section 2.3.1, the SDO_GEOR_COLORMAP object type in Section 2.3.2, SDO_GEOR_GRAYSCALE object type in Section 2.3.3, and the SDO_GEOR_SRS object type in Section 2.3.5.

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 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 supports both the functional fitting model (explained in Section 1.6.1) and the stored function model (explained in Section 1.6.2) for georeferencing. Rigorous models are not supported. When a GeoRaster object is georeferenced with the functional fitting model, 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 and georeferenced with the functional fitting model, the `isRectified` value in its metadata will be `TRUE`; otherwise, it should be `FALSE`. If a GeoRaster object is orthorectified and georeferenced with the functional fitting model, the `isOrthoRectified` value in its metadata will be `TRUE`; otherwise, it should be `FALSE`.

To georeference a GeoRaster object, see Section 3.5.

1.6.1 Functional Fitting Georeferencing Model

GeoRaster defines a generic functional fitting georeferencing model that is stored in the GeoRaster metadata. It includes several widely used geometric models, and it enables many non-rectified GeoRaster objects to be georeferenced.

This model supports transformations between two-dimensional or three-dimensional ground coordinates and two-dimensional cell coordinates, or between

two-dimensional cell coordinates and two-dimensional or three-dimensional ground coordinates. The following equations describe the model:

$r_n = p(X_n,Y_n,Z_n) / q(X_n,Y_n,Z_n)$

$c_n = r(X_n,Y_n,Z_n) / s(X_n,Y_n,Z_n)$

In these equations:

- r_n = Normalized row index of the cell in the raster
- c_n = Normalized column index of the cell in the raster
- X_n , Y_n , Z_n = Normalized ground coordinate values

The polynomials $p(X_n , Y_n , Z_n)$, $q(X_n , Y_n , Z_n)$, $r(X_n , Y_n , Z_n)$, and $s(X_n , Y_n , Z_n)$ have the form shown in Figure 1–6:

Figure 1–6 Polynomials Used for Georeferencing

$$\sum_{i=0}^{m1}\sum_{j=0}^{m2}\sum_{k=0}^{m3} a_{ijk}\, Xn^{i} yn^{j} Zn^{k}$$

In the polynomial form shown in Figure 1–6, a_{ijk} are the coefficients for the polynomial.

Each of the four polynomials can be different, and each polynomial is described independently by the following:

- `pType` = Polynomial type (1 or 2)
- `nVars` = Total number of variables (ground coordinate dimensions; 0, 2, or 3)
- `order` = Maximum order of power for each variable or maximum total order of power for each polynomial term (up to 5)
- `nCoefficients` = Total number of coefficients (must be derived from the preceding three numbers)

The `pType` indicates the meaning of the maximum total order of the polynomial, and thus affects the total number of terms in the polynomial. `pType = 1` indicates that the maximum order is the maximum total order of all variables in each polynomial term. `pType = 2` indicates that the maximum order is the maximum order of each variable in all polynomial term. The `nVars` indicates whether or not the ground coordinate system is 2D (X, Y) or 3D (X,Y,Z). The cell coordinate systems are always 2D. For example, it supports 2D-to-2D affine transformation and 3D-to-2D DLT and RPC models.

The total number and sequential ordering of the polynomial terms and their coefficients are determined by the logic in the following looping pseudocode:

```
n = 0;
For (k = 0; k <= order; k++)
   For (j = 0; j <= order; j++)
      For (i = 0; i <= order; i++)
        {
              if (pType == 1 & (i+j+k) > order )
                   break;
               polynomialCoefficients[n]=COEF[ijk];
               n++;
         }
```

In the preceding pseudocode, assume `i` is the order of X, `j` is the order of Y and `k` is the order of Z, and `n` is the index of the coefficients inside the GeoRaster metadata element `<polynomialCoefficients>`. Thus, `COEF[ijk]` is the coefficient of the term `x(i)y(j)z(k)` of numerator `p` or denominator `q`; `polynomialCoefficients[n]` is the *n*th double number of the `<polynomialCoefficients>` element (a list type of doubles) inside the XML metadata; and `COEF[ijk]` and `polynomialCoefficients[n]` have a one-to-one match.

Normalized values, rather than actual values, may or may not be stored and used in order to minimize introduction of errors during the calculations, depending on the data itself. The transformation between row and column values (row,column) and normalized row and column values (r_n, c_n), and between the model coordinate (x,y,z) and normalized model coordinate (X_n , Y_n , Z_n), is defined by a set of normalizing translations (offsets) and scales:

- r_n = (row - rowOff) / rowScale
- c_n = (column - columnOff) / columnScale
- X_n = (x - xOff) / xScale
- Y_n = (y - yOff) / yScale
- Z_n = (z - zOff) / zScale

The coefficients, scales, and offsets are stored in the GeoRaster SRS metadata, and are described in Section 2.3.5.

This functional fitting model is generic. It includes specific geometric models, such as Affine Transformation, Quadratic Polynomial, Cubic Polynomial, Direct Linear Transformation (DLT), Quadratic Rational, and Rational Polynomial Coefficients (RPC, also called Rapid Positioning Coefficients). The coefficients of those standard models are converted to the sequential ordering described in this section, for storage in GeoRaster.

You can use the SDO_GEOR.setSRS procedure to directly set the spatial reference information of a GeoRaster object, and the SDO_GEOR.getGeoreferenceType function to find out the specific georeferencing model type in a GeoRaster object.

The simplest georeferencing model type is a special affine transformation, as follows:

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

In the preceding formulas, if `c` is not zero, the raster data is considered rectified, and the `isRectified` value in its metadata will be `TRUE`.

For the Affine Transformation, `pType` can be either 1 or 2. `nVars` is 2, `order` is 1, and `nCoefficients` is 3 for the `p` and `r` polynomials; and `nVars` is 0, `order` is 0, and `nCoefficients` is 1 for the `q` and `s` polynomials.

For the Quadratic Polynomial model, `pType` is 1. `nVars` is 2, `order` is 2, and `nCoefficients` is 6 for the `p` and `r` polynomials; and `nVars` is 0, `order` is 0, and `nCoefficients` is 1 for the `q` and `s` polynomials.

For the Cubic Polynomial model, `pType` is 1. `nVars` is 2, `order` is 3, and `nCoefficients` is 10 for the `p` and `r` polynomials; and `nVars` is 0, `order` is 0, and `nCoefficients` is 1 for the `q` and `s` polynomials.

For the DLT model, `pType` can be either 1 or 2. `nVars` is 3, `order` is 1, and `nCoefficients` is 4 for all polynomials. In addition, the q and s polynomials must be identical.

For the Quadratic Rational model, `pType` is 1. `nVars` is 3, `order` is 2, and `nCoefficients` is 10 for all polynomials.

For the RPC model, `pType` is 1. `nVars` is 3, `order` is 3, and `nCoefficients` is 20 for all polynomials.

For detailed information about the DLT, RPC, and other geometric models, see any relevant third-party documentation.

1.6.2 Georeferencing Using GCPs

GeoRaster supports ground control point (GCP) storage and georeferencing. A **ground control point** (GCP) is a point for which you know its coordinates (X,Y or X,Y,Z) in some reference coordinate system, as well as its corresponding location (row, column) in cell space in the GeoRaster object. The reference coordinate system can be any valid Oracle Spatial coordinate system, including SRID 999999 for an "unknown" coordinate system. A collection of GCPs and its associated geometric model (functional fitting method) are also referred to as (called) the **stored function** georeferencing model in GeoRaster.

You can use GCPs that are either stored in the GeoRaster SRS or specified in parameters to generate the Functional Fitting model. For more information, see the SDO_GEOR.georeference function.

The guidelines for selecting GCPs include the following:

- The points should be easy to identify both in the GeoRaster object and in the reference coordinate system.
- The points should be evenly distributed within the area covered by the GeoRaster object, to ensure that results are not skewed.
- The points should not be on a line, so that the results can be stable.

GCPs or the stored function are specified using the SDO_GEOR_GCP object type (see Section 2.3.6), the SDO_GEOR_GCP_ COLLECTION collection type (see Section 2.3.7), and the SDO_GEOR_GCPGEOREFTYPE object type (see Section 2.3.8).

To georeference using GCPs, you must also select the geometric model, that is, how the relationship between the GeoRaster object's cell space and the reference coordinate system should be mathematically modeled. In GeoRaster, the following geometric models are supported with GCP georeferencing: Affine (the default model), Quadratic Polynomial, Cubic Polynomial, DLT, Quadratic Rational, and RPC. Affine, Quadratic Polynomial, and Cubic Polynomial are two-dimensional polynomial models with polynomial order 1, 2, and 3, respectively; DLT, Quadratic Rational, and RPC are three-dimensional rational polynomial models with polynomial order 1, 2, and 3, respectively. All the polynomials have polynomial type `pType=1`. (See Section 1.6.1 for more information about the georeferencing model types.)

In georeferencing using GCPs, the cell and model coordinates of the GCPs are used in the formula of the polynomial or rational polynomial model, and then a linear equation system is formed. No weight is used in the formula, that is, all points have equal weight 1.0. The linear equation system is solved by the least square method, which generates the coefficients for the model that best fits the given control points. Only GCPs with type Control Point are involved in the solution calculation; the GCP with type Check Point is used to check the positioning accuracy of the solved model.

The solution accuracy is evaluated based on the residuals of the cell coordinates of those control points involved in the solution.

Different geometric models require different model coordinate dimensions and a different minimum number of GCPs. For two-dimensional geometric models, the model coordinates must be 2D (X,Y); and for three-dimensional geometric models, the model coordinates must be 3D (X, Y, Z). The minimum number of GCPs required for the geometric models are as follows: Affine: 3, Quadratic Polynomial: 6, Cubic Polynomial: 10, DLT: 7, Quadratic Rational: 19, and RPC: 39. However, you should generally use more than the minimum number of GCPs to do georeferencing.

1.6.3 Cell Coordinate and Model Coordinate Transformation

Through the functional fitting georeferencing model, GeoRaster assigns ground coordinates to cell coordinates, and cell coordinates to ground coordinates. As a special case, a cell's integer coordinate (the array index of a cell in the cell matrix) can be transformed into a model coordinate, which identifies an exact location of a point in the model space. 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.

Similarly, a model coordinate can be transformed into a cell coordinate through georeferencing. However, the resulting cell coordinate from the direct solution of the functional fitting georeferencing model is mostly in floating numbers. The type of the cell space coordinate system, which is decided by the modelCoordinateLocation element, determines which cell the floating coordinate refers to, as described in Section 1.3. Effective with Oracle Spatial 11*g*, GeoRaster supports both floating (subcell) cell coordinates and integer cell coordinates in all parts of its API.

Cell coordinate and model coordinate transformations are based on the functional fitting model of the GeoRaster spatial reference system (SRS). Both before and after transformation using the GeoRaster SRS, the (row, column) coordinate values of a cell are relative to the GeoRaster cell space, not necessarily relative to the upper-left corner of the raster data itself. 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).

Any application that defines the upper-left corner of a raster data as the origin (0, 0) of its own cell space, as in many image file formats, must convert the (row, column) derived from the GeoRaster SRS to be relative to that origin, if the value of GeoRaster ULTCoordinate (row0, column0) is not (0, 0). This conversion must take the GeoRaster ULTCoordinate into consideration, as shown in the following formulas:

```
row = row0 + m
column = column0 + n
```

In these formulas:

- row = Row index of the cell relative to the origin of the GeoRaster cell space.
- column = Column index of the cell relative to the origin of the GeoRaster cell space.
- row0 = Row index of the ULTCoordinate relative to the origin of the GeoRaster cell space.
- column0 = Column index of the ULTCoordinate relative to the origin of the GeoRaster 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 column0 = 0), in which case m = row and n = column.

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.

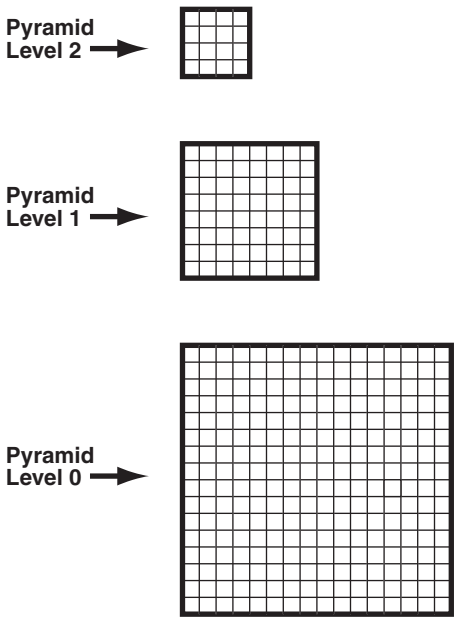
Pyramid levels represent reduced or increased resolution images that require less or more storage space, respectively. (GeoRaster currently supports only reduced resolution pyramids.) A pyramid level of 0 indicates the original raster data; that is, there is no reduction in the image resolution and no 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 1–7 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 1–7 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 addressing of cells in the pyramid uses the same type of cell addressing as that defined for the original raster data, as described in Section 1.3. Each pyramid level has its own cell space; however, all cell spaces of the pyramid levels have the same type of cell coordinate system (either center-based or upper-left based) as that of the original level (level zero). The cells are squares with equal size and the unit is 1 cell. The upper-left corner cell in each pyramid level has the same ULTCoordinate as that of the original raster data, registered in the metadata. Based on this cell space definition and the pyramid levels, the cell coordinates in one pyramid level can be converted to another.

There is no separate SRS defined for each pyramid level in the GeoRaster metadata. The model coordinates of the cells in the pyramid are derived by first converting the cell coordinates of different pyramid level into cell coordinates of pyramid level zero and then applying the GeoRaster SRS. Conversely, the cell coordinates of ground points in the pyramid are derived by first obtaining the cell coordinates of those ground points in pyramid level zero using the GeoRaster SRS, and then converting them into a specific pyramid level. Effective with Oracle Spatial 11*g*, GeoRaster supports subcell addressing of pyramids in all parts of its API.

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 Bitmap Masks

A **bitmap mask** is a special one-bit deep rectangular raster grid with each pixel having either the value of 0 or 1. It is used to define an irregularly shaped region inside another image. The 1-bits define the interior of the region, and the 0-bits define the exterior of the region.

A bitmap mask can be attached to or removed from a nonblank GeoRaster object. Each band or layer of a nonblank GeoRaster object can also have a separate bitmap mask associated with it. Thus, there can be at most *n*+1 bitmap masks associated with a nonblank GeoRaster object, where *n* is the total number of sublayers of the GeoRaster object. A bitmap mask can also be edited or updated independently.

If a bitmap mask is associated with the object layer, it also becomes the default bitmap mask for all sublayers. A bitmap mask associated with a sublayer overrides the default bitmap mask associated with the object layer.

A bitmap mask attached to a raster layer must have the same number of rows and columns as any other raster layers in the image, and must precisely cover the same area. It uses the same ULTCoordinate and SRS as that of the GeoRaster object itself. Logically, it is not an integral part of the raster image itself, but rather an ancillary piece of information; however, physically, it is stored inside the GeoRaster object.

The physical storage of bitmap masks is similar to that of a GeoRaster object's raster data. Bitmap masks are stored in the raster data table of the associated GeoRaster object, with exactly the same blocking attributes. However, the `bandBlockNumber` of a bitmap mask entry is always set to the layer number with which the bitmap mask is associated. For information about the relationship between bands and layers, see Section 1.5.

The `pyramidLevel` value starts with the value -99999 instead of 0, and it increases by 1 for each upper pyramid level. Pyramids are built on bitmap masks along with pyramids on the regular raster data, and bitmap masks can be scaled together with the associated GeoRaster object with the SDO_GEOR.scaleCopy procedure, but the

resampling method used for bitmap masks is always NN (Nearest Neighbor). Bitmap masks are compressed or decompressed when its associated GeoRaster object is compressed or decompressed, and bitmap masks are always compressed with the DEFLATE method (lossless). A bitmap mask can also be sparse and thus can contain empty blocks, with the missing cell values indicating 0.

Bitmap masks are generally used by applications in either or both of the following ways:

- When used as a transparency mask, a bitmap mask can be used by a display application to determine which part of the image to display. For example, main image pixels that correspond to 1-bits in the bitmap mask are imaged to the screen or printer, but main image pixels that correspond to 0-bits in the mask are not displayed or printed. It can also be used as the alpha channel of the image, and so the 0 and 1 values can be mapped to different transparency values for display.
- When used as a NODATA mask in a GIS application, a bitmap mask tells the application to treat pixels that correspond to the exterior (0-bits) of the mask as NODATA. For this purpose, it can be registered as a special type of NODATA in the GeoRaster metadata, as explained in Section 1.9.

Several PL/SQL subprograms perform operations on bitmap masks such as attaching a bitmap mask to a GeoRaster object, replacing an existing bitmap mask, removing a bitmap mask, checking whether a GeoRaster object has a certain bitmap mask, and extracting an entire bitmap mask, a subset of it, or a single cell value of it.

1.9 NODATA Values and Value Ranges

A NODATA value is used for cells whose values are either not known or meaningless. Each individual raster layer can have multiple NODATA values or NODATA value ranges, or both, associated with it. The GeoRaster metadata schema stores the NODATA information with each raster layer. Specifically, the NODATA values and value ranges associated with the object layer apply to any other sublayers. The NODATA values and value ranges for a sublayer is the union of those for the object layer and any NODATA metadata present in the sublayer. When you delete NODATA values or value ranges from a sublayer, any values or value ranges present in the object layer cannot be removed.

NODATA values and value ranges can be considered during resampling, for example, when pyramids are generated or when an image is generated by scaling. NODATA cells are by default treated as regular cells in those processes, to avoid dilations or erosions. However, when NODATA values or value ranges are chosen to be considered and the resampling method is `BILINEAR`, `AVERAGE4`, `AVERAGE16` or `CUBIC`, then whenever a cell value involved in the resampling calculation is a NODATA value, the result of the resampling is also a NODATA value. The resulting NODATA value is the first NODATA value inside each resampling window, where the cell values are ordered row by row from the upper-left corner to the lower-right corner.

If you have GeoRaster objects from before release 11*g* with NODATA metadata stored in the raster description, that metadata is still valid for backward compatibility. The old NODATA value is considered to be object-wide, and it is moved to the object layer when you call the SDO_GEOR.addNODATA procedure on the object layer or when you call the SDO_GEOR.deleteNODATA procedure on the object layer without deleting the old NODATA value.

A NODATA value or value range is described using the SDO_RANGE_ARRAY type, which is defined as `VARRAY(1048576) OF SDO_RANGE`; the SDO_RANGE type specifies a lower and upper bound and is defined as (`LB NUMBER, UB NUMBER`).

- To specify a single number in an SDO_RANGE definition, specify LB as the number and UB as null. The following example specifies 2 as the NODATA value: `SDO_RANGE_ARRAY(SDO_RANGE(2,NULL))`
- SDO_RANGE(*LB, UB*) where *LB=UB* is considered the same as SDO_RANGE(*LB*, NULL).
- A real NODATA value range (where UB is not NULL and LB is less than UB) is inclusive at the lower bound and exclusive at the upper bound.
- You can specify multiple NODATA value ranges and individual NODATA values. The following example specifies one single NODATA value (5) and two NODATA value ranges (1,3) and (7,8): `SDO_RANGE_ARRAY(SDO_RANGE(1,3), SDO_RANGE(5,NULL), SDO_RANGE(7,8))`

Several PL/SQL subprograms perform operations (such as adding, removing, and querying) on NODATA values and value ranges associated with a GeoRaster layer.

In GeoRaster, a bitmap mask can be treated as a special type of NODATA, that is, a NODATA mask specifying one or more irregular areas as NODATA areas. In this case, the bitmap mask is not only identified in the `bitmapMask` element of the `layerInfo` metadata, but is also registered with the `NODATA` element of the `layerInfo` metadata. However, bitmap mask NODATA values are not considered during any resampling processing and statistical analysis.

1.10 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 1.10.1)
- DEFLATE compression of GeoRaster objects (Section 1.10.2)
- Decompression of GeoRaster objects (Section 1.10.3)
- Third-party plug-ins for compression of GeoRaster objects (Section 1.10.4)
- Oracle Advanced Compression (Section 1.10.5)

1.10.1 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. JPEG compression is not supported for GeoRaster objects with a colormap.

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.

1.10.2 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.

1.10.3 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 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.

1.10.4 Third-Party Plug-ins for Compression

GeoRaster provides a plug-in architecture for third-party compression solutions. LizardTech Corporation provides a plug-in that enables users to compress and store raster imagery, in MrSID and JPEG 2000 compression types, natively in Oracle Spatial GeoRaster. For more information, including a link to a guide to using LizardTech wavelet compression with Oracle Spatial GeoRaster, visit the Oracle Spatial page at

http://www.oracle.com/technology/products/spatial/htdocs/spatial_partners_downloads.html

1.10.5 Oracle Advanced Compression

You can use the Oracle Database Advanced Compression Option to achieve lossless compression of GeoRaster tables or raster data tables (RDTs). The compression is transparent to GeoRaster, and thus no application changes are required. However, you should avoid using Advanced Compression on the RDT raster blocks if you are also using any GeoRaster-specific compression types (such as JPEG, DEFLATE, or a third-party plug-in) on these blocks.

For more information, visit the Oracle Advanced Compression page at

http://www.oracle.com/database/advanced-compression.html

1.11 GeoRaster and Database Management

GeoRaster enables you to perform database management tasks. It also performs many management tasks automatically, and enforces several guidelines to facilitate its automatic management operations.

GeoRaster provides several subprograms for users who need to perform specialized management tasks:

- SDO_GEOR_ADMIN.isRDTNameUnique checks for the uniqueness of an RDT name, and SDO_GEOR_UTL.renameRDT renames the RDT in the database to solve conflicts, which might happen during data migration.
- SDO_GEOR_ADMIN.checkSysdataEntries and SDO_GEOR_ADMIN.maintainSysdataEntries check for and fix corrupt SYSDATA entries in the current schema or the database, depending on the privileges associated with the database connection.
- The following subprograms check the status of existing GeoRaster objects and related objects in the current schema or the database, depending on the privileges associated with the database connection: SDO_GEOR_ADMIN.listGeoRasterObjects, SDO_GEOR_ADMIN.listGeoRasterColumns, SDO_GEOR_ADMIN.listGeoRasterTables, SDO_GEOR_ADMIN.listRDT, SDO_GEOR_ADMIN.listRegisteredRDT, and SDO_GEOR_ADMIN.listUnregisteredRDT.
- The following subprograms enable you to register existing GeoRaster objects in the current schema or the database, depending on the privileges associated with the database connection: SDO_GEOR_ADMIN.registerGeoRasterObjects and SDO_GEOR_ADMIN.registerGeoRasterColumns.
- SDO_GEOR_ADMIN.upgradeGeoRaster checks for and corrects errors after a database upgrade.

To ensure the reliability of GeoRaster data and metadata, the following actions are performed and the following guidelines are enforced:

- GeoRaster triggers are maintained by GeoRaster, and they cannot be dropped or altered by SQL statements issued by users directly.
- The name pattern `GRDMLTR_*` is reserved for GeoRaster triggers. Users must not create any triggers whose names start with `GRDMLTR_`.
- The associated GeoRaster metadata entries are updated automatically in all of the following cases: if a GeoRaster table is dropped, truncated, renamed, or altered; if a GeoRaster column is dropped; or if a schema is dropped. (However, if you use

the ALTER TABLE statement to add one or more GeoRaster columns, you must call the SDO_GEOR_UTL.createDMLTrigger procedure to create the DML trigger on each added GeoRaster column.)

- A raster data table (RDT) cannot be dropped or directly renamed using standard SQL statement as long as any GeoRaster object references that RDT. To rename RDT, call the SDO_GEOR_UTL.renameRDT procedure.

1.12 GeoRaster PL/SQL API

GeoRaster provides the SDO_GEOR, SDO_GEOR_UTL, and SDO_GEOR_ADMIN PL/SQL packages, which contain subprograms (functions and procedures) to work with GeoRaster data and metadata. Most of these subprograms fit into one of the following logical categories reflecting the purpose of the subprogram:

- Create, load, and export GeoRaster data
- Georeference and validate GeoRaster objects
- Query and update GeoRaster metadata
- Query and update GeoRaster cell data
- Process GeoRaster objects
- Perform GeoRaster administrative functions

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

Reference chapters provide detailed information about the subprograms in the SDO_GEOR (Chapter 4), SDO_GEOR_ADMIN (Chapter 5), and SDO_GEOR_UTL (Chapter 6) PL/SQL packages. The subprograms are presented in alphabetical order in those chapters. Chapter 3 describes operations that involve the use of many of those subprograms, including the general steps for calling them.

1.13 GeoRaster Java API

The Oracle Spatial GeoRaster Java API consists of interfaces and classes that support features available with the GeoRaster feature of Oracle Spatial. This API provides a complete mapping of the SDO_GEORASTER object type and its metadata to Java objects, and it offers Java methods to manipulate GeoRaster objects.

This API includes the following packages:

- The `oracle.spatial.georaster` package is the core of this API. It provides a complete mapping of the SDO_GEORASTER object type and its metadata to Java objects, and it offers Java methods to manipulate GeoRaster objects. It is in pure Java and does not depend upon JAI.
- The `oracle.spatial.georaster.sql` package provides support for wrapping some of the GeoRaster PL/SQL subprograms that do not have support included in the `oracle.spatial.georaster` package.
- The `oracle.spatial.georaster.image` package provides support for generating Java images from a GeoRaster object or a subset of a GeoRaster object, and for processing the images. This package depends upon and leverages JAI.

For detailed information about these packages, see *Oracle Spatial Java API Reference* (Javadoc).

The Spatial Java class libraries are in .jar files under the <ORACLE_HOME>/md/jlib/ directory. The GeoRaster Java API .jar file is $ORACLE_HOME/md/jlib/georasterapi.jar.

The support for GeoRaster themes in Oracle MapViewer is based on the GeoRaster Java API. You can use the MapViewer Java and XML APIs to access GeoRaster data. MapViewer is documented in *Oracle Fusion Middleware User's Guide for Oracle MapViewer*.

1.14 GeoRaster Tools: Viewer, Loader, Exporter

Note: GeoRaster provides limited external file format support for loading and exporting data and limited support for visualization. Instead, Oracle works closely with third parties to provide comprehensive ETL (extract, transform, load) tools for loading and exporting various raster data formats and to provide visualization clients to display GeoRaster objects.

See the Spatial partner solutions information at http://www.oracle.com/technology/products/spatial/spatial_partners.htm and the open source components for Spatial at http://www.oracle.com/technology/products/spatial/htdocs/spatial_opensource.html; see also the *$ORACLE_HOME*/md/demo/georaster/java/README file, including any limitations.

GeoRaster includes several client-side tools. To use these client-side tools, you must install the demo files from the Oracle Database Examples media (see *Oracle Database Examples Installation Guide*). After the installation, these tools are in the following .jar file (assuming the default Spatial installation directory of *$ORACLE_HOME*/md):

```
$ORACLE_HOME/md/demo/georaster/java/georaster_tools.jar
```

The $ORACLE_HOME/md/demo/georaster/java/README file includes helpful usage information and 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. GeoRaster can load the following image formats: TIFF, GeoTIFF, JPEG, BMP, GIF, PNG, and JP2. Georeferencing information can be loaded from ESRI world files, GeoTIFF files and Digital Globe RPC text files.

You can use the GeoRaster loader 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. The GeoRaster exporter tool supports the following destination image file formats: TIFF, GeoTIFF, JPEG, BMP, GIF, PNG, and JP2. Georeferencing information can be exported to ESRI world files, GeoTIFF files and Digital Globe RPC text files.

  You can use this as an alternative to the SDO_GEOR.exportTo procedure, which is documented in Chapter 4. Note, however, that the GeoRaster exporter tool does not support GIF as a destination file format; the SDO_GEOR.exportTo procedure does not support GIF, JPEG, or JP2 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.

Some restrictions on load and export operations may apply regarding image size and type; see the `README` file for the GeoRaster tools.

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.

To load or export GeoTIFF images with the GeoRaster client-side tools, add the following libraries to your CLASSPATH definition:

- `xtiff-jai.jar` (available from the SourceForge Extensible-TIFF-JAI group)
- `geotiff-jai.jar` (available from the SourceForge GeoTIFF-JAI group)

To load or export JP2 images, add the following library to your CLASSPATH definition: `jai-imageio.jar` (available from the Sun Java Advanced Imaging Image I/O Tools download page).

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.

1.15 GeoRaster PL/SQL and Java Demo Files

GeoRaster includes several PL/SQL and Java demo files that show common operations. If you installed the demo files from the Oracle Database Examples media

(see *Oracle Database Examples Installation Guide*), these files are in the following directories under the Spatial installation directory (which by default is *$ORACLE_HOME*/md):

```
/demo/georaster/plsql
/demo/georaster/java
```

The PL/SQL examples demonstrate basic operations using the GeoRaster PL/SQL API to initialize, import, insert, delete, query, process, update, and export GeoRaster objects.

The Java examples demonstrate how to use the GeoRaster Java API to develop GeoRaster ETL (extract, transform, load) tools and applications.

1.16 README File for Spatial and Related Features

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

$ORACLE_HOME/md/doc/README.txt

For GeoRaster, see also the *$ORACLE_HOME*/md/demo/georaster/java/README file.

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"

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, not necessarily in the GeoRaster model space, and can be directly updated by a SQL UPDATE statement specifying a geometry. However, the spatial extent geometry 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 1.14).

You can call SDO_CS.transform to convert it to any other supported coordinate system. 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 Developer's Guide*.

The GeoRaster spatial extent is generally used to build a Spatial R-tree index on the GeoRaster column. For example, you can use a geodetic SRID for all the spatial extents when all GeoRaster objects are in different local projections, and then build a whole-Earth based spatial index on the GeoRaster table and spatially search GeoRaster objects globally. 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 3.7.

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.

This attribute must be a valid nonquoted identifier without any period separators, and all the alphanumeric characters must be uppercase.

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.

This attribute and the `bandBlockNumber` attribute (described in Section 2.2.3) are also used to indicate bitmap masks and their pyramids. For more information about bitmap masks, bitmap mask pyramids, and how the `pyramidLevel` and `bandBlockNumber` attributes are used, see Section 1.8.

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. For more information about how the `bandBlockNumber` attribute is used with bitmap masks and their pyramids, see Section 1.8.

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. It is also used to store bitmap masks of the GeoRaster object. The `rasterBlock` attribute is of type BLOB.

2.3 Other GeoRaster Types

In addition to SDO_GEORASTER (described in Section 2.1), SDO_RASTER (described in Section 2.2), and SDO_RANGE_ARRAY and SDO_RANGE (described in Section 1.9), 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

In GeoRaster, the histogram is stored in the GeoRaster metadata using the XML schema defined in Appendix A. The SDO_GEOR_HISTOGRAM object type is used in the PL/SQL API to contain the histogram data of a GeoRaster object or a layer. The layers have the same histogram data structure. Each cell has a value, and for each cell value or a value range there may be any number of cells having that value or falling in that range.

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 cell value or cell value range.

The histogram contains the cell values (and the implied value ranges) and the total number of cells related to each cell value or each cell value range. For example, if (cellValue1, count1) and (cellValue2, count2) are the two adjacent entries in ascending order in the histogram, the implied value range is [cellValue1, cellValue2) and the total number of cells in this range is count1. The cell value range is always inclusive in its lower boundary and exclusive in the upper boundary. The size of each range does not necessarily have to be the same. Using this example, the range is equal to or greater than cellValue1 and less than cellValue2. For a lower cell depth (for example, 1-bit to 8-bit integers), the cell value ranges are typically the same as the cell values.

2.3.2 SDO_GEOR_COLORMAP Object Type

In GeoRaster, the color information is stored in the GeoRaster metadata using the XML schema defined in Appendix A. The SDO_GEOR_COLORMAP object type is used in the PL/SQL API to contain 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 or are in a specific value range. 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

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.

The colormap contains the cell values (and the implied value ranges) and the red, green, blue, and/or alpha values related to each cell value or each cell value range. For example, if (cellValue1, red1, green1, blue1, alpha1) and (cellValue2, red2, green2, blue2, alpha2) are the two adjacent entries in ascending order in the colormap, the implied value range is [cellValue1, cellValue2), and the color components associated with all cells in this range are (red1, green1, blue1, alpha1). The cell value range is always inclusive in its lower boundary and exclusive in the upper boundary. The size of each range does not necessarily have to be the same. In this example, the range is equal to or greater than cellValue1 and less than cellValue2. For a lower cell depth (for example, 1-bit to 8-bit integers), the cell value ranges are typically the same as the cell values.

Table 2–2 SDO_GEOR_COLORMAP Object Type Attributes

Attribute	Description
cellValue	Array of cell values. The values must be stored in ascending order.
red	Array of red component values for pseudocolor display of cells that have the values or value ranges 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 or value ranges 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 or value ranges 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 or value ranges in `cellValue`. Must be integer values from 0 to 255.

2.3.3 SDO_GEOR_GRAYSCALE Object Type

In GeoRaster, the grayscale information is stored in the GeoRaster metadata using the XML schema defined in Appendix A. The SDO_GEOR_GRAYSCALE object type is used in the PL/SQL API to contain grayscale information for identifying the grayscale value to be used to display cells that have a specific value or fall into a specific value range. 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.

The grayscale contains the cell values (and the implied value ranges) and the gray values related to each cell value or each cell value range. For example, if (cellValue1,

gray1) and (cellValue2, gray2) are the two adjacent entries in ascending order in the grayscale table, the implied value range is [cellValue1, cellValue2), and the gray color associated with all cells in this range is gray1. The cell value range is always inclusive in its lower boundary and exclusive in the upper boundary. The size of each range does not necessarily have to be the same. Taking the same example, the range is equal to or greater than cellValue1 and less than cellValue2. For a lower cell depth (for example, 1-bit to 8-bit integers), the cell value ranges are typically the same as the cell values.

Table 2–3 SDO_GEOR_GRAYSCALE Object Type Attributes

Attribute	Description
cellValue	Array of cell values. The values must be stored in ascending order.
gray	Array of gray component values for grayscale display of cells that have the values or value ranges 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

In GeoRaster, the spatial reference system (SRS) information is stored in the GeoRaster metadata using the XML schema defined in Appendix A. The SDO_GEOR_SRS object type is used in the PL/SQL API to contain information related to the spatial referencing of a GeoRaster object. The metadata and the object type contain the same information. You can use the object type to retrieve the SRS information from GeoRaster objects or to load and update the SRS information in GeoRaster objects.

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,
xRMS               NUMBER,
yRMS               NUMBER,
zRMS               NUMBER,
modelTotalRMS      NUMBER,
GCPgeoreferenceModel SDO_GEOR_GCPGEOREFTYPE);
```

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 control of the precision.
coordLocation	The model coordinate location defines the type of the cell space, which represents either upperleft-based (that is, coordLocation=1) or center-based (that is, coordLocation=0). For more information about model space and cell (raster) space, see Section 1.3.
rowOff	Row offset value.
columnOff	Column offset value.
xOff	X offset value.
yOff	Y offset value.
zOff	Z offset value.
rowScale	Row scaling factor value.
columnScale	Column scaling factor value.
xScale	X scaling factor value.
yScale	Y scaling factor value.
zScale	Z scaling factor value.
rowRMS	The row-dimension accuracy. It is computed using control points if you call SDO_GEOR.georeference using GCPs.
columnRMS	The column-dimension accuracy. It is computed using control points if you call SDO_GEOR.georeference using GCPs
totalRMS	The total row and column accuracy. It is computed using control points if you call SDO_GEOR.georeference using GCPs

Table 2–4 (Cont.) SDO_GEOR_SRS Object Type Attributes

Attribute	Description
rowNumerator	pType, nVars, order, nCoefficients, and all coefficients of the numerator of the row polynomial, where pType=1 or 2; nVars=0, 2, or 3; 0<=order<=5; and nCoefficients is derived from pType, nVars, and order. The polynomials are explained in Section 1.6.1.
rowDenominator	pType, nVars, order, nCoefficients, and all coefficients of the denominator of the row polynomial, where pType=1 or 2; nVars=0, 2, or 3; 0<=order<=5; and nCoefficients is derived from pType, nVars, and order. The polynomials are explained in Section 1.6.1.
columnNumerator	pType, nVars, order, nCoefficients, and all coefficients of the numerator of the column polynomial, where pType=1 or 2; nVars=0, 2, or 3; 0<=order<=5; and nCoefficients is derived from pType, nVars, and order. The polynomials are explained in Section 1.6.1.
columnDenominator	pType, nVars, order, nCoefficients, and all coefficients of the denominator of the column polynomial, where pType=1 or 2; nVars=0, 2, or 3; 0<=order<=5; and nCoefficients is derived from pType, nVars, and order. The polynomials are explained in Section 1.6.1.
xRMS	The x-dimension accuracy. It is computed using check points if you call SDO_GEOR.georeference using GCPs.
yRMS	The y-dimension accuracy. It is computed using check points if you call SDO_GEOR.georeference using GCPs.
zRMS	The z-dimension accuracy. It is computed using check points if you call SDO_GEOR.georeference using GCPs.
modelTotalRMS	The total model accuracy. It is computed using check points if you call SDO_GEOR.georeference using GCPs.
GCPgeoreferenceModel	The stored function model information, that is, all information about the GCP-based georeferencing model. For information about GCP-based georeferencing model information, see Section 2.3.8, "SDO_GEOR_GCPGEOREFTYPE Object Type".

However, when the direct and inverse solutions are derived from the functional fitting model, the accuracy values listed in Table 2–4 are not considered in GeoRaster internal cell coordinate and model coordinate transformation computations for the current release.

The SDO_GEOR_SRS object type has two constructors:

- One constructor takes no parameters and creates an instance of the type with the isReferenced attribute set to FALSE and the other attributes set to null values. This constructor allows you to set up either the functional fitting model or the stored function (GCP) model, or to set up both of them together.
- The other constructor takes all the attributes of this object type as parameters, except those related to the stored function (GCP) model.

For examples of how to use the SDO_GEOR_SRS constructor, see the reference section for the SDO_GEOR.setSRS procedure in Chapter 4.

2.3.6 SDO_GEOR_GCP Object Type

In GeoRaster, the ground control point (GCP) information is stored in the GeoRaster metadata using the XML schema defined in Appendix A. The SDO_GEOR_GCP object type is used in the PL/SQL API to contain GCP information related to the georeferencing of a GeoRaster object. The metadata and the object type contain the same information. You can use the object type to retrieve the GCP information from GeoRaster objects or to load and update the GCP information in GeoRaster objects.

The SDO_GEOR_GCP object type is defined as:

```
CREATE TYPE sdo_geor_gcp AS OBJECT (
     pointID          VARCHAR2(32),
     description      VARCHAR2(256),
     pointType        NUMBER,
     cellDimension    NUMBER,
     cellCoordinates  SDO_NUMBER_ARRAY,
     modelDimension   NUMBER,
     modelCoordinates SDO_NUMBER_ARRAY,
     accuracy         SDO_NUMBER_ARRAY,
     status           NUMBER
);
```

Table 2–5 describes the attributes of the SDO_GEOR_GCP object type.

Table 2–5 SDO_GEOR_GCP Object Type Attributes

Attribute	Description
pointID	Unique ID of the control point. Must not more 32 characters.
description	Descriptive information about the control point.
pointType	Point type: 1 (control point) or 2 (check point).
cellDimension	Dimensionality (number of dimensions) of the cell coordinates: 2 or 3.
cellCoordinates	Array of cell coordinates for the control points; (row, column) or (row, column, vertical) for each point.
modelDimension	Dimensionality (number of dimensions) of the model coordinates: 2 or 3.
modelCoordinates	Array of model coordinates for the control point, corresponding to the points in cell space; (X,Y) or (X,Y,Z) for each point.
accuracy	Accuracy of the control point, expressed as the values of (xRMS, yRMS) or (xRMS, yRMS, zRMS).
status	Status of the GCP: `Measured`, `Removed`, `Estimated`, `Validated`, or `Invalid`. The value of this column is informational only, and it has no effect on the usage of the GCP by GeoRaster.

The SDO_GEOR_GCP constructor can be used to create an empty instance of this object type. You should then fill in the necessary data before you use this instance.

2.3.7 SDO_GEOR_GCP_ COLLECTION Collection Type

The SDO_GEOR_GCP_ COLLECTION collection type is used to store an array (a collection) of ground control points (GCPs).

The SDO_GEOR_GCP_ COLLECTION collection type is defined as:

```
CREATE TYPE sdo_geor_gcp_collection VARRAY(1048576) OF SDO_GEOR_GCP;
```

The SDO_GEOR_GCP type is described in Section 2.3.6.

2.3.8 SDO_GEOR_GCPGEOREFTYPE Object Type

In GeoRaster, the GCP-based georeferencing model information is stored in the GeoRaster metadata using the XML schema defined in Appendix A. The SDO_GEOR_GCPGEOREFTYPE object includes the georeferencing functional fitting method (that is, the geometric model), control points for solving the model parameters, and solution accuracy. The SDO_GEOR_ GCPGEOREFTYPE object type is used in the PL/SQL API to contain georeferencing model information related to the GCP-based georeferencing of a GeoRaster object. The metadata and the object type contain the same information. You can use the object type to retrieve the georeferencing model information from GeoRaster objects or to load and update the georeferencing model information in GeoRaster objects.

The SDO_GEOR_GCPGEOREFTYPE object type is defined as:

```
CREATE TYPE sdo_geor_gcpgeoreftype AS OBJECT (
     FFMethodType     VARCHAR2(32),
     numberGCP        NUMBER,
     GCPs             SDO_GEOR_GCP_COLLECTION,
     solutionAccuracy SDO_NUMBER_ARRAY
);
```

Table 2–6 describes the attributes of the SDO_GEOR_GCPGEOREFTYPE object type.

Table 2–6 SDO_GEOR_GCPGEOREFTYPE Object Type Attributes

Attribute	Description
FFMethodType	Functional fitting method. Must be one of the following: `Affine`, `QuadraticPolynomial`, `CubicPolynomial`, `DLT`, `QuadraticRational`, or `RPC`.
numberGCP	Number of ground control points in the GCP collection (`GCPs` parameter).
GCPs	The GCP collection, of type SDO_GEOR_GCP_COLLECTION (described in Section 2.3.7).
solutionAccuracy	Array storing the accuracy of the georeferencing solution in the following format: (rowRMS, columnRMS, totalRMS, xRMS, yRMS, zRMS, modelTotalRMS). The first three RMS numbers are computed using control points, and the last four RMS numbers are computed using check points (if any). This information is for output only; do not store or modify values in this attribute.

The SDO_GEOR_GCPGEOREFTYPE object type has one constructor. The constructor takes no parameters, and it creates an instance of the type with the `FFMethodType` attribute set to `Affine` and the other attributes set to null values.

2.4 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)

GeoRaster uses a system data table (also called the sysdata 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 information, such as ground control points (GCPs) 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 sysdata table is under the MDSYS schema. Most of the information in the GeoRaster system data table is available for retrieval through system data views, and thus it can be used as a dictionary or a catalog of all GeoRaster objects in a GeoRaster database. 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 sysdata 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 is automatically created on each SDO_GEORASTER column in each GeoRaster table.

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

GeoRaster defines an XML schema to store and manage the GeoRaster metadata. The definition of this XML schema is included in Appendix A. The namespace defined by the GeoRaster XML schema is `http://xmlns.oracle.com/spatial/georaster`, and it is reserved for use by Oracle. You must refer to this namespace if you want to manipulate a GeoRaster metadata document using the SQL XML functions or the XMLType methods.

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 2–7 describes the columns of the SDO_GEOR_XMLSCHEMA_TABLE table.

Table 2–7 SDO_GEOR_XMLSCHEMA_TABLE Table Columns

Column Name	Data Type	Description
id	NUMBER	ID number, assigned by Oracle. Values `1` through `50` are reserved for use by Oracle.
georasterFormat	VARCHAR2(1024)	GeoRaster format identifier, assigned by Oracle. The value `GEORASTER` is reserved for use 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 for GeoRaster XML schema information.

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` through `50` or a GEORASTERFORMAT column value of `GEORASTER`, because these column values are reserved for use by Oracle. If you specify an XMLSCHEMA column value, you should

choose a unique namespace for your own XML schema and register it using a corresponding schema URL that will also be unique in the database. (For more information, see *Oracle XML DB Developer's Guide*.)

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 and raster data table (see Section 3.1).
2. Initialize or create GeoRaster objects (see Section 3.2).
3. Load raster imagery or grids (see Section 3.3).
4. Validate GeoRaster objects, if they have not already been validated (see Section 3.4).
5. Georeference the GeoRaster objects, if necessary (see Section 3.5).
6. Set the spatial extents of the GeoRaster objects (see Section 3.6).
7. Create spatial indexes or other indexes, or both (see Section 3.7).
8. Change and optimize the GeoRaster storage format, if necessary (see Section 3.8).
9. Copy GeoRaster objects (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. Use template-related subprograms to develop GeoRaster applications (see Section 3.17).
18. Use GeoRaster with Workspace Manager and Label Security (see Section 3.18).
19. Maintain efficient tablespace use by GeoRaster objects (see Section 3.19).
20. Maintain GeoRaster objects and system data in the database (see Section 3.20).
21. Transfer GeoRaster data between databases (see Section 3.21).
22. Use the Oracle Database transportable tablespaces feature with GeoRaster data (see Section 3.22).

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. You must update the GeoRaster object after you delete or edit the metadata or cell data of the GeoRaster object and before you commit the changes (see Section 3.16). For some examples of these operations, see the demo files described in Section 1.15 and the examples in Chapter 4.

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

Subsequent chapters contain detailed reference information about the SDO_GEOR, SDO_GEOR_ADMIN, and SDO_GEOR_UTL packages, which contains subprograms (functions and procedures) to work with GeoRaster data and metadata.

3.1 Creating the GeoRaster Table and Raster Data Tables

Before you can work with GeoRaster objects, you must create a GeoRaster table and one or more raster data tables, if they do not already exist.

3.1.1 Creating a GeoRaster Table

A GeoRaster table is any table that includes a column of type SDO_GEORASTER. This column can be an attribute column of another user-defined object type. 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 PRIMARY KEY, image_description
VARCHAR2(50), image SDO_GEORASTER);
```

For more information about GeoRaster tables, see Section 1.4.

3.1.2 Creating Raster Data Tables

After creating a GeoRaster table, you should create one or more raster data tables (RDTs) to be used with the objects in the GeoRaster table. You can create a raster data table using the original LOB storage paradigm BasicFile Lobs (BasicFiles) or using the LOB storage format SecureFile LOBs (SecureFiles) introduced in Release 11.1. Using SecureFiles significantly improves the performance of GeoRaster operations.

Example 3–2 creates a raster data table named CITY_IMAGES_RDT using BasicFiles. The RDT 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 3–2 Creating a Raster Data Table Using BasicFiles

```
CREATE TABLE city_images_rdt OF SDO_RASTER
  (PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber,
    rowBlockNumber, columnBlockNumber))
  TABLESPACE im_tbs_1
  LOB(rasterBlock) STORE AS lobseg
    (
     CHUNK 32768
     CACHE READS
     PCTVERSION 0
     STORAGE (PCTINCREASE 0)
    );
```

Example 3–3 creates a raster data table with the same name as in Example 3–2, but using SecureFiles.

Example 3–3 Creating a Raster Data Table Using SecureFiles

```
CREATE TABLE city_images_rdt OF SDO_RASTER
  (PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber,
    rowBlockNumber, columnBlockNumber))
  TABLESPACE im_tbs_2
  LOB(rasterBlock) STORE AS SECUREFILE lobseg
    (NOCACHE);
```

The CREATE TABLE statement for a raster data table must include the following clause (which is included in the preceding examples):

```
  (PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber,
    rowBlockNumber, columnBlockNumber))
```

This PRIMARY KEY clause creates a B-tree index on the raster data table, and this index is essential for optimal query performance.

When you use BasicFiles, you can specify a larger CHUNK size (16 or 32 KB) for the LOB storage to improve performance. With SecureFiles, there is no need to specify the CHUNK size parameter, and there are few other storage parameters to consider. Raster data tables using SecureFile LOBs must be created in a tablespace with the automatic segment space management option. For information about using Oracle SecureFiles and performance considerations for BasicFile LOBs, see *Oracle Database SecureFiles and Large Objects Developer's Guide*.

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

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

3.1.3 GeoRaster DML Trigger

To ensure the consistency and integrity of internal GeoRaster tables and data structures, GeoRaster automatically creates a unique DML trigger for each GeoRaster column whenever a user creates a GeoRaster table (that is, a table with at least one GeoRaster column), with the following exception: if you use the ALTER TABLE statement to add one or more GeoRaster columns, you must call the SDO_GEOR_UTL.createDMLTrigger procedure to create the DML trigger on each added GeoRaster column.

The trigger is fired 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.

GeoRaster automatically performs the following actions when the trigger is fired:

- 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 and insert it into a GeoRaster table before you start working on it. To create a new GeoRaster 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 the GeoRaster DML trigger ensures 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, including examples, see Section 3.1.2.

To avoid potential GeoRaster data problems (some of which are described in Section 3.20), you must **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. Any GeoRaster operations that need to manipulate the raster data table raise an exception if the source or target GeoRaster object is not registered.

3.3 Loading Raster Data

To load and export imagery or raster data, always consider third-party ETL tools (see the note in Section 1.14)

If you use features in GeoRaster to load raster 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 1.14.

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 or a Digital Globe RPC text file (.rpb) 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.
- Load a GeoTIFF format file with georeferencing, with or without raster data.

After loading raster data into a GeoRaster object, you must ensure that the object is valid by calling the SDO_GEOR.validateGeoRaster function, as explained in Section 3.4.

Because an ESRI world file or .rpb file does not contain coordinate system information, you can specify the SRID value of a coordinate reference system for the load operation. However, if you do not specify an SRID, the model SRID of the GeoRaster objects is set to 0 (zero) by the loader, which means that the GeoRaster object is invalid, and therefore you must use the SDO_GEOR.setModelSRID procedure to specify a valid model space for this object. If 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 Developer's Guide*.

3.3.1 Reformatting the Source Raster Before Loading

The GeoRaster loader does not support source raster files in BSQ interleaving, and it might raise an "insufficient memory" error if the files are too big, and it might have other restrictions. To avoid such problems, you can reformat and reblock the source files so that they can be properly loaded.

As an example, one way to do this is to use GDAL, an Open Source raster transformation library available from `http://www.gdal.org`, to reformat or reblock the image or raster file so that JAI (Java Advanced Imaging) can handle it. GDAL supports GeoRaster natively and can import and export GeoRaster objects directly, and can also process GeoRaster objects; for more information, see `http://www.oracle.com/technology/products/spatial/htdocs/spatial_opensource.html`. You can also use GDAL to generate TFW files. For example, execute commands such as the following two (each command on a single line) using the GDAL command line or (for batch conversion) shell:

```
gdal_translate -of GTiff -co "TFW=YES" -co "INTERLEAVE=PIXEL" -co "TILED=YES"
D:\my_image.tif D:\my_new_image.tif

gdal_translate -of GTiff -co "TILED=YES" -co "TFW=YES" D:\my_image.ecw D:\my_new_
image.tif
```

In the preceding example, the first command generates a TFW file, changes the interleaving to BIP (which is supported by JAI), and reblocks the image to 256x256. The second command converts ECW to TIFF, generates TFW, and reblocks the image.

Then use the GeoRaster loader tool (described in Section 1.14) , specifying reblocking so that the image can be loaded successfully and later retrieved from the database efficiently, as in the following example (a single command):

```
java -Xmx1024m oracle.spatial.georaster.tools.GeoRasterLoader mymachine db11 6521
georaster georaster thin 32 T globe image "blocking=true, blocksize=(512,512,3)"
"D:my_image.tif,2,RDT_15, D:\my_image.tfw,82213"
```

If you receive an "insufficient memory" error when calling SDO_GEOR.importFrom to load a very large image, try loading the image with a different blocking size parameter or reblock the image into smaller internal tile sizes using GDAL before loading. For extremely large images, you can also use GDAL to tile the image into multiple smaller

image files with sizes that JAI can handle, or you use GDAL to load and export the images directly.

3.4 Validating GeoRaster Objects

Before you use a GeoRaster object or after you manually edit the raster data and metadata of a GeoRaster object, you should ensure that the object is valid. Validation for a GeoRaster object includes checking the registration of the GeoRaster object, checking the metadata and the raster cell data, and making sure that the metadata and data 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 1.14, 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.
- SDO_GEOR.validateBlockMBR validates the `blockMBR` geometry associated with each raster block stored in the raster data table. If there are any invalid `blockMBR` geometries, call the SDO_GEOR.generateBlockMBR procedure to regenerate them.

3.5 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 or `.rpb` file containing RPC coefficients you can use the SDO_GEOR.importFrom procedure to load an ESRI world file or `.rpb` 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 1.14) to load an ESRI world file or `.rpb` file from a file, along with the image file itself.

  Because an ESRI world file or `.rpb` 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 1.14); 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.
- If the original image is a georeferenced GeoTIFF image, you can use the SDO_GEOR.importFrom procedure to load the image with georeferencing, by

specifying GEOTIFF as the input format. To load only the georeferencing information from a GeoTIFF image, without the raster image data, into an existing GeoRaster object, add the raster=false storage parameter. You can specify a backup SRID with the srid storage parameter, in case the GeoTIFF configuration values do not match any SRID recognized by Oracle Spatial.

The GeoTIFF PixelIsArea raster space is equivalent to the GeoRaster upperleft-based cell coordinate system. An export to GeoTiff is always in PixelIsArea raster space, with a half-pixel adjustment of the affine transformation if the GeoRaster object is in center-based cell coordinate system. An import from GeoTIFF is always to the GeoRaster center-based cell coordinate system, with a half-pixel adjustment of the affine transformation if the GeoTIFF file is specified in PixelIsArea raster space.

You can also use the GeoRaster client-side loader tool (described in Section 1.14) to load GeoTIFF images with georeferencing, using the storage parameter geotiff=true. If you omit this parameter or specify geotiff=false, the image is loaded as a simple TIFF image without georeferencing. The raster and srid storage parameters also apply to the client-side loader tool.

To load or export GeoTIFF images with the GeoRaster client-side tools, add the following GeoTIFF libraries to your CLASSPATH definition:

- xtiff-jai.jar (available from the SourceForge Extensible-TIFF-JAI group)
- geotiff-jai.jar (available from the SourceForge GeoTIFF-JAI group)

To load or export GeoTIFF images with the SDO_GEOR.importFrom or SDO_GEOR.exportTo procedure, load these libraries into the MDSYS schema by connecting to the database as the SYSTEM user, editing *$ORACLE_HOME*/md/admin/sdoldgtf.sql as needed to reflect the paths to the xtiff-jai.jar and geotiff-jai.jar files, and running the sdoldgtf.sql SQL*Plus script. As an alternative to using the sdoldgtf.sql script, you can enter the following commands:

```
loadjava -user system/password@database -resolve -force -synonym -schema MDSYS
-grant PUBLIC xtiff-jai.jar

loadjava -user system/password@database -resolve -force -synonym -schema MDSYS
-grant PUBLIC geotiff-jai.jar
```

If the database is downgraded to a release before Oracle Database 11*g*, these libraries should be uninstalled according to the script in *$ORACLE_HOME*/md/admin/sdormgtf.sql, editing it as needed to reflect the paths to the xtiff-jai.jar and geotiff-jai.jar files, and either running the sdormgtf.sql script or entering the following commands:

```
dropjava -user system/password@database -resolve -force -synonym -schema MDSYS
-grant PUBLIC xtiff-jai.jar

dropjava -user system/password@database -resolve -force -synonym -schema MDSYS
-grant PUBLIC geotiff-jai.jar
```

- You can use the SDO_GEOR.setSRS procedure to add, modify, and delete georeferencing information by directly accessing the GeoRaster SRS metadata. 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. You can use the SDO_GEOR.setSRS procedure to set up the spatial reference information for all supported functional fitting georeferencing models.

Examples of setting up the SRS information from an existing DLT model and from an existing RPC model are included in reference section for the SDO_GEOR.setSRS procedure.

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.

- If the GeoRaster object can be georeferenced using an affine transformation, you can call the SDO_GEOR.georeference procedure to georeference a GeoRaster object directly. As described in the reference information for the SDO_GEOR.georeference, this procedure takes the coefficients `A`, `B`, `C`, `D`, `E`, `F` 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.
- If you have ground control points (GCPs) and want to calculate georeferencing information using a geometric model, you can call the SDO_GEOR.georeference procedure to find the solution. To get and set the GCP-based georeferencing model, use the SDO_GEOR.getGCPGeorefModel function and the SDO_GEOR.setGCPGeorefModel procedure. To get and set only the geometric model, use the SDO_GEOR.getGCPGeorefMethod function and the SDO_GEOR.setGCPGeorefMethod procedure. To manipulate GCPs, use the SDO_GEOR.getControlPoint function and the SDO_GEOR.setControlPoint and SDO_GEOR.deleteControlPoint procedures.

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.

3.6 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 3.7), 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 1.14), 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 3.1.1) 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;
```

3.6.1 Special Considerations if the GeoRaster Table Has a Spatial Index

If you create a spatial R-tree index on the GeoRaster spatial extents (as described in Section 3.7), all spatial extent geometries must have the same SRID value. However, the GeoRaster objects may have different model SRIDs, and most GeoRaster operations automatically generate a spatial extent for the output GeoRaster objects based on the model SRID of the source GeoRaster object or objects. This can cause problems when the resulting GeoRaster object with a spatial extent is updated into a GeoRaster table, which might already have a spatial index built on its `spatialExtent` attribute but using a different SRID.

In such cases, you must transform the spatial extent to the same SRID as that of the spatial index before the insert or update operation. The following example performs a mosaic operation, but then transforms the spatial extent of the resulting GeoRaster object to SRID 8307 before updating the GeoRaster table with that object.

```
DECLARE
  gr sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM mosaic_test WHERE georid=1 FOR UPDATE;
  sdo_geor.mosaic('mosaic_data', 'georaster', gr, 'blocksize=(512,512)');
  -- Transform the spatial extent geometry, if ncessary.
  -- In this example example, the modelSRID of the mosaic is 27302,
  -- but the SRID of the spatial index on mosaic_test is 8307.
  gr.spatialExtent := sdo_cs.transform(gr.spatialExtent, 8307);
  UPDATE mosaic_test SET georaster=gr WHERE georid=1;
END;
/
```

If a spatial R-tree index exists, a commit operation after an insert or update operation causes the index to be updated if the inserted or updated GeoRaster object has a spatial extent geometry. This could slow some operations if you perform a commit after each operation, particularly for batch jobs such as batch image loading. It is usually more efficient to balance the performance of index updates with GeoRaster operations, and to commit only in batches after the operations.

For example, image data loading (the SDO_GEOR.importFrom procedure and the GeoRaster loader) is followed by an internal commit operation, so it would be inefficient to load while generating spatial extents by specifying `spatialExtent=TRUE`. Instead, you should probably specify `spatialExtent=FALSE`, and then update the `spatialExtent` attribute afterward, to speed the loading process.

3.7 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); however, see Section 3.6.1 for special considerations if the GeoRaster table has a spatial index. For information about creating spatial indexes, see *Oracle Spatial Developer's Guide*.

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

You should also create a single B-tree index on the `rasterId`, `pyramidLevel`, `bandBlockNumber`, `rowBlockNumber`, and `columnBlockNumber` columns of each raster data table.

3.8 Changing and Optimizing Raster Storage

You can change or specify some aspects of the way raster image data is or will be stored: the raster blocking size, cell depth, interleaving type, and other aspects. To make such changes on an existing GeoRaster object, use the SDO_GEOR.changeFormatCopy procedure, and specify the desired storage parameter values with the `storageParam` parameter. You can specify storage parameters with most subprograms that load and process a GeoRaster object to create another GeoRaster object. That is, you can specify the output format when you call functions or procedures such as SDO_GEOR.importFrom, SDO_GEOR.subset, SDO_GEOR.mosaic,

SDO_GEOR.reproject, SDO_GEOR.scaleCopy, SDO_GEOR.mergeLayers, and SDO_GEOR.createTemplate.

The storageParam parameter for the resulting GeoRaster objects should be based on factors such as the data size, dimension sizes, and application needs, as you determine them. However, the block sizes can also be optimized automatically based on the dimension sizes of the GeoRaster object and the desired output required by users, so that each GeoRaster object uses only minimum padding space but still meets the application requirements.

For more information, see Section 1.4.1, especially Table 1–1, " storageParam Keywords for Raster Data".

3.9 Copying GeoRaster Objects

To copy a GeoRaster object, you must either copy it into an empty GeoRaster object or overwrite an existing valid GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) To make an identical copy of the source GeoRaster object, use the SDO_GEOR.copy procedure; to make a copy that includes storage format changes, use the SDO_GEOR.changeFormatCopy procedure (see Section 3.8).

To copy a GeoRaster object using an empty GeoRaster object, follow these steps:

1. Initialize an empty GeoRaster object while inserting it into the destination table, returning the empty GeoRaster object.
2. Use the SDO_GEOR.copy or SDO_GEOR.changeFormatCopy procedure to copy the GeoRaster object into the returned empty GeoRaster object.
3. Use UPDATE statement to update the desired row in the destination table so that its GeoRaster column contains the copied GeoRaster object.
4. When you are ready to commit the transaction, use the COMMIT statement.

For an example of copying using an empty GeoRaster object, see the example for the SDO_GEOR.copy procedure inChapter 4.

To copy a GeoRaster object so that it overwrites (replaces) an existing GeoRaster object, follow these steps:

1. Select the existing GeoRaster object for update.
2. Use the SDO_GEOR.copy or SDO_GEOR.changeFormatCopy procedure to copy the selected GeoRaster object into either a valid existing GeoRaster object or an empty GeoRaster object.
3. Use UPDATE statement to update the desired row in the destination table so that its GeoRaster column contains the copied GeoRaster object.
4. When you are ready to commit the transaction, use the COMMIT statement.

For an example of copying to replace an existing GeoRaster object and to change its storage format, see the example for the SDO_GEOR.changeFormatCopy procedure inChapter 4.

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).

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.

Note the following recommendations, requirements, and restrictions:

- Most GeoRaster metadata can also be retrieved and modified using XMLType methods or XML-specific SQL functions, such as `extract` and `updateXML`. 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. If you do call XMLType methods or XML-specific SQL functions to update metadata, you should validate the GeoRaster object before you commit the transaction.
- Never directly set the metadata to be null.
- Do not directly update the rasterType attribute of a GeoRaster object; instead, call the SDO_GEOR.setRasterType procedure.
- To change the raster data table name, use the SDO_GEOR_UTL.renameRDT procedure.
- In general, you should not directly update the attributes of a GeoRaster object, except for the `spatialExtent` attribute.

3.11 Querying and Updating GeoRaster 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 cell values of one or multiple layers or bands for a specified location.
- SDO_GEOR.evaluateDouble evaluates a direct location based on neighboring cell values by using a specified interpolation method, and returns the raster values (double precision numbers) for the specified bands or layers for that location.
- SDO_GEOR.getRasterSubset creates a single BLOB object containing all cells of a precise subset of the GeoRaster object (as specified by a rectangular window or a clipping polygon geometry, layer or band numbers, and pyramid level). This BLOB object contains only raster cells and no related metadata.
- SDO_GEOR.reproject not only transforms a whole GeoRaster object from one projected coordinate system to another, but can also include the same capability as SDO_GEOR.getRasterSubset by directly transforming the query result (a single BLOB) into a different coordinate system.

- 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 update or change the value of raster cells in a specified window to a single value, you can use the SDO_GEOR.changeCellValue procedure. You can call the SDO_GEOR.updateRaster procedure to update a specified pyramid of a specified area, or the overlapping parts of one GeoRaster object, with a specified pyramid and specified bands or layers of another GeoRaster object. Both the SDO_GEOR.changeCellValue and the SDO_GEOR.updateRaster procedures support all pyramid levels, including the original raster data (that is, pyramid level 0).

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.

If you want to change the raster data table name, the attributes of a GeoRaster object, or any other metadata, see the recommendations, requirements, and restrictions noted in Section 3.10, "Querying and Updating GeoRaster Metadata".

3.12 Processing GeoRaster Objects

You can perform a variety of processing operations on GeoRaster data, including changing the format, subsetting (cropping), scaling, reprojecting (from one coordinate system to another), generating pyramids, and generating statistics and histograms. Some relevant subprograms are SDO_GEOR.changeFormatCopy, SDO_GEOR.subset, SDO_GEOR.mosaic, SDO_GEOR.reproject, SDO_GEOR.generatePyramid, SDO_GEOR.deletePyramid, SDO_GEOR.scaleCopy, SDO_GEOR.mergeLayers, and SDO_GEOR.generateStatistics.

See also the GeoRaster PL/SQL demo files, described in Section 1.15, 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 1.10, including information about support for third-party compression solutions in Section 1.10.4.

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 1.14).

With the GeoRaster viewer tool, you can select a 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. Some of the basic GeoRaster metadata is also displayed.

The GeoRaster viewer tool provides a set of image processing operators for enhanced display of the GeoRaster objects, especially for those whose cell depth is greater than 8 or is a floating-point number. It can also display and apply bitmap masks on the GeoRaster objects if they have bitmap masks.

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 load and export imagery or raster data, always consider third-party ETL tools (see the note in Section 1.14)

If you use features in GeoRaster 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 1.14.

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 Using Template-Related Subprograms to Develop GeoRaster Applications

The SDO_GEOR.createTemplate and SDO_GEOR.getRasterBlockLocator subprograms enable you to develop GeoRaster applications, such as ETL tools and image processing systems that work with GeoRaster objects, by reading and writing GeoRaster metadata and binary raster data without dealing directly with the Oracle XMLType, the GeoRaster XML schema, and Oracle BLOBs.

After you create a new GeoRaster object (explained in Section 3.2), you can use the SDO_GEOR.createTemplate function to populate the metadata of the GeoRaster object with basic information, such as raster type, dimension sizes, ultCoordinates, cell depth, interleaving type, blocking and block size, pyramid resampling method and reducing level, and compression method and quality. This function can optionally populate the raster data table with the correct number of rows and row data consisting of raster blocks containing empty BLOBs.

The XML metadata generated by the SDO_GEOR.createTemplate function conforms to the GeoRaster metadata schema. You can then use other GeoRaster subprogams to query or update the metadata (see Section 3.10).

You can use the SDO_GEOR.getRasterBlockLocator procedure to get the raster block locator by specifying the pyramid level and block number. If you have the raster block locator, you can then use the OCI or Java JDBC LOB interfaces to read and write the binary raster data. (The SDO_GEOR.getRasterBlockLocator procedure does not itself read or process LOB data.) To use this approach, you must understand the physical storage of the raster data (explained in Section 1.4), and you must compress and decompress the data as necessary before reading from or writing to the BLOB.

3.18 Using GeoRaster with Workspace Manager and Label Security

To use GeoRaster with Oracle Workspace Manager or Oracle Label Security, you must define an object view of SDO_RASTER type and use the object view as the raster storage. The object view must be defined on a single relational table (the base raster data table) that has all the necessary columns and the necessary primary key definition. The following example shows the general form of statements to create the table and the view:

```
CREATE TABLE <rdt_base_table>
(  RASTERID NUMBER,
   PYRAMIDLEVEL NUMBER,
   BANDBLOCKNUMBER NUMBER,
   ROWBLOCKNUMBER NUMBER,
   COLUMNBLOCKNUMBER NUMBER,
   BLOCKMBR SDO_GEOMETRY,
   RASTERBLOCK BLOB,
   PRIMARY KEY (RASTERID,PYRAMIDLEVEL,BANDBLOCKNUMBER,
                ROWBLOCKNUMBER,COLUMNBLOCKNUMBER)
) LOB (RASTERBLOCK) STORE AS (NOCACHE);

CREATE VIEW <rdt_view> OF SDO_RASTER
 WITH OBJECT OID (RASTERID,PYRAMIDLEVEL,BANDBLOCKNUMBER,
                ROWBLOCKNUMBER,COLUMNBLOCKNUMBER)
 AS SELECT RASTERID,PYRAMIDLEVEL,BANDBLOCKNUMBER,ROWBLOCKNUMBER,COLUMNBLOCKNUMBER,
           BLOCKMBR,RASTRBLOCK
      FROM <rdt_base_table>;
```

Use the name of the object view to create GeoRaster objects. For example (general format):

```
INSERT INTO georaster_table (georid, georaster)
  VALUES (0, SDO_GEOR.init(<rdt_view>));
```

The name of the object view is used in the RDT_TABLE_NAME column of the GeoRaster system data views (see Section 2.4). Grant the same privileges on the object view as you would for a raster object table. For example (general format)

```
GRANT SELECT,INSERT,UPDATE,DELETE on <rdt_view> TO SCOTT;
```

3.18.1 Using GeoRaster with Workspace Manager

With Workspace Manager, you can conveniently manage changes to the raster data by saving different raster data versions and making modifications in different workspaces. To use GeoRaster with Workspace Manager, you must use raster data views for raster storage and version-enable the underlying base raster data tables. For example (general format):

```
EXECUTE DBMS_WM.EnableVersioning (<rdt_base_table>, 'VIEW_WO_OVERWRITE');
```

> **Note:** You can version-enable only raster data tables. Do not version-enable any GeoRaster tables, where GeoRaster objects are stored, and do not perform any operations that will require a GeoRaster table to be modified while you are in a workspace.

After you version-enable a base RDT, you can use the subprograms in the DBMS_WM package to manage changes to the raster data. If you need to directly modify a raster block, call the DBMS_WM.copyForUpdate procedure before the operation, as shown in the following example:

```
declare
  geor sdo_georaster;
  cond varchar2(1000);
  lb   blob;
  r1   raw(1024);
  amt  number;
begin
```

```
  r1 := utl_raw.copies(utl_raw.cast_to_raw('0'),1024);

  select georaster into geor from georaster_table where georid=1;
  cond := 'rasterId=' || geor.rasterId || ' AND pyramidLevel=0 AND ' ||
          ' bandBlockNumber=0 AND rowBlockNumber=0 AND columnBlockNumber=0';
  dbms_wm.copyForUpdate(geor.rasterDataTable, cond);
  sdo_geor.getRasterBlockLocator(geor, 0, 0, 0, 0, lb, null, 'TRUE');
  amt := 1024;
  dbms_lob.write(lb, amt, 1, r1);
end;
/
```

However, if you modify raster data using GeoRaster subprograms, you do *not* need to call the DBMS_WM.copyForUpdate procedure beforehand.

For information about Workspace Manager, see *Oracle Database Workspace Manager Developer's Guide*.

3.18.2 Using GeoRaster with Label Security

Oracle Label Security provides row-level access control for sensitive data based on a user's level of security clearance. To use GeoRaster with Label Security, follow these basic steps:

1. Create the GeoRaster table, RDT base table or tables, and an object view of SDO_RASTER type for each RDT.
2. Create an Oracle Label Security policy and define the label components.
3. Create labeling functions for the GeoRaster table and the RDT base table or tables.

   The labels for rows in a GeoRaster table should be generated according to the application's requirements. Use the same label for both the row that stores a GeoRaster object and for the GeoRaster object's raster rows in the associated RDT base table; otherwise, the GeoRaster objects might be invalid or have an inconsistent status.

   The following example creates the labeling function for an RDT base table:

```
CREATE OR REPLACE FUNCTION gen_rdt_label(rdt_view_name varchar2, rid number)
  RETURN LBACSYS.LBAC_LABEL
AS
  tabname varchar2(80);
  schema  varchar2(32);
  grcol   varchar2(1024);
  colname varchar2(30);
  label   NUMBER;
BEGIN
  EXECUTE IMMEDIATE
   'SELECT v.owner, v.table_name, v.column_name grcol, p.column_name ' ||
   '  FROM all_sdo_geor_sysdata v, all_sa_policies p, all_sa_table_policies t '
   || ' WHERE v.rdt_table_name=:1 AND v.raster_id=:2 AND ' ||
   ' v.owner=t.schema_name AND v.table_name=t.table_name AND ' ||
   ' p.policy_name=t.policy_name '
   INTO schema, tabname, grcol, colname
   USING upper(rdt_view_name), rid;
  EXECUTE IMMEDIATE
    'SELECT t.' || colname  ||
     ' FROM ' || schema || '.' || tabname || ' t ' ||
     ' WHERE t.' || grcol || '.rasterdatatable=:1 AND ' ||
     '       t.' || grcol || '.rasterid=:2'
    INTO label
```

```
    USING upper(rdt), rid;
  RETURN LBACSYS.LBAC_LABEL.NEW_LBAC_LABEL(label);
END;
/
```

4. Apply the Label Security policy to a GeoRaster table and its associated RDT base table or tables.

   The following example (general format) applies a Label Security policy to an RDT base table using the labeling function example from the preceding step.

```
BEGIN
  SA_POLICY_ADMIN.REMOVE_TABLE_POLICY(<policy_name>,<schema_name>,<rdt_base_
table>);
  SA_POLICY_ADMIN.APPLY_TABLE_POLICY(
    POLICY_NAME => <policy_name>,
    SCHEMA_NAME => <schema_name>,
    TABLE_NAME  => <rdt_base_table>,
    TABLE_OPTIONS => 'READ_CONTROL,WRITE_CONTROL,CHECK_CONTROL',
    LABEL_FUNCTION => '<schema_name>.gen_rdt_label(<rdt_view>,:new.rasterid)',
    PREDICATE => NULL);
END;
/
```

5. Create and authorize users, and complete other administrative tasks related to Label Security.

You can load GeoRaster data before or after applying the policy to the tables.

The ALL_SDO_GEOR_SYSDATA view (described in Section 2.4) contains system data for all GeoRaster objects accessible by the current user, and accessibility in this case is determined by the user's privileges as defined in the context of discretionary access control (DAC).

After the label for a GeoRaster table row is updated, ensure that the related data labels in the base RDT are updated, so that the labels are synchronized.

For information about Label Security, see *Oracle Label Security Administrator's Guide*.

3.19 Maintaining Efficient Tablespace Use by GeoRaster Objects

After delete or rollback operations, unused space allocated to a raster data table might not be promptly returned to the underlying tablespace. This could result in wasted tablespace area, and it can be a significant issue if the amount of raster data is large. If the raster data table is created using BasicFile LOBs in an automatic segment space management tablespace, you can explicitly shrink the `rasterBlock` LOB segment or the raster data table by altering the raster data table, as shown in Example 3–5 and Example 3–6.

Example 3–5 Shrinking a BasicFile RasterBlock LOB Segment

```
ALTER TABLE city_images_rdt MODIFY LOB (rasterBlock) (SHRINK SPACE);
```

Example 3–6 Shrinking a Raster Data Table

```
ALTER TABLE city_images_rdt ENABLE ROW MOVEMENT;
ALTER TABLE city_images_rdt SHRINK SPACE CASCADE;
```

If you are using SecureFiles, or if you are using BasicFiles allocated in a manual segment space management tablespace, you cannot reclaim unused space using the ALTER TABLE statements as shown in the preceding examples. Instead, you should

create some working (for temporary use) raster data tables and try to put any intermittent results in these RDTs, and then drop these working RDTs after they are no longer needed.

3.20 Maintaining GeoRaster Objects and System Data in the Database

Although GeoRaster provides internal database mechanism to prevent the creation of invalid GeoRaster objects and system data, sometimes such GeoRaster objects and system data might exist in the database, especially after an upgrade from a previous release, or after some user errors in operations on GeoRaster system data. Examples of such invalid objects and system data include the following:

- An entry in the GeoRaster system data views (*xxx*_SDO_GEOR_SYSDATA, described in Section 2.4) refers to a nonexistent GeoRaster table or column.
- Two or more GeoRaster objects have the same pair of RDT name and raster ID values.
- Some GeoRaster objects, tables, columns, or RDTs not registered.
- An RDT name is not unique.
- A GeoRaster object is non-empty or nonblank, but an associated RDT does not exist.

After a database upgrade, you should call the SDO_GEOR_ADMIN.isUpgradeNeeded function to check for any invalid GeoRaster objects and invalid system data for the current version. If there are any errors or invalid data, call the SDO_GEOR_ADMIN.upgradeGeoRaster function to have the problems automatically corrected. If you connect as user MDSYS, the SDO_GEOR_ADMIN.upgradeGeoRaster function upgrades all GeoRaster objects in the database; otherwise, it upgrades only GeoRaster objects in the schema of the current user. (See the reference and usage information about SDO_GEOR_ADMIN.upgradeGeoRaster in Chapter 5.)

For regular maintenance due to possible user errors, several functions and procedures will be helpful in checking for and correcting invalid GeoRaster objects and system data entries:

- To check for errors, call SDO_GEOR_ADMIN.checkSysdataEntries and SDO_GEOR_ADMIN.listUnregisteredRDT.
- To check for dangling raster data, call SDO_GEOR_ADMIN.listDanglingRasterData.
- To correct all invalid system data entries, call SDO_GEOR_ADMIN.maintainSysdataEntries.
- To create correct DML triggers for all GeoRaster columns, call SDO_GEOR_ADMIN.registerGeoRasterColumns.
- To register all existing GeoRaster objects in the sysdata table, call SDO_GEOR_ADMIN.registerGeoRasterObjects.

See the reference and usage information about these procedures and functions in Chapter 5.

3.21 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 do not 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; however, you should use the SDO_GEOR.validateGeoRaster function to check the validity of imported GeoRaster objects before you perform any operations on these 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 3.21.1.
2. Perform the data transfer, as explained in Section 3.21.2.

3.21.1 Checking for and Resolving Conflicts

For a successful import of GeoRaster data into a target database, 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.

  For example, you might plan to import a GeoRaster object by creating its raster data table (RDT) in the target schema, but an existing RDT in the target schema might already have the same name. In this case, you should use the SDO_GEOR_ADMIN.listRDT or SDO_GEOR_ADMIN.isRDTNameUnique function to check both source database and target database to see if there are RDT name conflicts; and if there are any conflicts, use the SDO_GEOR_UTL.renameRDT procedure to rename the RDT to a different name in the target database to solve the conflicts before you import the GeoRaster objects.

- 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.

  For example, if you import RDT data by appending to an existing RDT in the target database, this conflict might occur. In this case, before importing the data into the target database, use the SDO_GEOR_ADMIN.listGeoRasterObjects function to list all GeoRaster objects defined in the target schema, and make sure that there are no conflicts in the combination of RDT name and raster ID between existing GeoRaster data and the GeoRaster data to be imported. If there are any conflicts, change the raster ID of the GeoRaster object in the target schema to resolve the conflicts; otherwise, those GeoRaster objects with conflicts in the dump file will get rejected when you perform import process.

If you need to check the raster data table (RDT) name and raster ID (RID) information in the dump file, you have the following options: check the information in the source database; request the information from the provider of the dump file; load the dump file into a separate test database and check the information there; or (if you cannot use a separate database for testing) load the dump file into a test schema in the current database and check the information. To load the dump file into a test schema in the current database and check the information, follow these steps:

1. Create a test schema in the target database.
2. Load all GeoRaster tables into this test schema from the dump file, using the Data Pump Import utility with the `CONTENT = METADATA_ONLY` parameter.
3. Connect to the database as the MDSYS user, and disable all DML triggers on the GeoRaster tables that were loaded in the preceding step.

4. Load the data into the GeoRaster tables, using the Data Pump Import utility with the `CONTENT = DATA_ONLY` parameter.
5. Retrieve the RDT/RID (raster data table name and raster ID) pairs directly from the GeoRaster tables in the test schema.

After you resolve conflicts, you should ensure the integrity of GeoRaster metadata and data (see Section 3.20). You should also validate any fixed GeoRaster objects before performing a commit or any other operation.

For general information about resolving conflicts during import operations, see the description of the `TABLE_EXISTS_ACTION` parameter in the Data Pump Import chapter of *Oracle Database Utilities*.

3.21.2 Performing the GeoRaster Data Transfer

When you export GeoRaster data from one database and import it into another, the GeoRaster database management system ensures that the necessary DML triggers and system data entries are automatically generated after the GeoRaster tables and objects are imported into the target database.

To export GeoRaster data, do as you would for other types of data. For example:

```
expdp scott schemas=scott directory=dump_dir dumpfile=exp.dmp
Enter password: password
```

To import GeoRaster data, do as you would for other types of data, but exclude the GeoRaster internal DML triggers. For example:

1. Ensure that no conflicts exist between the GeoRaster data to be imported and the existing GeoRaster data in the target database, as explained in Section 3.21.1.

   If any conflicts are not resolved, some exceptions will be raised and only non-conflicted GeoRaster data will be imported into the target database.

2. Import GeoRaster data as you would for other types of data, but exclude the GeoRaster internal DML triggers (whose names start with `GRDMLTR_`). For example:

   ```
   impdp scott schemas=scott directory=dump_dir dumpfile=exp.dmp
   parfile=exclude.par
   Enter password: password
   ```

   where the `exclude.par` file contains the following:

   ```
   exclude=trigger:"like 'GRDMLTR_%'"
   ```

3.22 Using Transportable Tablespaces with GeoRaster Data

You can use the Oracle Database transportable tablespaces feature with GeoRaster data.

If a tablespace to be transported contains any spatial indexes on the GeoRaster tables or raster data tables (RDTs), you may have to take some preparatory steps. See the Usage Notes for the SDO_UTIL.PREPARE_FOR_TTS and SDO_UTIL.INITIALIZE_INDEXES_FOR_TTS procedures in *Oracle Spatial Developer's Guide* for more information about using the transportable tablespace feature with spatial data.

For a successful import of GeoRaster data into a target database, there must be no conflicts in the target schema's GeoRaster system data. Before you transport the tablespace to another database or schema, it is recommended (but not required) that you check for and resolve such conflicts by following the procedure described in

Section 3.21.1. For this reason, you should design GeoRaster tables and RDT tables so as to avoid such foreseeable conflicts before you use such transportable tablespaces in the source database.

Regardless or whether a transported tablespace has any spatial indexes, after transporting the tablespace that contains GeoRaster objects, do the following:

1. Call the SDO_GEOR_ADMIN.registerGeoRasterObjects procedure (described in Chapter 5) to register all GeoRaster objects in the current schema or new database.
2. Before you use the transported GeoRaster data, perform the "regular maintenance" operations described in Section 3.20, to maintain GeoRaster objects and system data and to ensure all GeoRaster objects are correctly transported and properly registered.
3. If you find any conflicts, call the SDO_GEOR_UTL.renameRDT or SDO_GEOR_UTL.makeRDTNamesUnique procedure to solve such conflicts, and validate again.

For detailed information about transportable tablespaces and transporting tablespaces to other databases, see *Oracle Database Administrator's Guide*.

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 1.12. 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.addNODATA

Format

```
SDO_GEOR.addNODATA(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   nodata         IN NUMBER);
```

or

```
SDO_GEOR.addNODATA(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   nodata         IN SDO_RANGE_ARRAY);
```

Description

Adds one or more NODATA values or value ranges, to represent NODATA cells in one layer or all layers in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Layer number in the GeoRaster object. A value of 0 (zero) indicates the object layer.

nodata
Either a single numeric value, or an array of numbers or number ranges. Any NODATA value range is inclusive at the lower bound and exclusive at the upper bound.

The SDO_RANGE_ARRAY type is described in Section 1.9

Usage Notes

Some cells of a GeoRaster object may have no meaningful value assigned or collected. Such cells contain a NODATA value are thus called NODATA cells, which means that those cells are not semantically defined. The application is responsible for defining the meaning or significance of cells identified as NODATA cells. For more information about NODATA values and value ranges, see Section 1.9.

Any NODATA values or value ranges associated with the object layer apply to all sublayers. For an explanation of layers, the object layer, and sublayers, see Section 1.5.

NODATA values must be in the valid cell value range. Both the lower bound and the upper bound of a NODATA value range must be valid cell values as specified by the cell depth. Because NODATA value ranges are exclusive at the upper bound, if you want to specify the maximum valid cell value as NODATA, you must specify the maximum valid cell value as a single numeric NODATA value.

This procedure associates NODATA values or value ranges with a raster layer incrementally. It removes duplicate values or value ranges and combines adjacent

values or value ranges to form a compact representation in the metadata whenever feasible. However, a single numeric NODATA value that is equal to the upper bound of a NODATA value range will not be combined together with the value range because it is not always feasible to calculate the new exclusive upper bound.

To delete one or more NODATA values or value ranges, use the SDO_GEOR.deleteNODATA procedure. To return the NODATA values for a GeoRaster object, use the SDO_GEOR.getNODATA function.

Examples

The following example specifies that cells with values that are greater than or equal to 5 and less than 7, or that are equal to 9, are to be considered NODATA cells for the object layer (and thus all sublayers) of a specified GeoRaster object.

```
DECLARE
  gr sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=1 FOR UPDATE;
  SDO_GEOR.addNODATA(gr, 0, sdo_range_array(sdo_range(5,7), sdo_range(9,null)));
  UPDATE georaster_table SET georaster=gr WHERE georid=1;
  COMMIT;
END;
/
```

SDO_GEOR.addSourceInfo

Format

```
SDO_GEOR.addSourceInfo(
   georaster   IN OUT SDO_GEORASTER,
   sourceInfo  IN VARCHAR2);
```

Description

Adds to the source information for a GeoRaster object.

Parameters

georaster
GeoRaster object.

sourceInfo
String with source information. Cannot exceed 4096 characters.

Usage Notes

The specified `sourceInfo` string is added to the `<sourceInfo>` element in the metadata for the GeoRaster object (described in Appendix A). You can call this procedure as many times as needed to put multiple string values in the `<sourceInfo>` element or to add string values to any existing values.

If you want to replace any existing source information value or values, use the SDO_GEOR.setSourceInfo procedure.

Examples

The following example sets and adds some source information for a specified GeoRaster object, and then retrieves the information.

```
declare
 gr sdo_georaster;
begin
 select georaster into gr from georaster_table where georid=1 for update;
 sdo_geor.setSourceInfo(gr, 'Copyright (c) 2002, 2007, Oracle Corporation.');
 sdo_geor.addSourceInfo(gr, 'All rights reserved.');
 update georaster_table set georaster=gr where georid=1;
end;
/

select * from table(select sdo_geor.getSourceInfo(georaster) from georaster_table
where id=1);

COLUMN_VALUE
--------------------------------------------------------------------------------
Copyright (c) 2002, 2007, Oracle Corporation.
All rights reserved.
```

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 1.10.

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,
   bgValues        IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

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

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.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

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 the `window` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial Developer's Guide*.

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.changeFormatCopy

Format

```
SDO_GEOR.changeFormatCopy(
   inGeoRaster     IN SDO_GEORASTER,
   storageParam  IN VARCHAR2,
   outGeoRaster    IN OUT SDO_GEORASTER,
   bgValues        IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.changeFormatCopy(
   inGeoRaster     IN SDO_GEORASTER,
   pyramidLevel    IN NUMBER,
   storageParam  IN VARCHAR2,
   outGeoRaster    IN OUT SDO_GEORASTER,
   bgValues        IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

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.

pyramidLevel
A number specifying the pyramid level of the source GeoRaster object.

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

outGeoRaster
The SDO_GEORASTER object to hold the copy. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as `inGeoRaster`.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

Usage Notes

This procedure creates a new GeoRaster object that has the specified changes, based on the original GeoRaster object or a specified pyramid level of it. After you use this procedure, you can check to ensure that the desired changes were made in the copy, and then discard the original GeoRaster object if you wish.

If you use the format that does not include the `pyramidLevel` parameter, the copy is based on the original GeoRaster object (pyramidLevel=0).

If the copy is to be made from a pyramid of the original GeoRaster object (`pyramidLevel` > 0), and if the original GeoRaster object is georeferenced, georeferencing information is generated for the resulting GeoRaster object only when the georeference is a valid polynomial transformation. The resulting object's row and column ultCoordinates are set to (0,0).

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.

`inGeoRaster` and `outGeoRaster` must be different GeoRaster objects.

If pyramid data exists for `inGeoRaster`, any upper level 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. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as `inGeoRaster`.

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.

`inGeoRaster` and `outGeoRaster` must be different GeoRaster objects.

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; the object must then be registered in the xxx_SDO_GEOR_SYSDATA views (see the Usage Notes)

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. Must not contain spaces, period separators, or mixed-case letters in a quoted string; the name is always converted to uppercase when stored in an SDO_GEORASTER object. The RDT should be in the same schema as its associated GeoRaster table. 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

After creating the blank GeoRaster object and before performing any operations on the object, you must register it in the xxx_SDO_GEOR_SYSDATA views by inserting the empty GeoRaster object into a GeoRaster table. (The xxx_SDO_GEOR_SYSDATA views are described in Section 2.4. GeoRaster operations are described in Chapter 3.)

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.createTemplate

Format

```
SDO_GEOR.createTemplate(
   georaster      IN OUT SDO_GEORASTER,
   rasterType     IN INTEGER,
   rasterSpec     IN VARCHAR2,
   maskLayers     IN VARCHAR2 DEFAULT NULL,
   initRDTEntry   IN VARCHAR2 DEFAULT NULL
   ) ;
```

Description

Populates a GeoRaster object with metadata of a general pattern, and optionally inserts entries with empty raster blocks into its raster data table.

Parameters

georaster
The GeoRaster object to be updated.

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

rasterSpec
A string with raster specification parameters, as explained in the Usage Notes.

maskLayers
A string identifying the logical layer numbers on which there are associated bitmap masks. Use commas to delimit the values, and a hyphen to indicate a range (for example, `2-4` for layers 2, 3, and 4).

initRDTEntry
The string `TRUE` causes the raster data table to be populated; the string `FALSE` causes the raster data table not to be populated. If you do not specify this parameter, the raster data table is not populated.

Usage Notes

This function populates a GeoRaster object with metadata of a general pattern and optionally inserts proper rows (with empty raster blocks) into its raster data table. If the raster data table is to be populated, the raster data table must exist and the GeoRaster object must have been registered in the GeoRaster SYSDATA table.

In general, only use this procedure with an empty GeoRaster object to populate its XML metadata and raster blocks. If you use an existing (good) GeoRaster object, the GeoRaster object will be replaced with the new template object upon update.

The `rasterSpec` parameter must be a quoted string that contains one or more of the following keyword-value pairs:

- `dimSize` (for example, `dimSize=(256,256,3)`): Specifies the row, column, and band dimension sizes. This keyword must be specified and must be consistent with the `rasterType` parameter.
- `ultCoord` (for example, `ultCoord=(0,0,0)`): Specifies the upper-left coordinate integer values for the GeoRaster object. The default value is 0 for all the dimensions. The value for the band dimension must be 0.
- `cellDepth` (for example, `cellDepth=8BIT_S`): Specifies the cell depth of the GeoRaster object. Must be one of the following values (with `_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`. The default value is `8BIT_U`.
- `interleaving` (for example, `interleaving=BIP`): Specifies the interleaving type. Must be one of the following values: `BSQ`, `BIL`, or `BIP`. The default value is `BSQ`.
- `blocking` (for example, `blocking=TRUE`): Specified whether the GeoRaster object is blocked. Must be either `TRUE` or `FALSE`. The default value is `FALSE` if `blocksize` is not specified.
- `blocksize` (for example, `blocksize=(128,128,3)`): Specifies the block size for each dimension of the GeoRaster object. The values must be non-negative integers. A zero block size value means the corresponding dimension size is the real value. Is `blocking` is set to `TRUE` but `blocksize` is not specified, the GeoRaster object's blocksize is (256,256,*B*), where *B* is the band dimension size.
- `rLevel` (for example, `rLevel=2`): Specifies the maximum pyramid reduction level. Must be a positive integer. If you specify this keyword, the pyramid type is set to `DECREASE` in the metadata; otherwise the pyramid type is set to `NONE`.
- `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). This keyword is ignored if `rLevel` is not set.
- `compression` (for example, `compression=JPEG-B`): Specifies the compression type of the GeoRaster object. Must be one of the following: `NONE`, `DEFLATE`, `JPEG-B`, or `JPEG-F`. The default value is `NONE`.
- `quality` (for example, `quality=75`): 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). This keyword is ignored when the compression keyword is not specified or is not set to one of the JPEG types.

For more information about using this function in developing GeoRaster applications, see Section 3.17.

Examples

The following example populates a GeoRaster object with metadata and initial raster data table rows.

```
DECLARE
  gr sdo_georaster;
BEGIN
```

```
  INSERT INTO georaster_table (georid, georaster)
    VALUES (1, sdo_geor.init('RDT_1'))
    RETURNING georaster into gr;
  sdo_geor.createTemplate(gr, 21001,
                       'dimSize=(128,128,3) blocking=false rlevel=2',
                       null, 'TRUE');
  UPDATE georaster_table set georaster=gr where georid=1;
  COMMIT;
END;
/
```

SDO_GEOR.deleteControlPoint

Format

```
SDO_GEOR.deleteControlPoint (
   inGeoraster      IN SDO_GEORASTER,
   controlPointID  IN VARCHAR2);
```

Description

Deletes a ground control point (GCP) that has the specified control point ID value.

Parameters

inGeoraster
GeoRaster object.

controlPointID
Control point ID for `inGeoraster`. Must be a string not more than 32 characters.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

If the controlPointID is null, empty or not found in the existing GCPs stored in the GeoRaster object metadata, an exception is raised. If a GCP with the specified point ID is found, that GCP is deleted from the georeferencing model.

Examples

The following example deletes the GCP that has the ID value 23 in a specified GeoRaster object.

```
DECLARE
   gr1  sdo_georaster;
BEGIN
   SELECT georaster INTO gr1 from herman.georaster_table WHERE georid=10 FOR
UPDATE;
   sdo_geor.deleteControlPoint(gr1, '23');
   UPDATE georaster_table SET georaster=gr1 WHERE georid=10;
   COMMIT;
END;
/
```

SDO_GEOR.deleteNODATA

Format

```
SDO_GEOR.deleteNODATA(
   georaster      IN OUT SDO_GEORASTER
   layerNumber  IN NUMBER
   nodata         IN NUMBER);
```

or

```
SDO_GEOR.deleteNODATA(
   georaster      IN OUT SDO_GEORASTER
   layerNumber  IN NUMBER
   nodata         IN SDO_RANGE_ARRAY);
```

Description

Deletes one or more NODATA values or value ranges.

Parameters

georaster
GeoRaster object.

layerNumber
Layer number in the GeoRaster object. A value of 0 (zero) indicates the object layer.

nodata
Either a single numeric value, or an array of numbers or number ranges. Any NODATA value range is inclusive at the lower bound and exclusive at the upper bound.

The SDO_RANGE_ARRAY type is described in Section 1.9

Usage Notes

When a NODATA value or value range is deleted, the cell depth of the GeoRaster object is taken into consideration to generate the correct new ranges. If the cell depth specifies floating cell values, you can only remove existing single numeric NODATA values or remove a sub-range from an existing NODATA value range.

For information about NODATA values and value ranges, see Section 1.9.

To add one or more NODATA values or value ranges, use the SDO_GEOR.addNODATA procedure. To return the NODATA values for a GeoRaster object, use the SDO_GEOR.getNODATA function.

Examples

The following example removes cell value 9 from the NODATA metadata associated with the object layer.

```
DECLARE
  gr sdo_georaster;
BEGIN
```

```
  SELECT georaster INTO gr FROM georaster_table WHERE georid=0 FOR UPDATE;
  SDO_GEOR.deleteNODATA(gr, 0, 9);
  UPDATE georaster_table SET georaster=gr WHERE georid=0;
  COMMIT;
END;
/
```

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.evaluateDouble

Format

```
SDO_GEOR.evaluateDouble(
   georaster           IN SDO_GEORASTER,
   pyramidLevel        IN NUMBER,
   row                 IN NUMBER,
   collumn             IN NUMBER,
   bands               IN VARCHAR2,
   interpolationMethod IN VARCHAR2
   ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.evaluateDouble(
   georaster           IN SDO_GEORASTER,
   pyramidLevel        IN NUMBER,
   ptGeom              IN SDO_GEOMETRY,
   layers              IN VARCHAR2,
   interpolationMethod IN VARCHAR2
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Evaluates a direct location using a specified interpolation method, and returns the raster values (double precision numbers) for the specified bands or layers for that location.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level containing the location whose raster values are to be returned.

row
The row coordinate of the location whose raster values are to be returned. This can be a floating point number.

column
The column coordinate of the location whose raster values are to be returned. This can be a floating point number.

bands
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).

ptGeom
Point geometry that identifies the direct location whose raster values are to be returned.

layers
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). (As mentioned in Section 1.5, the logical layer number is the physical band number plus 1.)

interpolationMethod
A quoted string containing one or more keywords, each with an appropriate value. See the Usage Notes for information about the available keywords and values.

Usage Notes

This function returns interpolated raster values in double precision. In GeoRaster, the original cell values are always associated with the center of the cells, regardless of whether the cell coordinate system type is center-based or upperleft-based.

Identify the location in the GeoRaster object either by specifying its row, column, and band numbers in cell coordinate space, or by specifying a point geometry in either model coordinate space or cell coordinate space.

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

- `interpolationMethod` (for example, `interpolationMethod=NN`): Specifies the interpolation 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).
- `nodata` (for example, `nodata=TRUE`): Specifies whether NODATA values and value ranges should be considered during the procedure. Must be either `TRUE` (NODATA values and value ranges should be considered) or `FALSE` (NODATA values and value ranges should not be considered). The default value is `FALSE`. If the value is `TRUE` and the interpolation method is `BILINEAR`, `AVERAGE4`, `AVERAGE16`, or `CUBIC`, whenever a cell value involved in the interpolation calculation is a NODATA value, the result of the interpolation is also a NODATA value. The resulting NODATA value is the minimum NODATA value associated with the current raster layer, if multiple NODATA values or value ranges exist.

If `interpolationMethod` is specified as `'interpolationMethod=NN'`, this function is equivalent to calling the SDO_GEOR.getCellValue function.

Examples

The following examples return the raster values for a specified location in 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.

The examples show the two function formats, and they return the same values for the same location specified in either cell space or model space.

```
SELECT SDO_GEOR.evaluateDouble(a.georaster, 0,
    10.2, 10.3,
```

```
    '0-2',
    'interpolationMethod=BILINEAR')
  FROM georaster_table a WHERE georid=21;

SDO_
GEOR.EVALUATEDOUBLE(A.GEORASTER,0,10.2,10.3,'0-2','interpolationMethod=BILINEAR')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(86.68, 135.68, 31.72)

1 row selected.

SELECT SDO_GEOR.evaluateDouble(a.georaster, 0,
    SDO_GEOMETRY(2001, 82394, SDO_POINT_TYPE(18492.775, 1012881.9, NULL),
                 NULL, NULL),
    '1-3',
    'interpolationMethod=BILINEAR')
  FROM georaster_table a WHERE georid=21;

SDO_GEOR.EVALUATEDOUBLE(A.GEORASTER,0,SDO_GEOR.GETMODELCOORDINATE(A.GEORASTER,0,
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(86.68, 135.68, 31.72)

1 row selected.
```

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  OUT NOCOPY BLOB);
```

or

```
SDO_GEOR.exportTo(
    georaster       IN SDO_GEORASTER,
    subsetParam   IN VARCHAR2,
    r_destFormat   IN VARCHAR2,
    r_destBLOB     IN  OUT NOCOPY BLOB,
    h_destFormat  IN VARCHAR2 DEFAULT NULL,
    h_destCLOB    IN  OUT NOCOPY 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 = (startRow, startCol, endRow, endCol). It identifies the upper-left (startRow, startCol) and lower-right (endRow, endCol) coordinates of a rectangular window to be exported, and raster space is assumed. 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, GeoTIFF, 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 1.14, 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.

The GeoTIFF PixelIsArea raster space is equivalent to the GeoRaster upperleft-based cell coordinate system. An export to GeoTiff is always in PixelIsArea raster space, with a half-pixel adjustment of the affine transformation if the GeoRaster object is in center-based cell coordinate system.

To load GeoTIFF images with the SDO_GEOR.importFrom procedure, you will need the `xtiff-jai.jar` and `geotiff-jai.jar` libraries. For more information about these GeoTIFF libraries, see Section 3.5.

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',
   '/mydirectory/myimages/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',
  '/mydirectory/myimages/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.generateBlockMBR

Format

```
SDO_GEOR.generateBlockMBR(
   georaster  IN SDO_GEORASTER);
```

Description

Computes the minimum bounding rectangle (MBR) for each block in a GeoRaster object, and sets the `blockMBR` attribute for each raster block in the raster data table.

Parameters

georaster
GeoRaster object.

Usage Notes

This procedure does not change the GeoRaster object. It sets the value of the `blockMBR` attribute (described in Section 2.2.6) in each row of the raster data table associated with the GeoRaster object.

If you created the GeoRaster object as described in Section 3.2, the `blockMBR` attribute values were automatically calculated and they should not need to be validated or generated. However, if the GeoRaster object was generated by a third party, you should validate the `blockMBR` attribute values using the SDO_GEOR.validateBlockMBR function; and if any are not valid, call the SDO_GEOR.generateBlockMBR procedure.

Examples

The following example computes the MBR for a specified GeoRaster object and sets its `blockMBR` attribute.

```
DECLARE
  gr sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=1 FOR UPDATE;
  sdo_geor.generateBlockMBR(gr);
  COMMIT;
END;
/
```

SDO_GEOR.generatePyramid

Format

```
SDO_GEOR.generatePyramid(
   georaster        IN OUT SDO_GEORASTER,
   pyramidParams  IN VARCHAR2,
   bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

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.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

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 maximum reduction level: 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).

- nodata (for example, nodata=TRUE): Specifies whether NODATA values and value ranges should be considered during the procedure. Must be either TRUE (NODATA values and value ranges should be considered) or FALSE (NODATA values and value ranges should not be considered). The default value is FALSE. If the value is TRUE and the resampling method is BILINEAR, AVERAGE4, AVERAGE16, or CUBIC, whenever a cell value involved in the resampling calculation is a NODATA value, the result of the resampling is also a NODATA value. The resulting NODATA value is the minimum NODATA value associated with the current raster layer, if multiple NODATA values or value ranges exist.

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,
   height     IN NUMBER DEFAULT NULL
   ) RETURN SDO_GEOMETRY;
```

Description

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

Parameters

georaster
GeoRaster object.

height
Number specifying the Z value for three-dimensional (X, Y, Z) georeferencing.

Usage Notes

The returned SDO_GEOMETRY object is based on the model coordinate system 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; otherwise, the SDO_SRID value of the SDO_GEOMETRY object is the model SRID. Specifically:

- If the GeoRaster object is not georeferenced or if the model coordinate system is projected, the spatial extent object is a single polygon derived from eight boundary points.
- If the model coordinate system is geodetic, the spatial extent is densified according to the object's spatial footprint. If the area of the footprint is not larger than half of the Earth's surface, the result is a single geodetic polygon. Otherwise, a geodetic MBR is returned as the generated spatial extent object, and this returned object will be an invalid geometry according to Oracle Spatial validation rules, but index and query operations will work on this returned object.

The footprint is automatically adjusted, based on the GeoRaster object's model coordinate location (`CENTER` or `UPPERLEFT`), to cover the whole area in the model space. `CENTER` is the default model coordinate location for non-georeferenced cases.

If the model coordinate system is three-dimensional, the generated spatial extent is a three-dimensional geometry. To build a spatial index based on the generated value, you may need to convert it into a two-dimensional geometry before saving it in the `spatialExtent` attribute of the GeoRaster object. For more information about cross-dimensionality transformations, see *Oracle Spatial Developer's Guide*.

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 3.6.

If `georaster` is null, this function returns a null SDO_GEOMETRY object. If `georaster` is not valid, an exception is raised.

Examples

The following example generates a three-dimensional spatial extent, with a Z or height dimension value of 10, in the geographic 3D coordinate system 4327 (the model SRID). (The output is slightly reformatted.)

```
SELECT SDO_GEOR.generateSpatialExtent(georaster,10) spatialExtent
  FROM georaster_table where georid=10;

SPATIALEXTENT(A.GEORASTER,10)(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_IN
--------------------------------------------------------------------------------
SDO_GEOMETRY(3003, 4327, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_
ARRAY(.181818182, 1.1627907, 10, 12.1228111, 1.07010227, 10, 19.3902574,
1.07010229, 10, 25.1482989, 1.07010229, 10, 30.0714774, 1.07010229,
10, 34.4500035, 1.07010229, 10, 38.3920079, 1.07010229, 10, 42.0490801,
1.07010229, 10, 45.4612165, 1.07010229, 10, 48.6719786, 1.07010229, 10,
53.6193472, 1.07010229, 10, 53.6193472, 12.346373, 10, 53.6178888, 15.3903048,
10, 53.6178888, 18.3032341, 10, 50.6322061, 18.3032341, 10, 47.5331761,
18.3032341, 10, 44.2541078, 18.3032341, 10, 40.7594212, 18.3032341, 10, 37,
18.3032341, 10, 32.9046537, 18.3032341, 10, 28.3630834, 18.3032341, 10,
23.1869539, 18.3032341, 10, 17, 18.3032341, 10, -2.220E-16, 18.3032341, 10, 0,
16.3247208, 10, -2.220E-16, 13.6133114, 10, .181818182, 1.1627907, 10))
```

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.generateStatistics

Format

```
SDO_GEOR.generateStatistics(
    georaster          IN OUT SDO_GEORASTER,
    samplingFactor     IN VARCHAR2,
    samplingWindow IN SDO_NUMBER_ARRAY,
    histogram          IN VARCHAR2,
    layerNumbers       IN VARCHAR2 DEFAULT NULL,
    useBin             IN VARCHAR2 DEFAULT 'TRUE',
    binFunction        IN SDO_NUMBER_ARRAY DEFAULT NULL,
    nodata             IN VARCHAR2 DEFAULT 'FALSE'
    ) RETURN VARCHAR2;
```

or

```
SDO_GEOR.generateStatistics(
    georaster          IN OUT SDO_GEORASTER,
    samplingFactor     IN VARCHAR2,
    samplingWindow IN SDO_GEOMETRY,
    histogram          IN VARCHAR2,
    layerNumbers       IN VARCHAR2 DEFAULT NULL,
    useBin             IN VARCHAR2 DEFAULT 'TRUE',
    binFunction        IN SDO_NUMBER_ARRAY DEFAULT NULL,
    nodata             IN VARCHAR2 DEFAULT 'FALSE'
    ) RETURN VARCHAR2;
```

Description

Computes and sets statistical data associated with one or more layers.

Parameters

georaster
GeoRaster object.

samplingFactor
Sampling factor in the format `'samplingFactor=n'`, with the denominator *n* in 1/(*n***n*) representing the number of cells skipped in both row and column dimensions in computing the statistics. For example, if `samplingFactor` is 4, one-sixteenth of the cells are sampled; but if `samplingFactor` is 1, all cells are sampled. The higher the value, the less accurate the statistics are likely to be, but the more quickly they will be computed.

samplingWindow
Sampling window: a rectangular window for which to set statistics, specified either as a numeric array with the lower-left and upper-right coordinates or as an SDO_GEOMETRY object. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`. The window must be inside the extent in cell space. The default for this parameter is the entire image.

A sampling window for which to generate statistics, specified either as a numeric array or as a SDO_GEOMETRY object. If the data type is SDO_NUMBER_ARRAY (defined as `VARRAY(1048576) OF NUMBER`), 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, it is transformed into raster space if it is in model space, and then the minimum bounding rectangle (MBR) of the geometry object in raster space is used as the window. The default value is the entire image. In both cases, the intersection of the MBR of the sampling window in raster space and the MBR of the GeoRaster object in raster space is used for computing statistics.

histogram
Specify `TRUE` to cause a histogram to be computed and stored, or `FALSE` to cause a histogram not to be computed and stored. Histograms are discussed in Section 2.3.1. The XML definitions of the `<histogram>` element and the `histogramType` complex type are included in Appendix A.

layerNumbers
Numbers of the layers for which to compute the statistics. This is a string that can include numbers, number ranges indicated by hyphens (-), and commas to separate numbers and number ranges. For example, `'1,3-5,7'` specifies layers 1, 3, 4, 5, and 7. Layer 0 (zero) indicates the object layer.

useBin
Specifies whether or not to use a provided bin function (specified in the `binFunction` parameter) when generating statistics. `TRUE` (the default) causes a bin function to be used as follows: (1) the bin function specified by the `binFunction` parameter, if it is not null; otherwise, (2) the bin function specified by the `<binFunction>` element in the GeoRaster XML metadata, if one is specified; otherwise, (3) a dynamically generated bin function, as explained in the Usage Notes. `FALSE` causes a dynamically generated bin function to be used, and causes the `binFunction` parameter and `<binFunction>` element to be ignored.

For information about bin functions, see the Usage Notes for the SDO_GEOR.setBinFunction procedure.

binFunction
Bin function as an array whose elements specify the bin type, total number of bins, first bin number, minimum cell value, and maximum cell value. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`. For more information about the bin function for SQO_GEOR.generateStatistics, see the Usage Notes. For information about bin functions and an example, see the Usage Notes for the SDO_GEOR.setBinFunction procedure.

nodata
Specifies whether or not to compare each cell values with NODATA values defined in the metadata when computing statistics. `TRUE` causes all pixels with a NODATA value not to be considered; `FALSE` (the default) causes pixels with NODATA values to be considered as regular pixels. NODATA values and value ranges are discussed in Section 1.9.

Usage Notes

This function computes and sets the statistical data described by the <statisticDatasetType> element in the GeoRaster metadata XML schema, which is described in Appendix A.

If histogram is TRUE, this function determines the range of each bin based on the bin function being used, and within each range it computes the count of each pixel value. The histogram and the bin function are related as follows: each bin is mapped to a (value, count) pair of the histogram, and the lower boundary of each bin is mapped to corresponding value of histogram (value, count) pair, with the following exceptions:

- If Min_r < Min, then one more pair (Min_r, count) is added as the first pair of the histogram. (Min_r is the real minimum value of the data set computed by this function, and Min is the min value specified in the bin function.)
- If Max_r > Max, then one more pair (Max_r, count) is added as the last pair of the histogram. (Max_r is the real maximum value of the data set computed by this function, and Max is the max value specified in the bin function.)
- Leading and trailing count=0 pairs in the histogram are suppressed. For example:

```
(1,0) (2,0) (3,11) (4,12) becomes (3,11) (4,12)
(1,11) (2,12) (3,0) (4,0) becomes (1,11) (2,12)
```

If histogram is TRUE, any existing histogram in the XML metadata is replaced by the new generated histogram.

SQO_GEOR.generateStatistics supports only LINEAR bin functions (binType = 0), not LOGARITHM or EXPLICIT bin functions. (The XML definitions of all bin function types are in Appendix A.) If the useBin parameter value is FALSE, this function ignores any binFunction parameter value and any <binFunction> element in the GeoRaster XML metadata, and it uses a LINEAR bin function in which the min and max values are the actual minimum and maximum values of the data set, and the numbins value depends on the cell depth, as follows:

- If cellDepth = 1, numbins = 2.
- If cellDepth = 2, numbins = 4.
- If cellDepth = 4, numbins = 8.
- If cellDepth >= 8, numbins = 256.

If the useBin parameter value is TRUE, this function uses a bin function as follows:

1. If the binFunction parameter specifies a valid bin function, it is used.
2. Otherwise, if a valid bin function is defined in the GeoRaster metadata, it is used.
3. Otherwise, the same bin function is used as when the useBin parameter value is FALSE.

Contrast this function, which causes GeoRaster to compute and set the statistics, with the SDO_GEOR.setStatistics procedure, in which you specify the statistics to be set.

To retrieve the statistical data associated with a layer, use the SDO_GEOR.getStatistics function.

Examples

The following example generates the statistical data and a histogram.

```
DECLARE
  gr SDO_GEORASTER;
```

```
  ret VARCHAR2(256);
  window SDO_NUMBER_ARRAY := SDO_NUMBER_ARRAY(20,10,50,50);
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=1 FOR UPDATE;
  SDO_GEOR.setBinFunction(gr,1,sdo_number_array(0,10,1,50,200));
  ret := SDO_GEOR.generateStatistics(gr, 'samplingFactor=5', window, 'TRUE',
    '1-1', 'TRUE');
  UPDATE georaster_table SET georaster=gr WHERE georid=1;
  COMMIT;
END;
/
```

SDO_GEOR.georeference

Format (procedure)

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

Format (function)

```
SDO_GEOR.georeference(
   georaster                IN OUT SDO_GEORASTER,
   FFMethodType             IN VARCHAR2,
   nGCP                     IN NUMBER,
   GCPs                     IN SDO_GEOR_GCP_COLLECTION,
   storeGCP                 IN VARCHAR2 DEFAULT 'TRUE',
   srid                     IN NUMBER DEFAULT NULL,
   modelCoordinateLocation IN NUMBER DEFAULT NULL);
   ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.georeference(
   georaster                IN OUT SDO_GEORASTER,
   gcpGeorefModel           IN SDO_GEOR_GCPGEOREFTYPE,
   storeGCP                 IN VARCHAR2 DEFAULT 'TRUE',
   srid                     IN NUMBER DEFAULT NULL,
   modelCoordinateLocation IN NUMBER DEFAULT NULL,
   ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.georeference(
   georaster                IN OUT SDO_GEORASTER,
   FFMethodType             IN VARCHAR2,
   srid                     IN NUMBER DEFAULT NULL,
   modelCoordinateLocation IN NUMBER DEFAULT NULL,
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

As a procedure, georeferences a GeoRaster object using specified cell-to-model transformation coefficients of an affine transformation. As a function, returns the solution of any one of the supported geometric models using ground control points (GCPs) that are either stored in the database or specified in parameters.

Parameters

georaster
The SDO_GEORASTER object to be georeferenced.

srid
Model coordinate system. For the procedure, must not be null or 0 (zero); for function, it can be null. 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 the Usage Notes.

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

FFMethodType
Polynomial or rational polynomial function used as georeference geometric model. Must be one of the following string values: `Affine`, `QuadraticPolynomial`, `CubicPolynomial`, `DLT`, `QuadraticRational`, or `RPC`.

gcpGeorefModel
Object containing the following: `FFMethodType`, `nGCP`, `GCPs`, `solutionAccuracy`.

nGCP
Number of ground control points in the GCP collection (`GCPs` parameter).

GCPs
The GCP collection, of type SDO_GEOR_GCP_COLLECTION (described in Section 2.3.7).

storeGCP
A flag indicating whether the GCPs should be stored in the GeoRaster metadata. The string `TRUE` (the default) stores the points in the GeoRaster metadata; the string `FALSE` does not store the points in the GeoRaster metadata.

Usage Notes

Notes for the Procedure Format

Use this procedure to georeference a GeoRaster object based on an existing affine transformation. Georeferencing is explained in Section 1.6 and Section 3.5.

This procedure assumes that in the original georeferencing information in the source data, such as in an ESRI world file, the transformation formulas are the following:

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

Specify the preceding A, B, C, D, E, and F coefficients to the SDO_GEOR.georeference procedure. They are automatically adjusted internally to produce the correct georeferencing result: a, b, c, d, e, and f coefficients, as in the following formulas:

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

In these formulas:

- row = Row index of the cell in raster space.
- column = 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 GeoRaster SRS metadata.
- b*f – c*e should not be equal to 0 (zero).

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

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

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 or a Digital Globe RPC file from a file or from a CLOB object. It also loads geometadata from a GeoTIFF file.
- The GeoRaster loader tool (described in Section 1.14) can load an ESRI world file, a Digital Globe RPC file, or a GeoTIFF file.

Notes for the Function Formats (for Use with GCPs)

This function calculates the solution of the specified geometric model (the `FFMethodType`) using the GCPs that are either stored in the database or specified in parameters, and it stores the solution in the GeoRaster functional fitting model.

The returned array contains RMS values and residuals, which have has the following order: the solution accuracy (rowRMS, colRMS, totalRMS) computed using control points, the ground positioning accuracy (xRMS, yRMS, zRMS, modelTotalRMS) computed using check points, the ground positioning accuracy (xRMS, yRMS, zRMS, modelTotalRMS) computed using control points, and the (xResidual, yResidual) for each control point (not for check points). The ordering of the residuals is the same as the control points stored in the XML metadata (not necessarily in the sequential order of the control point ID values if the ID values are numbers).

There are always at least 17 values returned (assuming at least 3 control points). A positioning accuracy (RMS) value of –1.0 means that value does not exist. For a

two-dimensional geometric model, the zRMS value is always –1.0; otherwise, zRMS values are always 0 in the current release.

The GCPs can either be retrieved from the GeoRaster metadata or provided using the GCP-related object types.

For the interface without GCP information (that is, the format without the `gcpGeorefModel` parameter), the GCPs are assumed to be stored in the GeoRaster object's metadata. If no GCPs are stored or if not enough GCPs are stored for the specified model, an exception is raised.

After this function call, the GeoRaster object is georeferenced and the coefficients of the functional fitting model are set in the GeoRaster SRS metadata component.

For more information about georeferencing using GCPs, see Section 1.6.2.

Examples

The following example georeferences a GeoRaster object directly using the cell-to-model coefficients of an affine transformation. (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, 0,
                        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))
```

If the original raster data is rectified and if the model coordinate of the center point of the upper-left corner cell is (*x0, y0*) and its spatial resolution is *s*, you can directly use the preceding example code to georeference the GeoRaster object by replacing 28.5 with *s*, 1232804.04 with *x0*, and 13678.09 with *y0*. If you have other information about the GeoRaster object, such as a well-defined precise envelope of the raster or the model coordinates of the center point, you can compute the (*x0, y0*) and the spatial resolution *s*, and then use the same approach to georeference the object.

The following example georeferences a GeoRaster object, using ground control point (GCP) information.

```
DECLARE
   gr1                 sdo_georaster;
   gr2                  sdo_georaster;
   georefModel   SDO_GEOR_GCPGEOREFTYPE;
   GCPs              SDO_GEOR_GCP_COLLECTION;
   rms                  sdo_number_array;
BEGIN
   SELECT georaster INTO gr1 from georaster_table WHERE georid=10 FOR UPDATE;

   GCPs := SDO_GEOR_GCP_COLLECTION(
                          SDO_GEOR_GCP('1', '', 1,
                             2, sdo_number_array(25.625000, 73.875000),
                             2, sdo_number_array(237036.937500, 897987.187500),
                             NULL, NULL),
                             SDO_GEOR_GCP('2', '', 1,
                             2, sdo_number_array(100.625000, 459.125000),
                             2, sdo_number_array(237229.562500, 897949.687500),
                             NULL, NULL),
                             SDO_GEOR_GCP('3', '', 1,
                             2, sdo_number_array(362.375000, 77.875000),
                             2, sdo_number_array(237038.937500, 897818.812500),
                             NULL, NULL),
                             SDO_GEOR_GCP('4', '', 1,
                             2, sdo_number_array(478.875000, 402.125000),
                             2, sdo_number_array(237201.062500, 897760.562500),
                             NULL, NULL),
                             SDO_GEOR_GCP('5', '', 2,
                             2, sdo_number_array(167.470583,  64.030686),
                             2, sdo_number_array(237032.015343, 897916.264708),
                             NULL, NULL),
                             SDO_GEOR_GCP('6', '', 2,
                             2, sdo_number_array(101.456177,  257.915534),
                             2, sdo_number_array(237128.957767, 897949.271912),
                             NULL, NULL)
                       );

   georefModel := SDO_GEOR_GCPGEOREFTYPE('Affine',
GCPs.count, GCPs, NULL);

   rms := sdo_geor.georeference(gr1, georefModel, 'FALSE', 26986, 1);
   UPDATE georaster_table SET georaster=gr1 WHERE georid=10;
   COMMIT;
END;
/
```

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.getBinFunction

Format

```
SDO_GEOR.getBinFunction(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Gets the bin function 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 the bin function as an array whose elements specify the bin type, total number of bins, first bin number, minimum cell value, and maximum cell value. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`.

If the bin type is `EXPLICIT`, an external bin table is used and this function returns a null value.

For a more detailed explanation of the bin function format, see the Usage Notes for the SDO_GEOR.setBinFunction procedure.

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

Examples

The following example gets the bin function for layer 3 of a specified GeoRaster object.

```
SELECT sdo_geor.getBinFunction(georaster,3) FROM georaster_table WHERE georid=4;

SDO_GEOR.GETBINFUNCTION(GEORASTER,3)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0, 10, 1, 0, 511)
```

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 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.getBitmapMask

Format

```
SDO_GEOR.getBitmapMask(
   georaster      IN SDO_GEORASTER,
   layerNumber    IN NUMBER,
   storageParam   IN VARCHAR2,
   mask           IN OUT SDO_GEORASTER);
```

Description

Gets the bitmap mask that is associated with a GeoRaster object or a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
The layer on which to perform the operation. A value of 0 (zero) indicates the object layer.

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

mask
The resulting GeoRaster object to hold the bitmap mask. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as the input GeoRaster object (`georaster` parameter).

Usage Notes

If no bitmap mask is associated with the specified layer of the GeoRaster object, the resulting GeoRaster object is not updated.

If the `storageParam` parameter is null, the resulting GeoRaster object has a `celldepth` value of 1BIT celldepth, has `DEFLATE` compression if the input GeoRaster object is compressed, and has the same interleaving type and blocking size as the input GeoRaster object. It also contains all the bitmap mask pyramids if the input GeoRaster object has pyramids built on it.

A bitmap mask may have empty raster blocks (see Section 1.4.4). If there is reblocking that leads to partially empty raster blocks, any cells in a partially empty block that are derived from an empty raster block are filled with the value 0 (zero).

For an explanation of bitmap masks, see Section 1.8.

Examples

The following example retrieves the bitmap mask associated with the object layer of the specified GeoRaster object.

```
DECLARE
```

```
  gr sdo_georaster;
  mk sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=4;
  INSERT INTO georaster_table (georid, georaster)
    VALUES (100, sdo_geor.init('rdt_1', 100))
    RETURNING georaster INTO mk;
  sdo_geor.getBitmapMask(gr, 0, 'compression=none', mk);
  UPDATE georaster_table SET georaster=mk WHERE georid=100;
  COMMIT;
END;
/
```

SDO_GEOR.getBitmapMaskSubset

Format

```
SDO_GEOR.getBitmapMaskSubset(
    georaster       IN SDO_GEORASTER,
    layerNumber   IN NUMBER,
    pyramidLevel   IN VARCHAR2,
    window          IN SDO_NUMBER_ARRAY,
    rasterBlob      IN OUT NOCOPY BLOB,
    storageParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.getBitmapMaskSubset(
    georaster       IN SDO_GEORASTER,
    layerNumber   IN NUMBER,
    pyramidLevel   IN VARCHAR2,
    inWindow       IN SDO_NUMBER_ARRAY,
    rasterBlob      IN OUT NOCOPY BLOB,
    outWindow      OUT SDO_NUMBER_ARRAY,
    storageParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.getBitmapMaskSubset(
    georaster       IN SDO_GEORASTER,
    layerNumber   IN NUMBER,
    pyramidLevel   IN VARCHAR2,
    window          IN SDO_GEOMETRY,
    rasterBlob      IN OUT NOCOPY BLOB,
    storageParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.getBitmapMaskSubset(
    georaster      IN SDO_GEORASTER,
    layerNumber  IN NUMBER,
    pyramidLevel  IN VARCHAR2,
    inWindow       IN SDO_GEOMETRY,
    rasterBlob      IN OUT NOCOPY BLOB,
    outWindow      OUT SDO_NUMBER_ARRAY,
    storageParam IN VARCHAR2 DEFAULT NULL);
```

Description

Gets a subset of a bitmap mask.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer on which to perform the operation. A value of 0 (zero) indicates the object layer.

pyramidLevel
Pyramid level containing the specified cell.

window, inWindow
A rectangular window for the subset, specified either as a numeric array with the lower-left and upper-right coordinates or as an SDO_GEOMETRY object. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`.

rasterBlob
BLOB to hold the output (the resulting subset).

outWindow
An SDO_NUMBER_ARRAY object identifying the coordinates of the upper-left and lower-right corners of the output window in the cell space.

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 the `storageParam` parameter is null, the resulting GeoRaster object has a `celldepth` value of 1BIT celldepth, has `DEFLATE` compression if the input GeoRaster object is compressed, and has the same interleaving type as the input GeoRaster object.

Usage Notes

If there is no bitmap associated with the specified GeoRaster object at the specified raster layer, or the specified input window does not intersect with the spatial extent of the GeoRaster object, the procedure returns with `rasterBlob` truncated to length zero and the `outWindow` set to a null value.

This procedure operates on a single GeoRaster object. The procedure has four formats, depending on whether the input window is specified as a geometry object or as the upper-left and lower-right corners of a box, and on whether the `outWindow` parameter is used to return the coordinates of the output window.

If the `window` or `inWindow` parameter data type is SDO_GEOMETRY, the SDO_SRID value must be one of the following: null (to specify raster space) or a value from the SRID column of the MDSYS.CS_SRS table.

If the SDO_SRID values for the `window` or `inWindow` parameter geometry and the model space are different, the geometry parameter 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 geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial Developer's Guide*.

After the procedure completes, the rasterBLOB parameter contains the cell (pixel) data in the cropped window without tiling. The cropped window is the overlapping portion of the specified window of interest and the source GeoRaster object's spatial extent. If the `outWindow` parameter is specified, after the procedure completes it contains the coordinates of the cropped window in the cell space.

A bitmap mask may have empty raster blocks (see Section 1.4.4). Any cells in the output window that are derived from an empty raster block are filled with the value 0 in the output BLOB.

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 regardless of whether the input GeoRaster object is compressed or not. 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 1.10.

For an explanation of bitmap masks, see Section 1.8.

Examples

The following example retrieves a subset of a bitmap mask associated with the object layer of a specified GeoRaster object.

```
DECLARE
  gr sdo_georaster;
  lb blob;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=4;
  dbms_lob.createTemporary(lb, TRUE);
  sdo_geor.getBitmapMaskSubset(gr, 0, 0, sdo_number_array(0,0,99,99), lb,
'compression=none');
  dbms_lob.freeTemporary(lb);
END;
/
```

SDO_GEOR.getBitmapMaskValue

Format

```
SDO_GEOR.getBitmapMaskValue(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER,
   pyramidLevel  IN VARCHAR2,
   rowNumber   IN NUMBER,
   colNumber     IN NUMBER
   ) RETURN NUMBER;
```

or

```
SDO_GEOR.getBitmapMaskValue(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER,
   pyramidLevel  IN VARCHAR2,
   ptGeom         IN SDO_GEOMETRY
   ) RETURN NUMBER;
```

Description

Gets the value of a single cell from a bitmap mask.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer on which to perform the operation. A value of 0 (zero) indicates the object layer.

pyramidLevel
Pyramid level containing the specified cell.

rowNumber
Row number in cell space.

colNumber
Column number in cell space.

ptGeom
Point geometry in cell space or model space.

Usage Notes

You can specify the cell by its row and column numbers or by a point geometry object.

If there is no bitmap associated with the specified GeoRaster object at the specified raster layer, or the specified cell is in an empty raster block, the function returns a null value.

For an explanation of bitmap masks, see Section 1.8.

Examples

The following example gets the value of four cells from the bitmap mask associated with a specified GeoRaster object.

```
SELECT sdo_geor.getBitmapMaskValue(georaster,0,0,0,0) c1,
       sdo_geor.getBitmapMaskValue(georaster,0,0,9,9) c2,
       sdo_geor.getBitmapMaskValue(georaster,0,0,9,10) c3,
       sdo_geor.getBitmapMaskValue(georaster,0,0,10,9) c4
  FROM georaster_table WHERE georid=0;
```

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,
    subCell             IN VARCHAR2 DEFAULT NULL,
    height              IN NUMBER DEFAULT NULL,
    vert_id             IN NUMBER DEFAULT NULL,
    ellipsoidal         IN VARCHAR2 DEFAULT NULL
    ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.getCellCoordinate(
    georaster           IN SDO_GEORASTER,
    pyramidLevel        IN NUMBER,
    modelCoordinate     IN SDO_GEOMETRY,
    cellCoordinate      OUT SDO_GEOMETRY,
    subCell             IN VARCHAR2 DEFAULT NULL,
    height              IN NUMBER DEFAULT NULL,
    vert_id             IN NUMBER DEFAULT NULL,
    ellipsoidal         IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.getCellCoordinate(
    georaster               IN SDO_GEORASTER,
    sourcePyramidLevel      IN NUMBER,
    sourceCellCoordinate    IN SDO_NUMBER_ARRAY,
    targetPyramidLevel      IN NUMBER,
    subCell                 IN VARCHAR2 DEFAULT NULL,
    ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.getCellCoordinate(
    georaster               IN SDO_GEORASTER,
    sourcePyramidLevel      IN NUMBER,
    sourceCellCoordinate    IN SDO_GEOMETRY,
    targetPyramidLevel      IN NUMBER,
```

```
    subCell                 IN VARCHAR2 DEFAULT NULL,
) RETURN SDO_GEOMETRY;
```

Description

Returns the coordinates in the cell (raster) coordinate system associated with the geometry at the specified model (ground) coordinates (first two formats), or converts cell coordinates between pyramid levels (last two formats).

Note that the second format is a procedure; the other formats are functions.

Parameters

georaster
GeoRaster object.

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

modelCoordinate
The geometry that is to be converted.

cellCoordinate
The output geometry in the cell space of the GeoRaster object.

sourcePyramidLevel (last two formats)
Pyramid level with which the input cell coordinate is associated.

sourceCellCoordinate (last two formats)
Input cell coordinates to be converted. Must be a two-dimensional geometry, and its SDO_SRID value must be null.

targetPyramidLevel (last two formats)
Pyramid level of the returned (target) GeoRaster object.

subCell
String (`TRUE` or `FALSE`) specifying whether to return the cell coordinates in sub-pixel (floating) values.

height
Number specifying the Z value for three-dimensional (X, Y, Z) georeferencing.

vert_id
Number specifying the vertical reference ID.

ellipsoidal
String specifying whether the vertical reference system is ellipsoidal (`TRUE`) or not ellipsoidal (`FALSE`).

Usage Notes

The first two formats of this function return the coordinates in the cell (raster) coordinate system associated with the geometry at the specified model (ground) coordinates:

- Use the first format (a function without the `cellCoordinate` parameter) to transform a point in the ground coordinate system (a longitude, latitude pair) to the location of a point on the GeoRaster image.

- Use the second format (a procedure with the `cellCoordinate` parameter) to transform a geometry in the ground coordinate system to the location of a geometry in the raster space of the GeoRaster object. The conversion is done by converting the coordinates of each vertex of the input geometry from the ground coordinate system to the raster space of the GeoRaster object.

The last two formats of this function convert cell coordinates between pyramid levels. If the type of the `sourceCellCoordinate` parameter is SDO_NUMBER_ARRAY, it specifies the <row,column> pair for a point in the cell space at the source pyramid level. If the type of the sourceCellCoordinate parameter is SDO_GEOMETRY, it specifies a geometry in the cell space at the source pyramid level. The coordinates of each vertex of the input geometry are converted according to the specified pyramid levels.

- Use the first format (without the `cellCoordinate` parameter) to transform a point in the ground coordinate system (a longitude, latitude pair) to the location of a point on the GeoRaster image.
- Use the second format (with the `cellCoordinate` parameter) to transform a geometry in the ground coordinate system to the location of a geometry in the raster space of the GeoRaster object. The conversion is done by converting the coordinates of each vertex of the input geometry from the ground coordinate system to the raster space of the GeoRaster object.

If the SDO_SRID value of the `modelCoordinate` geometry is null, the parameter specifies a geometry in the raster space; otherwise, it specifies a point in a ground coordinate system. If the ground coordinate system is different from the model coordinate system, the `modelCoordinate` parameter geometry is automatically transformed to the coordinate system of the model space before the operation is performed.

Contrast this function with SDO_GEOR.getModelCoordinate, 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)
```

The following example returns the geometry at pyramid level 0 that is associated with the specified geometry at pyramid level 2, assuming the geometry is not georeferenced (the model coordination location is CENTER) and the ultCoordinate is (100,-100,0).

```
SELECT sdo_geor.getCellCoordinate(georaster, 2,
               sdo_geometry(2003,NULL,NULL,sdo_elem_info_array(1,1003,3),
                            sdo_ordinate_array(100.8,-100.2,220.15,0.3)),
               0, 'true') coord
FROM georaster_table WHERE georid=1;
```

```
COORD
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR
AY(104.7, -99.3, 582.1, 302.7))
```

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,
   rowNumber   IN NUMBER,
   colNumber     IN NUMBER,
   bands          IN VARCHAR2
   ) RETURN SDO_NUMBER_ARRAY;
```

or

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

or

```
SDO_GEOR.getCellValue(
   georaster      IN SDO_GEORASTER,
   pyramidLevel IN NUMBER,
   ptGeom         IN SDO_GEOMETRY,
   layers         IN VARCHAR2
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

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

If the specified cell is in an empty raster block, the function returns a null value.

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.

bands
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).

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.)

layers
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). (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 do any interpolation of cell values. (To evaluate a point location using an interpolation method, use the SDO_GEOR.evaluateDouble function.) 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 = (a0 + a1 * cellvalue) / (b0 + b1 * 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
```

The following example returns the values of the cells in bands 0, 1, and 2 for row number 10, column number 10 of the GeoRaster object (GEORASTER column) in the row with the GEORID column value of 1 in the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.

```
SELECT sdo_geor.getcellvalue(a.georaster,0,10,10,'0-2')
  FROM georaster_table a WHERE georid=1;

SDO_GEOR.GETCELLVALUE(A.GEORASTER,0,10,10,'0-2')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 137, 32)
```

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 1.10.

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.getControlPoint

Format

```
SDO_GEOR.getControlPoint (
   inGeoraster      IN SDO_GEORASTER,
   controlPointID  IN VARCHAR2
   ) RETURN SDO_GEOR_GCP;
```

Description

Returns the ground control point (GCP) that has the specified control point ID value.

Parameters

inGeoraster
GeoRaster object.

controlPointID
Control point ID of `inGeoraster`. Must be a string not more than 32 characters.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

This function returns an object of type SDO_GEOR_GCP, which is described in Section 2.3.6.

In the control point ID is null, empty, or missing in `inGeoraster`, an exception is raised.

Examples

The following example returns the GCP that has the ID value 25 in a specified GeoRaster object.

```
SELECT sdo_geor.getControlPoint(georaster, '25') FROM georaster_table
  WHERE georid =10;

SDO_GEOR.GETCONTROLPOINT(GEORASTER,'25')(POINTID, DESCRIPTION, POINTTYPE, CELLDI
--------------------------------------------------------------------------------
SDO_GEOR_GCP('25', NULL, 2, 2, SDO_NUMBER_ARRAY(167.470583, 64.030686), 2, SDO_N
UMBER_ARRAY(237032.015, 897916.265), NULL, NULL)
```

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.getGCPGeorefMethod

Format

```
SDO_GEOR.getGCPGeorefMethod(
   inGeoraster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the ground control point (GCP)-based georeferencing geometric model type of a GeoRaster object.

Parameters

inGeoraster
GeoRaster object.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

If `inGeoraster` does contains valid georeferencing model information, it returns one of the following values: `Affine`, `QuadraticPolynomial`, `CubicPolynomial`, `DLT`, `QuadraticRational`, or `RPC`.

If `inGeoraster` does not contain any georeferencing model information, this function returns a null value.

Examples

The following example returns the GCP-based georeferencing model information in a specified GeoRaster object. (The output is reformatted for readability.)

```
SELECT sdo_geor.getGCPGeorefMethod(georaster) FROM georaster_table
  WHERE georid =10;

SDO_GEOR.GETGCPGEOREFMETHOD(GEORASTER)
--------------------------------------------------------------------------------
Affine
```

SDO_GEOR.getGCPGeorefModel

Format

```
SDO_GEOR.getGCPGeorefModel(
   inGeoraster  IN SDO_GEORASTER
   ) RETURN SDO_GEOR_GCPGEOREFTYPE;
```

Description

Returns all information about the ground control point (GCP)-based georeferencing model in a GeoRaster object.

Parameters

inGeoraster
GeoRaster object.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

The SDO_GEOR_GCPGEOREFTYPE object type is defined in Section 2.3.8.

If `inGeoraster` does not contain any georeferencing model information, this function returns a null value.

Examples

The following example returns the GCP-based georeferencing model information in a specified GeoRaster object. (The output is reformatted for readability.)

```
SELECT sdo_geor.getGCPGeorefModel(georaster) FROM georaster_table WHERE georid=10;

SDO_GEOR.GETGCPGEOREFMODEL(GEORASTER)(FFMETHODTYPE,
NUMBERGCP, GCPS(POINTID, DES...
--------------------------------------------------------------------------------
SDO_GEOR_GCPGEOREFTYPE('Affine', 6,
SDO_GEOR_GCP_COLLECTION(
SDO_GEOR_GCP('21', NULL, 1, 2,SDO_NUMBER_ARRAY(25.625, 73.875), 2, SDO_NUMBER_
ARRAY(237036.938,897987.188), NULL, NULL),
SDO_GEOR_GCP('22', NULL, 1, 2,SDO_NUMBER_ARRAY(100.625, 459.125), 2,SDO_NUMBER_
ARRAY(237229.563, 897949.688), NULL, NULL),
SDO_GEOR_GCP('23', NULL, 1, 2, SDO_NUMBER_ARRAY(362.375, 77.875), 2,
SDO_NUMBER_ARRAY(237038.938, 897818.813), NULL, NULL),
SDO_GEOR_GCP('24', NULL, 1, 2, SDO_NUMBER_ARRAY(478.875, 402.125), 2,
SDO_NUMBER_ARRAY(237201.063, 897760.563), NULL, NULL),
SDO_GEOR_GCP('25', NULL, 2, 2, SDO_NUMBER_ARRAY(167.470583,
64.030686), 2, SDO_NUMBER_ARRAY(237032.015, 897916.265), NULL, NULL),
SDO_GEOR_GCP('26', NULL, 2, 2, SDO_NUMBER_ARRAY(101.456177,
257.915534), 2, SDO_NUMBER_ARRAY(237128.958, 897949.272), NULL, NULL)),
NULL)
```

SDO_GEOR.getGeoreferenceType

Format

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

Description

Returns a number that indicates the georeference type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns one of the following numbers to indicate the georeference type: 1 for unknown type or null GeoRaster object, 2 for affine transform, 3 for direct linear transform (DLT), 4 for rational polynomial coefficient (RPC), 5 for cubic polynomial, 6 for quadratic rational polynomial, or 7 for quadratic polynomial.

For an explanation of georeferencing, see Section 1.6.

Examples

The following example returns the georeference type for the GeoRaster objects in a table named GEORASTER_TABLE. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
SELECT georid,sdo_geor.getGeoreferenceType(a.georaster)
  FROM georaster_table a ORDER BY georid;

    GEORID SDO_GEOR.GETGEOREFERENCETYPE(A.GEORASTER)
---------- ----------------------------------------
         1                                        1
         2                                        1
         3                                        1
         4                                        1
         5                                        1
         7                                        1
         8                                        2
         9                                        1
        10                                        1
        12                                        1
        13                                        1
        14                                        2
        15                                        1
        16                                        1
        17                                        1
        18                                        1
        19                                        2
        20                                        2
        21                                        4
        22                                        4
```

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 1–5 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 1–5 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,
   height           IN NUMBER DEFAULT NULL,
   ) RETURN SDO_GEOMETRY;
```

or

```
SDO_GEOR.getModelCoordinate(
   georaster          IN SDO_GEORASTER,
   pyramidLevel       IN NUMBER,
   cellCoordinate     IN SDO_GEOMETRY,
   modelCoordinate OUT SDO_GEOMETRY,
   height             IN NUMBER DEFAULT NULL);
```

Description

Returns a geometry associated with the specified cell (raster) coordinates at the specified pyramid level.

Parameters

georaster
GeoRaster object.

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

cellCoordinate
If the type is SDO_NUMBER_ARRAY, `cellCoordinate` is an 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. If the type is SDO_GEOMETRY, `cellCoordinate` specifies a geometry in the cell coordinate system

modelCoordinate
The output geometry.

height
Number specifying the Z value for three-dimensional (X, Y, Z) georeferencing.

Usage Notes

SDO_GEOR.getModelCoordinate has two formats:

- Use the first format (a function without the `modelCoordinate` parameter) to transform the location of a point in the GeoRaster object's raster space.

- Use the second format (a procedure with the `modelCoordinate` parameter) to transform a geometry in the raster space of the GeoRaster object. The conversion is done by converting the coordinates of each vertex of the input geometry. Use an appropriate input geometry so that the output geometry will be valid. For example, if the model coordinate system is geodetic, the input geometry should not contain any arcs.

Use SDO_GEOR.getModelCoordinate 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.

If the GeoRaster object is georeferenced, the output geometry contains the coordinates in the model (ground) coordinate system. If the GeoRaster object is not georeferenced, the output geometry contains cell coordinates at the original image level.

If the GeoRaster object is georeferenced, the SDO_SRID value of the output geometry is the same as the model SRID of the GeoRaster object.

Contrast SDO_GEOR.getModelCoordinate with SDO_GEOR.getCellCoordinate, 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.getModelCoordLocation

Format

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

Description

Returns the model coordinate location value for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns a null value if the GeoRaster object is not georeferenced or if the `modelCoordinateLocation` element is not specified in the SRS metadata. Otherwise, it returns the `modelCoordinateLocation` element value specified in the SRS metadata.

A null return value or a value of `CENTER` means that the cell coordinate system is center-based. A value of `UPPERLEFT` means that the cell coordinate system is based on the upper-left corner.

To set or delete the model coordinate location value for a GeoRaster object, use the SDO_GEOR.setModelCoordLocation procedure.

For an explanation of georeferencing using GCPs, see Section 1.6.2.

Examples

The following example returns the model coordinate location of a specified GeoRaster object.

```
SELECT sdo_geor.getModelCoordLocation(georaster) modelCoordLocation
  FROM georaster_table
  WHERE georid = 1;
```

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;
```

or

```
SDO_GEOR.getNODATA(
   georaster     IN SDO_GEORASTER,
   layerNumber IN NUMBER
   ) RETURN SDO_RANGE_ARRAY;
```

Description

Returns the values or value ranges that represent NODATA cells in a GeoRaster object (in ascending order, without duplicates).

Parameters

georaster
GeoRaster object.

layerNumber
Layer number in the GeoRaster object. A value of 0 (zero) indicates the object layer.

Usage Notes

> **Note:** The first function format (returning a number) is deprecated, and will not be supported in future releases. You are encouraged to use the second format (returning an SDO_RANGE_ARRAY object).

Some cells of a GeoRaster object may have no meaningful value assigned or collected. Such cells contain a NODATA value are thus called NODATA cells, which means that those cells are not semantically defined. The application is responsible for defining the meaning or significance of cells identified as NODATA cells. For more information about NODATA values and value ranges, see Section 1.9.

This function returns all the NODATA values and value ranges associated with a specified raster layer of the specified GeoRaster object, in ascending order and in a compact form with duplicates eliminated. The set of NODATA values and value ranges associated with a sublayer (`layerNumber` > 0) is always a superset of the values and value ranges of the object layer (`layerNumber` = 0). The result for a sublayer is the combination of the NODATA metadata entries for the specified sublayer, the object layer, and any pre-release 11*g* NODATA metadata stored as part of the raster description information.

If the specified GeoRaster object or layer has more than one NODATA value, you must use the function format that returns an SDO_RANGE_ARRAY object. The SDO_RANGE_ARRAY type is described in Section 1.9.

If this function returns a null value, it means that all cells of the GeoRaster object or of the specified layer are defined and have a meaningful cell value.

To specify the NODATA values for a GeoRaster object, use the SDO_GEOR.addNODATA procedure.

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 SDO_GEOR.getNODATA(georaster, 0) NODATA FROM georaster_table WHERE
georid=0;

NODATA
------------
SDO_RANGE_ARRAY(SDO_RANGE(5,7))
```

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.getRasterBlockLocator

Format

```
SDO_GEOR.getRasterBlockLocator(
   georaster             IN SDO_GEORASTER,
   pyramidLevel          IN NUMBER,
   bandBlockNumber       IN NUMBER,
   rowBlockNumber        IN NUMBER,
   columnBlockNumber IN NUMBER,
   loc                   IN OUT NOCOPY BLOB,
   isBitmapMask          IN VARCHAR2 DEFAULT NULL,
   lock_for_write        IN VARCHAR2 DEFAULT NULL);
```

Description

Gets the LOB locator of a specified raster block.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the block.

bandBlockNumber
Band number of the block.

rowBlockNumber
Row number of the block.

columnBlockNumber
Column number of the block.

loc
LOB locator.

isBitmapMask
The string `TRUE` specifies that a bitmap mask block will be accessed; the string `FALSE` specifies that a regular raster block will be accessed. If you do not specify this parameter, a regular raster block will be accessed. For an explanation of bitmap masks, see Section 1.8.

lockForWrite
The string `TRUE` locks the row in the raster data table so that other users cannot lock or update that row until the current transaction ends; the string `FALSE` does not lock the row in the raster data table. If you do not specify this parameter, the row is not locked.

Usage Notes

This procedure gets the raster block locator using the specified parameters. The LOB locator is not opened, and no data is read or processed. You should use standard LOB operations to open and close the LOB locator and to read data from and write data to the LOB locator.

To ensure that data is read or written correctly, you must understand the physical storage of the raster data (described in Section 1.4), and you must compress and decompress the raster data as needed.

For information about LOB locators, see *Oracle Database SecureFiles and Large Objects Developer's Guide*.

Examples

The following example gets the LOB locators of two raster blocks, the first a regular raster block and the second a bitmap mask block. Both calls to the SDO_GEOR.getRasterBlockLocator procedure lock the row in the raster data table.

```
declare
gr sdo_georaster;
lb blob;
r1 raw(1024);

ln number;
begin
r1 := utl_raw.copies(utl_raw.cast_to_raw('0'),1024);


select georaster into gr from georaster_table where georid=1;
sdo_geor.getRasterBlockLocator(gr, 0, 0, 0, 0, lb, null, 'TRUE');
ln := 1024;
dbms_lob.write(lb, ln, 1, r1);

sdo_geor.getRasterBlockLocator(gr, 0, 0, 0, 0, lb, 'TRUE', 'TRUE');
ln := 128;
dbms_lob.write(lb, ln, 1, r1);
end;
/
```

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 NOCOPY BLOB,
   storageParam    IN VARCHAR2 DEFAULT NULL,
   bgValues        IN SDO_NUMBER_ARRAY 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.

bgValues
Background values for filling sparse data. The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

This parameter is useful when the source has empty raster blocks (see Section 1.4.4). If this parameter is not specified, any cells that are derived from an empty raster block are filled with the value 0 in the output BLOB.

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 1.10.

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 NOCOPY BLOB,
    storageParam     IN VARCHAR2 DEFAULT NULL,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL);
or
SDO_GEOR.getRasterSubset(
    georaster        IN SDO_GEORASTER,
    pyramidLevel     IN NUMBER,
    inWindow         IN SDO_NUMBER_ARRAY,
    bandNumbers      IN VARCHAR2,
    rasterBlob       IN OUT NOCOPY BLOB,
    outWindow        OUT SDO_NUMBER_ARRAY,
    storageParam     IN VARCHAR2 DEFAULT NULL,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL);
or
SDO_GEOR.getRasterSubset(
    georaster        IN SDO_GEORASTER,
    pyramidLevel     IN NUMBER,
    window           IN SDO_GEOMETRY,
    layerNumbers     IN VARCHAR2,
    rasterBlob       IN OUT NOCOPY BLOB,
    storageParam     IN VARCHAR2 DEFAULT NULL,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL,
    polygonClip      IN VARCHAR2 DEFAULT NULL);
or
SDO_GEOR.getRasterSubset(
    georaster        IN SDO_GEORASTER,
    pyramidLevel     IN NUMBER,
    inWindow         IN SDO_GEOMETRY,
```

```
layerNumbers  IN VARCHAR2,
rasterBlob    IN OUT NOCOPY BLOB,
outWindow     OUT SDO_NUMBER_ARRAY,
storageParam  IN VARCHAR2 DEFAULT NULL,
bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
polygonClip   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 either a specified rectangular window or polygon geometry object.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level on which to perform the operation.

window, inWindow
A rectangular window or a polygon geometry object 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 and the `polygonClip` value is `FALSE`, the MBR of the geometry object is used as the window; if the data type is SDO_GEOMETRY and the `polygonClip` value is `TRUE`, the polygon geometry object (if valid) is used as the window. If the data type is SDO_GEOMETRY, see also the Usage Notes for SDO_SRID requirements.

If `window` or `inWindow` is of type SDO_NUMBER_ARRAY, use the `bandNumbers` parameter to specify one or more band numbers; if `window` or `inWindow` 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.

outWindow
An SDO_NUMBER_ARRAY object identifying the coordinates of the upper-left and lower-right corners of the output window in the cell space.

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.

bgValues
Background values for filling sparse data. The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

This parameter is useful when the source has empty raster blocks and the output window intersects any empty raster blocks (see Section 1.4.4). If this parameter is not specified, any cells in the output window that are derived from an empty raster block are filled with the value 0 in the output BLOB.

polygonClip
The string `TRUE` causes the `window` or `inWindow` geometry object to be used for the subset operation; the string `FALSE` or a null value causes the MBR (minimum bounding rectangle) of the `window` or `inWindow` geometry object to be used for the subset operation.

Usage Notes

This procedure has four formats, depending on whether the input window is specified as a geometry object or as the upper-left and lower-right corners of a box, and on whether the `outWindow` parameter is used to return the coordinates of the output window.

If the `window` or `inWindow` 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` or `inWindow` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial Developer's Guide*.

After the procedure completes, the `rasterBLOB` parameter contains the cell (pixel) data in the cropped window without tiling. The cropped window is the overlapping portion of the specified window of interest and the source GeoRaster object's spatial extent. If the `outWindow` parameter is specified, after the procedure completes it contains the coordinates of the cropped window in the cell space.

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.

If `polygonClip` is `TRUE`, and if this procedure creates a rectangular image subset but the geometry is not a rectangle, check the validity of the `inWindow` geometry object with the function SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT. For an invalid geometry, this procedure operates as if the `polygonClip` value is `FALSE` or a null value.

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 1.10.

If you want to get a subset and reproject it to another coordinate system, do not use this procedure, but instead use the SDO_GEOR.reproject procedure using a format that includes the `rasterBlob` parameter, so that this BLOB holds the desired subset.

Examples

The following example retrieves raster data of a specified pyramid level inside a specified window. (It refers to the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr sdo_georaster;
  lb blob;
  win sdo_number_array;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=4;
  dbms_lob.createTemporary(lb, TRUE);
  win := sdo_number_array(-21,100,100,200);
  sdo_geor.getRasterSubset(gr, 0, win, null, lb);
  dbms_lob.freeTemporary(lb);
END;
/
```

The following example demonstrates how to get the window for the cropping. (It refers to the GEORASTER_TABLE table, whose definition is presented after Example 1–1 in Section 1.4.1.)

```
DECLARE
  gr sdo_georaster;
  lb blob;
  win1 sdo_geometry;
  win2 sdo_number_array;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=4;
  dbms_lob.createTemporary(lb, TRUE);
  win1 := sdo_geometry(2003,82263,null,sdo_elem_info_array(1,1003,3),
                       sdo_ordinate_array(1828466,646447,1823400,642512));
  sdo_geor.getRasterSubset(gr, 0, win1, '1-3', lb, win2, 'compression=NONE');
  dbms_lob.freeTemporary(lb);
  IF win2 IS NOT NULL THEN
    dbms_output.put_line('output window: (' || win2(1) || ',' ||
                         win2(2) || ',' || win2(3) || ',' || win2(4) || ')');
  END IF;
END;
```

```
/
```

The following example demonstrates how to do clipping while querying a subset using a polygon.

```
DECLARE
  gr sdo_georaster;
  lb blob;
  win1 sdo_geometry;
  win2 sdo_number_array;
BEGIN
  dbms_lob.createTemporary(lb, TRUE);
  SELECT georaster INTO gr FROM rstpoly_table WHERE georid=1;
  -- querying/clipping polygon
  win1 := sdo_geometry(2003, 26986, null, sdo_elem_info_array(1,1003,1),
                sdo_ordinate_array(237040,   897924,
                                   237013.3, 897831.6,
                                   237129,   897840,
                                   237182.5, 897785.5,
                                   237239.9, 897902.7,
                                   237223,   897954,
                                   237133,   897899,
                                   237040,   897924));
  sdo_geor.getRasterSubset(gr, 0, win1, '1-3',
                           lb, win2, NULL, NULL, 'TRUE');
  -- Then work on the resulting subset stored in 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.getSourceInfo

Format

```
SDO_GEOR.getSourceInfo(
   georaster   IN OUT SDO_GEORASTER,
   ) RETURN SDO_STRING2_ARRAY;
```

Description

Gets the source information for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns the source information stored in the `<sourceInfo>` element in the metadata for the GeoRaster object (described in Appendix A).

The SDO_STRING2_ARRAY type is defined as `VARRAY(2147483647) OF VARCHAR2(4096)`.

To replace or delete source information, use the SDO_GEOR.setSourceInfo procedure. To add source information, use the SDO_GEOR.addSourceInfo procedure.

Examples

The following example sets and adds some source information for a specified GeoRaster object, and then retrieves the information.

```
declare
 gr sdo_georaster;
begin
 select georaster into gr from georaster_table where georid=1 for update;
 sdo_geor.setSourceInfo(gr, 'Copyright (c) 2002, 2007, Oracle Corporation.');
 sdo_geor.addSourceInfo(gr, 'All rights reserved.');
 update georaster_table set georaster=gr where georid=1;
end;
/

select * from table(select sdo_geor.getSourceInfo(georaster) from georaster_table
where id=1);

COLUMN_VALUE
--------------------------------------------------------------------------------
Copyright (c) 2002, 2007, Oracle Corporation.
All rights reserved.
```

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) , NULL,
NULL, NULL, NULL, NULL)
```

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.hasBitmapMask

Format

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

Description

Checks if a GeoRaster object or layer has an associated bitmap mask.

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 the string `TRUE` if the GeoRaster object or layer has an associated bitmap mask, or `FALSE` if it does not have an associated bitmap mask.

For an explanation of bitmap masks, see Section 1.8.

Examples

The following example checks if layers 0 through 4 of a specified GeoRaster object have associated bitmap masks.

```
SELECT substr(sdo_geor.hasBitmapMask(georaster,0),1,12) BM0,
       substr(sdo_geor.hasBitmapMask(georaster,1),1,12) BM1,
       substr(sdo_geor.hasBitmapMask(georaster,2),1,12) BM2,
       substr(sdo_geor.hasBitmapMask(georaster,3),1,12) BM3
  FROM georaster_table WHERE georid=0;
```

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 the string `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.hasNODATAMask

Format

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

Description

Checks if a GeoRaster object or layer has an associated NODATA bitmap mask.

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 the string `TRUE` if the GeoRaster object or layer has an associated NODATA bitmap mask, or `FALSE` if it does not have an associated NODATA bitmap mask.

For an explanation of bitmap masks, see Section 1.8.

Examples

The following example checks if layers 0 through 4 of a specified GeoRaster object have associated NODATA bitmap masks.

```
SELECT substr(sdo_geor.hasNODATAMask(georaster,0),1,12) BM0,
       substr(sdo_geor.hasNODATAMask(georaster,1),1,12) BM1,
       substr(sdo_geor.hasNODATAMask(georaster,2),1,12) BM2,
       substr(sdo_geor.hasNODATAMask(georaster,3),1,12) BM3
  FROM georaster_table WHERE georid=0;
```

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 the string `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.)

- raster: TRUE (the default) causes the raster image data in a GeoTIFF format file to be loaded along with the georeferencing information; FALSE causes only the georeferencing information to be loaded from the GeoTIFF format file, without the raster image data, into an existing GeoRaster object.
- 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.
- srid: Coordinate system SRID numeric value, identifying an optional backup SRID, relevant when loading a GeoTIFF format file. This SRID value is used if the GeoTIFF configuration values do not match any SRID values recognized by Oracle Spatial.

r_sourceFormat
Raster source format. Must be one of the following: TIFF, GIF, BMP, GeoTIFF, or PNG. (JPEG is not supported for this procedure; however, you can use the client-side GeoRaster loader tool, described in Section 1.14, 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 raster data, see Section 3.3.

If you receive an "insufficient memory" error when loading a very large image, see Section 3.3.1.

When loading an image into a GeoRaster database, you should always specify a block size, and it should generally be 256x256 or larger.

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 1.14, to import a JPEG file.

The GeoTIFF `PixelIsArea` raster space is equivalent to the GeoRaster upperleft-based cell coordinate system. An import from GeoTIFF is always to the GeoRaster center-based cell coordinate system, with a half-pixel adjustment of the affine transformation if the GeoTIFF file is specified in `PixelIsArea` raster space.

To load GeoTIFF images with the SDO_GEOR.importFrom procedure, you will need the `xtiff-jai.jar` and `geotiff-jai.jar` libraries. For more information about these GeoTIFF libraries, see Section 3.5.

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',
  '/mydirectory/myimages/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',
   '/mydirectory/myimages/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 must then be registered n the xxx_SDO_GEOR_SYSDATA views (see the Usage Notes).

Parameters

rasterDataTable
Name of the object table of type SDO_RASTER that stores the cell data blocks. Must not contain spaces, period separators, or mixed-case letters in a quoted string; the name is always converted to uppercase when stored in an SDO_GEORASTER object. The RDT should be in the same schema as its associated GeoRaster table. 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

After initializing the empty GeoRaster object and before performing any operations on the object, you must register it in the xxx_SDO_GEOR_SYSDATA views by inserting the empty GeoRaster object into a GeoRaster table. (The xxx_SDO_GEOR_SYSDATA views are described in Section 2.4. GeoRaster operations are described in Chapter 3.)

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.mergeLayers

Format

```
SDO_GEOR.mergeLayers(
   targetGeoRaster        IN OUT SDO_GEORASTER,
   sourceGeoRaster        IN SDO_GEORASTER,
   sourceLayerNumbers IN VARCHAR2 DEFAULT NULL,
   bgValues               IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.mergeLayers(
   source1GeoRaster        IN SDO_GEORASTER,
   source1LayerNumbers  IN VARCHAR2,
   source2GeoRaster        IN SDO_GEORASTER,
   source2LayerNumbers  IN VARCHAR2,
   storageParam            IN VARCHAR2,
   outGeoRaster             IN OUT SDO_GEORASTER,
   bgValues                IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Merges the layers of two GeoRaster objects, either by appending source layers to a target GeoRaster object (first format) or by performing a union operation (second format).

Parameters

targetGeoRaster
GeoRaster object to which layers in `sourceGeoRaster` are to be appended. Cannot be the same GeoRaster object as `sourceGeoRaster`. (Be sure to make a copy of this object before calling this procedure.)

sourceGeoRaster
GeoRaster object in which specified layers are to be appended to `targetGeoRaster`.

sourceLayerNumbers
String specifying one or more layer numbers of layers in `sourceGeoRaster` to be appended to `targetGeoRaster`. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'1,3-5,7'` for layers 1, 3, 4, 5, and 7.

source1GeoRaster
One GeoRaster object in which specified layers are to be joined in a union operation with layers from `source2GeoRaster` in the output GeoRaster object `outGeoRaster`.

source1LayerNumbers
String specifying one or more layer numbers of layers in `source1GeoRaster` to be joined in a union operation with layers from `source2GeoRaster` in the output

GeoRaster object `outGeoRaster`. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'1,3-5,7'` for layers 1, 3, 4, 5, and 7.

source2GeoRaster
One GeoRaster object in which specified layers are to be joined in a union operation with layers from `source1GeoRaster` in the output GeoRaster object `outGeoRaster`.

source2LayerNumbers
String specifying one or more layer numbers of layers in `source2GeoRaster` to be joined in a union operation with layers from `source1GeoRaster` in the output GeoRaster object `outGeoRaster`. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'1,3-5,7'` for layers 1, 3, 4, 5, and 7.

storageParam
A string specifying storage parameters to be applied in creating `outGeoRaster`. Storage parameters are explained in Section 1.4.1.

outGeoRaster
The new SDO_GEORASTER object that reflects the results of the union operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as `source1GeoRaster` or source2GeoRaster.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

Usage Notes

> **Note:** Be sure to make a copy of the `targetGeoRaster` object before you call this procedure, because the changes made to this GeoRaster object might not be reversible after the procedure completes.

The resulting GeoRaster object (`georaster` or `outGeoRaster` parameter) must not be the same GeoRaster object as `sourceGeoRaster`, `source1GeoRaster`, or `source2GeoRaster`.

The two GeoRaster objects to be appended or unioned together must have the same spatial dimension sizes and cover the same area. If one of the GeoRaster objects is georeferenced, the other one must also be georeferenced, have the same model SRID and spatial resolutions, and cover the same area in the model space. If neither GeoRaster object is georeferenced, their ultCoordinates must be the same.

Examples

The following example merges specified layers of two GeoRaster objects into a third GeoRaster object, by performing a union operation.

```
declare
 gr1 sdo_georaster;
 gr2 sdo_georaster;
 gr3 sdo_georaster;
begin
 select georaster into gr1 from georaster_table where georid=1;
 select georaster into gr2 from georaster_table where georid=2;
 insert into georaster_table(georid, georaster) values (3, sdo_geor.init('RDT_1'))
    returning georaster into gr3;
 sdo_geor.mergeLayers(gr1, '3', gr2, '2,1', 'blocking=false', gr3);
 update georaster_table set georaster=gr3 where georid=3;
 commit;
end;
/
```

SDO_GEOR.mosaic

Format

```
SDO_GEOR.mosaic(
   georasterTableName     IN VARCHAR2,
   georasterColumnName  IN VARCHAR2,
   georaster                  IN OUT SDO_GEORASTER,
   storageParam             IN VARCHAR2,
   bgValues                   IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Mosaics GeoRaster objects into one GeoRaster object.

Parameters

georasterTableName
Name of the table or view containing all source GeoRaster objects.

georasterColumnName
Column of type SDO_GEORASTER in `georasterTableName`.

georaster
GeoRaster object to hold the result of the mosaic operation. Cannot be the same as any GeoRaster object in `georasterColumnName` in `georasterTableName`.

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. However, it is recommended that you specify the storage parameters, particularly the blocking size, as appropriate for the size of the output mosaic, unless you want the mosaic to have the same storage parameters as those of the upper-left corner GeoRaster object to be mosaicked.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the current operation leads to partially empty raster blocks (see Section 1.4.4), which could happen when the source GeoRaster objects have empty raster blocks or when the source GeoRaster objects do not cover the whole area. The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

Usage Notes

The source GeoRaster objects must be prepared images or raster data so that they can be mosaicked directly. The GeoRaster objects to be mosaicked must:

- Not be a mixture of georeferenced and nongeoreferenced objects. Either all of the objects are georeferenced, or none of the objects is georeferenced.
- Have the same SRID value if the objects are georeferenced, and the georeferencing method must be affine transformation. The affine transformations of the GeoRaster objects must have the same set of coefficients (A, B, D and E) or (b, c, e, f). This means that the images must have the same X resolution and Y resolution (although the X and Y resolutions do not have to be the same), the same rotation angle, and the same skewing factor; in other words, the images must have the same resolutions, and be rotated and skewed in the same way if the images are rotated and skewed.
- Have the same number of layers or bands. There is no restriction on the row and column dimension sizes of the source objects; for example, they do not need to be a power of 2.
- Have the same mapping between band number and layers.

If the GeoRaster objects to be mosaicked are georeferenced, they are co-located according to their georeferencing information. If the GeoRaster objects are not georeferenced, they are co-located according to their ULTCoordinate values. (The ULTCoordinate is explained in Section 1.3.)

If applicable, the resulting GeoRaster object takes the spatial reference metadata information from the upper-left corner source GeoRaster object in the model space. It also takes the cell space and any default storage attributes from the upper-left corner source GeoRaster object in the model space.

If the source GeoRaster objects have empty raster blocks or do not cover the whole area, the mosaicked result GeoRaster object may have empty or partially empty raster blocks (see Section 1.4.4). A result raster block that is not covered by any of the source GeoRaster objects is kept empty. Any partially empty raster blocks are filled with the values specified in the `bgValues` parameter, or with 0 if the `bgValues` parameter is not specified.

If the source GeoRaster objects overlap, data of the overlapping area comes from the source object that covers it and that has the largest `ultCoordinate` in the cell space where all the source objects are co-located.

Any bitmap masks associated with the source GeoRaster objects are not considered, and the `bitmapmask` parameter is ignored if it is specified in the `storageParam` string.

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.

The GeoRaster object to contain the results of the mosaic operation (`georaster` parameter) must not be any of the source GeoRaster objects (the objects on which the mosaic operation is performed).

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 by

maintaining GeoRaster objects and system data in the database, 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.reproject

Format

```
SDO_GEOR.reproject(
    inGeoRaster        IN SDO_GEORASTER,
    resamplingParam IN VARCHAR2,
    storageParam      IN VARCHAR2,
    outSRID             IN NUMBER,
    outGeoraster       IN OUT SDO_GEORASTER,
    bgValues            IN SDO_NUMBER_ARRAY DEFAULT NULL);
or
SDO_GEOR.reproject(
    inGeoRaster        IN SDO_GEORASTER,
    pyramidLevel       IN NUMBER,
    cropArea            IN SDO_GEOMETRY,
    layerNumbers       IN VARCHAR2,
    resamplingParam IN VARCHAR2,
    storageParam      IN VARCHAR2,
    outSRID             IN NUMBER,
    outGeoraster       IN OUT SDO_GEORASTER,
    bgValues            IN SDO_NUMBER_ARRAY DEFAULT NULL);
or
SDO_GEOR.reproject(
    inGeoRaster        IN SDO_GEORASTER,
    pyramidLevel       IN NUMBER,
    cropArea            IN SDO_NUMBER_ARRAY,
    bandNumbers       IN VARCHAR2,
    resamplingParam IN VARCHAR2,
    storageParam      IN VARCHAR2,
    outSRID             IN NUMBER,
    outGeoraster       IN OUT SDO_GEORASTER,
    bgValues            IN SDO_NUMBER_ARRAY DEFAULT NULL);
or
SDO_GEOR.reproject(
    inGeoRaster        IN SDO_GEORASTER,
    pyramidLevel       IN NUMBER,
```

```
    cropArea            IN SDO_GEOMETRY,
    layerNumbers        IN VARCHAR2,
    resamplingParam IN VARCHAR2,
    storageParam        IN VARCHAR2,
    outSRID             IN NUMBER,
    rasterBlob          IN OUT NOCOPY BLOB,
    outArea            OUT SDO_GEOMETRY,
    outWindow          OUT SDO_NUMBER_ARRAY,
    bgValues            IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.reproject(
    inGeoRaster         IN SDO_GEORASTER,
    pyramidLevel        IN NUMBER,
    cropArea            IN SDO_NUMBER_ARRAY,
    bandNumbers         IN VARCHAR2,
    resamplingParam IN VARCHAR2,
    storageParam        IN VARCHAR2,
    outSRID             IN NUMBER,
    rasterBlob          IN OUT NOCOPY BLOB,
    outArea            OUT SDO_GEOMETRY,
    outWindow          OUT SDO_NUMBER_ARRAY,
    bgValues            IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Reprojects all or part of a GeoRaster object to a different Oracle Spatial coordinate system (specified by the outSRID parameter). The resulting object can be a new GeoRaster object (for persistent storage) or a BLOB (for temporary use).

Parameters

inGeoRaster
The SDO_GEORASTER object on which the reprojection operation is to be performed to create the new object.

pyramidLevel
A number specifying the pyramid level of the source GeoRaster object.

cropArea
Crop area definition. 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.

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.

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).

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).

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

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. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as `inGeoRaster`.

rasterBlob
BLOB to hold the output reflecting the new coordinate system. It must exist or have been initialized before the reprojection operation.

outArea
An SDO_GEOMETRY object containing the MBR (minimum bounding rectangle) in the model coordinate system of the resulting object.

outWindow
An SDO_NUMBER_ARRAY object identifying the coordinates of the upper-left and lower-right corners of the output window in the cell space.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

Usage Notes

This procedure has two general kinds of interfaces:

- The first three formats generate a persistent GeoRaster object for storage in the database.

- The remaining formats generate a BLOB for temporary storage for immediate use, such as to display data on the screen.

`inGeoRaster` should be georeferenced and have a SRID value from the SRID column of the MDSYS.CS_SRS table. `outSRID` should be different from the SRID of `inGeoRaster`. In some cases, the reprojection is inappropriate, such as reprojecting a GeoRaster object in NAD83, Massachusetts Mainland (SRID = 26986) to coordinate system NAD 27, UTM zone 49N (SRID = 2032649). In this case, the reprojection would result in a large distortion and thus is not performed.

`inGeoRaster` and `outGeoRaster` must be different GeoRaster objects. After the operation, the ULT coordinates of the resulting GeoRaster object are set to zero (0).

If the source or destination object has a three-dimensional coordinate system, the height (Z) values are set to zero (0).

If you use the format that includes the `pyramidLevel` parameter and you specify a value greater than zero (0), the reprojection is based on the specified pyramid level of the source GeoRaster object; otherwise, the reprojection is based on the original GeoRaster object (pyramidLevel=0). The output GeoRaster object has no pyramid data.

If the `cropArea` parameter data type is SDO_GEOMETRY, its SDO_SRID value must be a value from the SRID column of the MDSYS.CS_SRS table. If the SDO_SRID values for the `cropArea` parameter geometry and the `inGeoraster` object model space are different, the `cropArea` 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 `cropArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian. (For information about geodetic MBRs, see *Oracle Spatial Developer's Guide*.) Only the overlapping portion of the specified crop area and the spatial extent of the source GeoRaster object is reprojected.

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

- `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).
- `nodata` (for example, `nodata=TRUE`): Specifies whether NODATA values and value ranges should be considered during the procedure. Must be either `TRUE` (NODATA values and value ranges should be considered) or `FALSE` (NODATA values and value ranges should not be considered). The default value is `FALSE`. If the value is `TRUE` and the resampling method is `BILINEAR`, `AVERAGE4`, `AVERAGE16`, or `CUBIC`, whenever a cell value involved in the resampling calculation is a NODATA value, the result of the resampling is also a NODATA value. The resulting NODATA value is the minimum NODATA value associated with the current raster layer, if multiple NODATA values or value ranges exist.

Examples

The following example reprojects a GeoRaster object into the coordinate system defined by SRID 32618. The result is another GeoRaster object.

```
DECLARE
```

```
   gr1 sdo_georaster;
   gr2 sdo_georaster;
BEGIN
  SELECT georaster INTO gr1 from georaster_table WHERE georid=10;
  INSERT INTO reproject_table VALUES (21, 'WGS 84 / UTM zone 18N',
                                      SDO_GEOR.init('rdt_5', 21))
  RETURNING georaster INTO gr2;
  sdo_geor.Reproject(gr1, SDO_NUMBER_ARRAY(0, 0, 517, 517),
                                  null, null, 'blocking=true,
blocksize=(256,256,3),
                                  interleaving=BSQ', 32618, gr2);
   UPDATE georaster_table SET georaster=gr2 WHERE georid=21;
   COMMIT;
END;
/
```

The following example reprojects a GeoRaster object into the coordinate system defined by SRID 32618. The result is temporary BLOB containing data in JPEG-F format.

```
DECLARE
   gr1 sdo_georaster;
   lob1 BLOB;
   outArea SDO_Geometry;
   outWindow SDO_NUMBER_ARRAY;
BEGIN
   SELECT georaster INTO gr1 from georaster_table WHERE georid=10;
   dbms_lob.createTemporary(lob1, TRUE);
   sdo_geor.Reproject(gr1,
                                  SDO_NUMBER_ARRAY(0, 0, 120, 300),
                                   '0',  null, 'compression = JPEG-F', 32618,
                                   lob1, outArea, outWindow);

   dbms_lob.freeTemporary(lob1);
   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,
   bgValues          IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.scaleCopy(
   inGeoRaster       IN SDO_GEORASTER,
   pyramidLevel      IN NUMBER,
   scaleParam        IN VARCHAR2,
   resampleParam  IN VARCHAR2,
   storageParam     IN VARCHAR2,
   outGeoRaster      IN OUT SDO_GEORASTER,
   bgValues          IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

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`).

pyramidLevel
A number specifying the pyramid level of the source GeoRaster object.

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 containing the resampling parameters. See the Usage Notes for information about the available keywords and values.

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. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as `inGeoRaster`.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

Usage Notes

Use this procedure to create a new GeoRaster object reflecting the specified scaling, based on the original GeoRaster object or a specified pyramid level of the GeoRaster 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.

If you use the format that does not include the `pyramidLevel` parameter, the scaling is based on the original GeoRaster object (pyramidLevel=0).

If you need to get the scaled cell values, use the procedure described in the Usage Notes for the SDO_GEOR.getCellValue function.

`inGeoRaster` and `outGeoRaster` must be different GeoRaster objects.

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

- `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).
- `nodata` (for example, `nodata=TRUE`): Specifies whether NODATA values and value ranges should be considered during the procedure. Must be either `TRUE` (NODATA values and value ranges should be considered) or `FALSE` (NODATA values and value ranges should not be considered). The default value is `FALSE`. If

the value is TRUE and the resampling method is BILINEAR, AVERAGE4, AVERAGE16, or CUBIC, whenever a cell value involved in the resampling calculation is a NODATA value, the result of the resampling is also a NODATA value. The resulting NODATA value is the minimum NODATA value associated with the current raster layer, if multiple NODATA values or value ranges exist.

Any upper-level pyramid data in the input GeoRaster object is not considered during this operation, and the output GeoRaster object has no pyramid data.

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).

This procedure does not scale along the band dimension.

If the source GeoRaster object is georeferenced with a valid polynomial transformation, the georeferencing information for the resulting GeoRaster object is generated accordingly; otherwise, the result GeoRaster object contains no spatial reference information.

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 the string `TRUE` if the metadata is valid, a null value 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.setBinFunction

Format

```
SDO_GEOR.setBinFunction(
   georaster      IN SDO_GEORASTER,
   layerNumber    IN NUMBER
   binFunction    IN SDO_NUMBER_ARRAY);
```

Description

Sets the bin function 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.

binFunction
Bin function as an array whose elements specify the bin type, total number of bins, first bin number, minimum cell value, and maximum cell value. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`. See the Usage Notes for more information and an example.

Usage Notes

A bin function maps values or value ranges of the GeoRaster cells to specific bin numbers, which are all integers. If a bin function of type `LINEAR` is defined, it is used by the SDO_GEOR.generateStatistics function for calculating statistics on cell values. GeoRaster does not provide interfaces to manipulate and process bin functions.

The `binFunction` parameter specifies an array of five numbers, which have the following meaning:

- The first number identifies the bin type, and must be 0 (`LINEAR`) or 1 (`LOGARIGHM`).
- The second number identifies the total number of bins.
- The third number identifies the number of the first bin.
- The fourth number is the minimum cell value in the range.
- The fifth number is the maximum cell value in the range.

For example, if `binFunction` is `SDO_NUMBER_ARRAY(0,10,1,0,511)`, the bin type is `LINEAR`, there are 10 bins numbered 1 through 10 (that is, starting at 1), and cell values from 0 through 511 are uniformly distributed to bins 1 through 10.

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

Examples

The following example sets the bin function for layer 3 of a specified GeoRaster object, using the `binFunction` parameter value explained in the Usage Notes.

```
DECLARE
 gr sdo_georaster;
BEGIN
 SELECT georaster INTO gr FROM georaster_table WHERE georid=4 FOR UPDATE;
 sdo_geor.setBinFunction(gr, 3, sdo_number_array(0,10,1,0,511));
 UPDATE georaster_table SET georaster=gr WHERE georid=4;
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 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.setBitmapMask

Format

```
SDO_GEOR.setBitmapMask(
   georaster       IN OUT SDO_GEORASTER,
   layerNumber     IN NUMBER,
   mask            IN SDO_GEORASTER,
   NODATAMask  IN VARCHAR2 DEFAULT NULL);
```

Description

Attaches a bitmap mask to a GeoRaster object, or replaces or removes the current bitmap mask.

Parameters

georaster
GeoRaster object.

layerNumber
Layer with which to associate the bitmap mask. A value of 0 (zero) indicates the object layer.

mask
The bitmap mask to be attached to the GeoRaster object. If this parameter is null, any existing bitmap mask associated with the specified layer of the GeoRaster object is removed.

NODATAMask
Specifies 0 (zero, the default) if `mask` is not to be interpreted as a NODATA mask, or1 if `mask` is to be interpreted as a NODATA mask.

Usage Notes

If the specified GeoRaster object or layer already has an associated bitmap mask, this procedure replaces it with the specified mask, or removes it if the `mask` parameter specifies a null value.

If the `mask` parameter value is not null, the bitmap mask must be a valid 1BIT GeoRaster object with only one band, and the row and column dimension sizes of the bitmap mask must be the same as those of the target GeoRaster object. It cannot be the same GeoRaster object as the input GeoRaster object (`georaster` parameter).

This procedure raises an exception if the target GeoRaster object is a blank GeoRaster object.

For an explanation of bitmap masks, see Section 1.8.

Examples

The following example uses a GeoRaster object in the table GRTAB (where id=1) as the bitmap mask to be attached to layer number 1 of an GeoRaster object in the GEORASTER_TABLE (where georid=1). The bitmap mask will be interpreted as a NODATA mask.

```
declare
  gr sdo_georaster;
  mk sdo_georaster;
begin
  select georaster into gr from georaster_table where georid=1 for update;
  select grobj into mk from grtab where id=1;
  sdo_geor.setBitmapMask(gr, 1, mk, 'true');
  update georaster_table set georaster=gr where georid=0;
  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.setControlPoint

Format

```
SDO_GEOR.setControlPoint (
   inGeoraster  IN OUT SDO_GEORASTER,
   controlPoint  IN SDO_GEOR_GCP);
```

Description

Adds a ground control point (GCP) for the GeoRaster object, or replaces an existing GCP if it has the same ID value as the input control point.

Parameters

inGeoraster
GeoRaster object.

controlPoint
GCP to be added for `inGeoraster`. Must be an object of type SDO_GEOR_GCP, which is described in Section 2.3.6.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

If the `controlPoint` is null, the function returns without performing any action. If a GCP is found in the GeoRaster object metadata with the same point ID as defined in `controlPoint`, that GCP is replaced; otherwise, this GCP is added to the georeferencing model.

Examples

The following example adds a GCP for a specified GeoRaster object.

```
DECLARE
   gr1            sdo_georaster;
   GCP         SDO_GEOR_GCP;
BEGIN
   SELECT georaster INTO gr1 from georaster_table WHERE georid=10 FOR UPDATE;

   GCP := SDO_GEOR_GCP('21', 'Updated', 1,
                       2, sdo_number_array(25.625000, 73.875000),
                       2, sdo_number_array(237036.937500, 897987.187500),
                       NULL, NULL);
   sdo_geor.setControlPoint(gr1, GCP);
   UPDATE georaster_table SET georaster=gr1 WHERE georid=10;
   COMMIT;
END;
/
```

SDO_GEOR.setDefaultBlue

Format

```
SDO_GEOR.setDefaultBlue(
   georaster    IN OUT SDO_GEORASTER,
   defaultBlue  IN 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  IN 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.

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.

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.

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  IN 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    IN 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.setGCPGeorefMethod

Format

```
SDO_GEOR.setGCPGeorefMethod(
   inGeoraster         IN OUT SDO_GEORASTER
   gcpGeorefMethod IN VARCHAR2);
```

Description

Sets the GCP-based georeferencing geometric model type of a GeoRaster object.

Parameters

inGeoraster
GeoRaster object.

gcpGeorefMethod
Georeferencing geometric model type to set for the GeoRaster object. Its value must be one of following strings: `Affine`, `QuadraticPolynomial`, `CubicPolynomial`, `DLT`, `QuadraticRational`, or `RPC`.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

If `inGeoraster` does not contain GCP-based georeferencing information, no action is performed; otherwise, the existing model type is replaced with the specified `gcpGeorefMethod` value.

The procedure just set the model type value; no new solution is calculated. To get the solution for the newly set model type, use the SDO_GEOR.georeference function.

Examples

The following example sets the GCP-based georeferencing geometric model type of a specified GeoRaster object, and updates the object.

```
DECLARE
   gr1  sdo_georaster;
BEGIN
   SELECT georaster INTO gr1 from georaster_table WHERE georid=10 FOR UPDATE;
   sdo_geor.setGCPGeorefMethod(gr1, 'DLT');
   UPDATE georaster_table SET georaster=gr1 WHERE georid=10;
   COMMIT;
END;
/
```

SDO_GEOR.setGCPGeorefModel

Format

```
SDO_GEOR.setGCPGeorefModel(
   inGeoraster        IN OUT SDO_GEORASTER
   gcpGeorefModel  IN SDO_GEOR_GCPGEOREFTYPE);
```

Description

Sets the GCP-based georeferencing model information for a GeoRaster object.

Parameters

inGeoraster
GeoRaster object.

gcpGeorefModel
Object containing the following: `FFMethodType`, `nGCP`, `GCPs`, `solutionAccuracy`.

Usage Notes

For an explanation of georeferencing using GCPs, see Section 1.6.2.

The SDO_GEOR_GCPGEOREFTYPE object type is defined in Section 2.3.8.

This procedure stores the GCP information in the GeoRaster SRS metadata component. If `gcpGeorefModel` is null and if the GeoRaster object has a georeferencing model, this model information will be deleted.

If there are not enough GCPs specified in `gcpGeorefModel` for the geometric model specified, the function will still succeed, but an exception will be raised if the SDO_GEOR.georeference is called specifying this GeoRaster object.

Examples

The following example sets the GCP-based georeferencing model information in a specified GeoRaster object.

```
DECLARE
   gr1                 sdo_georaster;
   georefModel   SDO_GEOR_GCPGEOREFTYPE;
   GCPs              SDO_GEOR_GCP_COLLECTION;
   rms                 sdo_number_array;
BEGIN
   SELECT georaster INTO gr1 from herman.georaster_table WHERE georid=10 FOR
UPDATE;

   GCPs:=SDO_GEOR_GCP_COLLECTION(
                   SDO_GEOR_GCP('21', '', 1,
                           2, sdo_number_array(25.625000, 73.875000),
                           2, sdo_number_array(237036.937500, 897987.187500),
                           NULL, NULL),
                    SDO_GEOR_GCP('22', '', 1,
                           2, sdo_number_array(100.625000, 459.125000),
                           2, sdo_number_array(237229.562500, 897949.687500),
                           NULL, NULL),
                     SDO_GEOR_GCP('23', '', 1,
```

```
                              2, sdo_number_array(362.375000, 77.875000),
                              2, sdo_number_array(237038.937500, 897818.812500),
                                  NULL, NULL),
                          SDO_GEOR_GCP('24', '', 1,
                                2, sdo_number_array(478.875000, 402.125000),
                                2, sdo_number_array(237201.062500, 897760.562500),
                                NULL, NULL),
                           SDO_GEOR_GCP('25', '', 2,
                                2, sdo_number_array(167.470583,  64.030686),
                                2, sdo_number_array(237032.015343, 897916.264708),
                                NULL, NULL),
                            SDO_GEOR_GCP('26', '', 2,
                                2, sdo_number_array(101.456177,  257.915534),
                                2, sdo_number_array(237128.957767, 897949.271912),
                                NULL, NULL)
                );

   georefModel := SDO_GEOR_GCPGEOREFTYPE('Affine',

GCPs.count, GCPs, rms);
   sdo_geor.setGCPGeorefModel(gr1, georefModel);
   UPDATE georaster_table SET georaster=gr1 WHERE georid=10;

   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 1–5 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.setModelCoordLocation

Format

```
SDO_GEOR.setModelCoordLocation(
   georaster          IN OUT SDO_GEORASTER
   modelCoordLoc  IN VARCHAR2);
```

Description

Sets the model coordinate location value for a GeoRaster object, or deletes the current model coordinate location value (if any) if the `modelCoordLoc` parameter is specified as null.

Parameters

georaster
GeoRaster object.

modelCoordLoc
Model coordinate location to set for the GeoRaster object. It must be specified as either null (to delete any current model coordinate location value) or one of the following string values: `CENTER` (the cell coordinate system is center-based) or `UPPERLEFT` (the cell coordinate system is based on the upper-left corner).

Usage Notes

This procedure enables you to change the cell coordinate system from CENTER to UPPERLEFT or from UPPERLEFT to CENTER.

This procedure applies only to georeferenced GeoRaster objects, and it automatically adjusts the functional fitting coefficients of the GeoRaster SRS accordingly to reflect the change (to ensure that the relationship between cell coordinates and model coordinates does not change).

To get the model coordinate location value for a GeoRaster object, use the SDO_GEOR.getModelCoordLocation function.

For an explanation of georeferencing using GCPs, see Section 1.6.2.

Examples

The following example changes the cell coordinate system to CENTER for a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=4 FOR UPDATE;
  sdo_geor.setModelCoordLocation(grobj, 'CENTER');
  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 and the GeoRaster metadata does not contain spatial reference information.

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. The `srid` value cannot be 0 (zero).

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:

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

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.setSourceInfo

Format

```
SDO_GEOR.setSourceInfo(
   georaster   IN OUT SDO_GEORASTER,
   sourceInfo  IN VARCHAR2);
```

Description

Sets the source information for a GeoRaster object, or deletes the existing value if you specify a null `sourceInfo` parameter.

Parameters

georaster
GeoRaster object.

sourceInfo
String with source information. Cannot exceed 4096 characters.

Usage Notes

The specified `sourceInfo` string is stored in the `<sourceInfo>` element in the metadata for the GeoRaster object (described in Appendix A).

This procedure replaces any existing source information value or values. If you want to keep any existing values and add one or more values, use the SDO_GEOR.addSourceInfo procedure.

Examples

The following example sets and adds some source information for a specified GeoRaster object, and then retrieves the information.

```
declare
 gr sdo_georaster;
begin
 select georaster into gr from georaster_table where georid=1 for update;
 sdo_geor.setSourceInfo(gr, 'Copyright (c) 2002, 2007, Oracle Corporation.');
 sdo_geor.addSourceInfo(gr, 'All rights reserved.');
 update georaster_table set georaster=gr where georid=1;
end;
/

select * from table(select sdo_geor.getSourceInfo(georaster) from georaster_table
where id=1);

COLUMN_VALUE
--------------------------------------------------------------------------------
Copyright (c) 2002, 2007, Oracle Corporation.
All rights reserved.
```

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. The SDO_GEOR_SRS object type and its constructor are 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

You can use this procedure to set the GeoRaster SRS for any functional fitting georeferencing models, including the affine transformation, DLT, and RPC models.

For the stored function (GCP) model only, you may find it more convenient not to use this procedure, but instead to use the SDO_GEOR.setGCPGeorefModel procedure to set the stored function (GCP) model.

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 examples specify spatial reference attributes of a GeoRaster object, and updates the GeoRaster object. (They refer to a table named GEORASTER_TABLE, whose definition is presented after Example 1–1 in Section 1.4.1.) Notes explain the operations in more detail.

The first example shows how to set an affine transformation model to a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
  srs   sdo_geor_srs;

BEGIN

SELECT georaster INTO grobj FROM georaster_table WHERE georid=4;
```

```
srs := sdo_geor_srs('TRUE', 'TRUE', null, 82262,
                        sdo_number_array(28.5, 28.5),0.5,0,
                        0,0,0,0,0,1,1,1,1,1,0,0,0,
                        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, .03508772, 0),
                        SDO_NUMBER_ARRAY(1, 0, 0, 1, 1));
sdo_geor.setSRS(grobj, srs);

UPDATE georaster_table SET georaster = grobj WHERE georid=4;
COMMIT;
END;
/
```

In the preceding example, the GeoRaster object has the following affine transformation:

```
row = 32631.5614 + 0 * x + (-0.03508772) * y
col = -7894.7544 + 0.03508772 * x + 0 * y
```

To use the generic functional fitting georeferencing model described in Section 1.6.1, the values of SRS attributes are as follows:

```
xOff=yOff=zOff=0
rowOff=columnOff=0
xScale=yScale=zScale=1
rowScale=columnScale=1
polynomial p : pType=1, nVars=2, order=1, nCoefficients= 3
polynomial q : pType=1, nVars=0, order=0, nCoefficients= 1
polynomial r : pType=1, nVars=2, order=1, nCoefficients= 3
polynomial s : pType=1, nVars=0, order=0, nCoefficients= 1

rowNumerator = 32631.5614, 0, -0.03508772
rowDenominator =  1
columnNumerator = -7894.7544, 0.03508772, 0
columnDenominator = 1
```

In the SRS structure, the `rowNumerator`, `rowDenominator`, `columnNumerator`, and `columnDenominator` elements are used to specify `pType`, `nVars`, `order`, and `nCoefficients`, and the remaining elements are used to specify coefficients of each polynomial.

The second example shows how to set a DLT model to a GeoRaster object. In a typical photogrammetry application, the interior orientation parameters and exterior orientation parameters of an oriented digital aerial photo can be used to derive a DLT model, which is widely used to simplify and approximate the rigorous model. The following is an example of a DLT model derived from a standard frame camera model.

```
row = (-46507111.2127784 + 65.81484127*X + 13.13186856*Y - 49.62133265*Z) /
(-41.47013322 + 0.00004128*X + 0.00009740*Y - 0.00655704*Z)

col = (-5259855.00453679 - 12.07452653*X + 66.23319061*Y - 49.45792766*Z) /
(-41.47013322 + 0.00004128*X + 0.00009740*Y - 0.00655704*Z)
```

For this example, the corresponding GeoRaster SRS parameters and coefficients are:

```
rowOff=0, colOff=0; rowScale = colScale = 1;
xOff = 0, yOff = 0, zOff = 0; xScale = yScale = zScale =1;
polynomial p : pType=1, nVars=3, order=1, nCoefficients= 4
polynomial q : pType=1, nVars=3, order=1, nCoefficients= 4
polynomial r : pType=1, nVars=3, order=1, nCoefficients= 4
```

```
polynomial s : pType=1, nVars=3, order=1, nCoefficients= 4

rowNumerator = -5259855.00453679, -12.07452653, 66.23319061, -49.45792766
rowDenominator = -41.47013322, 0.00004128, 0.00009740, -0.00655704
columnNumerator = -46507111.2127784, 65.81484127, 13.13186856, -49.62133265
columnDenominator = -41.47013322, 0.00004128, 0.00009740, -0.00655704
```

The following sets up the DLT model for the GeoRaster object. It uses the SDO_GEOR_SRS constructor to create an instance of the SDO_GEOR_SRS type, and then sets the appropriate attributes.

```
DECLARE
   grobj sdo_georaster;
   srs sdo_geor_srs;
BEGIN
   SELECT georaster INTO grobj FROM georaster_table WHERE georid = 101;

   -- Initiate the SRS object.
   srs:=sdo_geor_srs();

   -- Fill the parameters of the SRS object.
   srs.isReferenced := 'TRUE';
   srs.isRectified := 'FALSE';
   srs.isOrthoRectified := 'FALSE';
   srs.srid := 7406;
   srs.spatialResolution := sdo_number_array(0.07, 0.07);
   srs.coordLocation := 1;
   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, 3, 1, 4, -5259855.00453679,
     -12.07452653,66.23319061,-49.45792766);
   srs.rowDenominator := SDO_NUMBER_ARRAY(1, 3, 1, 4, -41.4701332195,
     0.0000412763, 0.0000974018, -0.0065570398);
   srs.columnNumerator := SDO_NUMBER_ARRAY(1, 3, 1, 4,
     -46507111.2127784, 65.81484127, 13.13186856, -49.62133265);
   srs.columnDenominator := SDO_NUMBER_ARRAY(1, 3, 1, 4,
     -41.4701332195, 0.0000412763, 0.0000974018, -0.0065570398);

   -- Set the SRS metadata to the GeoRaster object.
   sdo_geor.setSRS(grobj,srs);
   UPDATE georaster_table SET georaster = grobj WHERE georid= 101;
   COMMIT;

END;
/
```

SDO_GEOR.setStatistics

Format

```
SDO_GEOR.setStatistics(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   statistics        IN SDO_NUMBER_ARRAY);
```

or

```
SDO_GEOR.setStatistics(
   georaster          IN OUT SDO_GEORASTER,
   layerNumber      IN NUMBER,
   statistics          IN SDO_NUMBER_ARRAY,
   histogram         IN SDO_GEOR_HISTOGRAM,
   samplingFactor   IN NUMBER DEFAULT 1,
   samplingWindow IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.setStatistics(
   georaster          IN OUT SDO_GEORASTER,
   layerNumber      IN NUMBER,
   statistics          IN SDO_NUMBER_ARRAY,
   histogram         IN SDO_GEOR_HISTOGRAM,
   samplingFactor   IN NUMBER DEFAULT 1,
   samplingWindow IN SDO_GEOMETRY DEFAULT NULL);
```

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. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`.

If this parameter is null, all statistical information associated with the layer is deleted.

histogram
Histogram of type SDO_GEOR_HISTOGRAM. Section 2.3.1 describes this object type and briefly discusses histograms.

samplingFactor
Sampling factor. The denominator n in $1/n$, representing the number of cells sampled in computing the statistics. For example, if `samplingFactor` is 4, one-fourth of the cells were sampled. The default is 1; that is, all cells were sampled. The higher the value, the less accurate the statistics are likely to be, but the more quickly they were computed.

samplingWindow
Sampling window: a rectangular window for which to set statistics, specified either as a numeric array with the lower-left and upper-right coordinates or as an SDO_GEOMETRY object. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`. The window must be inside the extent in cell space. The default for this parameter is the entire image.

Usage Notes

This procedure sets statistical data described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in Appendix A.

If `histogram` is specified as null, and if there is an existing histogram and you set the statistics using a different sampling factor or sampling window, the existing histogram is removed.

Contrast this procedure, in which you specify the statistics to be set, with the SDO_GEOR.generateStatistics function, which causes GeoRaster to compute and set the statistics.

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 or adjusts the cell coordinate values of the upper-left corner of a GeoRaster object.

Parameters

georaster
GeoRaster object.

ultCoord
An array of two numbers (row and column ordinates) if the rasterType value is 20001, or three numbers (row, column, and band ordinates) if the rasterType value is 21001. If you specify three numbers, the third one (band number) must be 0. For more information about the ULTCoordinate, see Section 1.3.

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, with logic to handle whether the rasterType value is 20001 or 21001. (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=1 FOR UPDATE;
 if grobj.rasterType = 20001 then
      sdo_geor.setULTCoordinate(grobj, sdo_number_array(0, 0));
 elsif grobj.rasterType = 21001 then
      sdo_geor.setULTCoordinate(grobj, sdo_number_array(0, 0, 0));
 end if;
 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,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL,
    polygonClip      IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.subset(
    inGeoRaster      IN SDO_GEORASTER,
    pyramidLevel     IN NUMBER,
    cropArea         IN SDO_GEOMETRY,
    layerNumbers     IN VARCHAR2,
    storageParam     IN VARCHAR2,
    outGeoRaster     IN OUT SDO_GEORASTER,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL,
    polygonClip      IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.subset(
    inGeoRaster      IN SDO_GEORASTER,
    cropArea         IN SDO_NUMBER_ARRAY,
    bandNumbers      IN VARCHAR2,
    storageParam     IN VARCHAR2,
    outGeoRaster     IN OUT SDO_GEORASTER,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.subset(
    inGeoRaster      IN SDO_GEORASTER,
    pyramidLevel     IN NUMBER,
    cropArea         IN SDO_NUMBER_ARRAY,
    bandNumbers      IN VARCHAR2,
    storageParam     IN VARCHAR2,
```

```
outGeoRaster    IN OUT SDO_GEORASTER,
bgValues        IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

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.

pyramidLevel
A number specifying the pyramid level of the source GeoRaster object.

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. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Section 1.4.3.) Cannot be the same GeoRaster object as `inGeoRaster`.

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

polygonClip
The string `TRUE` causes the clipping window (`cropArea` geometry object) to be used for the subset operation; the string `FALSE` or a null value causes the MBR (minimum bounding rectangle) of the clipping window to be used for the subset operation.

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 you use the format that includes the `pyramidLevel` parameter and specify a value greater than zero (0), the cropping is done based on the specified pyramid level of the source GeoRaster object; otherwise, the cropping is done based on the original source GeoRaster object (`pyramidLevel` = 0).

If the source GeoRaster object is georeferenced and the `pyramidLevel` parameter value is greater than 0, the georeferencing information is generated for the resulting GeoRaster object only when the georeference is a valid polynomial transformation.

Any upper-level pyramid data in the input GeoRaster object is not considered in this operation, and the output GeoRaster object has no pyramid data.

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.)

If the `cropArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial Developer's Guide*.

To be able to use the clipping window geometry object itself to subset the GeoRaster object, the geometry object must be a valid two-dimensional polygon geometry, simple or multipolygon, with an SDO_GTYPE value in the form 2*nn*3 or 2*nn*7. For any other SDO_GTYPE value, the MBR of the geometry object is used regardless of the value of the `polygonClip` parameter. (For an explanation of SDO_GTYPE values, see *Oracle Spatial Developer's Guide*.)

If the clipping window geometry object itself is applied to the subset process, all cells inside the polygon or touched by the polygon edges are returned; other cells within the MBR of the geometry object are clipped, that is, filled by the specified or default `bgValues` parameter values.

If `polygonClip` is `TRUE`, and if this procedure creates a rectangular image subset but the geometry is not a rectangle, check the validity of the `inWindow` geometry object with the function SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT. For an invalid geometry, this procedure operates as if the `polygonClip` value is `FALSE` or a null value.

`inGeoRaster` and `outGeoRaster` must be different GeoRaster objects.

Only the overlapping portion of the specified window of interest and the source GeoRaster object's spatial extent is copied.

If you want to reproject the output GeoRaster object to a different coordinate system, use the SDO_GEOR.reproject procedure.

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.
- The specified window of interest falls outside of the GeoRaster object's spatial extent.

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;
/
```

The following example demonstrates how to do clipping while subsetting a GeoRaster object using a polygon. (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;
  grsub sdo_georaster;
  win1 sdo_geometry;
BEGIN
  Delete from georaster_table where georid = 111;
  INSERT INTO georaster_table VALUES (111, 'ClippedImage',
     sdo_geor.init('georaster_RDT', 111))
     RETURNING georaster INTO grsub;
  SELECT georaster INTO gr FROM georaster_table WHERE georid=1;
  win1 := sdo_geometry(2003, 26986, null, sdo_elem_info_array(1,1003,1),
                sdo_ordinate_array(237040,   897924,
                                   237013.3, 897831.6,
                                   237129,   897840,
                                   237182.5, 897785.5,
                                   237239.9, 897902.7,
                                   237223,   897954,
```

```
                                    237133,   897899,
                                    237040,   897924));
  sdo_geor.subset(gr, 0, win1, '1-3',
                  'interleaving = BIL, compression=DEFLATE',
                  grsub, NULL, 'TRUE');
  UPDATE georaster_table SET georaster=grsub WHERE georid=111;
  COMMIT;
END;
/
```

SDO_GEOR.updateRaster

Format

```
SDO_GEOR.updateRaster(
    targetGeoRaster         IN OUT SDO_GEORASTER,
    targetPyramidLevel      IN NUMBER,
    targetLayerNumbers      IN VARCHAR2,
    targetArea              IN SDO_GEOMETRY,
    sourceGeoRaster         IN SDO_GEORASTER,
    sourcePyramidLevel      IN NUMBER,
    sourceLayerNumbers      IN VARCHAR2,
    updateUpperPyramids     IN VARCHAR2,
    bgValues                IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.updateRaster(
    targetGeoRaster         IN OUT SDO_GEORASTER,
    targetPyramidLevel      IN NUMBER,
    targetBandNumbers       IN VARCHAR2,
    targetArea              IN SDO_NUMBER_ARRAY,
    sourceGeoRaster         IN SDO_GEORASTER,
    sourcePyramidLevel      IN NUMBER,
    sourceBandNumbers       IN VARCHAR2,
    updateUpperPyramids     IN VARCHAR2,
    bgValues                IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Updates a specified pyramid of a specified area or the overlapping parts of one GeoRaster object with selected pyramid and selected bands or layers of another GeoRaster object.

Parameters

targetGeoRaster
GeoRaster object to be updated. (Be sure to make a copy of this object before you update it.)

targetPyramidLevel
Number specifying the pyramid level of the target GeoRaster object to be updated.

targetLayerNumbers
String specifying one or more layer numbers of layers in `targetGeoRaster` to be updated. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'1,3-5,7'` for layers 1, 3, 4, 5, and 7.

targetBandNumbers
String specifying one or more band numbers of bands in `targetGeoRaster` to be updated. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'0,3-5,7'` for bands 0, 3, 4, 5, and 7. Any bands that you specify for this parameter must be compatible with the bands to be updated in the target GeoRaster object.

targetArea
Area to be updated in `targetGeoRaster`: a rectangular window, specified either as a numeric array with the lower-left and upper-right coordinates or as an SDO_GEOMETRY object. The SDO_NUMBER_ARRAY type is defined as `VARRAY(1048576) OF NUMBER`.

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 target area; see also the Usage Notes for SDO_SRID requirements.

If `targetArea` is of type SDO_GEOMETRY, use the `targetLayerNumbers` and `sourceLayerNumbers` parameters to specify one or more layer numbers; if `targetArea` is of type SDO_NUMBER_ARRAY, use the `targetBandNumbers` and `sourceBandNumbers` parameters to specify one or more band numbers.

If the specified area does not intersect with the spatial extent of `targetGeoRaster`, no update is performed. If this parameter is specified as null, all of the overlapping area is updated.

sourceGeoRaster
GeoRaster object in which specified layers are to be used to update `targetGeoRaster`.

sourcePyramidLevel
Number specifying the pyramid level of the `sourceGeoRaster` object.

sourceLayerNumbers
String specifying one or more layer numbers of layers in `sourceGeoRaster` to be used to update `targetGeoRaster`. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'1,3-5,7'` for layers 1, 3, 4, 5, and 7.

Any layers that you specify for this parameter must be compatible with the layers to be updated in the target GeoRaster object.

sourceBandNumbers
String specifying one or more band numbers of bands in `sourceGeoRaster` to be used to update `targetGeoRaster`. Use commas to delimit numbers or ranges, and use a hyphen to indicate a range. Example: `'0,3-5,7'` for bands 0, 3, 4, 5, and 7.

Any bands that you specify for this parameter must be compatible with the bands to be updated in the target GeoRaster object.

updateUpperPyramids
String (`TRUE` or `FALSE`) specifying whether to update upper-level pyramids. (This parameter has no default value; you should always specify it.)

bgValues
Background values for filling partially empty raster blocks. It is only useful when the source GeoRaster object has empty raster blocks and the current operation leads to partially empty raster blocks (see Section 1.4.4). The number of elements in the SDO_

NUMBER_ARRAY object must be either one (same filling value used for all bands) or the band dimension size (a different filling value for each band, respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first band with 1, the second band with 5, and the third band with 10.

The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

Usage Notes

> **Note:** Be sure to make a copy of the `targetGeoRaster` object before you call this procedure, because the changes made to this GeoRaster object might not be reversible after the procedure completes.

If both GeoRaster objects are georeferenced, they must use the same coordinate system, have the same cell depth, and have the same spatial resolutions at the specified pyramid levels; however, the `targetPyramidLevel` and `sourcePyramidLevel` values can be different. If both GeoRaster objects are not georeferenced, the ULTCoordinates will be considered to co-locate them into each other.

The two GeoRaster objects can have different dimensions and sizes.

If the `targetArea` 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 `targetArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial Developer's Guide*.

Any existing bitmap masks are not updated.

If the source GeoRaster object is not large enough to fill in the target area, the uncovered area will not be updated.

If the target GeoRaster object has pyramids or is compressed, or both, the updates will be reflected in the pyramids and the compression.

To update upper-level pyramids, you must specify the `updateUpperPyramids` parameter as '`TRUE`'. (This parameter has no default value; you should always specify '`TRUE`' or '`FALSE`'.)

Examples

The following example updates a specified area in band 1 of the specified target GeoRaster object with band 0 of the same area of another GeoRaster object.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  area sdo_number_array := sdo_number_array(-200,-50,201,162);
```

```
BEGIN
  SELECT georaster INTO gr2 FROM georaster_table WHERE georid=0 FOR UPDATE;
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=1;
  SDO_GEOR.updateRaster(gr2, 0, '1', area, gr1, 0, '0', 'true');
  UPDATE GEORASTER_TABLE SET georaster=gr2 WHERE georid=0;
  COMMIT;
END;
/
```

SDO_GEOR.validateBlockMBR

Format

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

Description

Validates the `blockMBR` attribute of each block of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function checks the `blockMBR` attribute (described in Section 2.2.6) in each row of the raster data table associated with the specified GeoRaster object to see if its geometry is the actual minimum bounding rectangle (MBR) of that block.

This function returns the string `TRUE` if the `blockMBR` attribute is the MBR of each block, a null value if the GeoRaster object is null, an Oracle error code if the error is known, or `FALSE` for an unknown error.

If you created the GeoRaster object as described in Section 3.2, the `blockMBR` attribute values were automatically calculated and they should not need to be validated or generated. However, if the GeoRaster object was generated by a third party, you should validate the `blockMBR` attribute values using this function; and if any are not valid, call the SDO_GEOR.generateBlockMBR procedure.

Examples

The following example validates the `blockMBR` attribute of each block of a specified GeoRaster object.

```
SELECT sdo_geor.validateBlockMBR(georaster) FROM georaster_table WHERE georid=1;

SDO_GEOR.VALIDATEBLOCKMBR(GEORASTER)
--------------------------------------------------------------------------------
TRUE
```

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 the string `TRUE` if the GeoRaster object is valid, a null value 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, or of empty or not empty.
- 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.)
- The name of the raster data table must not contain spaces, period separators, or mixed-case letters in a quoted string, and all the alphanumeric characters must be uppercase.

- The raster data table must be an object table of SDO_RASTER type, and the table must exist if the GeoRaster object is not blank. To use GeoRaster with Oracle Workspace Manager or Oracle Label Security (OLS), you can define an object view of SDO_RASTER type and use the object view as the raster storage.
- There must be an entry for the GeoRaster object in the ALL_SDO_GEOR_SYSDATA view.
- Each associated bitmap mask must have the correct number of rows in the RDT.
- Any NODATA values and value ranges are in the valid cell value range as designated by the cell depth.
- 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. (A BLOB with zero length is valid; it is an empty raster block.)
- For an uncompressed GeoRaster object, the raster block size of each bitmap mask is checked, based on the blocking size and `1BIT` cell depth. (A BLOB with zero length is valid; it is an empty bitmap mask raster block.)
- A generic functional fitting polynomial model is supported, as described in Section 1.6.1. The limitations on offsets, scales, RMS values, pType, nVars, and number of coefficients of the polynomials are described in Section 1.6.1 and Table 2–4 in Section 2.3.5.
- The SRID in the GeoRaster SRS metadata is not checked against the CS_SRS table and is not validated. To validate the SRID, call SDO_GEOR.getModelSRID and SDO_CS.VALIDATE_WKT (the latter described in *Oracle Spatial Developer's Guide*). The `verticalSRID` value is not used in the current release.
- Ground control points (GCPs), as the StoredFunction georeferencing model, are supported. The `gcpGeoreferenceModel` in the metadata should follow the definition of the SDO_GEOR_GCPGEOREFTYPE type as described in Section 2.3.8, and each GCP should follow the specification of the SDO_GEOR_GCP type as described in Section 2.3.6. The number of GCPs is not checked against the `FFMethod` attribute, so you can have the flexibility to add GCPs gradually.
- The `RigorousModel` georeferencing model is not supported. If the functional polynomial coefficients are set, the `modelType` value must be set to `FunctionalFitting` and the `isReferenced` value is set as `TRUE`. If are GCPs are stored in the metadata, the `modelType` value must be set to `StoredFunction`. If both conditions are true, two `modelType` values are added to contain both `StoredFunction` and `FunctionalFitting` values.
- 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.
- Complex `cellDepth` values are not supported.
- This function does not check any external tables (such as a 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 Developer's Guide*.
- 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 the `blockMBR` geometries, use the SDO_GEOR.validateBlockMBR function.

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_ADMIN Package Reference

The MDSYS.SDO_GEOR_ADMIN package contains subprograms (functions and procedures) for administrative operations related to GeoRaster. This chapter presents reference information, with one or more examples, for each subprogram.

SDO_GEOR_ADMIN.checkSysdataEntries

Format

SDO_GEOR_ADMIN.checkSysdataEntries() RETURN SDO_STRING2_ARRAY;

Description

Checks the USER_SDO_GEOR_SYSDATA view for any invalid entries.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of GeoRaster system data entries that are invalid. It checks for errors such as the following:

- The RDT name is not unique.
- The GeoRaster table does not exist.
- The GeoRaster column does not exist.
- The GeoRaster objects does not exist.
- The GeoRaster object is non-empty or nonblank, but the RDT does not exist.
- Duplicate GeoRaster objects exist (that is, one or more non-unique combinations of RDT and raster ID).

If you call this function while connected as the MDSYS user, the entries in the ALL_SDO_GEOR_SYSDATA view instead of the USER_SDO_GEOR_METADATA view are checked.

The USER_SDO_GEOR_DATA and ALL_SDO_GEOR_SYSDATA views are described in Section 2.4.

Examples

The following example checks the USER_SDO_GEOR_SYSDATA view for invalid entries.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.checkSysdataEntries FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
The RDT name "RDT1" is not unique
The GeoRaster object GEOR_TEST1.TABLE1.GEOR: RDT=RDT2 RID=3 is associated with a
 non-existing RDT table!
The specification of GeoRaster column GEOR_TEST1.TABLE1.c1  is not correct.
The GeoRaster object GEOR_TEST1.TABLE1.geor: RDT=dt3 RID=2 doesn't exist!
The GeoRaster table GEOR_TEST1.t1  doesn't exist!
```

SDO_GEOR_ADMIN.isRDTNameUnique

Format

```
SDO_GEOR_ADMIN.isRDTNameUniquer(
   rdtName  VARCHAR2)
   RETURN  VARCHAR2;
```

Description

Checks if the specified raster data table (RDT) name is unique among RDT names in the database.

Parameters

rdtName
Name to be checked for uniqueness.

Usage Notes

You can use this function to check, before you create an RDT, if the RDT name that you plan to use is unique.

This function returns the string `TRUE` if the name is unique and the string `FALSE` if the name is not unique.

Examples

The following example checks if the name `MY_RDT` is unique.

```
SELECT SDO_GEOR_ADMIN.IsRDTNameUnique('MY_RDT')  FROM DUAL;

SDO_GEOR_ADMIN.ISRDTNAMEUNIQUE('MY_RDT')
--------------------------------------------------------------------------------
TRUE
```

SDO_GEOR_ADMIN.isUpgradeNeeded

Format

SDO_GEOR_ADMIN.isUpgradeNeeded() RETURN SDO_STRING2_ARRAY;

Description

Checks the GeoRaster system data entries and GeoRaster data for the current schema.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of GeoRaster system data entries and GeoRaster columns and objects that are invalid. It can report errors such as the following:

- System data entry error, the RDT name is not unique.
- System data entry error, the RDT/RID pair is not unique.
- System data entry error, the GeoRaster table does not exist.
- System data entry error, the GeoRaster column does not exist.
- System data entry error, the GeoRaster object does not exist.
- The GeoRaster object is non-empty or nonblank, but the RDT does not exist.
- Duplicate GeoRaster objects exist (that is, one or more non-unique combinations of RDT and raster ID).
- There is a non-registered pair of (GeoRaster column, GeoRaster object).

If you call this function while connected as the MDSYS user, the GeoRaster system data entries and GeoRaster data for the entire database are checked.

Examples

The following example checks the GeoRaster system data entries and GeoRaster data. It assumes that you are connected as the MDSYS user.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.isUpgradeNeeded FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
The following GeoRaster columns aren't registered:
     SCHEMA:GEOR_TEST TABLE:TABLE1 COLUMN:GEOR
The following GeoRaster objects aren't registered:
     SCHEMA:GEOR_TEST TABLE:TABLE1 COLUMN:GEOR RDT:RDT RID:3
     SCHEMA:GEOR_TEST TABLE:TABLE1 COLUMN:GEOR RDT:RDT RID:4
```

SDO_GEOR_ADMIN.listGeoRasterColumns

Format

SDO_GEOR_ADMIN.listGeoRasterColumns() RETURN SDO_STRING2_ARRAYSET;

Description

Lists the GeoRaster columns defined in the current schema.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of GeoRaster columns with their registration status. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- GeoRaster table name
- GeoRaster column name
- Status: registered (a DML trigger is created for the GeoRaster column) or unregistered (no DML trigger is created for the GeoRaster column)

If you call this function while connected as the MDSYS user, all GeoRaster columns defined in the database are listed.

Examples

The following example lists the GeoRaster columns defined in the current schema.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listGeoRasterColumns FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('TEST_TABLE1', 'GEOR', 'registered')
SDO_STRING2_ARRAY('TEST_TABLE2', 'GEOR', 'registered')
```

SDO_GEOR_ADMIN.listGeoRasterObjects

Format

SDO_GEOR_ADMIN.listGeoRasterObjects() RETURN SDO_STRING2_ARRAYSET;

Description

Lists the GeoRaster objects defined in the current schema.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of GeoRaster objects with their registration status. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- GeoRaster table name
- GeoRaster column name
- RDT name
- Raster ID
- Status: registered (the GeoRaster object has been registered is the SYSDATA table) or unregistered (the GeoRaster object has not been registered is the SYSDATA table)

If you call this function while connected as the MDSYS user, all GeoRaster objects defined in the database are listed.

Examples

The following example lists the GeoRaster objects defined in the current schema.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listGeoRasterObjects FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('TEST_TABLE1', 'GEOR', 'RDT_REGULAR_01', '1', 'registered')
SDO_STRING2_ARRAY('TEST_TABLE2', 'GEOR', 'RDT_REGULAR_01', '2', 'registered')
```

SDO_GEOR_ADMIN.listGeoRasterTables

Format

SDO_GEOR_ADMIN.listGeoRasterTables() RETURN SDO_STRING2_ARRAYSET;

Description

Lists the GeoRaster tables defined in the current schema.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of GeoRaster tables. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- GeoRaster table name

If you call this function while connected as the MDSYS user, all GeoRaster tables defined in the database are listed.

Examples

The following example lists the GeoRaster tables defined in the database. It assumes that you are connected as the MDSYS user.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listGeoRasterTables FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('GEOR_TEST', 'TEST_TABLE1')
SDO_STRING2_ARRAY('GEOR_TEST', 'TEST_TABLE2')
```

SDO_GEOR_ADMIN.listDanglingRasterData

Format

SDO_GEOR_ADMIN.listDanglingRasterData() RETURN SDO_STRING2_ARRAYSET;

Description

Checks the GeoRaster system data entries and GeoRaster data, and lists all dangling raster data.

Parameters

None.

Usage Notes

Raster data table (RDT) rows might exist for nonexistent GeoRaster objects or GeoRaster objects that are not referred to in the SYSDATA table. The raster blocks associated with such rows are referred to *dangling* blocks. The dangling raster blocks cause wasted disk space in the RDT although otherwise they do not present a problem as long as the necessary primary key is defined on the RDT. To find these dangling blocks in the current schema or in all schemas, call the SDO_GEOR_ADMIN.listDanglingRasterData function.

Before you call this function, you should call SDO_GEOR_ADMIN.registerGeoRasterObjects to register all existing GeoRaster objects.

To remove the dangling raster block data from an RDT, delete the rows associated with the problems discovered by the SDO_GEOR_ADMIN.listDanglingRasterData function.

This function returns an array of comma-delimited list of dangling raster data. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- RDT name
- Raster ID

If you call this function while connected as the MDSYS user, all dangling raster data in the database is listed.

Examples

The following example lists all dangling raster data in the current schema.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listDanglingRasterData FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('RDT11', '3')
```

SDO_GEOR_ADMIN.listRDT

Format

SDO_GEOR_ADMIN.listRDT() RETURN SDO_STRING2_ARRAYSET;

Description

Lists the raster data tables (RDTs) defined in the current schema.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of RDTs. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- RDT name

If you call this function while connected as the MDSYS user, all RDTs defined in the database are listed.

Examples

The following example lists the RDTs defined in the current schema.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listRDT FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('RDT_REGULAR_01')
SDO_STRING2_ARRAY('RDT_REGULAR_02')
```

SDO_GEOR_ADMIN.listRegisteredRDT

Format

SDO_GEOR_ADMIN.listRegisteredRDT() RETURN SDO_STRING2_ARRAYSET;

Description

Lists the registered raster data tables (RDTs) defined in the current schema. An RDT is *registered* if at least one entry in the SYSDATA table refers to it.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of RDTs. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- RDT name

If you call this function while connected as the MDSYS user, all registered RDTs defined in the database are listed.

Examples

The following example lists the registered RDTs defined in the current schema.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listRegisteredRDT FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('RDT1_REGULAR_01')
```

SDO_GEOR_ADMIN.listUnregisteredRDT

Format

SDO_GEOR_ADMIN.listUnregusteredRDT() RETURN SDO_STRING2_ARRAYSET;

Description

Lists the unregistered raster data tables (RDTs) defined in the current schema. An RDT is *unregistered* if no entries in the SYSDATA table refer to it.

Parameters

None.

Usage Notes

This function returns an array of comma-delimited list of RDTs. The list contains the following information:

- Schema name (only if you are connected as the MDSYS user)
- RDT name

If you call this function while connected as the MDSYS user, all unregistered RDTs defined in the database are listed.

Examples

The following example lists the unregistered RDTs defined in the current schema.

```
SELECT * FROM THE (SELECT SDO_GEOR_ADMIN.listUnregisteredRDT FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('RDT_REGULAR_02')
```

SDO_GEOR_ADMIN.maintainSysdataEntries

Format

SDO_GEOR_ADMIN.maintainSysdataEntries() RETURN SDO_STRING2_ARRAY;

Description

Checks the USER_SDO_GEOR_SYSDATA view for any invalid entries, and takes corrective action as appropriate.

Parameters

None.

Usage Notes

This function performs the same checks as the SDO_GEOR_ADMIN.checkSysdataEntries function, and it takes the corrective action that is appropriate (if any). For each of the following errors, the function does the following:

- The RDT name is not unique. If you are connected as a user other then MDSYS, no action is taken; if you are connected as user MDSYS, duplicate RDTs are renamed so that their names are unique.
- The GeoRaster table does not exist. The entry is deleted.
- The GeoRaster column does not exist. The entry is deleted.
- The GeoRaster objects does not exist. The entry is deleted.
- The GeoRaster object is non-empty or nonblank, but the RDT does not exist. The entry is deleted.
- Duplicate GeoRaster objects exist (that is, one or more non-unique combinations of RDT and raster ID). The entry is deleted.

If you call this function while connected as the MDSYS user, the entries in the ALL_SDO_GEOR_SYSDATA view instead of the USER_SDO_GEOR_METADATA view are checked.

The USER_SDO_GEOR_DATA and ALL_SDO_GEOR_SYSDATA views are described in Section 2.4.

Examples

The following example checks the USER_SDO_GEOR_SYSDATA view for invalid entries, and performs corrective action as appropriate.

```
DECLARE
  ret SDO_STRING2_ARRAY;
BEGIN
  ret:=sdo_geor_admin.MAINTAINSYSDATAENTRIES;
  for i in 1..ret.count loop
    dbms_output.put_line(ret(i));
  end loop;
END;
/
The RDT name GEOR_TEST1.RDT2 is renamed to GEOR_TEST1.RDT1!
The sysdata entry (SCHEMA=GEOR_TEST1 RDT=dt1 RID=1) is deleted!
```

```
PL/SQL procedure successfully completed.
```

SDO_GEOR_ADMIN.registerGeoRasterColumns

Format

SDO_GEOR_ADMIN.registerGeoRasterColumns;

Description

Creates DML triggers for all GeoRaster columns defined in the current schema.

Parameters

None.

Usage Notes

You should not normally need to execute this procedure. You should execute it only if some error or other condition has resulted in GeoRaster columns without associated DML triggers.

If you execute this procedure while connected as the MDSYS user, DML triggers are created for all GeoRaster columns defined in all schemas.

Examples

The following example creates DML triggers for all GeoRaster columns defined in the current schema.

```
EXECUTE sdo_geor_admin.registerGeoRasterColumns;
```

SDO_GEOR_ADMIN.registerGeoRasterObjects

Format

SDO_GEOR_ADMIN.registerGeoRasterObjects;

Description

Registers all GeoRaster objects defined in the current schema.

Parameters

None.

Usage Notes

If you execute this procedure while connected as the MDSYS user, all GeoRaster objects defined in all schemas are registered.

Examples

The following example registers all GeoRaster objects defined in the current schema.

```
EXECUTE sdo_geor_admin.registerGeoRasterObjects;
```

SDO_GEOR_ADMIN.upgradeGeoRaster

Format

SDO_GEOR_ADMIN.upgradeGeoRaster() RETURN SDO_STRING2_ARRAY;

Description

Checks the GeoRaster system data entries and GeoRaster data for the current schema, and performs any corrective action as appropriate.

Parameters

None.

Usage Notes

This function performs the same checks as the SDO_GEOR_ADMIN.isUpgradeNeeded function, and it takes the corrective action that is appropriate (if any) for the following errors:

- System data entry error, the RDT name is not unique.
- System data entry error, the RDT/RID pair is not unique.
- System data entry error, the GeoRaster table does not exist.
- System data entry error, the GeoRaster column does not exist.
- System data entry error, the GeoRaster object does not exist.
- The GeoRaster object is non-empty or nonblank, but the RDT does not exist.
- Duplicate GeoRaster objects exist (that is, one or more non-unique combinations of RDT and raster ID).
- There is a non-registered pair of (GeoRaster column, GeoRaster object).

If you call this function while connected as the MDSYS user, the GeoRaster system data entries and GeoRaster data for the entire database are checked, and any appropriate corrective actions are taken.

Examples

The following example checks the GeoRaster system data entries and GeoRaster data for the current schema, and performs any corrective action as appropriate.

```
SELECT SDO_GEOR_ADMIN.upgradeGeoRaster FROM DUAL;
```

6

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.calcOptimizedBlockSize

Format

```
SDO_GEOR_UTL.calcOptimizedBlockSize(
   dimensionSize  IN SDO_NUMBER_ARRAY,
   blockSize      IN OUT SDO_NUMBER_ARRAY,
   pyramidLevel   IN number default 0);
```

Description

Calculates an optimal `blockSize` value that will use less padding space in the GeoRaster object storage, based on the GeoRaster dimension sizes and the user-specified block size values.

Parameters

dimensionSize
Dimension size array of the GeoRaster object.

blockSize
Block size array, which holds the user-specified block size values and into which the procedure outputs the adjusted optimal block size values.

pyramidLevel
Maximum pyramid level. The default value is 0.

Usage Notes

This procedure enables you to give desired block size values (which may not be optimal), automatically adjust them, and then determine the block size array values for a specified GeoRaster dimension size array that will be optimal for reducing the amount of padding space in GeoRaster object storage. The adjustment is always made around the user-specified values. For more information, see the explanations of the `blocking` and `blockSize` keywords in Table 1–1, " storageParam Keywords for Raster Data" in Section 1.4.1.

An exception is generated if the input `dimensionSize` or `blockSize` parameter contains any invalid values.

Examples

The following example calculates and displays an optimal block size value, based on a specified dimension size array of (12371,11261,13) and a specified block size array of (512,512,5). Note that the optimal `rowBlockSize` value returned is 538 as opposed to the original value of 512, and the optimal `bandBlockSize` value returned is 1 as opposed to the original value of 5.

```
DECLARE
  dimensionSize    sdo_number_array;
  blockSize        sdo_number_array;
BEGIN
  dimensionSize:=sdo_number_array(12371,11261,13);
  blockSize:=sdo_number_array(512,512,5);
  sdo_geor_utl.calcOptimizedBlockSize(dimensionSize,blockSize);
  dbms_output.put_line('Optimized rowBlockSize = '||blockSize(1));
```

```
  dbms_output.put_line('Optimized colBlockSize = '||blockSize(2));
  dbms_output.put_line('Optimized bandBlockSize = '||blockSize(3));
END;
/
Optimized rowBlockSize = 538
Optimized colBlockSize = 512
Optimized bandBlockSize = 1
```

SDO_GEOR_UTL.calcRasterNominalSize

Format

```
SDO_GEOR_UTL.calcRasterNominalSize(
   geor          IN SDO_GEORASTER,
   padding       IN VARCHAR2 DEFAULT 'TRUE',
   pyramid       IN VARCHAR2 DEFAULT 'TRUE',
   bitmapMask  IN VARCHAR2 DEFAULT 'TRUE'
   ) RETURN NUMBER;
```

Description

Returns the total raster block length (in bytes) of a GeoRaster object, as if it were not compressed and did not contain any empty raster blocks.

Parameters

geor
GeoRaster object.

padding
The string `TRUE` (the default) causes padding in the raster blocks to be considered; the string `FALSE` causes padding in the raster blocks not to be considered.

pyramid
The string `TRUE` (the default) causes the size of any pyramids to be considered; the string `FALSE` causes the size of any pyramids not to be considered.

bitmapMask
The string `TRUE` (the default) causes any associated bitmap masks to be considered; the string `FALSE` causes any associated bitmap masks not to be considered. For an explanation of bitmap masks, see Section 1.8.

Usage Notes

This function does not consider any LOB storage overhead, so the result is only an approximation of the real storage requirements for the GeoRaster object.

The result of this function will be greater than or equal to the result of the SDO_GEOR_UTL.calcRasterStorageSize function on the same GeoRaster object. If this function returns a larger value than the SDO_GEOR_UTL.calcRasterStorageSize function on the same GeoRaster object, the difference in the values reflects the space saved by the use of compression or empty raster blocks, or both.

For information about GeoRaster compression, see Section 1.10.

Examples

The following example calculates the nominal raster size (in bytes) of a GeoRaster object, according to its current blocking scheme. The returned size includes (by default) any padding in the raster blocks, any associated bitmap masks, and any pyramids.

```
SELECT SDO_GEOR_UTL.calcRasterNominalSize(georaster) nsize FROM georaster_table
```

```
  WHERE georid=1;

     NSIZE
----------
    289150
```

SDO_GEOR_UTL.calcRasterStorageSize

Format

```
SDO_GEOR_UTL.calcRasterStorageSize(
   geor  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the actual length (in bytes) of all raster blocks of a GeoRaster object.

Parameters

geor
GeoRaster object.

Usage Notes

The function calculates the actual length of all raster blocks of a GeoRaster object. It does not consider any LOB storage overhead, so the result is only an approximation of the real storage size of the GeoRaster object. In essence, this function executes the following statement:

```
EXECUTE IMMEDIATE
'SELECT SUM(DBMS_LOB.getLength(rasterBlock)) FROM ' || geor.rasterDataTable || '
WHERE rasterId=' || geor.rasterId;
```

The result of this function will be less than or equal to the result of the SDO_GEOR_UTL.calcRasterNominalSize function on the same GeoRaster object. If this function returns a smaller value than the SDO_GEOR_UTL.calcRasterNominalSize function on the same GeoRaster object, the difference in the values reflects the space saved by the use of compression or empty raster blocks, or both.

Examples

The following example calculates ratio (as a decimal fraction) of the actual size to the nominal size of a specified GeoRaster object. In this example, the actual size is about one-twentieth (1/20) of the nominal size.

```
SELECT SDO_GEOR_UTL.calcRasterStorageSize(georaster)/
  SDO_GEOR_UTL.calcRasterNominalSize(georaster) ratio
    FROM georaster_table WHERE georid=1;

     RATIO
----------
.056198816
```

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

As explained in Section 3.1.3, to ensure the consistency and integrity of internal GeoRaster tables and data structures, GeoRaster automatically creates a unique DML trigger for each GeoRaster column whenever a user creates a GeoRaster table (that is, a table with at least one GeoRaster column), with the following exception: if you use the ALTER TABLE statement to add one or more GeoRaster columns. In this case, you must call the SDO_GEOR_UTL.createDMLTrigger procedure to create the DML trigger on each added GeoRaster column.

Otherwise, you usually do not need to call this procedure, although but it is still useful for re-creating the DML trigger in some scenarios, such as a database upgrade or a data migration.

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 registered 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 registered 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.

This procedure is not transactional, and the result cannot be rolled back.

Examples

The following example automatically renames some existing registered raster data tables that do not have unique names so that all registered 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 registered 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 registered 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 registered 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.

This procedure is not transactional, and the result cannot be rolled back.

Examples

The following example renames the registered raster data tables RDT_1 and RDT_2 to ST_RDT_1 and ST_RDT_2, respectively.

```
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 Raster Spatial Reference Systems
     Part 1.3.2: Data Types for Raster Temporal Reference Systems
     Part 1.3.3: Data Types for Raster Band Reference Systems
     Part 1.4: Data Types for Layer Metadata
     Part 2: GeoRaster Metadata Elements or Content Structure
     ============================================================
    </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="defaultPyramidLevel" 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:enumeration value="JP2-C"/>
    <xsd:enumeration value="JP2-F"/>
  </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 1.3.1: Data Types for GeoRaster Spatial Reference System

    Spatial extent (footprint) is recorded as an attribute of GeoRaster object.
    ts 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 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:attribute name="xRMS" type="xsd:double" use="optional"/>
    <xsd:attribute name="yRMS" type="xsd:double" use="optional"/>
    <xsd:attribute name="zRMS" type="xsd:double" use="optional"/>
    <xsd:attribute name="modelTotalRMS" type="xsd:double" use="optional"/>
    <xsd:anyAttribute/>
  </xsd:complexType>
  <xsd:annotation>
    <xsd:documentation>
    The following types and definitions are for GCP support. It stores
    GCP collection for the GeoRaster object. It also optionally specify
    the Functional Fitting method for generating FFM using the GCP collection.
    cellDimension can 2 or 3 (only 2 is supported in current release).
    modelDimension can be 2, 3, -2, -3. cellCoordinate must be (row, column)
    in current release.
    modelCoordinate must be (X, Y) or (X, Y, Z) in current release.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType name="gcpPointType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ControlPoint"/>
      <xsd:enumeration value="CheckPoint"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="gcpPointStatusType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Measured"/>
      <xsd:enumeration value="Removed"/>
      <xsd:enumeration value="Estimated"/>
      <xsd:enumeration value="Validated"/>
      <xsd:enumeration value="Invalid"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="GCPType">
    <xsd:attribute name="ID" type="xsd:string" use="optional"/>
    <xsd:attribute name="description" type="xsd:string" use="optional"/>
     <xsd:attribute name="type" type="gcpPointType" use="required"/>
    <xsd:attribute name="cellDimension" type="xsd:nonNegativeInteger"
use="required"/>
    <xsd:attribute name="row" type="xsd:double" default="0" />
    <xsd:attribute name="column" type="xsd:double" default="0" />
    <xsd:attribute name="vertical" type="xsd:integer" use="optional"/>
    <xsd:attribute name="modelDimension" type="xsd:nonNegativeInteger"
```

```
use="required"/>
    <xsd:attribute name="X" type="xsd:double" default="0"/>
    <xsd:attribute name="Y" type="xsd:double" default="0"/>
    <xsd:attribute name="Z" type="xsd:double" use="optional"/>
    <xsd:attribute name="xRMS" type="xsd:double" use="optional"/>
    <xsd:attribute name="yRMS" type="xsd:double" use="optional"/>
    <xsd:attribute name="zRMS" type="xsd:double" use="optional"/>
    <xsd:attribute name="status" type="gcpPointStatusType" use="optional"/>
    <xsd:anyAttribute/>
  </xsd:complexType>
  <xsd:simpleType name="FFMethodType">
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Affine"/>
                <xsd:enumeration value="QuadraticPolynomial"/>
                <xsd:enumeration value="CubicPolynomial"/>
                <xsd:enumeration value="DLT"/>
                <xsd:enumeration value="QuadraticRational"/>
                <xsd:enumeration value="RPC"/>
           </xsd:restriction>
        </xsd:simpleType>
  <xsd:complexType name="GCPGeoreferenceType">
    <xsd:sequence>
      <xsd:element name="gcp" type="GCPType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="FFMethod" type="FFMethodType" use="optional"/>
  </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:choice minOccurs="0">
         <xsd:element name="gcpGeoreferenceModel" type="GCPGeoreferenceType"/>
```

```
            <xsd:element name="gcpTableName" type="xsd:string" minOccurs="0"/>
        </xsd:choice>
        <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 info 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 sub-layer the layerNumber is a positive integer, i.e., 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="NODATAType">
          <xsd:sequence>
            <xsd:element name="value" type="xsd:double" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="range" type="extentType" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="mask" type="xsd:boolean" minOccurs="0"/>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
  <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 statistic dataset 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="bitmapMask" type="xsd:boolean"
minOccurs="0" default="false"/>
                        <xsd:element name="NODATA" type="NODATAType"
minOccurs="0"/>
      <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

O

P

Q

R

S

T

U

V

W

X

Z