Oracle® Spatial and Graph
GeoRaster Developer's Guide

12*c* Release 2 (12.2)

E85863-01

April 2017

ORACLE®

Oracle Spatial and Graph GeoRaster Developer's Guide, 12*c* Release 2 (12.2)

E85863-01

Contents

2   GeoRaster Data Types and Related Structures

3   GeoRaster Database Creation and Management

4   GeoRaster Data Query and Manipulation

5   Raster Algebra and Analytics

6   Image Processing and Virtual Mosaic

7 SDO_GEOR Package Reference

8   SDO_GEOR_ADMIN Package Reference

9   SDO_GEOR_AGGR Package Reference

10   SDO_GEOR_IP Package Reference

11   SDO_GEOR_RA Package Reference

12   SDO_GEOR_UTL Package Reference

A   GeoRaster Metadata XML Schema

Index

List of Figures

List of Tables

Preface

Oracle Spatial and Graph GeoRaster Developer's Guide provides usage and reference information for the GeoRaster feature of Oracle Spatial and Graph, 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 and Graph 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 and Graph.

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

Topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

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

You should be familiar with Oracle Spatial and Graph, 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 that have purchased support 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 and Graph 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.

Changes in This Release for Oracle Spatial and Graph GeoRaster Developer's Guide

This preface contains the following.

Topics:

Changes in Oracle Database 12*c* Release 2 (12.2.0.1)

The following are changes in *Oracle Spatial and Graph GeoRaster Developer's Guide* for Oracle Database 12*c* Release 2 (12.2.0.1).

Native JPEG 2000 Compression

GeoRaster can be natively stored in JPEG 2000 Compression now (see JPEG 2000 Compression of GeoRaster Objects). GDAL can be used to load and export JPEG 2000 files with or without decompression (see Loading JPEG and JPEG 2000 Images Without Decompression).

GeoRaster includes two new procedures for JPEG 2000 compression and decompression: SDO_GEOR.compressJP2and SDO_GEOR.decompressJP2. Most GeoRaster functions support reading or writing of JP2 compressed GeoRaster objects, including key operations such as SDO_GEOR.getRasterSubset, SDO_GEOR.subset, SDO_GEOR.rectify, large scale mosaicking, virtual mosaic, and raster algebra.

New Image Processing Capabilities

Mosaicking supports new color balancing methods: statistic matching and histogram matching either using a reference image or automatically detecting and using

overlapping area (see Color Balancing During Mosaicking and the STATISTICMATCHING and HISTOGRAMMATCHING values for colorBalance in the mosaicParam Keywords table for SDO_GEOR_AGGR.mosaicSubset). The previous limit of eight overlapping GeoRaster objects for a pixel in mosaicking operation is removed.

GeoRaster includes a new image processing package, SDO_GEOR_IP, which supports linear stretching, piecewise stretching, equalization, normalization, histogram matching, dodging, and filtering. See SDO_GEOR_IP Package Reference for details about the subprograms in that package.

GeoRaster also supports warping (see Image Warping and SDO_GEOR.warp) and affine transformation (see SDO_GEOR.affineTransform).

All new functions are parallelized.

New Raster Algebra Capabilities

GeoRaster adds over 25 new operators to the Raster Algebra language, including conditionalExpr (IF-THEN-ELSE), ^ (XOR), % (MODULO), POWER, FACTORIAL, and local statistical operators.

GeoRaster provides new raster algebra functions: SDO_GEOR_RA.stack to perform stack statistical analysis, and SDO_GEOR_RA.diff and SDO_GEOR_RA.over to perform logical operations.

The previous limit of eight GeoRaster objects in one raster algebra operation is removed.

New Core Subprograms and Enhancements

Several new core functions and enhancements are provided:

- GeoRaster provides a set of new memory control functions to allow users to control how much memory to use during GeoRaster operations: SDO_GEOR_UTL.getMaxMemSize, SDO_GEOR_UTL.setMaxMemSize, SDO_GEOR_UTL.getReadBlockMemSize, SDO_GEOR_UTL.setReadBlockMemSize, SDO_GEOR_UTL.getWriteBlockMemSize, and SDO_GEOR_UTL.setWriteBlockMemSize. Memory can be set to larger values to improve processing performance (see Managing Memory to Improve Performance).

- The new SDO_GEOR.getCellValues, SDO_GEOR.evaluateDoubles, SDO_GEOR.changeCellValues, and SDO_GEOR.getBitmapMaskValues functions allow multiple points to be queried or updated in a single function call, thus improving performance.

- GeoRaster provides two additional statistic analysis functions: SDO_GEOR.generateAreaWeightedMean to compute area weighted statistical mean value for the cells and sub-cells within a specific window, and SDO_GEOR_UTL.calcSurfaceArea to calculate the three–dimensional (3D) surface area represented by digital elevation model (DEM).

- Two color ramping functions, SDO_GEOR_UTL.generateColorRamp and SDO_GEOR_UTL.generateGrayRamp, allow you automatically generate a color gradient or grayscale gradient for grids and images.

- The new SDO_GEOR.generateBitmapPyramid procedure automates pyramiding on bitmap GeoRaster objects, which significantly improves the pyramid quality. (See Bitmap Pyramiding.)

- SDO_GEOR.getRasterSubset is enhanced to return cell values in an array directly.

- The performance of repeated single-cell query (calling SDO_GEOR.getCellValue repeatedly) over the same GeoRaster object is significantly improved.

- Rectification, JPEG and DEFLATE compression, and statistics generation operations are parallelized.

- For output GeoRaster objects, the default block size in `storageParam` is changed from 256 x 256 to 512 x 512. (The available parameters and keywords are described in Storage Parameters.)

Java API, Client-Side Viewer, and ETL Enhancements

All Release 12.2 server-side new functions and procedures are supported in the GeoRaster Java API. See *Oracle Spatial and Graph Java API Reference* for details.

GeoRaster also provides a new Virtual Mosaic Java API to enable faster integration and support advanced visualization applications. See *Oracle Spatial and Graph Java API Reference* for details.

The GeoRaster ETL and Viewer are integrated into one tool for ease of deployment and use. The virtual mosaic display is significantly enhanced in the Viewer. The GeoRaster Tool provides a new GUI to automate some PL/SQL tasks, such as creating GeoRaster tables, generating pyramids, and rectifying images for demonstration purposes.

Also, GDAL is distributed with Oracle Spatial and Graph. See Using GDAL from the Spatial and Graph Installation.

Web Coverage Service (WCS) 2.0.1 Support

In addition to WMS, GeoRaster supports the Open Geospatial Consortium (OGC) Web Coverage Services (WCS). You can use WCS to publish GeoRaster objects in the database and allow users to retrieve the raster data over the web, including subsetting, reprojection, and support for many file formats. (See GeoRaster Spatial Web Services.)

SDO_GEOR_IP Package

The SDO_GEOR_IP package contains subprograms for performing some image processing operations on GeoRaster objects. For reference and usage information about the subprograms, see SDO_GEOR_IP Package Reference.

Other Changes for Release 12.2.0.1

The following other changes are made in this book:

- The previous single chapter about basic GeoRaster operations is divided into two chapters: GeoRaster Database Creation and Management and GeoRaster Data Query and Manipulation.

- Many new topics have been added. These include:

  – GeoRaster Spatial Web Services

  – MapViewer and GeoRaster

  – Using GDAL from the Spatial and Graph Installation

  – Loading with Blocking and Optimal Padding

  – Loading JPEG and JPEG 2000 Images Without Decompression

  – Checking GeoRaster Tables and Objects in the Database

  – Subsetting GeoRaster Objects with Polygon Clipping

  – Performing Cross-Schema Operations

  – Managing Memory to Improve Performance

  – Updating GeoRaster Objects in a Loop

  – Stack Statistical Analysis

  – Logical Operations

  – Image Warping

  – Image Affine Transformation and Scaling

  – Image Stretching, Normalization, Equalization, Histogram Matching, and Dodging

  – Image Filtering

  – Color Balancing During Mosaicking

Changes in Oracle Database 12*c* Release 1 (12.1.0.2)

The following are changes in *Oracle Spatial and Graph GeoRaster Developer's Guide* for Oracle Database 12*c* Release 1 (12.1.0.2).

- GeoRaster Not Enabled by Default

- JAVA API Supports All Release 12c New Features

- Third-Party Plug-in Installation Change

- GDAL-Based ETL Tool Enhancements

- SDO_GEOR_AGGR Package Enhancements

- Other Changes for Release 12.1.0.2

GeoRaster Not Enabled by Default

By default the GeoRaster feature is disabled after the Oracle Spatial and Graph is initially installed. To enable GeoRaster, follow these steps:

1. Connect to the database as SYS AS SYSDBA.

2. Enter the following statement:

```
EXECUTE MDSYS.enableGeoRaster;
```

For more information, see Section A.2, "Ensuring That GeoRaster Works Properly After an Installation or Upgrade" in *Oracle Spatial and Graph Developer's Guide*.

JAVA API Supports All Release 12*c* New Features

The GeoRaster Java API is updated to support all new packages, functions, procedures and other features in Release 12.1. These include raster algebra, advanced image processing, relational RDT tables, and new metadata. The GeoRaster Java API is included in *Oracle Spatial and Graph Java API Reference*.

Third-Party Plug-in Installation Change

Before you install the LizardTech plug-in, you must follow some new steps to prepare for the installation. For more information, see Third-Party Plug-ins for Compression.

GDAL-Based ETL Tool Enhancements

The user interface of the GDAL-based ETL tool is enhanced to improve usability and some new loading options are added. It is also updated with the latest GDAL version. For more information, see the updated user's guide (`$ORACLE_HOME/md/demo/georaster/tool/georaster_etl_user_guide.pdf`).

SDO_GEOR_AGGR Package Enhancements

The SDO_GEOR_AGGR.mosaicSubset and SDO_GEOR_AGGR.getMosaicSubset procedures accept the new `resFilter` keyword to the `mosaicParam` parameter to control the usage of the resolution filter. In addition, `HIGHRES` is a new `commonPointRule` keyword option to select the highest resolution image at the overlapping area in the mosaic. These keywords are included in Table 9-1.

Other Changes for Release 12.1.0.2

The following changes are made in this book:

- The new Section Parallel Compression_ Copying_ Subsetting_ Reprojection_ and Rectification describes how to apply parallel processing to compress, copying, subsetting, and rectification. Information has also been added to Parallel Processing in GeoRaster.
- The new Adding Temporary Tablespaces for GeoRaster Users is added.

Changes in Oracle Database 12*c* Release 1 (12.1.0.1)

The following are changes in *Oracle Spatial and Graph GeoRaster Developer's Guide* for Oracle Database 12*c* Release 1 (12.1.0.1).

- New Features
- Deprecated Features
- Other Changes

New Features

The following features are new in this release.

- Raster Algebra and Analytics
- Advanced Image Processing and Virtual Mosaic
- GeoRaster Core Enhancements
- Other New Server-Side Subprograms and Features
- New and Enhanced Client-Side Tools and Java API
- SDO_GEOR_AGGR Package
- SDO_GEOR_RA Package

Raster Algebra and Analytics

This release includes support for raster algebra and analytics, which are used in raster data analysis and GIS modeling to define raster analyses as algebraic expressions, so that users can easily apply algebraic functions on raster data to derive new results. For more information, see Raster Algebra and Analytics.

Support for raster algebra and analytics includes the following:

- New raster algebra language, which is an extension to the PL/SQL language. It includes an expression language and functions of map algebra local function type. (The raster algebra language is described in Raster Algebra Language.)
- Support for any arithmetic operations using the raster algebra language (SDO_GEOR_RA.rasterMathOp procedure).
- Support for cell value-based conditional queries using the raster algebra language (SDO_GEOR_RA.findCells procedure).
- Support for cell value-based conditional updates (raster editing) using the raster algebra language (SDO_GEOR_RA.rasterUpdate procedure).
- Support for classification for raster segmentation using the raster algebra language (SDO_GEOR_RA.classify procedure).
- New statistics generation functions to support interactive analysis on-the-fly (SDO_GEOR.generateStatisticsMax, SDO_GEOR.generateStatisticsMean, SDO_GEOR.generateStatisticsMedian, SDO_GEOR.generateStatisticsMin, SDO_GEOR.generateStatisticsMode, SDO_GEOR.generateStatisticsSTD).
- Enhancement of SDO_GEOR.generateStatistics to support on-the-fly analysis.

Advanced Image Processing and Virtual Mosaic

This release includes support for advanced image processing and virtual mosaic. For more information, see Image Processing and Virtual Mosaic.

Support for advanced image processing and virtual mosaic includes the following:

- Rectification of georeferenced raw images (SDO_GEOR.rectify)
- Orthorectification of georeferenced raw images with a height or a DEM (SDO_GEOR.rectify)
- Large-scale image appending (SDO_GEOR_AGGR.append)
- Advanced and faster large-scale mosaicking of georeferenced raw images (SDO_GEOR_AGGR.mosaicSubset).

- Virtual mosaic of any number of georeferenced raw images and on-the-fly query and cropping over the virtual mosaic (SDO_GEOR_AGGR.mosaicSubset, SDO_GEOR_AGGR.getMosaicSubset, SDO_GEOR_AGGR.getMosaicExtent, SDO_GEOR_AGGR.getMosaicResolutions, SDO_GEOR_AGGR.validateForMosaicSubset, and Virtual Mosaic)

- Support for internal reprojection/rectification, common point rules, and simple color balancing with both advanced mosaicking and virtual mosaic (see Table 9-1 in the SDO_GEOR_AGGR.mosaicSubset section)

- Image pyramid parallel generation and partial update (see Image Pyramiding: Parallel Generation and Partial Update)

- Special bitmap raster pyramiding algorithm (see Bitmap Pyramiding)

- Image masking (SDO_GEOR.mask)

- Image linear stretching (see Image Stretching) and image segmentation (see Image Segmentation)

- NDVI computation (see Vegetation Index Computation) and Tasseled Cap Transformation (see Tasseled Cap Transformation)

GeoRaster Core Enhancements

This release includes the following enhancements to GeoRaster core capabilities:

- Parallel processing capabilities, which can lead to significant improvements in performance, added into mosaicking, pyramiding, and all raster algebra functions and procedures (see Parallel Processing in GeoRaster)

- Operation progress reporting added into many new subprograms, and many new utility procedures and functions are added for this purpose (see Reporting Operation Progress in GeoRaster)

- Support for relational RDT tables as an additional raster data storage (see Raster Data Table)

- Ability to specify default alpha channel and pyramid level in the metadata (defined in GeoRaster Metadata XML Schema)

- New biquadratic interpolation method to interpolate cell value for any point in the grid; biquadratic interpolation method also added as a new resampling type in image operations and pyramiding (see Resampling and Interpolation)

- "OTHER" resampling type added to indicate an unknown or external pyramid resampling type imported from files (see Resampling and Interpolation)

- Significant performance improvements for pyramiding, reprojection, and scaling

Other New Server-Side Subprograms and Features

The following other new server-side subprograms and features are added:

- New function to query raster range, that is, the minimum and maximum cell values (SDO_GEOR.getRasterRange)

- Two new SDO_GEOR.getRasterBlockLocator formats, which allow specifying a single cell's row, column, and band numbers or specifying a point geometry in either the model coordinate space or the cell coordinate space

- Resolution generation for pyramids (SDO_GEOR.generateSpatialResolutions)

- Creation of empty raster blocks from existing raster blocks (SDO_GEOR_UTL.emptyBlocks)
- Re-creation of DML triggers (SDO_GEOR_UTL.recreateDMLTriggers)

New and Enhanced Client-Side Tools and Java API

The following new and enhanced client-side tools and Java API features are added:

- New GDAL-based ETL wizard for concurrent batch loading and exporting of large numbers of raster files (see GDAL-Based ETL Wizard for Concurrent Batch Loading and Exporting).
- GeoRaster viewer enhancement to display a virtual mosaic defined as one or a list of GeoRaster tables or views (see JAI-Based Viewer_ Loader_ and Exporter)
- Java API enhancements to support features such as ground control point (GCP) storage and manipulation, GCP georeferencing, reprojection, and grid interpolations

SDO_GEOR_AGGR Package

The new SDO_GEOR_AGGR package contains subprograms to support virtual mosaic, large-scale physical mosaicking, and appending operations on GeoRaster objects. For reference and usage information about the subprograms, see SDO_GEOR_AGGR Package Reference.

SDO_GEOR_RA Package

The new SDO_GEOR_RA package contains subprograms for raster algebra and analytic operations related to GeoRaster. For reference and usage information about the subprograms, see SDO_GEOR_RA Package Reference.

Deprecated Features

The following features are deprecated in this release, and will be desupported in a future release:

- Support for JPEG-B compression in GeoRaster

  You are encouraged to use the JPEG-F support, as explained in JPEG-B Support Deprecated.

Other Changes

The following are additional changes in the release:

- The new Raster Algebra and Analytics describes the raster algebra language (PL/SQL and algebraic expressions) and related raster operations, including conditional queries, cell value-based updates or edits, mathematical operations, classify, on-the-fly statistical analysis, and their applications in cartographic modeling.
- The new Image Processing and Virtual Mosaic describes the advanced image processing capabilities, including advanced georeferencing, reprojection, rectification, orthorectification, raster update, raster appending, large scale

physical mosaicking, virtual mosaic, and on-the-fly spatial queries over virtual mosaics.

1

GeoRaster Overview and Concepts

GeoRaster is a feature of Oracle Spatial and Graph 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 and Graph, which are documented in *Oracle Spatial and Graph 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.

> **✏ Note:**
>
> By default the GeoRaster feature is disabled after Oracle Spatial and Graph is initially installed. To enable GeoRaster, follow these steps:
>
> 1. Connect to the database as SYS AS SYSDBA.
>
> 2. Enter the following statement:
>
>    ```
>    EXECUTE MDSYS.enableGeoRaster;
>    ```
>
> You must also 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 and Graph 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 Maintaining GeoRaster Objects and System Data in the Database.

This chapter describes the core concepts and features of GeoRaster, including the GeoRaster data model and storage schema, georeferencing models, metadata support, resampling algorithms, pyramids, compression, parallel processing, loading and exporting capabilities, and the Java API. It contains the following major sections.

Topics:

- Vector and Raster Data
  Geographic features can be represented in vector or raster format, or both.

- 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.

- GeoRaster Data Model
  Raster data can have some or all of the following elements.

- GeoRaster Physical Storage
  GeoRaster optimizes the physical storage of metadata and data.

- Bands, Layers, and Metadata
  In GeoRaster, *band* and *layer* are different concepts.

- 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.

- Resampling and Interpolation
  Many image and raster transformations and operations involve pixel or cell resampling and interpolation.

- Pyramids
  Pyramids are subobjects of a GeoRaster object that represent the raster image or raster data at differing sizes and degrees of resolution.

- 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.

- NODATA Values and Value Ranges
  A NODATA value is used for cells whose values are either not known or meaningless.

- Compression and Decompression
  GeoRaster provides the following types of native compression to reduce storage space requirements for GeoRaster objects: JPEG (JPEG-F), JPEG 2000, and DEFLATE.

- GeoRaster and Database Management
  GeoRaster enables you to perform database management tasks.

- Parallel Processing in GeoRaster
  There are two types of parallel processing with GeoRaster.

- Reporting Operation Progress in GeoRaster
  For some resource-intensive operations, GeoRaster enables you to monitor and report their execution progress.

- GeoRaster PL/SQL API
  GeoRaster provides the SDO_GEOR, SDO_GEOR_ADMIN, SDO_GEOR_AGGR, SDO_GEOR_RA, and SDO_GEOR_UTL PL/SQL packages, which contain subprograms (functions and procedures) to work with GeoRaster data and metadata.

- GeoRaster Java API
  The Oracle Spatial and Graph GeoRaster Java API consists of interfaces and classes that support features available with the GeoRaster feature of Oracle Spatial and Graph.

- GeoRaster Spatial Web Services
  A web service enables developers of Oracle Spatial and Graph GeoRaster applications to provide raster data and metadata to their application users over the web. GeoRaster supports Open Geospatial Consortium (OGC) web services, specifically, Web Coverage Services (WCS) and Web Map Services (WMS).

- MapViewer and GeoRaster
  Oracle Fusion Middleware MapViewer (MapViewer) is a programmable tool for rendering maps using spatial data managed by Oracle Spatial and Graph or Oracle Locator (also referred to as Locator). It fully supports GeoRaster data types and is the web-based mapping and visualization application platform for GeoRaster.

- GeoRaster Tools: Viewer, Loader, Exporter
  Oracle Spatial includes tools for viewing, loading, and exporting GeoRaster data.

- GeoRaster PL/SQL and Java Sample Files
  GeoRaster includes several PL/SQL and Java sample code files that show common operations.

- README File for Spatial and Graph and Related Features
  Oracle Spatial and Graph includes a `README.txt` file.

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.

- Remote Sensing
- Photogrammetry
- Geographic Information Systems
- Cartography
- Digital Image Processing
- Geology, Geophysics, and Geochemistry

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 and Graph 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), 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 defines a generic raster data model that is component-based, logically layered, and multidimensional. The core data in a raster is a multidimensional array or 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 Bands_ Layers_ and Metadata.) 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 GeoRaster Metadata XML Schema), 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 SDO_GEOR Package Reference), which checks the validity of the metadata for a GeoRaster object.

The GeoRaster object data types, described in GeoRaster Data Types and Related Structures, 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
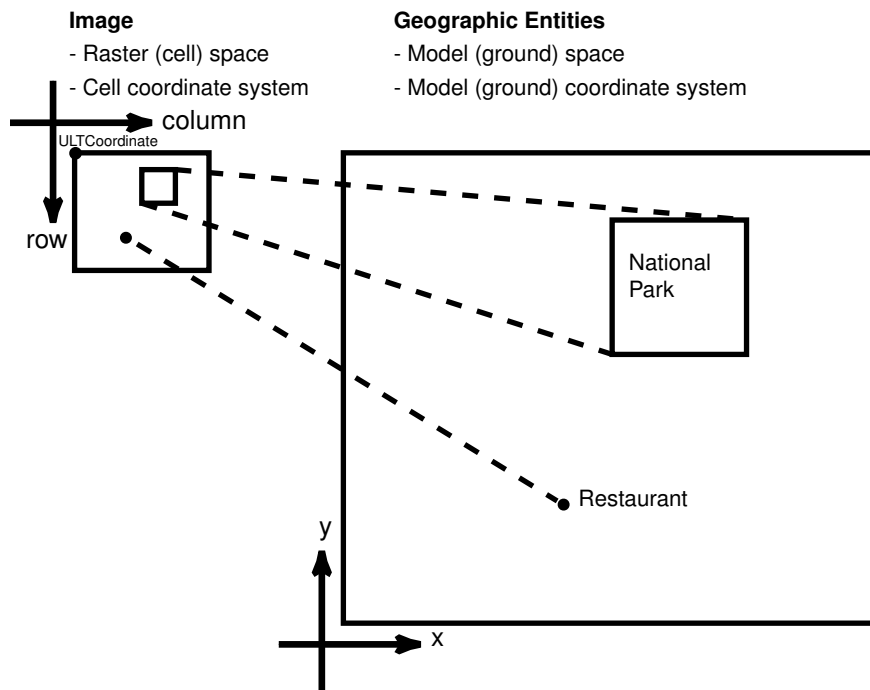
In Figure 1-1:

- 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 row dimension pointing downward and a column dimension pointing to the right, 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 row and column 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 row and column 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

Center-Based   **Upperleft-Based**
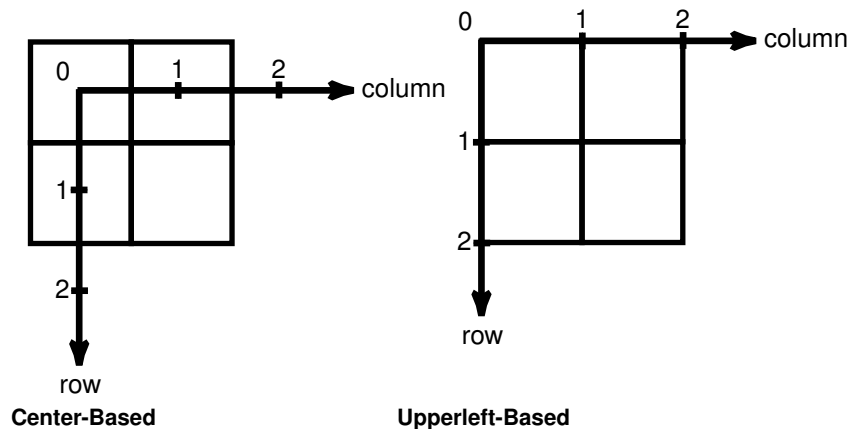
The default cell coordinate system has its origin at the *center* of a cell, and is called the center-based cell coordinate system. The other cell coordinate system has its origin at the upper-left corner of a cell, and is called the upperleft-based cell coordinate system. In both systems, the cells 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 and Graph Developer's Guide*.)

The GeoRaster model coordinate system is defined by an Oracle Spatial and Graph 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 4326 (EPSG WGS84), 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 Georeferencing.)

- **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 Bands_ Layers_ and Metadata.

1.4 GeoRaster Physical Storage

GeoRaster optimizes the physical storage of metadata and data.

As mentioned in GeoRaster Data Model, 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 GeoRaster Metadata XML Schema. 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 spatialExtent Attribute.

The GeoRaster metadata is stored using either the CLOB storage option or the binary XML storage option. The binary XML storage option for the GeoRaster metadata is the default, which saves disk space and improves performance. You can specify or change the storage option when you create a GeoRaster table.

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 Pyramids 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 Storage Parameters 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, or a relational table containing the same columns as the attributes of this object type, is called a **raster data table** (**RDT**).

The SDO_GEORASTER object stores and refers to an image or a raster data set. The SDO_RASTER object is an internal object for GeoRaster. The SDO_GEORASTER object fully encapsulates the raster data set's metadata and raster cell data, that is, a collection of SDO_RASTER objects. The relationship between the SDO_GEORASTER object and its SDO_RASTER objects is maintained by GeoRaster automatically. All interfaces of GeoRaster functions and procedures deal with the SDO_GEORASTER objects only; the SDO_RASTER objects of a SDO_GEORASTER object are *internally* handled automatically. The SDO_GEORASTER object is the major interface for users to build and manage a GeoRaster database; you only need to use the SDO_RASTER object to create raster data tables (RDTs).

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
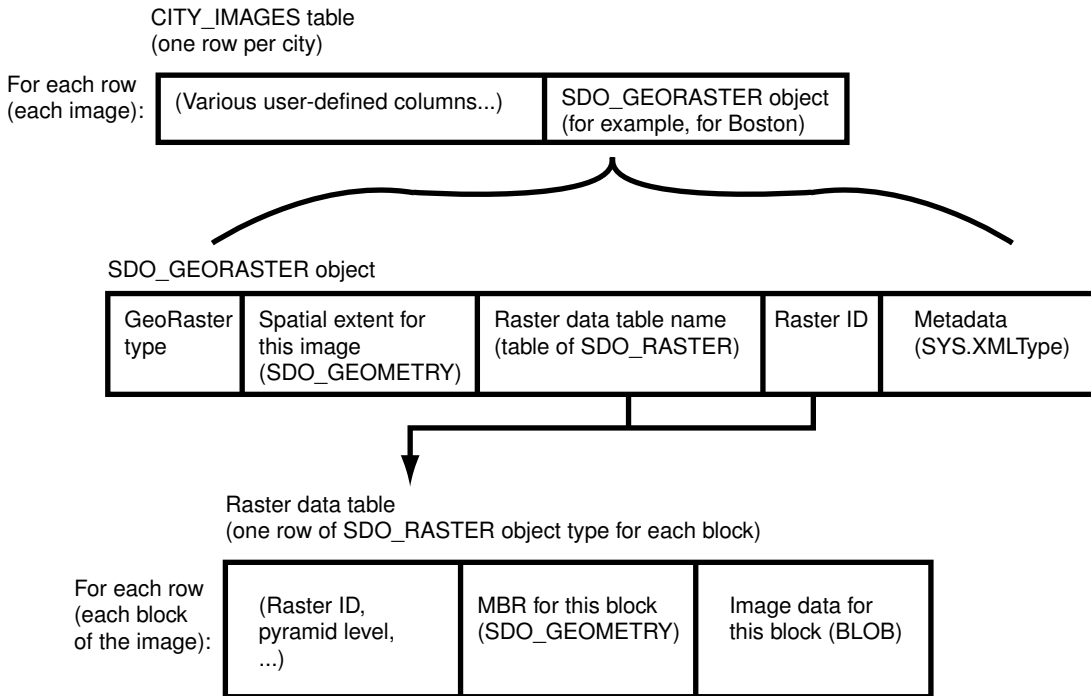
As shown in Figure 1-3:

- 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 Raster Data Table.

The SDO_GEORASTER and SDO_RASTER object types are described in detail in GeoRaster Data Types and Related Structures.

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
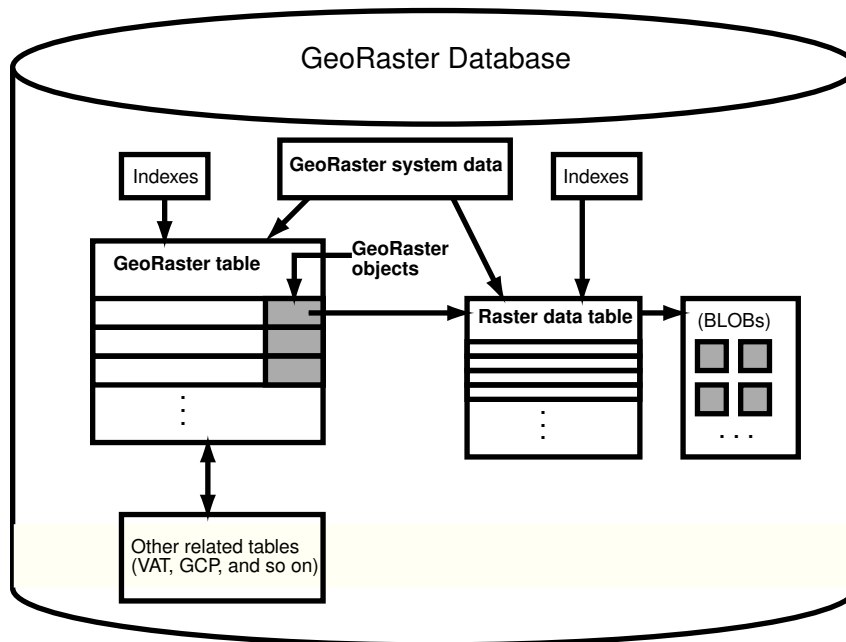
In Figure 1-4:

- 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 Storage Parameters) 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)) 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 Indexing GeoRaster Objects.

- Additional information, such as ground control points (GCPs) and value attribute tables (VATs), can be related to the GeoRaster objects.

You generally maintain a one-to-many relationship between a GeoRaster table and its associated raster data tables, even though they could have a many-to-many relationship. That is, let a raster data table only contain cell data of GeoRaster objects that belong to the same GeoRaster table. A GeoRaster table can contain a large number (potentially unlimited) of GeoRaster objects. An RDT should be used to contain the raster blocks of a limited number of GeoRaster objects, depending on the size of the rasters.

The following considerations apply to schema, table, and column names that are stored in any Oracle Spatial and Graph 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 Raster Data Table.

- Storage Parameters
- Raster Data Table
- Blank and Empty GeoRaster Objects
- Empty Raster Blocks
- Cross-Schema Support with GeoRaster

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 SDO_GEOR Package Reference.

> **Note:**
>
> For any numbers in string (VARCHAR2) parameters to GeoRaster subprograms, the period (.) must be used for any decimal points regardless of the locale.

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 (512,512,*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=(512,512)` specifies row and column block sizes of 512 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.
	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`

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

Keyword	Explanation
compression	Specifies the compression type to be applied to the GeoRaster object. Must be one of the following values: `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 Compression and Decompression. 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 Blank and Empty GeoRaster Objects.)
interleaving	Specifies the interleaving type. (Interleaving is explained in Bands_ Layers_ and Metadata.) Must be one of the following values: `BSQ` (band sequential), `BIL` (band interleaved by line), or `BIP` (band interleaved by pixel). Example: `interleaving=BSQ`
parallel	Specifies the degree of parallelism for the compression operation. (This parameter is ignored when a subprogram call specifies the `parallelParam` parameter.) If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. Must be used with the `compression` storage parameter. Parallelism is supported for the following compression operations:
	• From `NONE` to `JPEG-F`
	• From `NONE` to `DEFLATE`
	• From `JPEG-F` to `NONE`
	• From `DEFLATE` to `NONE`
	Parallelism is **not** supported for the following compression operations:
	• From `JPEG-F` to `DEFLATE`
	• From `DEFLATE` to `JPEG-F`

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

Keyword	Explanation
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 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 JPEG Compression of GeoRaster Objects. 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, 'blocking=OPTIMALPADDING
    blocksize=(512,512) 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 Updating GeoRaster Objects Before Committing.

Example 1-1 and many examples in SDO_GEOR Package Reference 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, or a relational table with the following column definitions:

```
rasterID          NUMBER,
pyramidLevel      NUMBER,
bandBlockNumber   NUMBER,
rowBlockNumber    NUMBER,
columnBlockNumber NUMBER,
blockMBR          SDO_GEOMETRY,
rasterBlock       BLOB
```

The raster data table, whether an object table or a relational table, 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.

Creating a raster data table enables you to control the placement and storage characteristics of the RDT (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; however, always consider sharing an RDT for a certain number of smaller GeoRaster objects to avoid creating too many RDTs. 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.

Never insert or delete any rows directly in a raster data table. The rows in the appropriate RDTs are automatically inserted or deleted when GeoRaster objects are created with raster data or deleted from a GeoRaster table.

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 smaller than or close to 4 KB (such as 64 by 64) is usually a bad choice, because 4 KB is the threshold for storing an Oracle BLOB out-of-line.

For information about creating object or relational raster data tables, see Creating Raster Data Tables.

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 Bitmap Masks) 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 or READ privilege on the GeoRaster tables and their associated raster data tables. For cross-schema update of GeoRaster objects, you must have the SELECT or READ privilege and the INSERT, UPDATE, and DELETE privileges on the GeoRaster tables and their associated raster data tables.

The ALL_SDO_GEOR_SYSDATA view (described in GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)) 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.

For examples of cross-schema GeoRaster operations, see Performing Cross-Schema Operations.

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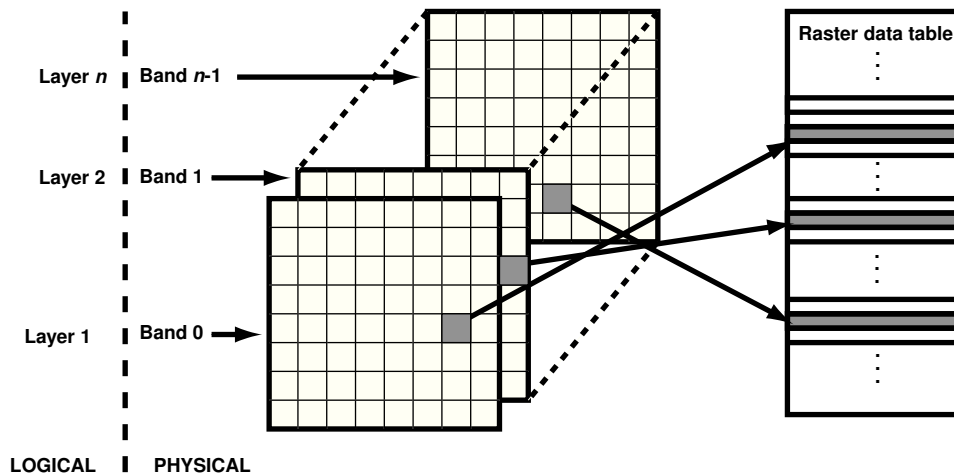
The GeoRaster XML metadata refers to the object layer and to layers. The **object layer** refers to the whole GeoRaster object, which may or may not contain multiple layers. If the GeoRaster object contains multiple layers, each layer is a sublayer of the object layer, and it refers to a single band.

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 GeoRaster Metadata XML Schema. the SDO_GEOR_HISTOGRAM object type in SDO_GEOR_HISTOGRAM Object Type, the SDO_GEOR_COLORMAP object type in SDO_GEOR_COLORMAP Object Type, SDO_GEOR_GRAYSCALE object type in SDO_GEOR_GRAYSCALE Object Type, and the SDO_GEOR_SRS object type in SDO_GEOR_SRS Object Type.

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 Functional Fitting Georeferencing Model) and the stored function model (explained in Ground Control Point (GCP) Georeferencing Model) 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 Georeferencing GeoRaster Objects and Advanced Georeferencing. To rectify and orthorectify a GeoRaster object, see Image Rectification and Image Orthorectification.

- Functional Fitting Georeferencing Model
- Ground Control Point (GCP) Georeferencing Model
- Cell Coordinate and Model Coordinate Transformation

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

$$\sum_{i=0}^{m1} \sum_{j=0}^{m2} \sum_{k=0}^{m3} a_{ijk} \, x_n^i y_n^j z_n^k$$

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 SDO_GEOR_SRS Object Type.

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 Ground Control Point (GCP) Georeferencing Model

GeoRaster supports ground control point (GCP) storage and georeferencing. A **ground control point** (GCP), or simply a **control point**, 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 and Graph 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 SDO_GEOR_GCP Object Type), the SDO_GEOR_GCP_ COLLECTION collection type (see SDO_GEOR_GCP_ COLLECTION Collection Type), and the SDO_GEOR_GCPGEOREFTYPE object type (see SDO_GEOR_GCPGEOREFTYPE Object Type).

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 Functional Fitting Georeferencing Model 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.

For more information, see Advanced 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

GeoRaster Data Model. 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 mth row, starting at 0 for the first row) of the cell relative to the ULTCoordinate.

- n = Column index (that is, the nth 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 Resampling and Interpolation

Many image and raster transformations and operations involve pixel or cell resampling and interpolation.

GeoRaster supports the following standard resampling and interpolation methods:

- Nearest neighbor (NN)

- Bilinear interpolation using 4 neighboring cells (BILINEAR)

- Biquadratic interpolation using 9 neighboring cells (BIQUADRATIC)

- Cubic convolution using 16 neighboring cells (CUBIC)

- Average using 4 neighboring cells (AVERAGE4)

- Average using 16 neighboring cells (AVERAGE16)

- OTHER

The keywords for these resampling types are defined in the `resamplingType` element definition in the GeoRaster XML metadata schema (described in GeoRaster Metadata XML Schema). Except for `OTHER`, the keywords can be used in several subprograms including the following:

- SDO_GEOR.generatePyramid

- SDO_GEOR.scaleCopy

- SDO_GEOR.reproject

- SDO_GEOR.rectify

- SDO_GEOR_AGGR.append

- SDO_GEOR_AGGR.getMosaicSubset

- SDO_GEOR_AGGR.mosaicSubset

The resampling type `OTHER` is used only to indicate an unknown or external resampling type when the pyramids of a GeoRaster object are generated or imported from external sources, such as a file.

Raster data deals with real world phenomena that vary continuously over space. This data is usually associated with **grid interpolation**, a method for interpolating values at spatial positions between the cells or within the cells. In GeoRaster, SDO_GEOR.evaluateDouble is the grid interpolation function. It uses the same keywords for interpolation methods as those for resampling.

1.8 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 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

Pyramid
Level 2 →

Pyramid
Level 1 →
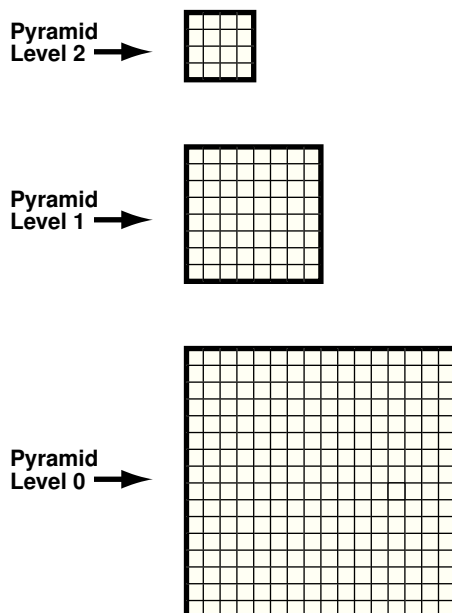
Pyramid
Level 0 →

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 GeoRaster Data Model. 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. 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 standard resampling methods described in Resampling and Interpolation.

The following subprograms 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.

1.9 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 Bands, Layers, and Metadata.

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 NODATA Values and Value Ranges.

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. You can also apply the masking operation inside the database using the SDO_GEOR.mask procedure. For more information about image masking, see Image Masking.

1.10 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`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, 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.

1.11 Compression and Decompression

GeoRaster provides the following types of native compression to reduce storage space requirements for GeoRaster objects: JPEG (JPEG-F), JPEG 2000, and DEFLATE.

- With JPEG (JPEG-F) and DEFLATE compression, each block of a GeoRaster object is compressed individually, as a distinct raster representation; and when a compressed GeoRaster object is decompressed, each block is decompressed individually

- With JPEG 2000 compression, each GeoRaster object is stored in a single BLOB as a JP2 file, in which the raster can be internally blocked.

For JPEG (JPEG-F) and DEFLATE compression, 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.

For JPEG 2000 compression, most GeoRaster operations can internally decompress the JP2 compressed GeoRaster object while performing the operation.

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 Storage Parameters. You can use the `compression` keyword in the `storageParam` parameter with all GeoRaster procedures. (For JPEG (JPEG-F) and DEFLATE compression, 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 Blank and Empty GeoRaster Objects.)

This section covers the following topics.

- JPEG (JPEG-F) Compression of GeoRaster Objects

- JPEG 2000 Compression of GeoRaster Objects

- DEFLATE Compression of GeoRaster Objects

- Decompression of GeoRaster Objects

- Third-Party Plug-ins for Compression

- Advanced LOB Compression

1.11.1 JPEG (JPEG-F) Compression of GeoRaster Objects

JPEG (JPEG-F) 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-F compression mode, which compresses objects in the full-format baseline JPEG format.

JPEG-F compression is described in the CCITT Rec. T.81 JPEG specification (or ICO/IEC IS 10918-1). 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.

JPEG-F is a lossy compression format. 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.

- JPEG-B Support Deprecated

1.11.1.1 JPEG-B Support Deprecated

GeoRaster support for JPEG-B compression, which compresses objects in the abbreviated baseline JPEG format, is deprecated, and will be desupported in a future release. If JPEG-B is specified in a parameter to a GeoRaster subprogram, JPEG-F compression is used instead. You are encouraged to use the JPEG-F support.

1.11.2 JPEG 2000 Compression of GeoRaster Objects

GeoRaster supports JPEG 2000 (JP2) compression on cell depth `8BIT_U` and `16BIT_U` raster images following the standard ISO/IEC 15444-1. A JPEG 2000 compressed GeoRaster object is stored in one raster block. The data in this raster block is in JP2 file format as described in standard ISO/IEC 15444-1 Annex I. The image contained in the JPEG 2000 compressed GeoRaster object can be internally tiled.

With JPEG 2000 compression, the pyramids are implicitly embedded in the JP2 compressed data, and thus there is no separate, explicit pyramid storage in the JP2 compressed GeoRaster object. The maximum level of pyramids that can be retrieved from a JP2 compressed GeoRaster object is `log2(min(tile_width, tile_height))`,

where `tile_width` and `tile_height` are the width and height of the internal tiles, respectively. Both lossy and lossless compressions are supported.

The SDO_GEOR.compressJP2 procedure is used to compress a GeoRaster object into JP2 compressed GeoRaster object. The SDO_GEOR.decompressJP2 procedure can be used to explicitly decompress a JP2 compressed GeoRaster object into another GeoRaster object. Other GeoRaster operations, such as rectification, mosaicking, and raster algebra – but not SDO_GEOR.changeCellValue, SDO_GEOR.reproject, SDO_GEOR.scaleScopy, and SDO_GEOR.mosaic – can internally decompress the JP2 compressed GeoRaster object while performing the operation.

Large images can be compressed, but the size is limited by memory and max number of tiles (max_mem_size / 20 * 65535). **To improve scalability and performance, always apply internal tiling.** The tile size can be specified using the `tileSize` keyword in the `compressParam` parameter of the SDO_GEOR.compressJP2 procedure. The maximum number of tiles supported is 65535.

1.11.3 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.

1.11.4 Decompression of GeoRaster Objects

You can decompress a compressed GeoRaster object in the database by specifying `compression=NONE` in the `storageParam` parameter. 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, Oracle 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.

1.11.5 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 and Graph GeoRaster.

Before you install the LizardTech plug-in, you must follow these steps:

1. Go to the `$ORACLE_HOME/md/admin` directory.

2. Connect to the database as `SYS AS SYSDBA`.

3. Enter the following SQL statement:

   ```
   SQL> @prvtgrlt.plb
   ```

To get the LizardTech plug-in and related information, contact LizardTech Corporation.

1.11.6 Advanced LOB Compression

You can use Oracle Database Advanced LOB Compression (described briefly in *Oracle Database SecureFiles and Large Objects Developer's Guide*) to achieve lossless compression of GeoRaster raster data tables (RDTs), thus compressing the GeoRaster objects. If you specify Advanced LOB Compression for LOB storage when you create a table (such as the rasterBlock column of an RDT), then the SecureFiles LOBs in all rows of that table are compressed using Advanced LOB Compression. The compression is transparent to GeoRaster, and thus no application changes are required. However, you should avoid using Advanced LOB 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.

The use of Advanced LOB Compression requires licensing for the Oracle Database Advanced Compression Option, which is described in *Oracle Database Licensing Information*. Note that the Oracle Database Advanced Compression Option is *not* required for GeoRaster compression operations that do not involve Advanced LOB Compression.

1.12 GeoRaster and Database Management

GeoRaster enables you to perform database management tasks.

These tasks are described in GeoRaster Database Creation and Management. 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.

For usage information related to the preceding subprograms, see Maintaining GeoRaster Objects and System Data in the Database.

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

- 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.

- 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.

- A raster data table (RDT) cannot be dropped or directly renamed using standard SQL statement as long as any GeoRaster object references that RDT.

For more information, see GeoRaster DML Trigger and Deleting GeoRaster Objects, and Performing Actions on GeoRaster Tables and RDTs.

1.13 Parallel Processing in GeoRaster

There are two types of parallel processing with GeoRaster.

- Parallel execution of SQL statements
- Parallelized GeoRaster procedures

Parallel execution of SQL statements allows most SQL statements, both query and DML, to run in parallel. When a SQL statement is executed, it is decomposed into individual steps or row-sources, which are identified as separate lines in an execution plan.

All GeoRaster read-only functions such as metadata-related query operations (that is, all GeoRaster metadata *get* functions and SDO_GEOR.validateGeoRaster) and all single-raster cell queries (SDO_GEOR.getCellValue and SDO_GEOR.evaluateDouble) are enabled for parallel query. This means that in a multi-CPU environment, if these functions are used to query many GeoRaster objects in one or more GeoRaster tables and if the SQL statement is made to run in parallel, the GeoRaster rows are automatically divided into multiple subsets, and multiple Oracle server processes will work simultaneously to process each subset to reduce the overall response time. By dividing the work to run a GeoRaster SQL statement among multiple processes, you can more quickly maintain spatial indexes and find GeoRaster objects based on their locations, various metadata, and attributes. You can also use the pipelined and parallel table function to implement more sophisticated procedures, including parallelizing some operations on a single GeoRaster object.

Parallelized GeoRaster procedures let you specify multiple subprocesses for simultaneous processing of a GeoRaster object. Some individual raster and image processing procedures are specifically implemented to support this type of parallelism. With these procedures, you simply specify an integer number for the degree of parallelism (DOP) as an input parameter, to cause the operation to be split into that number of subprocesses to process the subsets of a single GeoRaster object simultaneously. Each of those subprocesses runs independently. When all subprocesses are finished, the whole process is finished. The following procedures directly support this kind of parallel processing:

- SDO_GEOR.generatePyramid
- SDO_GEOR_RA.classify
- SDO_GEOR_RA.findCells
- SDO_GEOR_RA.rasterMathOp
- SDO_GEOR_RA.rasterUpdate
- SDO_GEOR_AGGR.mosaicSubset

Through the SDO_GEOR_AGGR.mosaicSubset procedure, other types of parallel operations are supported. These include parallel compression and decompression, parallel copying or change format copying, parallel subsetting, parallel reprojection, and parallel rectification. See Parallel Compression, Copying, and Subsetting for more information.

Imagery and raster data are typically very large, so the preceding operations can be time consuming. Therefore, when using multi-CPU or multicore servers, always consider using parallel processing to improve the performance.

1.14 Reporting Operation Progress in GeoRaster

For some resource-intensive operations, GeoRaster enables you to monitor and report their execution progress.

This capability applies to the execution of the following subprograms:

- SDO_GEOR_AGGR.getMosaicSubset
- SDO_GEOR_AGGR.mosaicSubset
- SDO_GEOR.generatePyramid
- SDO_GEOR.mosaic

To monitor and report on execution progress, you can use the following subprograms:

- SDO_GEOR_UTL.clearReportTable
- SDO_GEOR_UTL.createReportTable
- SDO_GEOR_UTL.disableReport
- SDO_GEOR_UTL.dropReportTable
- SDO_GEOR_UTL.enableReport
- SDO_GEOR_UTL.getAllStatusReport
- SDO_GEOR_UTL.getProgress
- SDO_GEOR_UTL.getStatusReport
- SDO_GEOR_UTL.isReporting
- SDO_GEOR_UTL.setClientID
- SDO_GEOR_UTL.setSeqID

For information about monitoring and reporting the progress of GeoRaster operations, see Monitoring and Reporting GeoRaster Operation Progress.

1.15 GeoRaster PL/SQL API

GeoRaster provides the SDO_GEOR, SDO_GEOR_ADMIN, SDO_GEOR_AGGR, SDO_GEOR_RA, and SDO_GEOR_UTL 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
- Format, transform, process, and analyze 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 (SDO_GEOR Package Reference), SDO_GEOR_ADMIN (SDO_GEOR_ADMIN Package Reference), SDO_GEOR_AGGR (SDO_GEOR_AGGR Package Reference), SDO_GEOR_RA (SDO_GEOR_RA Package Reference), and SDO_GEOR_UTL (SDO_GEOR_UTL Package Reference) PL/SQL packages. The subprograms are presented in alphabetical order in those chapters. Basic GeoRaster Operations, Raster Algebra and Analytics, and Image Processing and Virtual Mosaic describe operations that involve the use of many of those subprograms, including the general steps for calling them.

GeoRaster uses spatial indexing capabilities and related operations, which are described in *Oracle Spatial and Graph Developer's Guide*.

1.16 GeoRaster Java API

The Oracle Spatial and Graph GeoRaster Java API consists of interfaces and classes that support features available with the GeoRaster feature of Oracle Spatial and Graph.

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 major 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 also provides a virtual mosaic class to support advanced visualization applications. 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, a subset of a GeoRaster object, or a virtual mosaic, and for processing the images. This package depends upon and leverages JAI.

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

The Spatial and Graph 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`.

1.17 GeoRaster Spatial Web Services

A web service enables developers of Oracle Spatial and Graph GeoRaster applications to provide raster data and metadata to their application users over the web. GeoRaster supports Open Geospatial Consortium (OGC) web services, specifically, Web Coverage Services (WCS) and Web Map Services (WMS).

WCS offers multidimensional coverage data (imagery and gridded rasters) for access over the Internet. You can publish GeoRaster objects in the database and allow users to retrieve the raster data over the web, including subsetting, reprojection, and GeoTIFF format support. WCS is described in a chapter in the *Oracle Spatial and Graph Developer's Guide*.

MapViewer supports the rendering of data delivered using the OGC Web Map Service (WMS) protocol, specifically the WMS 1.1.1 and 1.3.0 implementation specifications. It supports any images and gridded rasters stored in GeoRaster. WMS is described in an appendix in the *User's Guide to Oracle MapViewer*.

1.18 MapViewer and GeoRaster

Oracle Fusion Middleware MapViewer (MapViewer) is a programmable tool for rendering maps using spatial data managed by Oracle Spatial and Graph or Oracle Locator (also referred to as Locator). It fully supports GeoRaster data types and is the web-based mapping and visualization application platform for GeoRaster.

MapViewer allows you to define GeoRaster themes (based on an individual GeoRaster object) and GeoRaster virtual mosaic themes (based on a collection of GeoRaster objects). You can use the Map Builder tool to define GeoRaster themes and virtual mosaic themes, and to specify image processing operations and rendering styles.

MapViewer also has a map tile server, which is a map image caching engine that fetches, caches, and serves pregenerated, fixed-size map image tiles. You can leverage it to cache GeoRaster images in the middle tier to speed up applications.

MapViewer is documented in the*User's Guide to Oracle MapViewer*.

1.19 GeoRaster Tools: Viewer, Loader, Exporter

Oracle Spatial includes tools for viewing, loading, and exporting GeoRaster data.

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 and Graph partner solutions information at `http://www.oracle.com/technetwork/database-options/spatialandgraph/learnmore/` and the open source GDAL support at `http://www.gdal.org/frmt_georaster.html`.

GeoRaster also includes the following client-side tools:

- JAI-based GeoRaster viewer, loader and exporter
- GDAL-based ETL wizard for concurrent batch loading and exporting of large numbers of image and raster files

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 and Graph installation directory of `$ORACLE_HOME/md`):

```
$ORACLE_HOME/md/demo/georaster/tool/georastertool.jar
```

In addition, GDAL itself is included with the Oracle Spatial and Graph installation.

- JAI-Based Viewer, Loader, and Exporter
- GDAL-Based ETL Wizard for Concurrent Batch Loading and Exporting
- Using GDAL from the Spatial and Graph Installation

1.19.1 JAI-Based Viewer, Loader, and Exporter

The GeoRaster JAI-based tools include a viewer, a loader, and an exporter. These tools are intended for DBAs and application developers. The viewer is especially useful for examining all types of GeoRaster objects and their metadata. It can also

display a virtual mosaic defined as one or a list of GeoRaster tables or views. The loader and exporter are lightweight tools for conveniently load and export a limited number of image and raster files one at a time. The `$ORACLE_HOME/md/demo/georaster/tool/README.txt` file includes helpful usage information and instructions for using the following tools:

- GeoRaster viewer displays GeoRaster objects and metadata, as well as virtual mosaics. 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 Viewing GeoRaster Objects.)

  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 `$ORACLE_HOME/md/demo/georaster/tool/JAI_based_tools_user_guide.txt` file.

- GeoRaster loader, which loads raster data into the GeoRaster objects. It 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 SDO_GEOR Package Reference. 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 SDO_GEOR Package Reference. 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 `$ORACLE_HOME/md/demo/georaster/tool/JAI_based_tools_user_guide.txt` 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 Oracle 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.

1.19.2 GDAL-Based ETL Wizard for Concurrent Batch Loading and Exporting

GeoRaster includes an ETL wizard tool to automate and enable concurrent batch loading and exporting of various image and raster files using GDAL. This powerful tool can load and export large numbers of raster and image files in batches and concurrently.

It defines an XML schema and provides a graphical user interface to create loading and exporting description files in XML. Each description file describes how to load or export a series of raster files into or from GeoRaster in a batch. After the XML description files are created, you can use the same wizard tool to invoke multiple description files to concurrently load and export raster files in batches. Any run-time failures are caught and logged, but they do not stop the batch loading or exporting processes. This tool supports all raster formats supported by GDAL.

To use this wizard, you must install the demo files from the Oracle Database Examples media (see *Oracle Database Examples Installation Guide*). After the installation, this wizard is in the following `.jar` file (assuming the default Spatial and Graph installation directory of *$ORACLE_HOME*/md):

`$ORACLE_HOME/md/demo/georaster/tool/georastertool.jar`

The `$ORACLE_HOME/md/demo/georaster/tool/README.txt` file describes how to set up GDAL and launch the wizard.

The `$ORACLE_HOME/md/demo/georaster/tool/GDAL_based_etl_user_guide.pdf` file describes the usage in detail.

1.19.3 Using GDAL from the Spatial and Graph Installation

The GeoSpatial Data Abstraction Library (GDAL)GeoSpatial Data Abstraction Library (GDAL) is an Open Source software library that supports many data formats and services. Oracle Spatial geometries and GeoRaster objects are supported by the GDAL library, command line tools, and programing interface.

GDAL is distributed with Oracle Spatial and Graph, where it is installed under `$ORACLE_HOME/md/gdal` on Linux systems and `%ORACLE_HOME% \md\gdal` on Windows systems. (GDAL is not distributed with Oracle Spatial and Graph on other platforms.)

To prepare GDAL for command line use, you must add the GDAL `bin`, `data`, `lib`, and `plugins` folders to the system environment variables.

The following examples set up GDAL on Linux:

```
setenv GDAL_HOME ${ORACLE_HOME}/md/gdal
setenv GDAL_DATA ${GDAL_HOME}/data
setenv GDAL_DRIVER_PATH ${GDAL_HOME}/lib/gdalplugins
setenv PATH ${GDAL_HOME}/bin:${PATH}
setenv LD_LIBRARY_PATH ${GDAL_HOME}/lib:${LD_LIBRARY_PATH}
```

The following examples set up GDAL on Windows:

```
set GDAL_HOME=%ORACLE_HOME%\md\gdal
set GDAL_DATA=%GDAL_HOME%\data
set GDAL_DRIVER_PATH=%GDAL_HOME%\bin\gdalplugins
set PATH=%GDAL_HOME%\bin;%PATH%
```

The preceding examples assume that Oracle OCI shared libraries are already configured in the system. Oracle OCI shared libraries can be found in the Oracle Database or Instant Client installation.

The following example adds Oracle Instant Client to the Windows PATH variable:

```
set PATH=C:\instantclient_12_1;%PATH%
```

The scripts to automatically set up GDAL are `setup_gdal.conf` and `setup_gdal.bat`, which can be found in the following folder: `$ORACLE_HOME/md/demo/georaster/tool`

Loading Raster Data and its subtopics provide explanations and examples of how to use GDAL to load raster files into GeoRaster.

1.20 GeoRaster PL/SQL and Java Sample Files

GeoRaster includes several PL/SQL and Java sample code files that show common operations.

If you installed the example files from the Oracle Database Examples media (see *Oracle Database Examples Installation Guide*), these sample code files are in the following directories under the Spatial and Graph installation directory (which by default is *$ORACLE_HOME*/md):

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

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

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

1.21 README File for Spatial and Graph and Related Features

Oracle Spatial and Graph includes a `README.txt` file.

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

`$ORACLE_HOME/md/doc/README.txt`

2

GeoRaster Data Types and Related Structures

The object-relational implementation of GeoRaster consists of a set of object data types for storing data and system data.

Each image or gridded raster data is stored in a column of type SDO_GEORASTER, and the blocks in that raster data are stored in a raster data table of type SDO_RASTER, as explained and illustrated in GeoRaster Physical Storage. This chapter contains the following major sections.

Topics:

- 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 at least one column of type SDO_GEORASTER are referred to as GeoRaster tables.

- 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.

- Other GeoRaster Types
  GeoRaster also porvides some other data types.

- 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.

- GeoRaster XML Schema
  GeoRaster defines an XML schema to store and manage the GeoRaster metadata.

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 at least one column 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.

- rasterType Attribute

- [spatialExtent Attribute](#)

- [rasterDataTable Attribute](#)

- [rasterID Attribute](#)

- [metadata 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 [Bands, Layers, and Metadata](#).)

- `[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 and Graph 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 [GeoRaster Tools: Viewer, Loader, Exporter](#)).

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 and Graph 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 metadata Attribute. For information about indexing GeoRaster data, see Indexing GeoRaster Objects.

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 or a relational table that includes all columns defined by object type SDO_RASTER. It contains a row 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 SDO_RASTER Object Type and the Raster Data Table.

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 GeoRaster Metadata XML Schema. 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.

The default storage option for GeoRaster metadata is binary XML.

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 rasterDataTable Attribute. 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 or as a relational table that includes all columns defined by object type SDO_RASTER. 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.

- rasterID Attribute
- pyramidLevel Attribute
- bandBlockNumber Attribute
- rowBlockNumber Attribute
- columnBlockNumber Attribute
- blockMBR Attribute
- rasterBlock 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 rasterID Attribute.) 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 Pyramids.

This attribute and the `bandBlockNumber` attribute (described in bandBlockNumber Attribute) 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 Bitmap Masks.

2.2.3 bandBlockNumber Attribute

The `bandBlockNumber` attribute identifies the block number along the band dimension. For information about bands and layers, see Bands_ Layers_ and Metadata. For more information about how the `bandBlockNumber` attribute is used with bitmap masks and their pyramids, see Bitmap Masks.

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

GeoRaster also porvides some other data types.

In addition to SDO_GEORASTER, SDO_RASTER, and SDO_RANGE_ARRAY and SDO_RANGE, 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.

• SDO_GEOR_HISTOGRAM Object Type

• SDO_GEOR_COLORMAP Object Type

• SDO_GEOR_GRAYSCALE Object Type

• SDO_RASTERSET Collection Type

• SDO_GEOR_SRS Object Type

• SDO_GEOR_GCP Object Type

• SDO_GEOR_GCP_ COLLECTION Collection Type

• SDO_GEOR_GCPGEOREFTYPE Object Type

Related Topics:

- SDO_GEORASTER Object Type

- SDO_RASTER Object Type and the Raster Data Table

- NODATA Values and Value Ranges

2.3.1 SDO_GEOR_HISTOGRAM Object Type

In GeoRaster, the histogram is stored in the GeoRaster metadata using the XML schema defined in GeoRaster Metadata XML Schema. 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 GeoRaster Metadata XML Schema. 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

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

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

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

The SDO_GEOR_COLORMAP object type is defined as:

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

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

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 GeoRaster Metadata XML Schema. 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;
```

Related Topics:

• SDO_RASTER Object Type and the Raster Data Table

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 GeoRaster Metadata XML Schema. 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.

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

Attribute	Description
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 GeoRaster Data Model.
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
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 Functional Fitting Georeferencing Model.

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

Attribute	Description
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 Functional Fitting Georeferencing Model.
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 Functional Fitting Georeferencing Model.
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 Functional Fitting Georeferencing Model.
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 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 SDO_GEOR Package Reference.

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 GeoRaster Metadata XML Schema. 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).

Table 2-5   (Cont.) SDO_GEOR_GCP Object Type Attributes

Attribute	Description
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;
```

Related Topics:

•   SDO_GEOR_GCP Object Type

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 GeoRaster Metadata XML Schema. 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 SDO_GEOR_GCP_ COLLECTION Collection Type).
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.

- TABLE_NAME Column
- COLUMN_NAME Column
- METADATA_COLUMN_NAME Column
- RDT_TABLE_NAME Column
- RASTER_ID Column
- OTHER_TABLE_NAMES Column

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.)

Related Topics:

- SDO_RASTER Object Type and the Raster Data Table

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 GeoRaster Metadata XML Schema. 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.

Table 2-7   (Cont.) SDO_GEOR_XMLSCHEMA_TABLE Table Columns

Column Name	Data Type	Description
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 GeoRaster Metadata XML Schema.

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 Database Creation and Management

This chapter describes how to perform important GeoRaster database creation and management operations. A typical workflow to build and manage a GeoRaster database consists of most or all of the steps described.
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 insert, update, reformat, compress, decompress, or delete the metadata or cell data of the GeoRaster object and before you commit the changes (see Updating GeoRaster Objects Before Committing). For some examples of these operations, see the demo files described in GeoRaster PL/SQL and Java Demo Files and the examples in SDO_GEOR Package Reference.

See also the operations in GeoRaster Data Query and Manipulation.

Other chapters in this book cover advanced topics (Raster Algebra and Analytics and Image Processing and Virtual Mosaic), and provide detailed reference information about GeoRaster PL/SQL packages (SDO_GEOR Package Reference, SDO_GEOR_ADMIN Package Reference, SDO_GEOR_AGGR Package Reference, SDO_GEOR_RA Package Reference, and SDO_GEOR_UTL Package Reference).

Topics:

- Adding Data Files and Temporary Tablespaces for GeoRaster Users
- Creating the GeoRaster Table and Raster Data Tables
- Creating New GeoRaster Objects
- Loading Raster Data
- Validating GeoRaster Objects
- Georeferencing GeoRaster Objects
- Generating and Setting Spatial Extents
- Indexing GeoRaster Objects
- Viewing GeoRaster Objects
- Exporting GeoRaster Objects
- Using GeoRaster with Workspace Manager and Label Security
- Maintaining Efficient Tablespace Use by GeoRaster Objects
- Checking GeoRaster Tables and Objects in the Database
- Maintaining GeoRaster Objects and System Data in the Database
- Transferring GeoRaster Data Between Databases

- Using Transportable Tablespaces with GeoRaster Data

3.1 Adding Data Files and Temporary Tablespaces for GeoRaster Users

A GeoRaster database is typically very large. For storage and performance reasons, a database schema should use one or more user tablespaces for GeoRaster data storage (avoid using the system tablespace for storing GeoRaster data), and you should add data files to the tablespaces appropriately. If Oracle Automatic Storage Management (Oracle ASM) or a bigfile tablespace is not being used, you should create many data files for each tablespace and distribute the data files on different disks if possible. You also should create data files or alter existing data files, so that they automatically increase in size when more space is needed in the database.

A GeoRaster table can contain a large (potentially almost unlimited) number of GeoRaster objects. A raster data table (RDT) should be used to contain the raster blocks of a limited number of GeoRaster objects, depending on the size of the rasters. In contrast with GeoRaster tables, an RDT should not grow too large, unless partitioning is to be applied. Also, RDTs can be created on different tablespaces, so that the raster blocks are distributed to different disks. (See also Creating the GeoRaster Table and Raster Data Tables.)

A GeoRaster database may use a temporary tablespace for some operations. When compression is involved in GeoRaster operations, particularly for large scale mosaicking operations, some temporary spaces are needed to store intermediate compressed or uncompressed data. If the GeoRaster user does not have a temporary tablespace, the database system temporary tablespace is used. This is not efficient and may slow down the mosaicking and other operations. Therefore, you should always create temporary tablespaces for GeoRaster users. For example:

```
CONNECT system/<password>;
CREATE TEMPORARY TABLESPACE geor_temp TEMPFILE 'geor_temp_1.f' SIZE 1G AUTOEXTEND
ON;
ALTER USER <georaster_user> TEMPORARY TABLESPACE geor_temp;
```

In general, the amount of temporary space needed is limited. However, for large scale mosaicking, if the result is to be compressed, the temporary space needed is equal to the uncompressed image size of the result. Therefore, specify AUTOEXTEND ON when you create temporary tablespaces for GeoRaster users.

3.2 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.

- Creating a GeoRaster Table
- Creating Raster Data Tables
- GeoRaster DML Trigger

3.2.1 Creating a GeoRaster Table

A GeoRaster table is any table that includes at least one column of type SDO_GEORASTER. The 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 GeoRaster Physical Storage.

3.2.2 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 as an object table or as a relational table. You should use the LOB storage format SecureFiles LOBs (SecureFiles) when creating RDTs. Using SecureFiles significantly improves the performance of GeoRaster operations, compared to using the original LOB storage paradigm BasicFiles LOBS (BasicFiles).

Example 3-2 creates a raster data table using SecureFiles. The RDT will be used to store all raster blocks of one or many GeoRaster objects in the CITY_IMAGES table or other GeoRaster tables. (The association between a GeoRaster object and a raster data table is not made until you create a GeoRaster object, as explained in Creating New GeoRaster Objects.)

Example 3-2   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
    (CACHE);
```

Example 3-3   Creating a Raster Data Table (Relational) Using SecureFiles

Example 3-3 creates a raster data table with the same name as in Example 3-2, also using SecureFiles, but creating it as a relational table instead of an object table.

```
CREATE TABLE city_images_rdt
  (rasterID NUMBER,
  pyramidLevel NUMBER,
  bandBlockNumber NUMBER,
  rowBlockNumber NUMBER,
  columnBlockNumber NUMBER,
  blockMBR SDO_GEOMETRY,
  rasterBlock BLOB,
  CONSTRAINT pkey PRIMARY KEY (rasterId, pyramidLevel, bandBlockNumber,
    rowBlockNumber, columnBlockNumber))
  LOB (rasterblock) STORE AS SECUREFILE(cache);
```

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 SecureFiles LOBs must be created in a tablespace with the automatic segment space management option. For information about using Oracle SecureFiles and performance considerations for BasicFiles 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 Raster Data Table.

3.2.3 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. In some scenarios, such as a database upgrade or a data migration, you can call the SDO_GEOR_UTL.recreateDMLTriggers procedure to re-create the DML triggers on all GeoRaster columns.

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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)). 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.

3.3 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 Creating Raster Data Tables.

To avoid potential GeoRaster data problems (some of which are described in Maintaining GeoRaster Objects and System Data in the Database), an initialized GeoRaster object must be **registered** in the GeoRaster system views, which is done automatically when you insert the GeoRaster object into a GeoRaster table. This should be done 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.

3.4 Loading Raster Data

To load and export imagery or raster data, you can consider third-party ETL tools (see the note in GeoRaster Tools: Viewer_ Loader_ Exporter). For example, you can use the gdal_translate command line and other GDAL utilities (http://www.gdal.org/gdal_utilities.html), which fully supports GeoRaster through the Oracle Spatial GeoRaster driver.

You can also use features in GeoRaster to load raster data, With GeoRaster, you have the following options:

- Use the GDAL based ETL tool for concurrent batch loading and exporting. This tool is described in GDAL-Based ETL Wizard for Concurrent Batch Loading and Exporting.

- Call the SDO_GEOR.importFrom procedure to load images into GeoRaster objects.

- Use the GeoRaster JAI-based loader tool or viewer tool, which are described in JAI-Based Viewer_ Loader_ and Exporter.

With the last two options (SDO_GEOR.importFrom and JAI-based tool), 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. To load and export the georeferencing information of GeoTIFF images, the GeoTIFF libraries are required. See Georeferencing GeoRaster Objects for instructions.

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 Validating GeoRaster Objects.

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 and Graph Developer's Guide*.

- Loading with Blocking and Optimal Padding
- Loading JPEG and JPEG 2000 Images Without Decompression
- Reformatting the Source Raster Before Loading

3.4.1 Loading with Blocking and Optimal Padding

Unless you want to load JPEG or JPEG2000 images and store them without any change, when you load an image or raster file into GeoRaster object, always consider and apply appropriate blocking of the data, because file formats might have very different blocking schemes. In general, blocking sizes should be 512x512 or larger. There is no absolute rule for the blocking sizes, but the larger the raster, the larger the blocking sizes you might use. For regular rasters, 512x512 to 2048x2048 is appropriate. For very small images (less than 1024x1024x3), no blocking may be a good choice. Avoid blocking sizes that are either too small (such as 64x64 and 128x128) or too large, and avoid extreme blocking sizes such as 0.5 (one-half), 1, or 8 rows of pixels per block. Generally, the rectangular shape of blocks should be a square or close to a square. For different applications, you might tune the blocking to balance efficient storage with optimal performance.

You should also always apply optimal padding during loading. In other words, specify `blocking=OPTIMALPADDING` in addition to specifying `blocksize`. GeoRaster applies padding to the right column and bottom row of blocks to make them the same size as other blocks. If the block size is not optimal for a specific raster, the default resulting padding would waste some storage space. When you specify`blocking=OPTIMALPADDING`, all GeoRaster procedures and the ETL tools automatically adjust the GeoRaster dimension size array so that it will be optimal for reducing the amount of padding in GeoRaster object storage. The adjustment is always made around the user-specified values. See the explanation of the `blocking` keyword in the table in the Usage Notes for the SDO_GEOR_UTL.calcOptimizedBlockSizeprocedure.

For how to apply optimal padding when using the GDAL command line, see the following example:

```
gdal_translate -of georaster /images/image_1.tif \
      georaster:georaster/georaster@my_db, image_table, raster \
      -co "insert=(id,label,raster) values (1, 'image_1', sdo_geor.init('rdt_table',
1)" \
```

```
-co blockxsize= 512 \
-co blockysize=512 \
-co blockbsize=3 \
-co blocking=optimalpadding \
-co interleave=BIP
```

For how to apply optimal padding when using the SDO_GEOR.importFrom procedure, see the examples in the reference topic for that procedure.

3.4.2 Loading JPEG and JPEG 2000 Images Without Decompression

GeoRaster supports JPEG compression, in which the GeoRaster blocks are stored as JPEG files. GeoRaster also supports JPEG 2000 compression, in which the GeoRaster has a single block stored as a JPEG 2000 file. There are some special cases where you can load and export JPEG or JPEG 2000 images *without* decompressing and recompressing, thus improving performance significantly.

For JPEG, you can use the JAI-based GeoRaster loader to load the image directly without decompression and recompression if the image file is a JPEG file, the GeoRaster object's compression type is specified as JPEG-F and no blocking is specified for the GeoRaster object's storage (that is, the GeoRaster object has only one block).

For JPEG 2000, you can use GDAL or the GDAL-based GeoRaster ETL tool to load the image directly without decompression and recompression – if the image file is a JPEG2000 file and if no parameters in use require any change to the internal structure of the JPEG 2000 file. For example, the following script loads the JPEG 2000 file directly without decompression.

```
gdal_translate -of georaster /images/image_3.jp2 \
     georaster:georaster/georaster@my_db,image_table,raster \
     -co "insert=(id,label,raster) values (3, 'image_3', sdo_geor.init('rdt_table',
3)" \
     -co compress=jp2-f
```

However, if any of the parameter in use require changing the internal structure of the JPEG 2000 data, direct loading will not be possible. The following example requires decompression and recompression, resulting in a substantial increase of the loading time.

```
gdal_translate -of georaster /images/image_4.jp2 \
     georaster:georaster/georaster@my_db,image_table,raster \
     -co "insert=(id,label,raster) values (4, 'image_4', sdo_geor.init('rdt_table',
4)" \
     -co compress=jp2-f \
     -co blockxsize=1024 \
     -co blockysize=1024 \
     -srcwin 100 200 1000 1000 \
     -outsize 50% 50%
```

3.4.3 Reformatting the Source Raster Before Loading

The GeoRaster JAI-based 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. However, it is recommended that you use the GDAL-based ETL loader, which generally does not have such issues and requirements.

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/technetwork/database/enterprise-edition/getting-started-with-gdal-133874.pdf`. 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 GeoRaster Tools: Viewer_ Loader_ Exporter) , 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.

3.5 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 GeoRaster Tools: Viewer_ Loader_ Exporter, 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.

3.6 Georeferencing GeoRaster Objects

Georeferencing, as explained in Georeferencing, 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 GeoRaster Tools: Viewer_ Loader_ Exporter) 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 GeoRaster Tools: Viewer_ Loader_ Exporter); 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 and Graph.

  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 GeoRaster Tools: Viewer_ Loader_ Exporter) 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) or want to collect GCPs yourself, you can call the SDO_GEOR.georeference function to georeference the GeoRaster object. For more information, see Advanced Georeferencing.

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

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

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

3.7 Generating and Setting Spatial Extents

When a GeoRaster object is created, its spatial extent (`spatialExtent` attribute, described in spatialExtent Attribute) 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 Indexing GeoRaster Objects), 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 any 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 GeoRaster Tools: Viewer_ Loader_ Exporter), use the SQL UPDATE statement, or set the spatial extent during loading with GDAL. 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 Creating a GeoRaster Table) 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;
```

The following example updates the spatial extent geometry of all GeoRaster objects in the CITY_IMAGES table 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.spatialExtent is null;
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;
```

- Special Considerations if the GeoRaster Table Has a Spatial Index

3.7.1 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 Indexing GeoRaster Objects), 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 4326 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, 'blocking=OPTIMALPADDING,
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 4326.
```

```
  gr.spatialExtent := sdo_cs.transform(gr.spatialExtent, 4326);
  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.

3.8 Indexing GeoRaster Objects

GeoRaster data can be indexed in various ways. The most important index you can create on a GeoRaster object is a spatial (R-tree) index on the spatial extent (footprint) geometry of the GeoRaster object (`spatialExtent` attribute, described in spatialExtent Attribute). For large-scale geospatial image and raster databases, you should always create spatial indexes on the GeoRaster columns. The following are the basic steps to create a spatial index on GeoRaster column. (The examples assume that the GeoRaster table name is CITY_IMAGES and its GeoRaster column name is IMAGE.)

1. Insert a row into the USER_SDO_GEOM_METADATA view with the georaster table name (CITY_IMAGES in this example) and the spatial extent of the GeoRaster column name (IMAGE.SPATIALEXTENT). Be sure that the correct SRID value (3371 in this example) is registered.

    ```
    INSERT INTO user_sdo_geom_metadata
        (TABLE_NAME,
         COLUMN_NAME,
         DIMINFO,
         SRID)
    VALUES (
         'city_images',
         'image.spatialextent',
         SDO_DIM_ARRAY(
             SDO_DIM_ELEMENT('X', -1000000000, 1000000000, 0.005),
             SDO_DIM_ELEMENT('Y', -1000000000, 1000000000, 0.005)),
         3371
    );
    ```

2. Create a spatial index on the GeoRaster column, as in the following example which creates a spatial index named CITY_IMAGES_IDX on the spatial extents of the images using default values for all parameters.

    ```
    CREATE INDEX city_images_idx
         ON city_images (image.spatialextent)
         INDEXTYPE IS MDSYS.SPATIAL_INDEX;
    ```

    The preceding statement may fail if there are some invalid spatial extents or if the SRID values in the GeoRaster table do not match the SRID registered in the preceding step. If the statement fails, ensure that all GeoRaster objects have a

valid `spatialExtent` geometry attribute and that all `spatialExtent` geometries have the same SRID. (Null for the `spatialExtent` values is acceptable.) Then re-create the spatial index.

See also Special Considerations if the GeoRaster Table Has a Spatial Index for special considerations if the GeoRaster table already has a spatial index. For more information about creating spatial indexes and about advanced capabilities, see *Oracle Spatial and Graph 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. This should be done using `PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber,rowBlockNumber, columnBlockNumber)`, as shown in Example 3-2 and Example 3-3.

3.9 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 GeoRaster Tools: Viewer_ Loader_ Exporter).

- Use Oracle Fusion Middleware MapViewer or its associated Map Builder utility.

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 allows you to display a virtual mosaic defined as one or a list of GeoRaster tables or views.

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.

Visualization applications can leverage the default RGBA and default pyramid level specifications in the GeoRaster objects. You can set up different bands in a multiband

image as the default Red, Green, Blue, and Alpha channels by calling SDO_GEOR.setDefaultColorLayer or SDO_GEOR.setDefaultRed, SDO_GEOR.setDefaultGreen, SDO_GEOR.setDefaultBlue, and SDO_GEOR.setDefaultAlpha. For large images, you can call SDO_GEOR.setDefaultPyramidLevel to set up the best resolution (pyramid) level of an image for initial display in the applications. For example, for a complete overview of a whole image, it is best to set the top pyramid level as the default pyramid level.

3.10 Exporting GeoRaster Objects

To load and export imagery or raster data, always consider third-party ETL tools (see the note in GeoRaster Tools: Viewer_ Loader_ Exporter)

If you use features in GeoRaster to export GeoRaster objects to image files, you have the following options:

- Use the GDAL-based ETL tool for concurrent batch exporting, which is described in GDAL-Based ETL Wizard for Concurrent Batch Loading and Exporting.
- 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 GeoRaster Tools: Viewer_ Loader_ Exporter.

3.11 Using GeoRaster with Workspace Manager and Label Security

Oracle Workspace Manager provides a versioning capability for the raster blocks of a GeoRaster object. Oracle Label Security supports GeoRaster objects with enhanced security at the row level of raster blocks.

To use GeoRaster with Oracle Workspace Manager or Oracle Label Security, you should create a raster data table (RDT) as a relational table for the GeoRaster objects (see Example 3-3). You do not need to define an object view of SDO_RASTER type on the base relational RDT.

- Using GeoRaster with Workspace Manager
- Using GeoRaster with Label Security

3.11.1 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 relational raster data tables for raster storage and version-enable these relational raster data tables. For example (general format):

```
EXECUTE DBMS_WM.EnableVersioning (<rdt_relational_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 relational 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*.

3.11.2 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 and relational RDT or RDTs.

2. Create an Oracle Label Security policy and define the label components.

3. Create labeling functions for the GeoRaster table and the relational RDT or RDTs.

   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; otherwise, the GeoRaster objects might be invalid or have an inconsistent status.

   The following example creates the labeling function for a relational RDT:

```
CREATE OR REPLACE FUNCTION gen_rdt_label(rdt_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_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_name), rid;
  RETURN LBACSYS.LBAC_LABEL.NEW_LBAC_LABEL(label);
END;
/
```

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

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

```
BEGIN

SA_POLICY_ADMIN.REMOVE_TABLE_POLICY(<policy_name>,<schema_name>,<rdt_relational_t
able>);
  SA_POLICY_ADMIN.APPLY_TABLE_POLICY(
    POLICY_NAME => <policy_name>,
    SCHEMA_NAME => <schema_name>,
    TABLE_NAME  => <rdt_relational_table>,
    TABLE_OPTIONS => 'READ_CONTROL,WRITE_CONTROL,CHECK_CONTROL',
    LABEL_FUNCTION =>
'<schema_name>.gen_rdt_label(<rdt_relational_table>,: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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)) 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 RDT are updated, so that the labels are synchronized.

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

3.12 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 BasicFiles 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 3-4 and Example 3-5.

Example 3-4   Shrinking a BasicFile RasterBlock LOB Segment

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

Example 3-5   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.

3.13 Checking GeoRaster Tables and Objects in the Database

For database management purposes, you might need check on GeoRaster tables and objects in the whole database or under a specific schema. After the GeoRaster database is created, you have the following options for checking or listing existing GeoRaster tables, RDT tables, and GeoRaster objects.

*   Use 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 lists all GeoRaster objects defined in the current schema; or if you call this function while connected as the MDSYS user, all GeoRaster objects defined in the database are listed.

    SDO_GEOR_ADMIN.listGeoRasterColumns lists all GeoRaster columns defined in the current schema; or if you call this function while connected as the MDSYS user, all GeoRaster columns defined in the database are listed.

    SDO_GEOR_ADMIN.listGeoRasterTables lists all GeoRaster tables defined in the current schema; or if you call this function while connected as the MDSYS user, all GeoRaster tables defined in the database are listed.

    SDO_GEOR_ADMIN.listRDT lists all raster data tables (RDTs) defined in the current schema; or if you call this function while connected as the MDSYS user, all raster data tables (RDTs) defined in the database are listed.

SDO_GEOR_ADMIN.listRegisteredRDT lists all registered raster data tables (RDTs) defined in the current schema; or if you call this function while connected as the MDSYS user, all registered RDTs defined in the database are listed . An RDT is *registered* if at least one entry in the SYSDATA table refers to it.

SDO_GEOR_ADMIN.listUnregisteredRDT lists all unregistered raster data tables (RDTs) defined in the current schema; or if you call this function while connected as the MDSYS user, all unregistered RDTs defined in the database are listed.. An RDT is *unregistered* if no entry in the SYSDATA table refers to it.

- Run SQL queries directly against GeoRaster metadata views, and check or list GeoRaster tables and objects stored in the different schemas. This approach is more flexible than calling subprograms. It also enables some query results that cannot be returned by functions defined in the SDO_GEOR_ADMIN package. The following are some sample queries.

  List all GeoRaster objects that are defined in the schema HERMAN and MYTEST and accessible by the current schema.

  ```
  SELECT owner,TABLE_NAME,COLUMN_NAME,RDT_TABLE_NAME,RASTER_ID
  from all_sdo_geor_sysdata where owner='HERMAN' or owner='MYTEST';
  ```

  Count the total number of GeoRaster objects accessible by the current schema.

  ```
  SELECT count(*) from all_sdo_geor_sysdata;
  ```

  Count the total number of GeoRaster objects stored in the GeoRaster table GTF_TABLE in the current schema.

  ```
  SELECT count(*) from user_sdo_geor_sysdata where TABLE_NAME='GTF_TABLE';
  ```

  List all GeoRaster objects stored in the RDT table RDT_1 in the current schema.

  ```
  SELECT TABLE_NAME,COLUMN_NAME,RDT_TABLE_NAME,RASTER_ID from
  user_sdo_geor_sysdata where RDT_TABLE_NAME='RDT_1';
  ```

  Find out all GeoRaster tables that store some raster data in or reference the RDT table RDT_1 in the current schema.

  ```
  SELECT distinct TABLE_NAME from user_sdo_geor_sysdata where
  RDT_TABLE_NAME='RDT_1';
  ```

  List all RDT tables that are used by the GeoRaster table GTF_TABLE in the current schema.

  ```
  SELECT distinct RDT_TABLE_NAME from user_sdo_geor_sysdata where
  TABLE_NAME='GTF_TABLE';
  ```

3.14 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)) 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 SDO_GEOR_ADMIN Package Reference.)

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 SDO_GEOR_ADMIN Package Reference.

3.15 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)) 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 Checking for and Resolving Conflicts.

2. Perform the data transfer, as explained in Performing the GeoRaster Data Transfer.

- Checking for and Resolving Conflicts
- Performing the GeoRaster Data Transfer

3.15.1 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 Maintaining GeoRaster Objects and System Data in the Database). 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*.

3.15.2 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. Therefore, all GeoRaster internal DML triggers should be excluded in `expdp` and `impdp` operations; otherwise, some `impdp` errors such as the following will be raised, even though the errors can be safely ignored:

```
ORA-39083: Object type TRIGGER failed to create with error:
ORA-13391:  GeoRaster reserved names cannot be used to create regular triggers
```

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

```
expdp 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_%'"
```

To import GeoRaster data, do as you would for other types of data, but exclude the GeoRaster internal DML triggers (whose names start with `GRDMLTR_`) if you did not exclude them in the export operation. 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 Checking for and Resolving Conflicts.

   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_`) if you did not exclude them in the export operation. 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_%'"
```

3.16 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 and Graph 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 Checking for and Resolving Conflicts. 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 SDO_GEOR_ADMIN Package Reference) 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 Maintaining GeoRaster Objects and System Data in the Database, 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

GeoRaster Data Query and Manipulation

This chapter describes how to perform several important GeoRaster data query and manipulation operations. Typical GeoRaster data query and manipulation involve most or all of the operations described.
See also the operations in GeoRaster Database Creation and Management.

Other chapters in this book cover advanced topics (Raster Algebra and Analytics and Image Processing and Virtual Mosaic), and provide detailed reference information about GeoRaster PL/SQL packages (SDO_GEOR Package Reference, SDO_GEOR_ADMIN Package Reference, SDO_GEOR_AGGR Package Reference, SDO_GEOR_RA Package Reference, and SDO_GEOR_UTL Package Reference).

Topics:

- Querying and Searching GeoRaster Objects
- Changing and Optimizing Raster Storage
- Copying GeoRaster Objects
- Subsetting GeoRaster Objects with Polygon Clipping
- Querying and Updating GeoRaster Metadata
- Querying and Updating GeoRaster Cell Data
- Interpolating Cell Values
- Processing and Analyzing GeoRaster Objects
- Monitoring and Reporting GeoRaster Operation Progress
- Compressing and Decompressing GeoRaster Objects
- Deleting GeoRaster Objects, and Performing Actions on GeoRaster Tables and RDTs
- Performing Cross-Schema Operations
- Managing Memory to Improve Performance
- Updating GeoRaster Objects Before Committing
- Updating GeoRaster Objects in a Loop
- Using Template-Related Subprograms to Develop GeoRaster Applications

4.1 Querying and Searching GeoRaster Objects

GeoRaster tables are regular relational tables that can have various columns, such as an ID number, a name, a timestamp, and a unique description in the form of a string. These columns can be indexed, and GeoRaster objects can be queried using the standard database indexing and query statements, as shown in many examples in this manual.

After the GeoRaster tables are spatially indexed (see Indexing GeoRaster Objects), you can quickly query or search GeoRaster objects using a geometry as well. For

example, you may want to find all images (maybe hundreds or more) inside a specific region and then generate full pyramids for each image, as in the following example.

Example 4-1   Searching GeoRaster Objects and Generating Pyramids for Them

```
DECLARE
  type curtype is ref cursor;
  my_cursor curtype;
  stmt varchar2(1000);
  tid    number;
  gr    sdo_georaster;
  gm  sdo_geometry;
BEGIN
  -- 1. Define the query area in EPSG 4326 (WGS84) coordinate system
  gm := sdo_geometry(2003, 4326, null,
             sdo_elem_info_array(1,1003,3),
             sdo_ordinate_array(5,6,30,30));

  -- 2. Define the query statement on the GeoRaster table (city_images)using the
given geometry
  stmt := 'select id from city_images t ' ||
    'where sdo_inside(t.image.spatialextent, :1)=''TRUE''';

  -- 3. Spatially query all images INSIDE the query area
  --    and generate full pyramids for each of the images
  open my_cursor for stmt using gm;
  loop
    fetch my_cursor into tid;
    exit when my_cursor%NOTFOUND;
    -- retrieve the image to generate the pyramids
    select image into gr from city_images where id = tid for update;
    sdo_geor.generatePyramid(gr, 'resampling=bilinear', null, 'parallel=4');
    update city_images set image=gr
       where id = tid;
    commit;
  end loop;
  close my_cursor;
END;
```

You can also wrap up such blocks into a PL/SQL procedure and store it in the database, then call the stored procedure directly. These features enable you to organize complex processes and automate database administration tasks.

4.2 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. Such flexibility allows you to optimize the raster data storage format to save disk space and improve application performance.

To load and process a GeoRaster object to create another GeoRaster object, you can specify storage parameters with GeoRaster PL/SQL subprograms. That is, you can specify the output format when you call functions or procedures such as SDO_GEOR.importFrom, SDO_GEOR.subset, SDO_GEOR.rectify, SDO_GEOR_AGGR.append, SDO_GEOR.mergeLayers, SDO_GEOR.createTemplate, SDO_GEOR_RA.rasterMathOp, and SDO_GEOR_AGGR.mosaicSubset. You cannot directly make such changes on an existing GeoRaster object; however, you can use the SDO_GEOR.changeFormatCopyprocedure, and specify the desired storage

parameter values with the `storageParam` parameter, to make a copy of the existing GeoRaster object.

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. Depending on the raster dimension size and your desired blocking size, padding might waste some storage space, so you should always consider specifying `blocking=OPTIMALPADDING` in the `storageParam` parameter for the output GeoRaster when a GeoRaster procedure is called.

For more information, see Storage Parameters, especially Table 1-1. For examples of applying optimal padding, see the PL/SQL example at the end of Storage Parameters and the GDAL example in Loading with Blocking and Optimal Padding

4.3 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 Blank and Empty GeoRaster Objects.) 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 Changing and Optimizing Raster Storage).

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 in SDO_GEOR Package Reference.

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 in SDO_GEOR Package Reference.

Parallel copying and subsetting are supported with the
SDO_GEOR_AGGR.mosaicSubset procedure. For parallelized copying and change
format copying, See Example 6-24 in Parallel Compression, Copying, and Subsetting.

4.4 Subsetting GeoRaster Objects with Polygon Clipping

With GeoRaster, subsetting means cropping rasters spatially, extracting or duplicating
raster layers, or doing both together. To perform subsetting, use the
SDO_GEOR.subset procedure. For example, you can call this procedure to crop a
small area or obtain a subset of a few layers of a GeoRaster object, to duplicate
layers, to specify storage parameters such as blocking and interleaving for the
resulting object, and to perform polygon clipping.

For examples, see the SDO_GEOR.subset reference topic.

You can also use the SDO_GEOR_AGGR.mosaicSubset procedure to perform
subsetting with parallelism (see Parallel Compression, Copying, and Subsetting).

4.5 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 using XMLType methods or XML-
  specific SQL functions, such as extract, and be modified using XQuery Update.
  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.

- After updating a GeoRaster object's metadata or cell data (or both) and before you commit a database transaction, you should call the SQL UPDATE statement to update the GeoRaster object (see Updating GeoRaster Objects Before Committing).

4.6 Querying and Updating GeoRaster Cell Data

To query cell (pixel) data of a GeoRaster object for processing and visualization, you can query the raster data for a cell (pixel), a range of cells, or the entire raster of a GeoRaster object:

- SDO_GEOR.getCellValue returns cell values of one or multiple layers or bands for a specified location.

- SDO_GEOR.getCellValues returns the cell values of one or more cells in an array.

- 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. (See Interpolating Cell Values for more information.)

- SDO_GEOR.evaluateDoubles evaluates multiple locations using a specified interpolation method, and returns the raster values (double precision numbers) for the specified band or layer for those locations.

- SDO_GEOR.getRasterSubset creates a single BLOB object or a single in-memory SDO_NUMBER_ARRAY 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 or SDO_NUMBER_ARRAY object contains only raster cells and no related metadata.

- SDO_GEOR.getRasterData creates a single BLOB object containing all cells of the GeoRaster object at a specified pyramid level. This BLOB object contains only raster cells and no related metadata.

- SDO_GEOR.getRasterBlocks returns an object that includes all image data inside or touching a specified window. Specifically, it returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that are inside or touch a specified window.

- 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 into a different coordinate system on-the-fly.

- SDO_GEOR.rectify performs reprojection, rectification, or orthorectification on all or part of a georeferenced GeoRaster object based on a query window. The resulting object can be a new GeoRaster object (for persistent storage) or a BLOB (for temporary use).

- SDO_GEOR_RA.findCells generates a new GeoRaster object based on the cell values using the GeoRaster Raster Algebra language. (See Cell Value-Based Conditional Queries for more information.)

- SDO_GEOR_AGGR.mosaicSubset mosaics a number of GeoRaster objects into one GeoRaster object.

- SDO_GEOR_AGGR.getMosaicSubset lets you query a virtual mosaic and returns a mosaicked subset on-the-fly.

- SDO_GEOR.getBitmapMask, SDO_GEOR.getBitmapMaskSubset, SDO_GEOR.getBitmapMaskValue, and SDO_GEOR.getBitmapMaskValues let you query bitmap masks on the basis of a full raster, a window, or single cells.

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.

There are two types of raster updates: space-based and cell value-based

- Space-based raster update allows you update a GeoRaster object's raster data inside a specified window entirely using either a single value or another GeoRaster object.

  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. To change the value of raster cells specified by row/column arrays or by a multipoint geometry to new values, you can use the SDO_GEOR.changeCellValues 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).

  The SDO_GEOR_AGGR.append procedure can also be used to update an existing image with a new image (see Image Appending).

- Cell value-based raster update allows you update a GeoRaster object's raster data based on the cell values using the GeoRaster Raster Algebra language.

  SDO_GEOR_RA.rasterUpdate selects cells from the specified GeoRaster object based on Boolean strings specified in the `conditions` parameter, and updates corresponding cell values by calculating expression strings specified in the `vals` parameter. Both the `conditions` and `vals` parameters can be complicated expressions using the raster algebra language. (See Cell Value-Based Conditional Updates (Edits) for more information.)

If statistics are already set in the GeoRaster object when you perform space-based or raster cell value-based updates, the statistics are not removed or updated automatically after you run the raster update procedures. If necessary, you should remove or regenerate the statistics.

> **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 Querying and Updating GeoRaster Metadata.

After updating a GeoRaster object's metadata or cell data (or both) and before you commit a database transaction, you should call the SQL UPDATE statement to update the GeoRaster object (see Updating GeoRaster Objects Before Committing).

4.7 Interpolating Cell Values

GeoRaster objects are grid coverages. The "evaluate" operation of a grid coverage is also called **grid interpolation**, a method for interpolating cell values at point positions between the cells or within the cells. This operation in GeoRaster is performed by the SDO_GEOR.evaluateDouble function, which evaluates any point in the raster and returns a double number value for that location. You can use any one of the six different interpolation methods (listed in Resampling and Interpolation) to do the evaluation. For example, if a georaster object is a DEM layer, you can find out the elevation of a random point location, using the following example:

```
SELECT SDO_GEOR.evaluateDouble(a.georaster, 0,
    SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(112.704, 41.917, NULL),
                NULL, NULL),
    '1',
    'interpolationMethod=BILINEAR')
  FROM georaster_table a WHERE raster_name='myDEM';
```

If you call SDO_GEOR.evaluateDouble with `'interpolationMethod=NN'`, the GeoRaster object is treated as a discrete raster and the preceding is the same as calling SDO_GEOR.getCellValue, which gives you the same value (that is, the cell value) at a different point location inside a cell. In this case, you can directly call SDO_GEOR.getCellValue instead, particularly when you query only the cell values of a single band. Other interpolation methods treat the raster as a continuous surface and may give you different values at different point locations inside a cell.

4.8 Processing and Analyzing GeoRaster Objects

You can perform a variety of raster and image processing operations on GeoRaster data, including changing the internal raster storage format, subsetting (cropping), scaling, rotating, masking, stretching, filtering, dodging, reprojecting (from one coordinate system to another), rectifying, orthorectifying, warping, mosaicking, appending, and generating pyramids. GeoRaster also supports virtual mosaic. Some relevant subprograms are SDO_GEOR.changeFormatCopy, SDO_GEOR.subset, SDO_GEOR.reproject, SDO_GEOR.rectify, SDO_GEOR.generatePyramid, SDO_GEOR.deletePyramid, SDO_GEOR.scaleCopy, SDO_GEOR.mergeLayers, SDO_GEOR_AGGR.mosaicSubset, SDO_GEOR_AGGR.getMosaicSubset, and SDO_GEOR_AGGR.append. For detailed descriptions, see Image Processing and Virtual Mosaic, SDO_GEOR Package Reference, and SDO_GEOR_AGGR Package Reference.

For raster cell value-based algebraic operations and cartographic modeling and analysis, GeoRaster supports a raster algebra language (PL/SQL and Algebraic Expressions) and related raster operations, including conditional queries (SDO_GEOR_RA.findCells), cell value-based updates or edits (SDO_GEOR_RA.rasterUpdate), logical and mathematical operations (SDO_GEOR_RA.rasterMathOp), and image and raster segmentation

(SDO_GEOR_RA.classify). The SDO_GEOR.generateStatistics function supports polygon-based statistics and histogram generation. The following on-the-fly functions support interactive statistical analysis of a GeoRaster object or its layers: SDO_GEOR.generateStatisticsMax, SDO_GEOR.generateStatisticsMean, SDO_GEOR.generateStatisticsMedian, SDO_GEOR.generateStatisticsMin, SDO_GEOR.generateStatisticsMode, and SDO_GEOR.generateStatisticsSTD. For detailed descriptions, see Raster Algebra and Analytics and SDO_GEOR_RA Package Reference.

See also the GeoRaster PL/SQL demo files, described in GeoRaster PL/SQL and Java Demo Files, for examples and explanatory comments.

4.9 Monitoring and Reporting GeoRaster Operation Progress

GeoRaster lets you monitor and report the execution progress of many operations (listed in Reporting Operation Progress in GeoRaster). The following are the basic steps for reporting the progress of an operation:

1. Use the SDO_GEOR_UTL.createReportTable procedure to create the report table under the appropriate user's schema. (This must be called once before you can monitor any operations.)

```
EXECUTE SDO_GEOR_UTL.createReportTable;
```

2. In the user session where the operations are to be executed and monitored, perform the following actions:

   a. Use SDO_GEOR_UTL.enableReport to enable the monitoring. (You must call this procedure in order to be able to get the status report later.)

   ```
   EXECUTE SDO_GEOR_UTL.enableReport;
   ```

   b. Optionally, use SDO_GEOR_UTL.setClientID to set the client ID. The client ID is used to identify the user session that executes the operation. If this procedure is not called, the client ID defaults to the SQL session ID. For example:

   ```
   EXECUTE SDO_GEOR_UTL.setClientID(100);
   ```

   c. Optionally, use SDO_GEOR_UTL.setSeqID to set the sequence ID. The sequence ID is used to identify the repeated operations in the same SQL session. If this procedure is not called, the sequence ID defaults to 0. For example:

   ```
   EXECUTE SDO_GEOR_UTL.setSeqID(1);
   ```

   d. Execute the operation to be monitored. For example:

   ```
   -- Generate pyramid for georid=6. The progress of this generatePyramid call
   -- can be monitored by step 3.
   DECLARE
     gr sdo_georaster;
   BEGIN
      SELECT georaster INTO gr
       FROM georaster_table WHERE georid = 6 FOR UPDATE;
     sdo_geor.generatePyramid(gr, 'rLevel=5, resampling=NN');
     UPDATE georaster_table SET georaster = gr WHERE georid = 6;
      COMMIT;
   ```

```
END;
/
```

e. Optionally, repeat steps c and d for each additional operation to be monitored.
    For example:

```
EXECUTE SDO_GEOR_UTL.setSeqID(2);
-- Generate pyramid for georid=7. The progress of this generatePyramid call
-- can be monitored by step 3.
DECLARE
  gr sdo_georaster;
BEGIN
   SELECT georaster INTO gr
    FROM georaster_table WHERE georid = 7 FOR UPDATE;
  sdo_geor.generatePyramid(gr, 'rLevel=5, resampling=NN');
  UPDATE georaster_table SET georaster = gr WHERE georid = 7;
   COMMIT;
END;
/
```

f. Optionally, use SDO_GEOR_UTL.disableReport to disable the monitoring. If
    this procedure is not called, the monitoring is automatically stopped when the
    user session ends.

```
EXECUTE SDO_GEOR_UTL.disableReport;
```

3. From another session under the same user, retrieve the execution status report.

    To get the progress of a specific operation identified by client ID and sequence ID,
    use the SDO_GEOR_UTL.getProgress function. This function returns the progress
    as a number between 0 and 1 reflecting the percentage of completion. For
    example, the following query shows that the operation if 55% complete:

```
SELECT sdo_geor_utl.getProgress(100, 2) progress FROM DUAL;

PROGRESS
--------
    0.55

1 row selected.
```

    To get the status report of a specific operation identified by client ID and sequence
    ID, use the SDO_GEOR_UTL.getStatusReport function. This function returns an
    array of strings describing the progress and other information about the operation.
    For example:

```
-- Check the status of the generatePyramid on georid=6
SELECT sdo_geor_utl.getStatusReport(100, 1) FROM DUAL;
SDO_GEOR_UTL.GETSTATUSREPORT(100,1)
--------------------------------------------------------------------------
SDO_STRING2_ARRAY('31-OCT-11 02.20.04.854558 PM', 'GeneratePyramid',
'RDT:RDT_1', 'RID:6', '100% complete', 'operation completed')

1 row selected.

-- Check the status of the generatePyramid on georid=7
SELECT sdo_geor_utl.getStatusReport(100, 2) FROM DUAL;

SDO_GEOR_UTL.GETSTATUSREPORT(100,2)
--------------------------------------------------------------------------
SDO_STRING2_ARRAY('31-OCT-11 02.20.08.854558 PM', 'GeneratePyramid',
'RDT:RDT_1', 'RID:7', '55% complete', 'operation completed')
```

ORACLE®

```
1 row selected.
```

To get the status of all the monitored operations, enter the following statement:

```
SELECT * from the (select sdo_geor_utl.getAllStatusReport() FROM DUAL);
COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('Client:100', 'Sequence:1', '31-OCT-11 02.20.04.854558 PM',
'GeneratePyramid', 'RDT:RDT_1', 'RID:6', '100% complete', 'operation completed')
SDO_STRING2_ARRAY('Client:100', 'Sequence:2', '31-OCT-11 02.20.08.854558 PM',
'GeneratePyramid', 'RDT:RDT_1', 'RID:7', '55% complete', NULL)

2 rows selected.
```

If you need to clear or drop the report table, use the
SDO_GEOR_UTL.clearReportTable or SDO_GEOR_UTL.dropReportTable
procedure, respectively:

```
EXECUTE SDO_GEOR_UTL.clearReportTable;
-- or:
EXECUTE SDO_GEOR_UTL.dropReportTable;
```

4.10 Compressing and Decompressing GeoRaster Objects

You can reduce the storage space requirements for GeoRaster objects by
compressing them using JPEG-F, DEFLATE, or JPEG 2000 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.

For JPEG-F and DEFLATE, 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,
SDO_GEOR.subset, raster algebra (SDO_GEOR_RA) procedures, and
SDO_GEOR_AGGR.mosaicSubset. (For JPEG-F and DEFLATE compression, there
are no separate procedures for compressing and decompressing a GeoRaster object.)

For JPEG 2000, use the SDO_GEOR.compressJP2 and SDO_GEOR.decompressJP2
procedures to compress and decompress a GeoRastrer object, respectively. Most
other procedures and functions (except for SDO_GEOR.changeCellValue,
SDO_GEOR.reproject, sdo_geor.scaleScopy, and sdo_geor.mosaic) can internally
decompress the JP2 compressed GeoRaster object while performing the operation.

For more information about GeoRaster compression and decompression, see
Compression and Decompression, including information about support for third-party
compression solutions in Third-Party Plug-ins for Compression.

In addition, when JPEG-F or DEFLATE compression is used with GeoRaster objects,
some special usage considerations apply:

• If a large GeoRaster object is to be compressed and will have full pyramids built
  on it, it is faster to generate pyramids on the uncompressed GeoRaster object first,
  then apply compression.

- For large scale mosaicking, it is faster to mosaic without applying compression first, then generate pyramids, then apply compression.

- In some operations, GeoRaster uses temporary tablespaces to compress and decompress data, so adding temporary tablespaces for GeoRaster users is essential for performance (see Adding Temporary Tablespaces for GeoRaster Users).

Parallel compression and decompression for JPEG and DEFLATE are supported with the SDO_GEOR.changeFormatCopy procedure, if reformatting is not needed, by using the `parallel` keyword in the `storageParam` parameter. You can also call the SDO_GEOR_AGGR.mosaicSubset procedure, which provides better performance if you want to change the raster format while doing parallelized compression or decompression. Parallel compression and decompression significantly improve performance, which is especially useful for large images. See Example 6-23 in Parallel Compression, Copying, and Subsetting

If you want to store compressed GeoRaster objects, make sure you create a temporary tablespace for the users. For more information, see Adding Temporary Tablespaces for GeoRaster Users.

4.11 Deleting GeoRaster Objects, and Performing Actions on GeoRaster Tables and RDTs

GeoRaster automatically maintains the GeoRaster metadata and the relationship between GeoRaster tables and raster data tables (RDTs). Therefore, for most operations you can use the relevant traditional SQL statement.

- **To delete a GeoRaster object**, delete the row containing the object using the DELETE statement (for example, `DELETE FROM geor_table WHERE ...;`).

  After a GeoRaster object is deleted from a GeoRaster table, all related raster data stored in the RDT is deleted automatically. Never insert or delete any rows directly in a raster data table.

- **To drop a GeoRaster table**, use the DROP statement (for example, `DROP geor_table;`).

  After a GeoRaster table is dropped, all raster data associated with GeoRaster objects in the deleted GeoRaster table is deleted automatically.

- **To rename a GeoRaster table**, use the RENAME statement (for example, `RENAME geor_table1 TO geor_table2;`).

- **To add a GeoRaster column to a table**, use the ALTER TABLE statement.

  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. For example:

```
ALTER TABLE geor_table ADD (image SDO_GEORASTER);
CALL sdo_geor_utl.createdmltrigger('GEOR_TABLE','IMAGE');)
```

- **To drop a GeoRaster column in a table**, use the ALTER TABLE statement (for example, `ALTER TABLE geor_table DROP COLUMN image;`).

  Caution: Dropping a GeoRaster column will delete all GeoRaster objects in that column.

- **To drop an RDT**, you must first delete all GeoRaster objects that reference the RDT, after which you can use the DROP statement on the RDT.

  If you do not delete all GeoRaster objects that reference the RDT before attempting to drop the RDT, an exception is raised.

- **To rename an RDT**, use the SDO_GEOR_UTL.renameRDT procedure.

4.12 Performing Cross-Schema Operations

All GeoRaster operations can work on GeoRaster objects defined in schemas other than the current connection schema. In other words, GeoRaster fully supports cross-schema access, update, and processing.

For more information, see Cross-Schema Support with GeoRaster.

Example 4-2   Cross-Schema Copy Operation

In the following example, USER2 makes a copy of an image from USER1 and stores it in the USER2 schema. Assume that USER1 owns the GEORASTER_TABLE table and that USER2 owns the G_TABLE table.

```
--connect to user1 and grant permissons to user2
--assume user1 stores the image in georaster_table and the image's RDT table is
rdt_1
conn user1/pswd1
 grant select on georaster_table to user2;
grant select on rdt_1 to user2;

--connect to user2 and make a copy of a georaster from user1
conn user2/pswd2
SQL> DECLARE
   gr1 sdo_georaster;
   gr2 sdo_georaster;
BEGIN
   --select the image from georaster_table in user1
   select georaster into gr1 from user1.georaster_table where georid = 100;
   -- the copy is to be stored in g_table in user2, assuming the georaster object is
already initiated
   select geor into gr2 from g_table where id = 1 for update;
   sdo_geor.changeFormatCopy(gr1, 'blocking=OPTIMALPADDING blocksize=(512,512,3)
interleaving=BIP', gr2);
   update g_table set geor=gr2 where id=1;
   commit;
END;
/
```

Example 4-3   Cross-Schema Raster Algebra and Copy Operation

In the following example, USER2 runs a raster algebra operation on an image in the USER1 schema and stores the result in USER1. Assume that USER1 owns both the GEORASTER_TABLE and G_TABLE tables. The existing image is in GEORASTER_TABLE and the image's raster data table is RDT_1. The resulting image is stored in G_TABLE and its raster data table is RDT_2.

```
--connect to user1 and grant permissons to user2
conn user1/pswd1
grant select on georaster_table to user2;
grant select on rdt_1 to user2;
grant select, update, insert, delete on g_table to user2;
```

```
grant select, update, insert, delete on rdt_2 to user2;

--connect to user2 and run a raster algebra operation on an image in user1
conn user2/pswd2
DECLARE
   gr1 sdo_georaster;
   gr2 sdo_georaster;
BEGIN
   --select the image from georaster_table in user1
   select georaster into gr1 from user1.georaster_table where georid = 100;
   -- the result is to be stored in g_table in user1, assuming the georaster object
is already initiated
   select geor into gr2 from user1.g_table where id = 1 for update;
   sdo_geor_ra.rasterMathOp(gr1,sdo_string2_array('{0}','{1}','{2}'),
'blocking=OPTIMALPADDING blocksize=(512,512,3) interleaving=BIP',gr2);
   update user1.g_table set geor=gr2 where id=1;
   commit;
END;
/
```

4.13 Managing Memory to Improve Performance

GeoRaster has its own buffer system to read and write raster (LOB) data. This system is separate from the Oracle Database buffer system. The following table lists parameters that can be used to configure the GeoRaster buffer system, which is used for all I/O operations on GeoRaster objects.

Table 4-1   GeoRaster Buffering Parameters

Parameter Name	Description	Default Value
MemMaxSize	Upper limit size of the memory that can be used for GeoRaster buffering for each GeoRaster object.	17 MB
MemReadBlockSize	Internal data block size for read-only operations for caching raster data.	32 KB
MemWriteBlockSize	Internal data block size for read/write operations for caching raster data.	64 KB

You can get and set the values of these parameters using the following PL/SQL subprograms:

- SDO_GEOR_UTL.getMemMaxSize

- SDO_GEOR_UTL.setMemMaxSize

- SDO_GEOR_UTL.getMemReadBlockSize

- SDO_GEOR_UTL.setMemReadBlockSize

- SDO_GEOR_UTL.getMemWriteBlockSize

- SDO_GEOR_UTL.setMemWriteBlockSize

Because the parameters are set using PL/SQL, their values are defined for the duration of the database session. For any subsequent sessions, if you want to use any nondefault values for any of the parameters, you must set them using the appropriate procedures.

In general, using large values for the parameters improves performance for GeoRaster I/O operations. The following are some specific considerations and guidelines.

- Allocating more memory (increasing `MemMaxSize`) reduces disk access; and ideally, allocating big enough memory to hold an entire GeoRaster object will dramatically improve performance. However, Oracle Database allows multiple users and concurrent access, and so you should aim for balanced memory allocation in such an environment.

- Increasing the read block size (increasing `MemReadBlockSize`) reduces the number of OCI LOB read operations, thus improving performance. However, due to different interleaving between source and target GeoRaster objects in an operation, if the `MemReadBlockSize` value cannot hold the entire GeoRaster object, the read block size might be too large and cause frequent read block page-in and page-out operations, thus degrading performance.

- Almost all GeoRaster operations are write-driven, so that a larger write block size (increasing `MemWriteBlockSize`) will reduce number of OCI LOB write operations and thus improve performance.

4.14 Updating GeoRaster Objects Before Committing

Before you commit a database transaction that inserts, updates, reformats, compresses, decompresses, or deletes GeoRaster cell data or metadata, you should use the SQL UPDATE statement to 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 4-4, the UPDATE statement is required after the call to the SDO_GEOR.changeFormatCopy procedure and before the COMMIT statement.

Example 4-4   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;
/
```

4.15 Updating GeoRaster Objects in a Loop

When multiple GeoRaster objects are to be updated, a cursor is generally used. However, if the GeoRaster operation in the loop is parallel enabled or the COMMIT statement is executed after each UPDATE statement, the SQL query for the cursor cannot have the FOR UPDATE clause because the commit within the loop will invalidate the cursor with that clause.

Example 4-5   Updating GeoRaster Objects in a Loop With Parallel Processing Enabled

In the following example, pyramids were generated on all the GeoRaster objects in the table. An explicit commit is executed after each update, and parallel processing is enabled (`parallel=4`) for the SDO_GEOR.generatePyramid procedure. The query of the GeoRaster object with FOR UPDATE clause is executed within the loop, not in the cursor query statement.

```
DECLARE
    gr1 sdo_georaster;
BEGIN
    FOR rec in (SELECT georid FROM georaster_table ORDER BY georid) LOOP
       SELECT georaster INTO gr1 FROM georaster_table WHERE georid=rec.georid FOR
UPDATE;
       sdo_geor.generatePyramid(gr1, 'rlevel=5, resampling=bilinear', null,
'parallel=4');
       UPDATE georaster_table SET georaster=gr1 WHERE georid=rec.georid;
       COMMIT;
    END LOOP;
 END;
/
```

Example 4-6   Updating GeoRaster Objects in a Loop Without Parallel Processing Enabled

If the GeoRaster operation in the loop is not parallel enabled (as is the case in this example), instead of executing a COMMIT after each update, the COMMIT statement can be executed *outside* the cursor loop, to avoid invalidating the cursor with FOR UPDATE clause. Also note that the example adds NODATA to all the GeoRaster objects in the table.

```
BEGIN
FOR rec in (SELECT georid, georaster FROM georaster_table FOR UPDATE)
    LOOP
       sdo_geor.addNODATA(rec.georaster, 0, 0);
       UPDATE georaster_table SET georaster=rec.georaster
         WHERE georid=rec.georid;
END LOOP;
COMMIT;
 END;
/
```

4.16 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 Creating New GeoRaster Objects), 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 Querying and Updating GeoRaster Metadata).

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 GeoRaster Physical Storage), and you must compress and decompress the data as necessary before reading from or writing to the BLOB.

5
Raster Algebra and Analytics

This chapter describes the raster algebra language (PL/SQL and algebraic expressions) and related raster operations, including conditional queries, cell value-based updates or edits, mathematical operations, classify, on-the-fly statistical analysis, logical operations, and their applications in cartographic modeling.

It contains the following major sections.

Topics:

- Raster Algebra Language
  Raster algebra is commonly used in raster data analysis and GIS modeling. In GeoRaster, raster algebra is supported by the GeoRaster raster algebra language.

- Cell Value-Based Conditional Queries
  Using cell-based conditional queries, you can generate a new GeoRaster object based on a specified condition.

- Cell Value-Based Conditional Updates (Edits)
  You can update raster cell values based on conditions.

- Mathematical Operations
  A major use of raster algebra is to apply mathematical models to raster layers from different sources.

- Classification Operations
  Classification (segmentation) operations can be applied on source GeoRaster objects to generate new objects.

- Statistical Operations
  To apply statistical operations on one or more layers, which are from one or more GeoRaster objects, the following types of operations are available.

- Logical Operations
  A major use of raster algebra is to apply logical models to raster layers from different sources; that is, you can apply logical operations on one or more layers, from one or more GeoRaster objects, to generate a new GeoRaster object.

- Raster Data Scaling and Offsetting
  You can perform raster data scaling and offsetting operations.

- Raster Data Casting
  Raster data casting maps cell values from one data type to another.

- Cartographic Modeling
  Raster algebra is widely used in cartographic modeling and is considered an essential component of GIS systems. Using the PL/SQL and the raster algebra expressions and functions, you can conduct cartographic modeling over a large number of rasters and images of virtually unlimited size.

5.1 Raster Algebra Language

Raster algebra is commonly used in raster data analysis and GIS modeling. In GeoRaster, raster algebra is supported by the GeoRaster raster algebra language.

The GeoRaster raster algebra language is an extension to the Oracle PL/SQL language. PL/SQL provides declarations of variables and constants, general mathematical expressions, basic functions, statements, and programming capabilities. GeoRaster provides a raster algebra expression language and a set of raster algebra functions for raster layer operations. The raster algebra expression language includes general arithmetic, casting, logical, and relational operators and allows any combination of them. The raster algebra functions enable the usage of the expressions and support cell value-based conditional queries, mathematical modeling, classify operations, and cell value-based updates or edits over one or many raster layers from one or many GeoRaster objects.

This combination of the PL/SQL language and GeoRaster algebraic expressions and functions provides an easy-to-use, powerful way to define raster analyses as algebraic expressions, so that users can easily apply algebraic functions on raster data to derive new results. For example, a simple raster operation can use two or more raster layers with the same dimension sizes to produce a new raster layer by using algebraic operations (addition, subtraction, and so on), or a sophisticated raster operation to generate a Normalized Difference Vegetation Index (NDVI) from multiple bands of satellite imagery.

GeoRaster supports raster algebra local operations, so the raster algebra operations work on individual raster cells, or pixels.

The following is the GeoRaster raster algebra expression language definition:

```
<arithmeticExpr> ::=
            <unaryArithmeticExpr>
          | <binaryArithmeticExpr>
          | <functionalArithmeticExpr>
          | <conditionalExpr>
          | <castingExpr>
          | <booleanExpr>
          | <constantNumber>
          | <identifier>
          | ( <arithmeticExpr> )

<unaryArithmeticExpr> ::=
            ( <unaryArithmeticOp> <arithmeticExpr> )

<unaryArithmeticOp> ::=
            +
          | -

<binaryArithmeticExpr> ::=
            <arithmeticExpr> <binaryArithmeticOp> <arithmeticExpr>

<binaryArithmeticOp> ::=
            +
          | -
          | *
          | /
          | %
```

```
<functionalArithmeticExpr> ::=
                <statisticalFunction> ( )
              | <numericFunction_with_1_param> ( <arithmeticExpr> )
              | <numericFunction_with_2_param> ( <arithmeticExpr> ,
<arithmeticExpr> )
              | <numericFunction_with_3_param> ( <arithmeticExpr> ,
<arithmeticExpr> , <arithmeticExpr> )

<statisticalFunction> ::=
                min
              | max
              | mean
              | median
              | std
              | minority
              | majority
              | sum
              | variety

<numericFunction_with_1_param> ::=
                abs
              | sqrt
              | exp
              | exp2
              | exp10
              | log
              | ln
              | log2
              | sin
              | cos
              | tan
              | sinh
              | cosh
              | tanh
              | arcsin
              | arccos
              | arctan
              | arcsinh
              | arccosh
              | arctanh
              | ceil
              | floor
              | factorial

<numericFunction_with_2_param> ::=
                power
              | max2
              | min2

<numericFunction_with_3_param> ::=
                max3
              | min3

<conditionalExpr> ::=
            <conditionalFunction> ( <booleanExpr> , <arithmeticExpr> ,
<arithmeticExpr> )

<conditionalFunction> ::=
                condition

<castingExpr> ::=
```

```
            <castingFunction> ( <arithmeticExpr> )

<castingFunction> ::=
            castint
          | castonebit
          | casttwobit
          | castfourbit
          | casteightbit
          | castBoolean

<booleanExpr> ::=
            <unaryBooleanExpr>
          | <binaryBooleanExpr>
          | ( <booleanExpr> )

<unaryBooleanExpr> ::=
            <unaryBooleanOp> <booleanExpr>

<unaryBooleanOp> ::=
            !

<binaryBooleanExpr> ::=
            <booleanExpr> <binaryBooleanOp> <booleanExpr>
          | <arithmeticExpr> <comparisonOp> <arithmeticExpr>

<binaryBooleanOp> ::=
            &
          | |
          | ^

<comparisonOp> ::=
            =
          | <
          | >
          | >=
          | <=
          | !=

<constantNumber> ::=
            <double number>

<identifier> ::=
            { <ID> , <band> }
          | { <band> }

<ID> ::=
            <integer number>

<band> ::=
            <integer number>
```

The precedence of the algebraic operators (+, -, *, /, and so on) in the expression language complies with general conventions. However, in any case where the expression might be misinterpreted, you should use parentheses to clarify which interpretation is intended.

The booleanExpr can be used as arithmeticExpr, as defined in the GeoRaster raster algebra expression language. In this case, the TRUE and FALSE evaluation results of booleanExpr are cast to numeric values 1 and 0, respectively.

The `identifier` in the expression refers to a raster layer of a GeoRaster object. It is either a single `band` number if there is only one GeoRaster object involved, or a pair of (`ID`, `band`) where `ID` refers to one of GeoRaster objects in the expression and band refers to a specific layer of that GeoRaster object. The `band` number in this language refers to the ordinate number of a layer along the band dimension in the cell space, so it always starts with 0 (zero). The GeoRaster ID number always starts with 0 (zero).

The following procedures provide the main support for raster algebra operations:

- SDO_GEOR_RA.rasterMathOp takes `arithmeticExpr` to perform mathematical operations or modeling, `conditionalExpr` and `booleanExpr` to perform logical operations, and `statisticalFunction` expression to perform statistical analysis.

- SDO_GEOR_RA.findCells searches/masks cells based on a `booleanExpr` condition.

- SDO_GEOR_RA.classify applies `arithmeticExpr` to cells and then segments the raster.

- SDO_GEOR_RA.rasterUpdate updates cell values of a raster based on different `booleanExpr` conditions.

- SDO_GEOR_RA.diff and SDO_GEOR_RA.over perform logical operations without using expressions.

- SDO_GEOR_RA.stack and many other subprograms perform statistical analysis or generate statistics.

These raster algebra functions take many layers from one or many GeoRaster objects, apply `booleanExpr` and/or `arithmeticExpr` expressions over those layers, do the specific algebraic computation or modeling, and output a new GeoRaster object. The expressions can be defined in any way based on the syntax described earlier in this section.

All raster algebra functions require that the raster layers overlap each other and have the same dimension sizes and resolution if they are georeferenced, or have the same dimension sizes if they are not georeferenced. Before you apply raster algebra operations over two or more GeoRaster objects, you can use the SDO_GEOR_RA.isOverlap function to determine if the GeoRaster objects are of the same size and cover the same ground area.

Raster and image databases are generally very large. Querying and manipulating such databases are computationally intensive operations. To improve performance, all GeoRaster raster algebra functions are parallelized. You should always consider applying parallel processing when using multi-CPU or multicore servers.

- Examples of Raster Algebra Expressions

5.1.1 Examples of Raster Algebra Expressions

This section contains examples showing how to define raster algebra expressions.

Example 5-1   Finding Pixels Based on a Comparison (>)

Example 5-1 finds all pixels that meet the condition defined by algebra expression `'{1}>200'` , because there is only one GeoRaster object involved in the procedure, so {1} refers to the cell value of second layer ({0} would be for the first layer), and `'{1}>200'` means any pixels whose second layer value is greater than 200. The example assumes that the source GeoRaster object has at least two layers.

```
DECLARE
  geor        MDSYS.SDO_GEORASTER;
  geor1       MDSYS.SDO_GEORASTER;
BEGIN
  select georaster into geor from georaster_table where georid = 1;
  select georaster into geor1 from georaster_table where georid = 5 for update;
  sdo_geor_ra.findcells(geor, '{1}>200','blocking=optimalpadding, blocking=true,
blocksize=(512,512,3)',geor1);
  update georaster_table set georaster = geor1 where georid = 5;
  commit;
END;
/
```

Example 5-2   Generating a GeoRaster Object Based on an Expressions Array

Example 5-2 generates a new GeoRaster object `geor2` from two input GeoRaster objects `geor` and `geor1` based on the algebra expressions array `SDO_STRING2_ARRAY('{0,0}-0.5*{1,0}','{0,1}-0.5*{1,1}','{0,2}-0.5*{1,2}')`. The example assumes that both of the source GeoRaster objects are images with three bands.

```
DECLARE
  geor        MDSYS.SDO_GEORASTER;
  geor1       MDSYS.SDO_GEORASTER;
  geor2       MDSYS.SDO_GEORASTER;
  geo_array   MDSYS.SDO_GEORASTER_ARRAY;
BEGIN
  select georaster into geor from georaster_table where georid = 1;
  select georaster into geor1 from georaster_table where georid = 2;
  insert into georaster_table values (17, sdo_geor.init('rdt_1', 17)) returning
georaster into geor2;
  geo_array:=MDSYS.SDO_GEORASTER_ARRAY(geor,geor1);

sdo_geor_ra.rasterMathOp(geo_array,SDO_STRING2_ARRAY('{0,0}-0.5*{1,0}','{0,1}-0.5*{1,
1}','{0,2}-0.5*{1,2}'),null,geor2);
  update georaster_table set georaster = geor2 where georid = 17;
  commit;
END;
/
```

In the algebra expressions array in Example 5-2:

- `{0,0}` refers to the cell value of band 0 of the first input GeoRaster object `geor`.

- `{0,1}` refers to the cell value of band 1 of the first input GeoRaster object `geor`.

- `{0,2}` refers to the cell value of band 2 of the first input GeoRaster object `geor`.

- `{1,0}` refers to the cell value of band 0 of the second input GeoRaster object `geor1`.

- `{1,1}` refers to the cell value of band 1 of the second input GeoRaster object `geor1`.

- `{1,2}` refers to the cell value of band 2 of the second input GeoRaster object `geor1`.

In Example 5-2, then, the target GeoRaster object `geor2` will have three bands, and:

- The cell value of band 0 of target GeoRaster object `geor2` is: `{0,0}-0.5*{1,0}`

- The cell value of band 1 of target GeoRaster object `geor2` is: `{0,1}-0.5*{1,1}`

- The cell value of band 2 of target GeoRaster object `geor2` is: `{0,2}-0.5*{1,2}`

Example 5-3   Updating a GeoRaster Object Based on an Expressions Array

Example 5-3 updates cell values of the input GeoRaster object based on the algebra expression array `SDO_STRING2_ARRAY('(abs({0}-{1})=48)&({2}-{1}=-101)','2*{0}-{1}/3=108')`. The example assumes that the source GeoRaster object has three layers.

```
DECLARE
  geor       MDSYS.SDO_GEORASTER;
  geor1      MDSYS.SDO_GEORASTER;
begin
  select georaster into geor from georaster_table where georid = 1;
  sdo_geor_ra.rasterUpdate(geor,0,SDO_STRING2_ARRAY('(abs({0}-{1})=48)&({2}-
{1}=-101)','2*{0}-{1}/
3=108'),SDO_STRING2_ARRAYSET(SDO_STRING2_ARRAY('123','54','89'),SDO_STRING2_ARRAY('98
','56','123')));
END;
/
```

In Example 5-3, for each pixel:

* If `(abs({0}-{1})=48)&({2}-{1}=-101)` is true, then the cell values of the three layers will be updated to `('123','54','89')`.

* If `2*{0}-{1}/3=108` is true, then the cell values of the three layers will be updated to `('98','56','123')`.

5.2 Cell Value-Based Conditional Queries

Using cell-based conditional queries, you can generate a new GeoRaster object based on a specified condition.

In addition to their use in space-based queries on rasters (such as with SDO_GEOR.getRasterSubset), cell value-based queries are particularly useful in analytics and application modeling. To perform cell value based conditional queries and generate a new GeoRaster object based on the specified condition, you can use the SDO_GEOR_RA.findCells procedure and specify an appropriate `condition` parameter.

The `condition` parameter must specify a valid `booleanExpr` value (explained in Raster Algebra Language). The procedure computes the `booleanExpr` against each cell in the source GeoRaster object. If the result is `TRUE`, the original cell values are kept in the output GeoRaster object; if the result is `FALSE`, the `bgValues` are used to fill cell values in the output GeoRaster object. This can also be considered as a masking operation.

Example 5-4   Conditional Query

Example 5-4 calls the SDO_GEOR_RA.findCells procedure to find all pixels where the value of the second band is greater than 200. Because the bgValues parameter is not specified, the value 0 is used as the background value to fill all pixels that make the condition false. The example assumes that the source GeoRaster object is an image with more than two bands.

```
DECLARE
  geor    SDO_GEORASTER;
  geor1   SDO_GEORASTER;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  INSERT into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
georaster into geor1;
```

```
    sdo_geor_ra.findcells(geor, '{ 1 }>200', 'blocking=optimalpadding,
blocksize=(512,512,3)', geor1);
  UPDATE georaster_table set georaster = geor1 WHERE georid = 5;
  COMMIT;
END;
/

-- This pixel is set to (0,0,0) because the cell value of the
-- second band is 136, which is not greater than 200.
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1 OR georid=5 ORDER BY georid;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)
SDO_NUMBER_ARRAY(0, 0, 0)

2 rows selected.

-- This pixel keeps the original values because the cell value
-- of the second band is greater than 200.
SELECT sdo_geor.getcellvalue(georaster,0,132,116,'') FROM georaster_table WHERE
georid =1 OR georid=5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,132,116,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(242, 225, 233)
SDO_NUMBER_ARRAY(242, 225, 233)

2 rows selected.

-- This pixel keeps the original values because the cell value
-- of the second band is greater than 200.
SELECT sdo_geor.getcellvalue(georaster,0,261,185,'') FROM georaster_table WHERE
georid =1 OR georid=5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,261,185,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(255, 214, 2)
SDO_NUMBER_ARRAY(255, 214, 2)
```

Example 5-5   Conditional Query with nodata Parameter

Example 5-5 is basically the same as Example 5-4, except that the `nodata` parameter value is set to `'TRUE'`, so that all NODATA pixels keep their original values from the input GeoRaster object in the output GeoRaster object.

```
DECLARE
  geor   SDO_GEORASTER;
  geor1  SDO_GEORASTER;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  INSERT into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
georaster into geor1;
  sdo_geor_ra.findcells(geor, '{ 1 }>200', null, geor1, null, 'TRUE');
  UPDATE georaster_table set georaster = geor1 WHERE georid = 5;
  COMMIT;
END;
/

SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1;
```

```
SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)

1 row selected.

-- This pixel keeps its original cell values because it is nodata, even though
-- the cell value of the second band is not greater than 200.
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE
georid=5;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)

1 row selected.
```

Example 5-6   Conditional Query with parallelParam

Example 5-6 finds all pixels that meet all of the following conditions:

- The cell value of the first band is between (100,200).

- The cell value of the second band is between [50,250].

- The cell value of the third band is greater than 100.

In addition, because `parallelParam` is specified as `'parallel=4'`, the procedure in Example 5-6 will run in parallel with four processes.

```
DECLARE
  geor   SDO_GEORASTER;
  geor1  SDO_GEORASTER;
BEGIN
   SELECT georaster INTO geor FROM georaster_table WHERE georid = 2;
   INSERT into georaster_table values (10, sdo_geor.init('rdt_1', 10)) returning
georaster into geor1;
   sdo_geor_ra.findcells(geor,'({1}>=50)&({1}<=250)&({0}>100)&({0}<200)&{2}>100)
',null,geor1,null,'false','parallel=4');
   UPDATE georaster_table SET georaster = geor1 WHERE georid = 10;
   COMMIT;
END;
/
```

5.3 Cell Value-Based Conditional Updates (Edits)

You can update raster cell values based on conditions.

This section pertains to cell value-based raster updates and not space-based raster updates, both of which types of update are described in Querying and Updating GeoRaster Cell Data.

To update raster cell values based on conditions, you can use the SDO_GEOR_RA.rasterUpdate procedure and specify appropriate `condition` and `vals` parameters.

The `condition` parameter specifies an array of Boolean expressions, and the `vals` parameter specifies an array of arrays of math expressions. (See the raster algebra operation explanations in Raster Algebra Language). For each cell, if `condition` is TRUE,

its cell value is updated to the result of the corresponding math expression in the `vals` array.

Example 5-7   Cell Value-Based Update

Example 5-7 assumes that the GeoRaster object to be updated is an image with three bands, and it calls the SDO_GEOR_RA.rasterUpdate procedure to do the following:

*   For any pixels if abs(first_band_value - second_band_value)=48 and (third_band_value - second_band_value=-101), then the three band values will be updated to (123,54,89), respectively.

*   For any pixels if (2*first_band_value - second_band_value/3)=108, then the three band values will be updated to (98,56,123), respectively.

Example 5-7 also includes several calls to the SDO_GEOR.getCellValue function to show "before" and "after" values.

```
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,130,130,'') FROM georaster_table WHERE
georid =1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,130,130,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(64, 60, 48)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,230,230,'') FROM georaster_table WHERE
georid =1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,230,230,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(11,11, 11)

1 row selected.

DECLARE
  geor    SDO_GEORASTER;
  geor1   SDO_GEORASTER;
BEGIN

  SELECT georaster into geor FROM georaster_table WHERE georid = 1;
  sdo_geor_ra.rasterUpdate(geor,0,SDO_STRING2_ARRAY('(abs({0}-{1})=48)&({2}-
{1}=-101)','2*{0}-{1}/
3=108'),SDO_STRING2_ARRAYSET(SDO_STRING2_ARRAY('123','54','89'),SDO_STRING2_ARRAY('98
','56','123')));
END;
/

PL/SQL procedure successfully completed.

show errors;
No errors.
```

```
-- This pixel gets updated because it meets the first condition.
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(123, 54, 89)

1 row selected.

--This pixel gets updated because it meets the second condition.
SELECT sdo_geor.getcellvalue(georaster,0,130,130,'') FROM georaster_table WHERE
georid=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,130,130,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(98, 56, 123)

1 row selected.

-- This pixel keeps its original values because it does not meet any condition
-- in the "condition" array.
SELECT sdo_geor.getcellvalue(georaster,0,230,230,'') FROM georaster_table WHERE
georid =1;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,230,230,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(11,11, 11)

1 row selected.
```

Example 5-8   Cell Value-Based Update with nodata Parameter

Example 5-8 is basically the same as Example 5-7, except that the `nodata` parameter
value is set to `'TRUE'`, so that all NODATA pixels keep their original values from the
input GeoRaster object in the output GeoRaster object.

```
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
-------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,130,130,'') FROM georaster_table WHERE
georid =1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,130,130,'')
-------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(64, 60, 48)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,230,230,'') FROM georaster_table WHERE
georid =1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,230,230,'')
-------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(11,11, 11)
```

```
1 row selected.

DECLARE
  geor    SDO_GEORASTER;
  geor1   SDO_GEORASTER;
BEGIN

  SELECT georaster into geor FROM georaster_table WHERE georid = 1;
  sdo_geor.addNODATA(geor, 1,88);
  sdo_geor_ra.rasterUpdate(geor,0,SDO_STRING2_ARRAY('(abs({0}-{1})=48)&({2}-
{1}=-101)','2*{0}-{1}/
3=108'),SDO_STRING2_ARRAYSET(SDO_STRING2_ARRAY('123','54','89'),SDO_STRING2_ARRAY('98
','56','123')),null,'true');
END;
/

PL/SQL procedure successfully completed.

-- This pixel keeps its original values because it is a NODATA pixel.
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)

1 row selected.

--This pixel gets updated because it meets the second condition.
SELECT sdo_geor.getcellvalue(georaster,0,130,130,'') FROM georaster_table WHERE
georid=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,130,130,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(98, 56, 123)

1 row selected.
```

5.4 Mathematical Operations

A major use of raster algebra is to apply mathematical models to raster layers from different sources.

To apply mathematical operations on one or multiple layers, which could be from one or more GeoRaster objects, to generate a new GeoRaster object, you can use the SDO_GEOR_RA.rasterMathOp procedure.

For most formats of this procedure, the `operation` parameter specifies an array of `arithmeticExpr` strings used to calculate raster cell values in the output GeoRaster object. Each element of the array corresponds to a layer in the output GeoRaster object.

Note that `booleanExpr` can be also used as `arithmeticExpr`, as is done in Example 5-8.

Example 5-9   Mathematical Operations (1)

Example 5-9 calls the SDO_GEOR_RA.rasterMathOp procedure to generate a new 6-layer GeoRaster object from a 3-layer source GeoRaster object, and follows these rules to calculate cell values of the target GeoRaster object:

- The cell value of the first three layers of target GeoRaster object is equal to the value of the corresponding layer of source GeoRaster object, minus 10.

- The cell value of the last three layers of target GeoRaster object is equal to the value of the first three layers of the source GeoRaster object, respectively.

```
DECLARE
  geor   SDO_GEORASTER;
  geor1  SDO_GEORASTER;
  geor2  SDO_GEORASTER;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  INSERT into georaster_table values (16, sdo_geor.init('rdt_1', 16)) returning
georaster into geor1;

sdo_geor_ra.rasterMathOp(geor,SDO_STRING2_ARRAY('{0,0}-10','{0,1}-10','{0,2}-10','{0,
0}','{0,1}','{0,2}'),null,geor1);
  UPDATE georaster_table SET georaster = geor1 WHERE georid = 16;
  COMMIT;
END;
/

PL/SQL procedure successfully completed.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=1;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 163, 159)

1 row selected.

--  In the results of the next SELECT statement, note:
--  171=181-10
--  153=163-10
--  149=159-10
--  181=181
--  163=163
--  159=159
SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid =16;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(171, 153, 149, 181, 163, 159)

1 row selected.
```

Example 5-10   Mathematical Operations (2)

Example 5-10 applies an operation on a 2-element GeoRaster array (containing two 3-layer source GeoRaster objects) to generate a new 3-layer GeoRaster object.

```
DECLARE
  geor       SDO_GEORASTER;
  geor1      SDO_GEORASTER;
  geor2      SDO_GEORASTER;
  geo_array  SDO_GEORASTER_ARRAY;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  SELECT georaster INTO geor2 FROM georaster_table WHERE georid = 2;
  INSERT into georaster_table values (17, sdo_geor.init('rdt_1', 17)) returning
```

```
georaster into geor1;
  geo_array:=SDO_GEORASTER_ARRAY(geor,geor2);

sdo_geor_ra.rasterMathOp(geo_array,SDO_STRING2_ARRAY('{0,0}-0.5*{1,0}','{0,1}-0.5*{1,
1}','{0,2}-0.5*{1,2}'),null,geor1,'false',null,'parallel=4');
  UPDATE georaster_table SET georaster = geor1 WHERE georid = 17;
  COMMIT;
END;
/

PL/SQL procedure successfully completed.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=1 or georid=2;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 163, 159)
SDO_NUMBER_ARRAY(60, 80, 90)

2 rows selected.

-- In the results of the next SELECT statement, note:
-- 151=181-0.5*60
-- 123=163-0.5*80
-- 114=159-0.5*90
SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid =17;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(151, 123 114)

1 row selected.
```

Example 5-11   Mathematical Operations (3)

Example 5-11 applies a subtraction operation on two 3-layer input GeoRaster objects to generate a new GeoRaster object. The example also includes several calls to the SDO_GEOR.getCellValue function to show "before" and "after" values.

```
SELECT sdo_geor.getcellvalue(georaster,0,10,10,'0-2') FROM georaster_table WHERE
georid=1 OR georid=5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,10,10,'0-2')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 137, 32)
SDO_NUMBER_ARRAY(98, 147, 42)

2 rows selected.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'0-2') FROM georaster_table WHERE
georid=1 OR georid=5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'0-2')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 163, 159)
SDO_NUMBER_ARRAY(191, 173, 169)

2 rows selected.
```

```
DECLARE
  geor0  SDO_GEORASTER;
  geor   SDO_GEORASTER;
  geor1  SDO_GEORASTER;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  SELECT georaster INTO geor0 FROM georaster_table WHERE georid = 5;
  INSERT into georaster_table values (6, sdo_geor.init('rdt_1', 6)) returning
georaster into geor1;
  sdo_geor_ra.rasterMathOp(geor0,geor,null,sdo_geor_ra.OPERATOR_SUBTRACT,null,geor1);
  UPDATE georaster_table SET georaster = geor1 WHERE georid = 6;
  COMMIT;
END;
/

PL/SQL procedure successfully completed.

SELECT sdo_geor.getcellvalue(georaster,0,10,10,'0-2') FROM georaster_table WHERE
georid=6;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,10,10,'0-2')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(10, 10, 10)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'0-2') FROM georaster_table WHERE
georid=6;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'0-2')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(10, 10, 10)

1 row selected.
```

5.5 Classification Operations

Classification (segmentation) operations can be applied on source GeoRaster objects to generate new objects.

To apply simple classification operations on source GeoRaster objects and generate new GeoRaster objects based on your specifications, you can use the SDO_GEOR_RA.classify procedure and specify the `expression`, `rangeArray`, and `valueArray` parameters. This classification procedure is also called *segmentation*.

The `expression` parameter is used to compute values that are used to map into the value ranges defined in the `rangeArray` parameter. The `rangeArray` parameter specifies a number array that defines ranges for classifying cell values, and this array must have at least one element. The `valueArray` parameter is a number array that defines the target cell value for each range, and its length must be the length of `rangeArray` plus one.

Example 5-12   Classification

Example 5-12 calls the SDO_GEOR_RA.classify procedure to apply a segmentation operation on the value of the first band of the input GeoRaster object. The example assumes that the GeoRaster object is an image.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  rangeArray SDO_NUMBER_ARRAY;
  valueArray SDO_NUMBER_ARRAY;
BEGIN
  rangeArray:=sdo_number_array(70,80,90,100,110,120,130,140,150,160,170,180);
  valueArray:=sdo_number_array(70,80,90,100,110,120,130,140,150,160,170,180,190);
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  INSERT into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
georaster into geor1;
  sdo_geor_ra.classify(geor,'{0}',rangeArray,valueArray,null,geor1);
  UPDATE georaster_table SET georaster = geor1 WHERE georid = 5;
  COMMIT;
END;
/

PL/SQL procedure successfully completed.

-- In the next statement, the target value is 90 because the value of the
-- first band of source GeoRaster object is 88, which is between 80 and 90.
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1 OR georid =5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)
SDO_NUMBER_ARRAY(90)

2 rows selected.

-- In the next statement, the target value is 190 because the value of the
-- first band of source GeoRaster object is 242, which is greater than 180.
SELECT sdo_geor.getcellvalue(georaster,0,132,116,'') FROM georaster_table WHERE
georid =1 OR georid =5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,132,116,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(242, 225, 233)
SDO_NUMBER_ARRAY(190)

2 rows selected.
```

Example 5-13   Classification with nodata and nodataValue Parameters

Example 5-13 calls the SDO_GEOR_RA.classify procedure to apply a segmentation
operation on the value of the first layer of the source GeoRaster object, and to set the
nodata parameter to 'TRUE' and the nodataValue parameter to 5, so that all NODATA
pixels will be set with a NODATA value of 5 in the target GeoRaster object.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  rangeArray SDO_NUMEBR_ARRAY;
  valueArray SDO_NUMEBR_ARRAY;
BEGIN
  rangeArray:=sdo_number_array(70,80,90,100,110,120,130,140,150,160,170,180);
  valueArray:=sdo_number_array(70,80,90,100,110,120,130,140,150,160,170,180,190);
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  sdo_geor.addNODATA(geor, 2,136);
  INSERT into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
```

```
georaster into geor1;
   sdo_geor_ra.classify(geor,'{0}',rangeArray,valueArray,null,geor1,'true',5);
   UPDATE georaster_table SET georaster = geor1 WHERE georid = 5;
END;
/

PL/SQL procedure successfully completed.

-- In the next statement, the target value of the cell is 5 because the value
-- of the second layer of the input GeoRaster object is 136, which is nodata.
SELECT sdo_geor.getcellvalue(georaster,0,30,30,'') FROM georaster_table WHERE georid
=1 OR georid =5 ORDER BY georid;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,30,30,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(88, 136, 35)
SDO_NUMBER_ARRAY(5)

2 rows selected.
```

5.6 Statistical Operations

To apply statistical operations on one or more layers, which are from one or more GeoRaster objects, the following types of operations are available.

- On-the-Fly Statistical Analysis
- Stack Statistical Analysis

5.6.1 On-the-Fly Statistical Analysis

Many applications require statistical analysis. GeoRaster provides statistical analysis functions that dynamically ("on the fly") compute complete statistical values for a GeoRaster object or the following individual statistical values: minimum, maximum, mean, median, mode, and standard deviation. You can do this without generating a histogram and updating the GeoRaster object metadata.

These subprograms support pyramids, band by band and the aggregation of specified band numbers. Each subprogram returns an SDO_NUMBER_ARRAY object or a number.

See the reference information for explanations and examples of these on-the-fly statistics computation subprograms:

- SDO_GEOR.generateStatistics
- SDO_GEOR.generateStatisticsMax
- SDO_GEOR.generateStatisticsMean
- SDO_GEOR.generateStatisticsMedian
- SDO_GEOR.generateStatisticsMin
- SDO_GEOR.generateStatisticsMode
- SDO_GEOR.generateStatisticsSTD

These subprograms do not modify the metadata in the GeoRaster object, except for some formats of SDO_GEOR.generateStatistics that set statistical data in the

GeoRaster object metadata and return a string value of TRUE or FALSE instead of an SDO_NUMBER_ARRAY object.

GeoRaster also provides statistical analysis functions that compute the area weighted statistical mean value for the cells and sub-cells within a specific window of the input GeoRaster object, and that calculate the three–dimensional (3D) surface area represented by digital elevation model (DEM) data that is stored in a GeoRaster object. See the reference information for explanations and examples of these on-the-fly statistics computation functions:

- SDO_GEOR.generateAreaWeightedMean
- SDO_GEOR_UTL.calcSurfaceArea

These two functions support irregular polygon clipping and sub-cell computation, thus providing very accurate results.

5.6.2 Stack Statistical Analysis

Stack statistical analysis generates a new one-layer GeoRaster object from one or more layers, which are from one or more GeoRaster objects, by computing one of the following statistical values for each cell: max, min, median, std, sum, minority, majority, or diversity.

To perform stack statistical analysis, you have the following options:

- Use the SDO_GEOR_RA.stack procedure.

  This option is more intuitive and does not require constructing raster algebra expressions (especially for GeoRaster objects with many layers), and it allows you to specify a list of layers instead of all layers.

- Use the SDO_GEOR_RA.rasterMathOp procedure.

  This option is more flexible and powerful, allowing you to perform more comlpicated statistical analysis.

Example 5-14   Using SDO_GEOR_RA.stack

This example uses the first option for performing stack statistical analysis. It calls the SDO_GEOR_RA.stack procedure to generate a new GeoRaster object by computing the maximum (max) value of layers 2 and 5 of two 3–layer source GeoRaster objects.

```
DECLARE
  geor       MDSYS.SDO_GEORASTER;
  geor1      MDSYS.SDO_GEORASTER;
  geor2      MDSYS.SDO_GEORASTER;
  geom       mdsys.sdo_geometry;
BEGIN
  geom:= sdo_geometry(2003,82394, NULL,
                      sdo_elem_info_array(1, 1003, 1),
                      sdo_ordinate_array(20283.775, 1011087.9,
                                         18783.775, 1008687.9,
                                         21783.775, 1008687.9,
                                         22683.775+0.001, 1009587.9+0.001,
                                         20283.775, 1011087.9));
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor2 from georaster_table where georid = 102;
  select georaster into geor1 from georaster_table where georid = 101 for update;

sdo_geor_ra.stack(SDO_GEORASTER_ARRAY(geor,geor2),geom,SDO_NUMBER_ARRAY(2,5),'max',nu
```

```
ll,geor1,'false',0,'TRUE');
  update georaster_table set georaster = geor1 where georid = 101;
END;
/

PL/SQL procedure successfully completed.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=100;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(121, 66, 181)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=102;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(33, 55, 56)

1 row selected.

--  In the results of the next SELECT statement, note:
--  max(181,56)  ==>  181

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid =101;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181)

1 row selected.
```

Example 5-15   Using SDO_GEOR_RA.rasterMathOp

This example uses the second option for performing stack statistical analysis. It calls
the sdo_GEOR_RA.rasterMathOp specifying a statistical operation (max) to perform
an operation similar to the preceding example, except that this example applies to all
layers.

```
DECLARE
  geor       MDSYS.SDO_GEORASTER;
  geor1      MDSYS.SDO_GEORASTER;
  geor2      MDSYS.SDO_GEORASTER;
  geo_array  MDSYS.SDO_GEORASTER_ARRAY;
BEGIN
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101;
  select georaster into geor2 from georaster_table where georid = 102 for update;
  geo_array:=MDSYS.SDO_GEORASTER_ARRAY(geor,geor1);
  sdo_geor_ra.rasterMathOp(geo_array,SDO_STRING2_ARRAY('max()'),null,geor2);
  update georaster_table set georaster = geor2 where georid = 102;
  commit;
END;
/

PL/SQL procedure successfully completed.
```

```
SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=100;
 SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 163, 159)
1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=101;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 122, 159) 1 row selected.

-- In the results of the next SELECT statement, note:
-- max(181,163,159,181,122,159)  ==> 181

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid =102;
 SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
 --------------------------------------------------------------------------------
 SDO_NUMBER_ARRAY(181)
1 row selected.
```

5.7 Logical Operations

A major use of raster algebra is to apply logical models to raster layers from different sources; that is, you can apply logical operations on one or more layers, from one or more GeoRaster objects, to generate a new GeoRaster object.

To apply logical operations, you can either use raster algebra procedures with logical expressions, which is more flexible and powerful and mostly be used for some complicated raster logical operations, or use raster algebra procedures only, which are straightforward and do not require constructing complicated logical expressions. However, using raster algebra procedures only (that is, without logical expressions) has some limitations and is mainly used for some specific raster logical operations.

- Using Raster Algebra Procedures with Logical Expressions
- Using Raster Algebra Functions Only

5.7.1 Using Raster Algebra Procedures with Logical Expressions

GeoRaster logical expressions can be conditional expressions, boolean expressions, or both, which can take any combination of unary and binary boolean operators (!, &, |, ^) and comparison operators (=, <, >, <=, >=, !=).

To apply logical expressions on the raster data, you must use raster algebra procedures defined in the SDO_GEOR_RA package and specify appropriate parameters with your constructed logical expressions.

Example 5-16   Using SDO_GEOR_RA.rasterMathOp with condition operators

This example implements logic described in the following pseudocode to implement 3–band raster data segmentation:

```
if ( (layer1 < 100)
     & (layer2< 1000)
     & (layer3< 500))
then output = 10
```

```
  elsif ( (layer1 < 200)
         & (layer2< 2000)
         & (layer3< 1000))
 then output = 20
 elsif ( (layer1 < 300)
         & (layer2< 3000)
         & (layer3< 1500))
 then output = 30
 elsif ( (layer1 < 400)
         & (layer2< 4000)
         & (layer3< 2000))
 then output = 40
 elsif ( (layer1 < 500)
         & (layer2< 5000)
         & (layer3< 2500))
 then output = 50
 else
     output = 0
```

The example calls the SDO_GEOR_RA.rasterMathOp procedure, as follows

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  mycursor    sys_refcursor;
  expr        varchar2(1024);
BEGIN
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101 for update;
  --construct logical expression
  expr :='condition(((({0}<100)&({1}<1000)&({2}<500)),'||
                    '10,'||
                    'condition((({0}<200)&({1}<2000)&({2}<1000)),'||
                              '20,'||
                              'condition((({0}<300)&({1}<3000)&({2}<1500)),'||
                                        '30,'||

'condition((({0}<400)&({1}<4000)&({2}<2000)),'||
                                                  '40,'||

'condition((({0}<500)&({1}<5000)&({2}<2500)),'||
                                                          '50,'||
                                                          '0)'||
                                                  ')'||
                                        ')'||
                              ')'||
                    ')';

  sdo_geor_ra.rasterMathOp(geor, sdo_string2_array(expr),null, geor1, 'true', 0,
'parallel=4');
  update georaster_table set georaster = geor1 where georid = 101;
  commit;
END;
/
```

Example 5-17   Using SDO_GEOR_RA.rasterMathOp with a condition operator

This example uses statistical functions and arithmetic operations to implement the simple logic described in the following pseudocode:

```
if (sum()>min()*3)
then
```

```
   output = sqrt(layer0+layer2)
else
   output = layer1*1.5
```

The example calls the SDO_GEOR_RA.rasterMathOp procedure, as follows

```
DECLARE
  geor       SDO_GEORASTER;
  geor1      SDO_GEORASTER;
  mycursor   sys_refcursor;
  expr       varchar2(1024);
BEGIN
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101 for update;
  --construct logical expression
  expr :='condition(sum()>min()*3,sqrt({0}+{2}),{1}*1.5)';
  sdo_geor_ra.rasterMathOp(geor, sdo_string2_array(expr),null, geor1, 'true', 0,
'parallel=4');
  update georaster_table set georaster = geor1 where georid = 101;
  commit;
END;
/
```

5.7.2 Using Raster Algebra Functions Only

To perform logical operations using only raster algebra functions, you have the following options

- Use the SDO_GEOR_RA.diff procedure.

  For example, if a cell value in raster A is different from the cell value in raster B, the cell value in raster A is returned. If the cell values are the same, the value 0 (zero) is returned.

- Use the SDO_GEOR_RA.over procedure.

  For example, if a cell value in raster A is not equal to 0 (zero), the cell value in raster A is returned. If the cell value in raster A is equal to 0, the cell value in raster B is returned.

Example 5-18   Using SDO_GEOR_RA.diff

This example calls the SDO_GEOR_RA.diff procedure to generate a new GeoRaster object from two 3–layer source GeoRaster objects.

```
DECLARE
  geor       SDO_GEORASTER;
  geor1      SDO_GEORASTER;
  geor2      SDO_GEORASTER;
  geom       sdo_geometry;
BEGIN
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101;
  select georaster into geor2 from georaster_table where georid = 102 for update;
  geom:=null;
  sdo_geor_ra.diff(geor,geor1,geom,null,geor2);
  update georaster_table set georaster = geor2 where georid = 102;
END;
/

PL/SQL procedure successfully completed.
```

```
SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=100;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 163, 159)

1 row selected.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=101;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 122, 159)

1 row selected.

--  In the results of the next SELECT statement, note:
--  181 =181  ==>   0
--  163!=122  ==>  163
--  159 =159  ==>   0

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid =102;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0, 163, 0)

1 row selected.
```

Example 5-19   Using SDO_GEOR_RA.over

This example calls the SDO_GEOR_RA.over procedure to generate a new GeoRaster
object from two 3–layer source GeoRaster objects.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  geor2       SDO_GEORASTER;
  geom        sdo_geometry;
BEGIN
  select georaster into geor from georaster_table where georid = 102;
  select georaster into geor1 from georaster_table where georid = 101;
  select georaster into geor2 from georaster_table where georid = 100 for update;
  geom:=null;
  sdo_geor_ra.over(geor,geor1,geom,null,geor2);
  update georaster_table set georaster = geor2 where georid = 100;
END;
/

PL/SQL procedure successfully completed.

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=102;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0, 163, 0)

1 row selected.
```

```
SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid=101;

SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 122, 159)

1 row selected.

-- In the results of the next SELECT statement, note:
-- 0 =0    ==> 181      result from georid=101
-- 163!=0  ==> 163      result from georid=102
-- 0 =0    ==> 159      result from georid=101

SELECT sdo_geor.getcellvalue(georaster,0,100,100,'') FROM georaster_table WHERE
georid =100;
SDO_GEOR.GETCELLVALUE(GEORASTER,0,100,100,'')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(181, 163, 159)

1 row selected.
```

5.8 Raster Data Scaling and Offsetting

You can perform raster data scaling and offsetting operations.

Raster algebra has many applications, such as cartographic modeling (see Cartographic Modeling), vegetation index computing (see Vegetation Index Computation), and tasseled cap transformation (see Tasseled Cap Transformation). Topics in this chapter and in Image Processing and Virtual Mosaic describe a few sample applications of the GeoRaster raster algebra.

Example 5-20   Converting DEM Data from Feet to Meters

The cell value of a GeoRaster object may represent a quantitative attribute of spatial objects, which could be in a specific unit. For example, the elevation data in a DEM GeoRaster object could be in the unit of feet. An application may require you to convert the elevations into another unit, such as meters, for georectification and other operations. You can use the raster algebra to scale the DEM data from feet into meters (that is, unit conversion), as shown in Example 5-20.

```
DECLARE
  geor1     SDO_GEORASTER;
  geor2     SDO_GEORASTER;
BEGIN
  --Source GeoRaster object with a single DEM layer
  select georaster into geor1 from georaster_table where georid = 1;
  --To store the output DEM layer
  select georaster into geor2 from georaster_table where georid = 2 for update;
  --Scale elevation from feet to meters using the unit factor
  sdo_geor_ra.rasterMathOp(geor1,SDO_STRING2_ARRAY('{0} * 0.3048'),null,geor2);
  --Commit changes to the output georaster object
  update georaster_table set georaster = geor2 where georid = 2;
  commit;
END;
/
```

Example 5-21   Offsetting DEM by Geoid Height

The cell data of a GeoRaster object may need to be offset by a constant for further processing. For example, a DEM layer may represent orthometric elevation instead of ellipsoidal elevation. To orthorectify a raw image georeferenced by an RPC model requires ellipsoidal elevation. Example 5-21 offsets the orthometric DEM by the geoid height, resulting in an ellipsoidal DEM.

```
DECLARE
  geor1    SDO_GEORASTER;
  geor2    SDO_GEORASTER;
BEGIN
  --Source GeoRaster object with a single orthometric DEM layer
  select georaster into geor1 from georaster_table where georid = 1;
  --To store the output DEM layer
  select georaster into geor2 from georaster_table where georid = 2 for update;
  --Offset elevation by geoid height to get ellipsoidal elevation
  sdo_geor_ra.rasterMathOp(geor1,SDO_STRING2_ARRAY('{0} - 28.8'),null,geor2);
  --Commit changes to the output GeoRaster object
  update georaster_table set georaster = geor2 where georid = 2;
  commit;
END;
/
```

Example 5-22   Converting (Scaling) and Offsetting

You can combine the operations of Example 5-20 and Example 5-21 into a single simple step, as shown in Example 5-22.

```
DECLARE
  geor1    SDO_GEORASTER;
  geor2    SDO_GEORASTER;
BEGIN
  --Source GeoRaster object with a single DEM layer
  select georaster into geor1 from georaster_table where georid = 1;
  --To store the output DEM layer
  select georaster into geor2 from georaster_table where georid = 2 for update;
  --Scale elevation from feet to meters and offset elevation by geoid height
  sdo_geor_ra.rasterMathOp(geor1,SDO_STRING2_ARRAY('{0} * 0.3048 -
28.8'),null,geor2);
  --Commit changes to the output georaster object
  update georaster_table set georaster = geor2 where georid = 2;
  commit;
END;
/
```

5.9 Raster Data Casting

Raster data casting maps cell values from one data type to another.

In GeoRaster, there are two types of casting operations: one uses the `cellDepth` keyword in the `storageParam` parameter of operations, and the other uses the `castingExpr` operation in the GeoRaster raster algebra. (`castingExpr` is one of the `arithmeticExpr` operations, as described in Raster Algebra Language.)

Whenever you apply an operation which stores the raster data result into a new GeoRaster object, you can use the `cellDepth` keyword in the `storageParam` parameter of that operation. (The `cellDepth` keyword and its values are described in Table 1-1.) If the `cellDepth` is specified, the target GeoRaster object will be created using that

`cellDepth` value, and the raster cell data will be automatically cast to that `cellDepth` value for storage. You can directly use `cellDepth` in the `storageParam` parameter to do the casting if the source data is in lower cell depth and the resulting data is in higher cell depth. In this case, the casting is transparent and fast.

However, if you specify a *lower* cell depth for data in higher cell depth, changing the cell depth using the `cellDepth` keyword in the `storageParam` parameter can cause loss or change of data and reduced precision or quality. To have better control of the precision and accuracy, you can use the Raster Algebra casting operator, `castingExpr`.

For example, assume you have a raster with a cell depth of `32BIT_REAL` and a value range in [0.0, 100.0). You can use Example 5-23 to perform linear segmentation of the raster into 10 different classes, each of which has a cell value that is a multiple of 10 (0, 10, 20, …, 90), using the `castint` operator. This operation casts all cell values to their closest lower multiple of 10; for example, all numbers from 60 to 69 are cast to 60.

Example 5-23   Linear Segmentation of a Raster

```
DECLARE
  geor1    SDO_GEORASTER;
  geor2    SDO_GEORASTER;
BEGIN
  --Source georaster object with cell value range [0.0,100.0)
  select georaster into geor1 from georaster_table where georid = 1;
  --Target georaster object to store the output layer
  select georaster into geor2 from georaster_table where georid = 2 for update;
  --Linearly segment the source raster into 10 classes and store in 8BIT cell depth
  sdo_geor_ra.rasterMathOp(geor1,
    SDO_STRING2_ARRAY('(castint({0}/10)*10'),
    'celldepth=8BIT',
    geor2);
  --Commit changes to the output georaster object
  update georaster_table set georaster = geor2 where georid = 2;
  commit;
END;
/
```

As shown in Example 5-23, you can combine the usage of the `cellDepth` keyword in the `storageParam` parameter with the raster algebra casting operator, so that the result can be calculated correctly as well as stored in an appropriate and concise way. In Example 5-23, the output cell values are integers equal to or less than 90, so the resulting raster can be stored using `8BIT` cell depth (instead of `32BIT_REAL`), which saves storage space.

5.10 Cartographic Modeling

Raster algebra is widely used in cartographic modeling and is considered an essential component of GIS systems. Using the PL/SQL and the raster algebra expressions and functions, you can conduct cartographic modeling over a large number of rasters and images of virtually unlimited size.

For example, a cartographic modeling process for wildfire evaluation might retrieve the elevation, slope, aspect, temperature, wetness, and other information from a series of raster layers and then evaluate the cells one-by-one to create a resulting raster map, which can be further classified to create a thematic map. Change analysis, site selection, suitability analysis, climate modeling, and oil field evaluation using the raster

layer overlay technique are other typical cartographic modeling processes. In those cases, arithmetic, relational, and logical operations may need to be combined.

Assume that a hypothetical cartographic model involves seven different raster layers and has an expression as follows. and that the modeling result is a raster map with 0 and 1 as cell values:

```
output = 1 if ( (100 < layer1 <= 500)
               & (layer2 == 3 or layer2 == 10)
               & ( (layer3+layer4) * log(Layer5) / sqrt(layer5) ) >= layer6)
               || (layer7 != 1) )
               is TRUE and
         0 if otherwise
```

Example 5-24 shows how to run the preceding cartographic model in GeoRaster and store the result as a bitmap.

Example 5-24   Cartographic Modeling

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  mycursor    sys_refcursor;
  expr        varchar2(1024);
BEGIN
  --7 source GeoRaster objects, each of which contains one source layer in the order
of 1 to 7
  OPEN mycursor FOR
    select georaster from georaster_table where georid >0 and georid <=7 order by
georid;
  --Output GeoRaster object to contain the result
  insert into georaster_table (georid, georaster) values (8, sdo_geor.init('RDT_1',
8))
    returning georaster into geor1;
  --Modeling using arithmeticExpr, booleanExpr, and rasterMathOp
  expr :=
   'condition(
        ( (100<{0,0}) & ({0,0}<=500) )
         & ( ({1,0}=3) | ({1,0}=10) )
         & ( ( ( ({2,0}+{3,0}) * log({4,0} ) / sqrt({4,0}) ) >= {5,0} ) | ({6,0}!
=1)
        ),
        1,
        0)';
  sdo_geor_ra.rasterMathOp(mycursor, sdo_string2_array(expr),
                          'celldepth=1BIT', geor1, 'true', 0, 'parallel=4');
  update georaster_table set georaster = geor1 where georid = 8;
  commit;
END;
/
```

The process in Example 5-24 considers NODATA and will assign 0 (zero) to any cell that is a NODATA cell in one or more source layers. It is also parallelized into four processes to leverage multiple CPUs of the database server to improve performance.

6

Image Processing and Virtual Mosaic

This chapter describes advanced image processing capabilities, including GCP georeferencing, reprojection, rectification, orthorectification, warping, image scaling, stretching, filtering, masking, segmentation, NDVI computation, Tasseled Cap Transformation, image appending, bands merging, and large-scale advanced image mosaicking.

This chapter also describes the concept and application of *virtual mosaic* within the context of a large-scale image database and on-the-fly spatial queries over it.

The operations in this chapter are most commonly used to process geospatial images, particularly raw satellite imagery and airborne photographs. However, those operations, just like the GeoRaster raster algebra, apply to all raster data types.

This chapter contains the following major sections.

Topics:

- Advanced Georeferencing
  In addition to spatial referencing capability, advanced georeferencing capabilities are available.

- Image Reprojection
  Image reprojection is the process of transforming an image from one SRS (spatial reference system, or coordinate system) to another.

- Image Rectification
  Most raster data originating from remote sensors above the ground is usually subject to distortion caused by the terrain, the view angles of the instrument, and the irregular shape of the Earth. Image rectification as explained in this section is the process of transforming the images to reduce some of that distortion.

- Image Orthorectification
  Orthorectification is a rectification transformation process where information about the elevation, the terrain, and the shape of the Earth is used to improve the quality of the output rectified image. Oracle GeoRaster supports single image orthorectification with average height value or DEM.

- Image Warping
  Image warping transforms an input GeoRaster object to an output GeoRaster object using the spatial reference information from a specified SDO_GEOR_SRS object.

- Image Affine Transformation and Scaling
  Affine transformation is the process of using geometric transformations of translation, scaling, rotation, shearing, and reflection on an image to produce another image.

- Image Stretching, Normalization, Equalization, Histogram Matching, and Dodging
  The color and contrast of images can be enhanced to improve their visual quality. The SDO_GEOR_IP package ("IP" for image processing) provides a set of subprograms for image enhancement, including performing image stretching, image normalization, image equalization, histogram matching, and image dodging.

- **Image Filtering**

  Image filtering is the process of applying a convolution filter on an image to achieve a specific purpose. For example, applying a low-pass filter on an image can smooth and reduce noise in an image, while applying a high-pass filter on an image can enhance the details of the image or even detect the edges inside the image.

- **Image Segmentation**

  Segmentation is a simple type of classification algorithm, and can be useful in classifying certain types of images into larger ground feature categories, such as land, cloud, water, or snow.

- **Image Pyramiding: Parallel Generation and Partial Update**

  Image pyramiding is one of the most commonly used processes in building large-scale image databases.

- **Bitmap Pyramiding**

  Bitmap pyramiding can produce high-quality pyramids in certain cases where traditional pyramiding is not adequate.

- **Vegetation Index Computation**

  In remote sensing, the Normalized Difference Vegetation Index (NDVI) is a widely used vegetation index, enabling users to quickly identify vegetated areas and monitor the growth and "condition" of plants.

- **Tasseled Cap Transformation**

  Tasseled Cap Transformation (TCT) is a useful tool for analyzing physical ground features using remotely sensed imagery.

- **Image Masking**

  To perform image masking, an application can query the GeoRaster database for bitmap masks, retrieve the desired bitmap mask or masks, and apply the masking operation on the target GeoRaster object for the purpose of displaying the object or performing some other processing.

- **Band Merging**

  For image classification, time series analysis, and raster GIS modeling, multiple bands or layers of different GeoRaster objects may need to be merged into a single GeoRaster object.

- **Image Appending**

  You can append one image to another image when the two images have the same number of bands.

- **Large-Scale Image Mosaicking**

  A large geospatial area typically consists of many smaller aerial photographs or satellite images. Large-scale image mosaicking can stitch these small geospatial images into one large image to get a better view of the whole spatial area.

- **Virtual Mosaic**

  A virtual mosaic treats a set of GeoRaster images as one large virtually mosaicked image.

- **Image Serving**

  Serving of image and raster data to clients or applications is supported through many features of the GeoRaster PL/SQL and Java APIs.

6.1 Advanced Georeferencing

In addition to spatial referencing capability, advanced georeferencing capabilities are available.

In GeoRaster, the spatial referencing capability is called SRS (spatial reference system) or georeferencing, which may or may not be related to geography or a geospatial scheme. Georeferencing is a key feature of GeoRaster and is the foundation of spatial query and operations over geospatial image and gridded raster data. See Georeferencing for a detailed description of the SRS models.

GeoRaster supports non-geospatial images, fine art photos, and multi-dimensional arrays, which might not be associated with any coordinate system. For those images and rasters, there is generally no need for georeferencing, but most of the GeoRaster operations still work on them, such as pyramiding, scaling, subsetting, band merging, stretching, and algebraic operations. In these cases, you address the pixels (cells) using the raster's cell space coordinates (that is, row, column, and band).

You can also create a user-defined coordinate system (a new SRID) that is not related to geography, and you can use that SRID as the model coordinate system for the rasters. Then, you can spatially reference these rasters to that SRID; that is, an SRS metadata component will be created for each of those rasters. Doing this causes those rasters to be spatially referenced, and thus co-located in that user-defined model coordinate system. After this is done for all related rasters, GeoRaster operations will work on those rasters as if they are georeferenced to a geographic coordinate system. For example, assume that an artist has painted a large mural on a wall, and that you want to be able to take many high-resolution photographs of different tiles of this wall and then stitch them together. You can spatially reference the tile images and then use the GeoRaster mosaicking capability to do the stitching.

If you do not define a new coordinate system, you can still co-locate the images in the cell space. That is, you can set up different ULT coordinates for the images by calling the SDO_GEOR.setULTCoordinate procedure, so that the images are aligned in the same coordinate system and then can be mosaicked.

Most geospatial image and raster files that you have are probably already georeferenced by other software tools, and thus they may come with georeferencing information. In those cases, the georeferencing information can be directly loaded with the rasters or afterward by using SDO_GEOR.importFrom, SDO_GEOR.setSRS, the GeoRaster loader tool, GDAL, or other third-party ETL tools. For more information, check GeoRaster Tools: Viewer_ Loader_ Exporter and Georeferencing GeoRaster Objects.

If a geospatial image does not have spatial reference information, you can use the GeoRaster Ground Control Point (GCP) support to georeference the image. GCPs are collected either automatically by the remote sensing system or manually afterward. For an image without GCP information, you can use a GeoRaster visualization tool to collection GCPs for the GeoRaster object. GCPs are described in Ground Control Point (GCP) Georeferencing Model.

After you have the GCPs and want to store them in the GeoRaster metadata, you can get and set the GCP-based georeferencing mode by using the SDO_GEOR.getGCPGeorefModel function and the SDO_GEOR.setGCPGeorefModel procedure. To get, set, and edit only GCPs, use the SDO_GEOR.getControlPoint function and the SDO_GEOR.setControlPoint and SDO_GEOR.deleteControlPoint

procedures. The GCPs can also be stored in the GeoRaster metadata when you call SDO_GEOR.georeference.

To get and set only the geometric model, use the SDO_GEOR.getGCPGeorefMethod function and the SDO_GEOR.setGCPGeorefMethod procedure. GeoRaster also allows you to store check points (pointType = 2), which are treated and manipulated in the same way as control points (pointType = 1) except that check points are not used to create the SRS coefficient when SDO_GEOR.georeference is called with the GCPs.

If you have ground control points (GCPs) that are either stored in the GeoRaster object or not, and if you want to calculate the functional fitting georeferencing model, you can call the SDO_GEOR.georeference procedure to find the solution. The functional fitting georeferencing model stores all coefficients in the GeoRaster SRS and enables the coordinate transformations between cell space and model space. To generate the functional fitting georeferencing model using GCP, you must specify an appropriate geometric model. The specific geometric models supported by SDO_GEOR.georeference are Affine Transformation, Quadratic Polynomial, Cubic Polynomial, DLT, Quadratic Rational, and RPC. These models are described in Functional Fitting Georeferencing Model.

Example 6-1   Setting Up the GCP Georeferencing Model

For example, if you have a Landsat image in a plain area and want to georeference it, you might choose the Quadratic Polynomial geometric model. For that purpose, assuming you have collected 9 GCPs (at least 6 GCPs in this case) and 3 check points, you can set up the GCPs and store them in the GeoRaster's metadata using the code in Example 6-1.

```
DECLARE
   gr1         sdo_georaster;
   georefModel SDO_GEOR_GCPGEOREFTYPE;
   GCPs        SDO_GEOR_GCP_COLLECTION;
BEGIN
   SELECT georaster INTO gr1 from georaster_table WHERE georid=1 FOR UPDATE;
   GCPs := SDO_GEOR_GCP_COLLECTION(
               SDO_GEOR_GCP('1', '', 1,
                   2, sdo_number_array(25, 73),
                   2, sdo_number_array(237036.9, 897987.2),
                   NULL, NULL),
               SDO_GEOR_GCP('2', '', 1,
                   2, sdo_number_array(100, 459),
                   2, sdo_number_array(237229.6, 897949.7),
                   NULL, NULL),
               SDO_GEOR_GCP('3', '', 1,
                   2, sdo_number_array(362, 77),
                   2, sdo_number_array(237038.9, 897818.8),
                   NULL, NULL),
               SDO_GEOR_GCP('4', '', 1,
                    2, sdo_number_array(478, 402),
                    2, sdo_number_array(237201.06, 897760.56),
                   NULL, NULL),
               SDO_GEOR_GCP('5', '', 1,
                   2, sdo_number_array(167, 64),
                   2, sdo_number_array(237032.02, 897916.26),
                   NULL, NULL),
               SDO_GEOR_GCP('6', '', 1,
                   2, sdo_number_array(101, 257),
                   2, sdo_number_array(237128.9, 897949.3),
                   NULL, NULL),
               SDO_GEOR_GCP('7', '', 1,
```

```
                            2, sdo_number_array(235, 501),
                            2, sdo_number_array(237250.9, 897882.2),
                            NULL, NULL),
                    SDO_GEOR_GCP('8', '', 1,
                            2, sdo_number_array(423, 214),
                            2, sdo_number_array(237107.3, 897788.0),
                            NULL, NULL),
                    SDO_GEOR_GCP('9', '', 1,
                            2, sdo_number_array(127, 178),
                            2, sdo_number_array(237089.0, 897936.5),
                            NULL, NULL),
                    SDO_GEOR_GCP('10', '', 2,
                            2, sdo_number_array(131, 425),
                            2, sdo_number_array(237212.8, 897934.2),
                            NULL, NULL),
                    SDO_GEOR_GCP('11', '', 2,
                            2, sdo_number_array(299, 111),
                            2, sdo_number_array(237055.7, 897850.4),
                            NULL, NULL),
                    SDO_GEOR_GCP('12', '', 2,
                            2, sdo_number_array(329, 253),
                            2, sdo_number_array(237126.9,897835.4),
                            NULL, NULL) );
    georefModel := SDO_GEOR_GCPGEOREFTYPE('QuadraticPolynomial', GCPs.count, GCPs,
NULL);
    -- Set and store the GCP georeference model into the GeoRaster object's metadata
    sdo_geor.setGCPGeorefModel(gr1, georefModel);
    UPDATE georaster_table SET georaster=gr1 WHERE georid=1;
    COMMIT;
END;
/
```

Example 6-2   Generating the Functional Fitting Model Using GCPs

After using the code in Example 6-1, you can generate the functional fitting model coefficients by using the code in Example 6-2.

```
DECLARE
    gr1  sdo_georaster;
    rms  sdo_number_array;
BEGIN
    SELECT georaster INTO gr1 from georaster_table WHERE georid=1 FOR UPDATE;
    -- georeference the image using the GCPs stored in the image's metadata
    rms := sdo_geor.georeference(gr1, null, 26986, 0, 'TRUE');
    UPDATE georaster_table SET georaster=gr1 WHERE georid=1;
    COMMIT;
END;
/
```

The steps in Example 6-1 and Example 6-2 can be combined without the need to pre-set the GCPs into the GeoRaster object's metadata (see the example for SDO_GEOR.georeference in SDO_GEOR Package Reference). The returned value array of SDO_GEOR.georeference in Example 6-2 contains RMS values and residuals for each GCP. Using these, you can examine the solution accuracy and identify erratic GCPs. If the accuracy is not satisfactory, recheck all GCPs to make sure they are accurate and add more GCPs as necessary, and then run the script or scripts again.

The GCP support in GeoRaster enables you to spatially reference any *non*-geospatial images and rasters also.

After geospatial images are georeferenced, you can process those images, such as applying rectification, reprojection, and mosaicking, and spatially querying and subsetting the rasters using geometry polygons in different coordinate systems.

6.2 Image Reprojection

Image reprojection is the process of transforming an image from one SRS (spatial reference system, or coordinate system) to another.

Reprojection is particularly useful with certain GeoRaster operations that combine two or more objects, because it requires that all the GeoRaster objects involved be in the same SRS.

Basic reprojection in GeoRaster is performed by the SDO_GEOR.reproject procedure and requires that the source GeoRaster SRID be different from the output SRID.

Example 6-3   Image Reprojection

Example 6-3 reprojects a raster image that had been loaded into a GeoRaster object with SRID 4326, but needs to be reprojected to have the same SRID as other images previously stored with SRID 23619.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  select raster into gr1 from georaster_load_table where georid = 10;
  delete from georaster_table where georid = 54;
  insert into georaster_table
        values(54,'reprojected', sdo_geor.init())
        returning georaster into gr2;
  sdo_geor.reproject(inGeoRaster   => gr1,
                     pyramidLevel  => 0,
                     cropArea      => null,
                     layerNumbers  => null,
                     resampleParam => 'resampling=BILINEAR',
                     storageParam  => null,
                     outSRID       => 32619,
                     outGeoraster  => gr2);
  update georaster_table set georaster = gr2 where georid = 54;
  commit;
END;
```

The same operation can be accomplished by the SDO_GEOR.rectify procedure, producing similar results. The SDO_GEOR.rectify procedure offers more capabilities and flexibility than SDO_GEOR.reproject; for example, the input and output SRID can be the same and users can specify the precise resolution of the output (see Image Rectification).

If a GeoRaster object does not have an associated SRS, the process for georeferencing and rectifying it is explained in Georeferencing GeoRaster Objects and Image Rectification.

Parallel reprojection is supported with the SDO_GEOR_AGGR.mosaicSubset procedure.

6.3 Image Rectification

Most raster data originating from remote sensors above the ground is usually subject to distortion caused by the terrain, the view angles of the instrument, and the irregular shape of the Earth. Image rectification as explained in this section is the process of transforming the images to reduce some of that distortion.

Rectification is performed by the SDO_GEOR.rectify procedure, and requires that the source GeoRaster object have at least a functional fitting georeferencing model. This means that the image does not need to be rectified, but it needs to have georeference information in the metadata (see Georeferencing GeoRaster Objects).

The SDO_GEOR.rectify procedure can use the information available in the source GeoRaster object to automatically establish the spatial extents, dimension, and SRID of the output GeoRaster, and users can also specify different values by using the appropriate parameters.

Example 6-4   Image Rectification

Example 6-3 rectifies an aerial image that had been loaded into GeoRaster and later georeferenced with GCPs (see Advanced Georeferencing). The image is rectified so that the output GeoRaster object has the same SRS and resolution of an existing GeoRaster object. The image is to be restricted to the area of existing GeoRaster object, and the pixels should be perfectly aligned with the existing GeoRaster object.

```
DECLARE
  gr_src sdo_georaster;
  gr_ref sdo_georaster;
  gr_out sdo_georaster;
BEGIN
  select raster into gr_src from georaster_load_table where georid = 15;
  select raster into gr_ref from georaster_table where georid = 1;
  delete from georaster_table where georid = 2;
  insert into georaster_table
         values(2, 'rectified', sdo_geor.init())
         returning georaster into gr_out;
  sdo_geor.rectify(inGeoRaster       => gr_src,
                   pyramidLevel      => null,
                   elevationParam    => null,
                   dem               => null,
                   outSRID           => sdo_geor.getModelSRID(gr_ref),
                   outModelCoordLoc  => null,
                   cropArea          => sdo_geor.generateSpatialExtent(gr_ref),
                   polygonClip       => null,
                   layerNumbers      => null,
                   outResolutions    => sdo_geor.getSpatialResolutions(gr_ref),
                   resolutionUnit    => 'unit=meters',
                   referencePoint    => sdo_geor.getModelCoordinate(gr_ref,
                                            0, sdo_number_array(-0.5,-0.5)),
                   resampleParam     => null,
                   storageParam      => null,
                   outGeoraster      => gr_out);
  update georaster_table set georaster = gr_out where georid = 2;
  commit;
END;
```

Rectification output can be significantly improved if information about elevation is passed to the SDO_GEOR.rectify procedure. (See Image Orthorectification for more information about elevation.)

Parallel rectification is supported with the SDO_GEOR_AGGR.mosaicSubset procedure.

6.4 Image Orthorectification

Orthorectification is a rectification transformation process where information about the elevation, the terrain, and the shape of the Earth is used to improve the quality of the output rectified image. Oracle GeoRaster supports single image orthorectification with average height value or DEM.

The orthorectification is done by the SDO_GEOR.rectify procedure and requires that the source GeoRaster have a 3D SRS. The SDO_GEOR.rectify procedure can execute orthorectification with just the average height of the area or with a detailed Digital Elevation Model (DEM).

- Orthorectification with Average Height
- Orthorectification with DEM

6.4.1 Orthorectification with Average Height

A GeoRaster object with a Digital Elevation Model (DEM) is optional for orthorectification. For relatively flat terrains, the 3D SRS together with the average height value might be sufficient to correct the distortion of the source image.

Example 6-5   Orthorectification with Average Height

Example 6-5 shows orthorectification with average height. For this example, the source image was acquired from DigitalGlobe with RPC. The DEM was not available, but the average elevation of the area is known to be 1748.0 meters.

```
DECLARE
  gr_src  sdo_georaster;
  gr_out  sdo_georaster;
BEGIN
  select georaster into gr_src from georaster_table where georid = 1;
  delete from georaster_table where georid = 3;
  insert into georaster_table values(3, 'orthorectified without DEM',
         sdo_geor.init('rdt_4',3))
         returning georaster into gr_out;
  sdo_geor.rectify(inGeoRaster     => gr_src,
                   pyramidLevel    => null,
                   elevationParam  => 'average=1748.8',
                   dem             => null,
                   outSRID         => 32613,
                   outModelCoordLoc => null,
                   cropArea        => null,
                   polygonClip     => null,
                   layerNumbers    => null,
                   outResolutions  => null,
                   resolutionUnit  => null,
                   referencePoint  => null,
                   resampleParam   => 'resampling=AVERAGE4',
                   storageParam    => null,
                   outGeoraster    => gr_out);
```

```
                update georaster_table set georaster = gr_out where georid = 3;
                commit;
END;
```

In Example 6-5, the `dem` parameter is null, and the `elevationParam` average elevation must be in the same unit as the SRS. Also, in `elevationParam` the `offset` and scale `keywords` are *not* specified because they are relevant only if DEM is specified.

6.4.2 Orthorectification with DEM

The use of a DEM (Digital Elevation Model) layer improves the accuracy of the rectification process and therefore produces a higher quality output GeoRaster object.

Orthorectification with DEM requires that the source GeoRaster have a 3D SRS. The DEM must cover all the target output area, and it should be in the same SRID as the output. The resolution of the DEM should be similar to the expected resolution of the output GeoRaster object.

For orthorectification with DEM, the `elevationParam average` keyword is optional; and if it is not specified, the procedure estimates elevation values based on sample values extracted from the DEM on the target area.

The `elevationParam offset` and `scale` values can be used to modify the values from the DEM. For example, `scale` can be used for unit conversion if the DEM values are in a unit other than that of the source GeoRaster SRS, and `offset` can be used to perform geoidal correction or other offsetting. However, these specifications do not apply the changes to DEM values in the GeoRaster object. An alternative is to pre-process the DEM values by applying the scaling and offsetting to the DEM data before the orthorectification, as explained in Raster Data Scaling and Offsetting.

Example 6-6   Orthorectification with DEM

Example 6-6 example performs orthorectification with DEM. The DEM covers all the output area in a resolution approximated to the resolution of the output GeoRaster. The DEM values are in meters but the source image SRS is in feet. There is also a geoid correction on that area of about -15.3 meters:

```
DECLARE
  gr_src  sdo_georaster;
  gr_dem  sdo_georaster;
  gr_out  sdo_georaster;
BEGIN
  select georaster into gr_src from georaster_table where georid = 1;
  select georaster into gr_dem from georaster_table where georid = 5;
  delete from georaster_table where georid = 6;
  insert into georaster_table values(5, 'orthorectified with DEM',
          sdo_geor.init('rdt_4',6))
          returning georaster into gr_out;
  sdo_geor.rectify(inGeoRaster      => gr_src,
                   pyramidLevel     => null,
                   elevationParam   => 'average=1748.8 offset=-15.3',
                   dem              => gr_dem,
                   outSRID          => 32613,
                   outModelCoordLoc => null,
                   cropArea         => null,
                   polygonClip      => null,
                   layerNumbers     => null,
                   outResolutions   => null,
                   resolutionUnit   => null,
```

```
                    referencePoint   => null,
                    resampleParam    => 'resampling=BILINEAR',
                    storageParam     => null,
                    outGeoraster     => gr_out);
    update georaster_table set georaster = gr_out where georid = 6;
    commit;
END;
```

Example 6-7   Orthorectification with Cropped DEM

Typically, the DEM covers an area much larger than the target area, and the resolution is coarser than the target resolution of the output GeoRaster object. Using this DEM "as is" would result in poor quality orthorectification. The solution to that common problem is to crop the DEM to the target area and rescale it to the desired resolution, as shown in Example 6-7. This example uses the SDO_GEOR.rectify procedure to transform the low-resolution DEM GeoRaster object into a second DEM GeoRaster object that has the same resolution as the orthorectified GeoRaster object generated by the second call to the SDO_GEOR.rectify procedure.

```
DECLARE
  height   number := 1748.8;
  gr_src   sdo_georaster;
  gr_out   sdo_georaster;
  gr_dem   sdo_georaster;
  gr_dem2  sdo_georaster;
  gm_area  sdo_geometry;
begin
  select georaster into gr_src from georaster_table where georid = 1;
  select georaster into gr_dem from georaster_table where georid = 2;
  -- Calculate crop area
  gm_area := sdo_cs.make_2d(
              sdo_geor.generateSpatialExtent(gr_src,height),
              sdo_geor.getModelSRID(gr_dem));
  -- Rectify dem ( re-project, crop area, re-escale and resample )
  delete from georaster_table where georid = 4;
  insert into georaster_table values(4,
              'rectified DEM',
              sdo_geor.init('rdt_4',4))
              returning georaster into gr_dem2;
  sdo_geor.rectify(inGeoRaster     => gr_dem,
                  pyramidLevel     => null,
                  elevationParam   => null,
                  dem              => null,
                  outSRID          => 32613,
                  outModelCoordLoc => null,
                  cropArea         => gm_area,
                  polygonClip      => null,
                  layerNumbers     => null,
                  outResolutions   => sdo_number_array(0.6,0.6),
                  resolutionUnit   => null,
                  referencePoint   => null,
                  resampleParam    => 'resampling=CUBIC',
                  storageParam     => null,
                  outGeoraster     => gr_dem2);
  update georaster_table set georaster = gr_dem2 where georid = 4;
  commit;
  -- Orthorectification with DEM
  select georaster into gr_dem2 from georaster_table where georid = 4;
  delete from georaster_table where georid = 5;
  insert into georaster_table
        values(5, 'orthorectified', sdo_geor.init('rdt_4',5))
```

```
            returning georaster into gr_out;
    sdo_geor.rectify(inGeoRaster      => gr_src,
                     pyramidLevel     => null,
                     elevationParam   =>
                            'average=' || height || ' offset=-15.588',
                     dem              => gr_dem2,
                     outSRID          => 32613,
                     outModelCoordLoc => null,
                     cropArea         => gm_area,
                     polygonClip      => null,
                     layerNumbers     => null,
                     outResolutions   => sdo_number_array(0.6,0.6),
                     resolutionUnit   => null,
                     referencePoint   => null,
                     resampleParam    => 'resampling=average16',
                     storageParam     => null,
                     outGeoraster     => gr_out);
  update georaster_table set georaster = gr_out where georid = 5;
  commit;
end;
/
```

6.5 Image Warping

Image warping transforms an input GeoRaster object to an output GeoRaster object using the spatial reference information from a specified SDO_GEOR_SRS object.

The reference SDO_GEOR_SRS object can be copied from an existing GeoRaster object or created using a constructor. (For more information, see SDO_GEOR_SRS Object Type.)

Warping is performed by the SDO_GEOR.warp procedure, and requires that the source GeoRaster object have at least a functional fitting georeferencing model. This means that the image does not need to be rectified, but it needs to have georeference information in the metadata (see Georeferencing GeoRaster Objects).

Example 6-8   Image Warping

The following example uses the SDO_GEOR_SRS information from one GeoRaster image (gr1) as a reference to transform an existing GeoRaster object (gr2) into a new (warped) GeoRaster object (gr3). Thus, the third GeoRaster object is a "copy" (actually, a transformation) of the second GeoRaster object, but reflects the same georeferencing as the first GeoRaster object.

```
DECLARE
  srs sdo_geor_srs;
  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;

  srs := sdo_geor.getSRS(gr1); -- get the SRS from image 1.

  insert into georaster_table values(3, 'Warped Object',
        sdo_geor.init('imagery_rdt'))
        returning georaster  into gr3;

  sdo_geor.warp( inGeoRaster     => gr2,
```

```
                    pyramidLevel      => null,
                    outSRS            => srs, -- apply SRS to warp transformation
                    cropArea          => null,
                    dimensionSize     => null,
                    layerNumbers      => null,
                    elevationParam    => null,
                    resampleParam     => 'resampling=AVERAGE4',
                    storageParam      => 'pyramid=true',
                    outGeoRaster      => gr3,
                    bgValues          => sdo_number_array(0,0,0),
                    parallelParam     => 'parellel=4' );

  update georaster_table set georaster = gr3 where georid = 3;
  commit;
END;
```

6.6 Image Affine Transformation and Scaling

Affine transformation is the process of using geometric transformations of translation, scaling, rotation, shearing, and reflection on an image to produce another image.

For details and examples, see the SDO_GEOR.affineTransform reference topic.

Image scaling is the process of enlarging or shrinking an image by changing the pixel size for the row and column dimensions of an image. Image scaling resamples the pixel values from the original image to construct the rescaled version of that image. Image scaling can be performed in several ways:

- Use the SDO_GEOR.scaleCopy procedure and specify for scaleParam a scaleFactor to be applied to the input image dimensions or a maxDimSize for the output image.

- Use the SDO_GEOR.rectify procedure and specify the resolution of the output image. (This procedure can be executed in parallel.)

- During affine transformation, use the scales parameter of the SDO_GEOR.affineTransform procedure. In that procedure, the scales parameter is a two-number array where you can specify a scale factor for rows and for columns independently. (This procedure can be executed in parallel.)

Example 6-9   Image Scaling Using SDO_GEOR.scaleCopy

This example performs rescaling by using SDO_GEOR.scaleCopy and specifying the scaleFactor value as 2. The input image will have 2 times more rows and 2 times more columns than the original, and the values will be resampled by the average16 algorithm. Note that the image will be 4 times larger than the original.

```
DECLARE
  gr_src  sdo_georaster;
  gr_out  sdo_georaster;
BEGIN
  select georaster into gr_src from georaster_table where georid = 7;
  -- Rescale
  delete from georaster_table where georid = 9;
  insert into georaster_table values(9, 're-scaled by scaleCopy',
              sdo_geor.init('rdt_4',9))
              returning georaster into gr_out;
  sdo_geor.scaleCopy(inGeoRaster      => null,
                     scaleFactor      => 'scaleFactor=2',
                     resampleParam    => 'resampling=AVERAGE16',
```

```
                      storageParam     => null,
                      outGeoraster     => gr_out);
  update georaster_table set georaster = gr_out where georid = 9;
  commit;
END;
/
```

Example 6-10   Image Scaling Using SDO_GEOR.rectify

This example performs rescaling by using SDO_GEOR.rectify and specifying the
`outResolutions` parameter. The input image is already rectified, and the output will
have the same SRID as the input.

```
DECLARE
  gr_src     sdo_georaster;
  gr_out     sdo_georaster;
BEGIN
  select georaster into gr_src from georaster_table where georid = 7;
  -- Rescale
  delete from georaster_table where georid = 10;
  insert into georaster_table values(10, 're-scaled by rectify',
              sdo_geor.init('rdt_4',10))
              returning georaster into gr_out;
  sdo_geor.rectify(inGeoRaster      => null,
                   pyramidLevel     => null,
                   elevationParam   => null,
                   dem              => null,
                   outSRID          => null,
                   outModelCoordLoc => null,
                   cropArea         => null,
                   polygonClip      => null,
                   layerNumbers     => null,
                   outResolutions   => sdo_number_array(1.2,1.2),
                   resolutionUnit   => null,
                   referencePoint   => null,
                   resampleParam    => 'resampling=CUBIC',
                   storageParam     => null,
                   outGeoraster     => gr_out,
                   parallelParam    => 'parallel=4');
  update georaster_table set georaster = gr_out where georid = 10;
  commit;
END;
/
```

Example 6-11   Rescaling Using SDO_GEOR.affineTransform

This example performs rescaling by using the SDO_GEOR.affineTransform procedure
and specifying he `scales` parameter as `sdo_number_array(2, 2)`, indicating that the
image will be enlarged 2 times on the rows dimension and 2 times on the columns
dimension.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  select georaster into gr1 from georaster_table where georid = 1;

  insert into georaster_table values(2, 'Rotated 90 left',
         sdo_geor.init('rdt0',2)) returning georaster into gr2;

  sdo_geor.affineTransform(inGeoRaster    => gr1,
```

```
                            translation   => null,
                            scales        => sdo_number_array(2,2),
                            rotatePt      => null,
                            rotateAngle   => null,
                            shear         => null,
                            reflection    => null,
                            storageParam  => null,
                            outGeoraster  => gr2,
                            parallelParam => 'parallel=4');

    update georaster_table set georaster = gr2 where georid = 2;
    commit;
END;
```

6.7 Image Stretching, Normalization, Equalization, Histogram Matching, and Dodging

The color and contrast of images can be enhanced to improve their visual quality. The SDO_GEOR_IP package ("IP" for image processing) provides a set of subprograms for image enhancement, including performing image stretching, image normalization, image equalization, histogram matching, and image dodging.

Linear stretching and **piecewise stretching** can stretch the image cell values linearly for all cells values based on the minimum and maximum cell values or at specified value range, to achieve better color and contrast. To perform image stretching, you can use the following procedures:

- SDO_GEOR_IP.stretch stretches GeoRaster objects in any supported cell depth (1BIT to 64BIT_REAL) to cell depth of 8BIT_U for display purposes.

- SDO_GEOR_IP.piecewiseStretch stretches GeoRaster objects of any supported cell depth to the GeoRaster objects in higher or lower cell depth, not limited to 8BIT_U.

Image **normalization** linearly stretches the image based on the statistics (mean and standard deviation) of the image cell values. To perform image normalization, use the SDO_GEOR_IP.normalize.

Image **equalization** enhances image contrast by equalizaing its histogram. To perform equalization, use the SDO_GEOR_IP.equalize procedure.

Image **histogram matching** stretches the image to match the specified histogram or the histogram of a reference image. To perform image histogram matching, use the SDO_GEOR_IP.histogramMatch procedure.

Image **dodging** balances image color by stretching the contrast of the image locally instead of globally. To perform image dodging, use the SDO_GEOR_IP.dodge procedure.

6.8 Image Filtering

Image filtering is the process of applying a convolution filter on an image to achieve a specific purpose. For example, applying a low-pass filter on an image can smooth and reduce noise in an image, while applying a high-pass filter on an image can enhance the details of the image or even detect the edges inside the image.

The SDO_GEOR_IP.filter procedure provides standard filters such as low-pass filter (LPF), high-pass filter (HPF), and high-boost filter (HBF). It also allows you to apply customized filters on images.

The following example performs image filtering by providing a customized 3-by-3 Laplacian filter on the image for edge detection.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_geometry;
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_ip.filter(gr1, 0, cropArea, null, 'filtertype=CUSTOM, kernelsize=(3,3)',
sdo_number_array(0, 1, 0, 1, -4, 1, 0, 1, 0 ), null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

6.9 Image Segmentation

Segmentation is a simple type of classification algorithm, and can be useful in classifying certain types of images into larger ground feature categories, such as land, cloud, water, or snow.

You can use the SDO_GEOR_RA.classify procedure to apply thresholding to images. **Thresholding** is the simplest segmentation, which classifies an image into two categories by using a single cell value as the threshold. The resulting image has only two values and can be cast into a binary bitmap mask directly in the same process.

You can also apply more image segmentation operations using the SDO_GEOR_RA.classify procedure, by first applying arithmetic operations on multiple bands and then classifying the results into a number of categories.

For examples of using the SDO_GEOR_RA.classify procedure, see Classification Operations.

6.10 Image Pyramiding: Parallel Generation and Partial Update

Image pyramiding is one of the most commonly used processes in building large-scale image databases.

This topic discusses some related techniques: pyramid generation in parallel, partial updating of pyramids, and batch and concurrent processing.

For working more efficiently with pyramids, you can generate pyramids in parallel and perform a partial update of a pyramid. (This section assumes you understand the concepts explained in Pyramids.)

Example 6-12   Parallel Generation of Pyramids

For faster pyramid generation, you can specify the `parallelParam` parameter with the SDO_GEOR.generatePyramid procedure. In Example 6-12, the degree of parallelism is set to 4. (The actual performance improvement for pyramid generation depends on the number of CPUs available to Oracle Database.)

```
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', null, 'parallel=4');

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

  COMMIT;
END;
/
```

To enable parallel processing of the pyramid generation, SDO_GEOR.generatePyramid performs an implicit commit operation. If an error during the call, the GeoRaster object may in an invalid state. If this occurs, use SDO_GEOR.deletePyramid to remove the newly generated and upper pyramid levels of the GeoRaster object.

Example 6-13   Partial Updating of Pyramids

You can partially update pyramids by using the SDO_GEOR.updateRaster procedure. In Example 6-13, the target GeoRaster object at a specified area (`targetArea` is specified as `area`) is updated by another GeoRaster object. The `updateUpperPyramids` parameter is set to `true`, so the upper pyramids of the target GeoRaster object are only partially updated at the specified area. In other words, the upper pyramid levels are not regenerated in full, but only the cells in that targetArea are regenerated, and thus performance is improved.

```
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, null, area, gr1, 0, null, 'true');
  UPDATE GEORASTER_TABLE SET georaster=gr2 WHERE georid=0;
  COMMIT;
END;
/
```

Other techniques to speed up and automate the pyramiding process include batch processing and concurrent processing can be used. To batch pyramid many images in a certain area, see the example in Querying and Searching GeoRaster Objects. To process many batches concurrently, you can start different database sessions

6.11 Bitmap Pyramiding

Bitmap pyramiding can produce high-quality pyramids in certain cases where traditional pyramiding is not adequate.

For most raster data types, image pyramiding as described in Image Pyramiding: Parallel Generation and Partial Update results in pyramids of great quality. However, for bitmap rasters of points, lines, or polylines, which are typically stored in 1-bit cell depth, the same pyramiding approach may not create high-quality pyramids. Distorted point patterns and dashed lines are commonly seen in those pyramids.

To solve such problems, you can use the SDO_GEOR.generateBitmapPyramid procedure, instead of SDOGEOR.generatePyramid, to perform pyramiding on bitmap GeoRaster objects. The SDO_GEOR.generateBitmapPyramid procedure significantly improves the pyramid quality by avoiding distorted patterns, particularly dashed lines or missing lines in a bitmap raster, such as a road raster map or utility network raster map.

6.12 Vegetation Index Computation

In remote sensing, the Normalized Difference Vegetation Index (NDVI) is a widely used vegetation index, enabling users to quickly identify vegetated areas and monitor the growth and "condition" of plants.

Using Landsat TM imagery, the standard NDVI computation formula is: (TM4 - TM3) / (TM4 + TM3).

Example 6-14   Vegetation Index Computation

Example 6-14takes a Landsat 7 ETM+ image and computes the NDVI with parallelism. The result is stored as another raster of floating number data type. Note that in the GeoRaster algebra language, band numbering starts with 0, so the formula translates into the expression: ({3}-{2})/({3}+{2}).

```
DECLARE
  geor1    SDO_GEORASTER;
  geor2    SDO_GEORASTER;
EBGIN
  -- Source ETM+ image
  select georaster into geor1 from georaster_table where georid = 2;
  -- Store NDVI
  select georaster into geor2 from georaster_table where georid = 3 for update;
  sdo_geor_ra.rasterMathOp(geor1,
      SDO_STRING_ARRAY('({3}-{2})/({3}+{2})'),
      'celldepth=32bit_real',geor2, null, null, 'parallel=4');
  update georaster_table set georaster = geor2 where georid = 3;
  commit;
end;
/
```

In addition to NDVI, there are many other vegetation indexes in the area of remote sensing. Many of these can be similarly computed using the GeoRaster raster algebra.

6.13 Tasseled Cap Transformation

Tasseled Cap Transformation (TCT) is a useful tool for analyzing physical ground features using remotely sensed imagery.

With various Landsat imagery, it uses 5 bands of either original digital number (DN) or reflectance data to generate 6 new bands, each of which represents different ground features. The 6 resulting bands are generally called (soil) brightness, (vegetation) greenness, (soil and canopy) wetness, haze, TC5, and TC6. Each one or a combination of them is useful for different applications, such as crop growth monitoring and analysis, biomass study, and agriculture planning.

Example 6-15   Tasseled Cap Transformation

Example 6-15 takes the DN data of a Landsat 5 TM image as input, executes the TCT using the GeoRaster raster algebra with parallelism, and creates a new image holding the results.

```
DECLARE
    gr1 sdo_georaster;
    gr2 sdo_georaster;
    ret varchar2(32);
BEGIN
    select georaster into gr1 from georaster_table where georid = 2;
    select georaster into gr2 from georaster_table where georid = 4 for update;
    sdo_geor_ra.rasterMathOp(
      gr1,
      SDO_STRING2_ARRAY(
        '0.3561*{0}+0.3972*{1}+0.3904*{2}+0.6966*{3}+0.2286*{4}+0.1596*{6}',
        '(-0.3344)*{0}-0.3544*{1}-0.4556*{2}+0.6966*{3}-0.0242*{4}-0.2630*{6}',
        '0.2626*{0}+0.2141*{1}+0.0926*{2}+0.0656*{3}-0.7629*{4}-0.5388*{6}',
        '0.0805*{0}-0.0498*{1}+0.1950*{2}-0.1327*{3}+0.5752*{4}-0.7775*{6}',
        '(-0.7252)*{0}-0.0202*{1}+0.6683*{2}+0.0631*{3}-0.1494*{4}-0.0274*{6}',
        '0.4000*{0}-0.8172*{1}+0.3832*{2}+0.0602*{3}-0.1095*{4}+0.0985*{6}' ),
      'celldepth=32BIT_REAL',
      gr2, null, null, 'parallel=4');
    update georaster_table set georaster = gr2 where georid = 4;
    commit;
END;
/
```

You can also use the same raster algebra language to add code in Example 6-15 to convert the 32-bit floating number image into an 8-bit integer image and to apply image stretching (described in Image Stretching) on the resulting TCT image to generate a new GeoRaster object for visualization and analysis.

In addition to using the optimized implementation of raster algebra algorithms and the embedded parallel processing, you can further take advantage of the Oracle grid computing infrastructure to quickly compute NDVI or apply TCT on thousands of images stored in the GeoRaster database.

6.14 Image Masking

To perform image masking, an application can query the GeoRaster database for bitmap masks, retrieve the desired bitmap mask or masks, and apply the masking

operation on the target GeoRaster object for the purpose of displaying the object or performing some other processing.

A bitmap mask (described in Bitmap Masks) can be stored as an independent GeoRaster object; it can also be stored as metadata inside a GeoRaster object and be associated with a single band or with the whole GeoRaster object.

You can also perform masking operations inside the database to generate new GeoRaster objects, using the SDO_GEOR.mask procedure.

6.15 Band Merging

For image classification, time series analysis, and raster GIS modeling, multiple bands or layers of different GeoRaster objects may need to be merged into a single GeoRaster object.

This operation is called band or layer merging in GeoRaster, and can be performed by using the SDO_GEOR.mergeLayers procedure or the SDO_GEOR_RA.rasterMathOp procedure. You can either append specified bands of a source GeoRaster object to a target GeoRaster object or merge different bands from two GeoRaster objects into a new GeoRaster object. By doing this merging or appending iteratively, you can merge an unlimited number of bands into a single GeoRaster object.

Example 6-16   Band Merging

Example 6-16 includes two examples. The first example assumes there are eight GeoRaster objects, each of which contains *only one* band loaded from a single-band Landsat ETM+ image file in GeoTIFF format. The number of the band in each GeoRaster object is the same as the GEORID column value for the GeoRaster object. The example merges all bands into a single GeoRaster object to create a complete ETM+ scene.

```
DECLARE
    gr1 sdo_georaster;
BEGIN
    select georaster into gr1 from georaster_table where georid = 1 for update;
    for rec in (select georaster from georaster_table
                    where georid >= 2 and georid <= 8
                    order by georid)
    loop
        sdo_geor.mergelayers(gr1, rec.georaster);
    end loop;
    update georaster_table set georaster = gr1 where georid = 1;
    commit;
END;
/
```

The second example assumes there are eight GeoRaster objects, each of which contains *three* bands. The example picks up one band from each GeoRaster object and merges them into a single 8-band GeoRaster object in parallel.

```
DECLARE
  geor       SDO_GEORASTER;
  geo_array  SDO_GEORASTER_ARRAY;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 0 for update;
  geo_array:=SDO_GEORASTER_ARRAY();
  for rec in (select georaster from georaster_table
                    where georid >= 1 and georid <= 8
```

```
                              order by georid)
      loop
         geo_array.extend(1);
         geo_array(geo_array.last):=rec.georaster;
      end loop;

   sdo_geor_ra.rasterMathOp(geo_array,SDO_STRING2_ARRAY('{0,0}','{1,1}','{2,2}','{3,0}',
'{4,1}','{5,2}','{6,0}','{7,1}',),null,geor,'false',null,'parallel=4');
     UPDATE georaster_table SET georaster = geor WHERE georid = 0;
     COMMIT;
   END;
   /
```

6.16 Image Appending

You can append one image to another image when the two images have the same
number of bands.

Image appending is useful when the geospatial images are collected at intervals and
the captured image later needs to be appended to the existing image to make a large
image of the whole spatial area. Image appending is also useful for updating the
existing image with a new image.

The SDO_GEOR_AGGR.append procedure implements image appending by partially
updating the existing GeoRaster object with another GeoRaster object. If the existing
GeoRaster object contains pyramids, the pyramids with blocking are partially updated
with the new data.

Example 6-17 appends one image to another, with pyramids with blocking are updated
at the same time. Because the `appendParam` parameter specifies `'nodata=true'`, the
NODATA values in the overlapping area are considered transparent.

Example 6-17   Appending One Image to Another Image

```
DECLARE
    gr1 sdo_georaster;
    gr1 sdo_georaster;
BEGIN
    select georaster into gr1 from georaster_table where georid = 1 for update;
    select georaster into gr2 from georaster_table where georid = 2;
    sdo_geor_aggr.append(gr1, gr2, 0, 'nodata=true');
    update georaster_table set georaster = gr1 where georid= 1;
    commit;
END;
/
```

6.17 Large-Scale Image Mosaicking

A large geospatial area typically consists of many smaller aerial photographs or
satellite images. Large-scale image mosaicking can stitch these small geospatial
images into one large image to get a better view of the whole spatial area.

GeoRaster provides large-scale mosaicking functions that allow gaps, overlaps, and
missing source GeoRaster objects. It supports both rectified and unrectified images. It
supports internal reprojection and rectification, common point rules, and simple color
balancing. You can also mosaic at a certain pyramid level. This mosaicking process
results in a single GeoRaster object, which is also called a **physical mosaic** as
opposed to *virtual mosaic* (For information about virtual mosaic, see Virtual Mosaic).

The SDO_GEOR.mosaic and SDO_GEOR_AGGR.mosaicSubset procedures provide support for image mosaicking; however, you are strongly encouraged to use SDO_GEOR_AGGR.mosaicSubset because it provides much more advanced features and options, and it is also implemented with parallelism. SDO_GEOR_AGGR.mosaicSubset can take a virtual mosaic, such as a list of GeoRaster tables, a database view with a GeoRaster column, or a REF CURSOR, as the source images.

The SDO_GEOR.mosaic procedure mosaics a set of source GeoRaster images that are rectified, are geospatially aligned under the same SRID, and have the same resolution. The result of the mosaic is another GeoRaster object. If there are overlaps between the source images, the mosaic result will have the last source image's content at the overlapping area. This procedure works well for preprocessed and perfectly aligned source images.

In the examples in this section, the source images are stored in source GeoRaster tables GRTAB, GRTAB1, and GRTAB2, which are defined with the following columns:

```
(id           NUMBER PRIMARY KEY,
cloud_cover   NUMBER      -- percentage of cloud coverage
last_update   TIMESTAMP  -- GeoRaster object's last update time
grobj         SDO_GEORASTER )
```

Oracle Spatial and Graph spatial indexes have been created on the `spatialExtent` attribute of the GeoRaster object in these tables.

In these examples, the mosaicked image is stored in GEORASTER_TABLE, which is defined in Storage Parameters.

Example 6-18   SDO_GEOR.mosaic (Table and Column Name)

Example 6-18 shows the SDO_GEOR.mosaic procedure.

```
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, 'blocking=optimalpadding
blocksize=(512,512,1)');
  UPDATE georaster_table SET georaster=gr WHERE id=12;
END;
/
```

In the real world, however, the source images are often collected under different circumstances so as to have different resolutions or large areas of overlap, or using a different georeference system. In such cases, you can use the SDO_GEOR_AGGR.mosaicSubset procedure to mosaic these source images into one uniform mosaicked image. Compared to SDO_GEOR.mosaic, the SDO_GEOR_AGGR.mosaicSubset procedure provides more features and options:

*   The source images do not have to be in the same coordinate system (SRID) and have the same georeferencing information or resolutions.

*   The source images can be mosaicked on a user-specified pyramid level.

*   The source images can be mosaicked on user-specified bands.

*   The output images can have a different coordinate system and resolution than the input images (`outSRID` and `outResolutions` parameters).

- You have more control on the output of the overlapping area through the `mosaicParam` parameter: `commonPointRule` can specify which cell value to use for the output at the overlapping area, and `NODATA` can indicate whether to consider the NODATA value at the overlapping area.

- The output mosaicked image can be aligned at a specified point (the reference point). The source image can be resampled in order to align with the reference point if the source image is out of alignment more than the `resampleTolerance` value specified in `mosaicParam`.

- If there is small gap between the source images that is less than 2 pixels wide, it can be filled using the neighboring pixel values when `fillGap` is `true` in `mosaicParam`.

- Limited color balancing (linear stretching and normalization) is supported.

- Parallel processing is supported to speed up the mosaicking process.

Example 6-19   SDO_GEOR_AGGR.mosaicSubset

Example 6-19 uses SDO_GEOR_AGGR.mosaicSubset to mosaic all the source images from two GeoRaster tables (GRTAB1 and GRTAB2) into a large mosaicked image in SRID 4326 with a resolution of 30 meters on the x and y dimensions.

```
DECLARE
  resolutions sdo_number_array;
  gr sdo_georaster;
BEGIN
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
         returning georaster into gr;

    resolutions := sdo_number_array(30, 30);
    sdo_geor_aggr.mosaicSubset('grtab1, grtab2', 'grobj, grobj',
                               0, 4326, null, null, null,
                               null, null, null, resolutions, 'unit=meter',
                               'commonPointRule = end, nodata=true,
resampleTolerance=0.2, resampling=bilinear, fillGap=true',
                               'blocking=optimalpadding blocksize=(512, 512, 3)',
gr, null, 'parallel=4');

      update georaster_table set georaster = gr where georid=10;
      commit;
END;
/
```

In Example 6-19:

- Any source image that is not rectified is rectified; any source image that is not in SRID 4326 is reprojected to SRID 4326.

- Any source image that has a resolution other than 30 meters is scaled to a resolution of 30 meters.

- The `nodata` keyword in the `mosaicParam` parameter is specified as `true`, which means the NODATA values in the overlapping area are not considered.

- The `resampleTolerance` keyword in the `mosaicParam` parameter is specified as 0.2, which means that if the source image is offset from the target by more than 0.2 pixel, the source image is resampled.

- The resampling method is specified as `bilinear` in the `mosaicParam` parameter.

- The degree of parallelism is specified as 4 in the `parallelParam` parameter.

You can call SDO_GEOR_AGGR.validateForMosaicSubset before calling SDO_GEOR_AGGR.getMosaicSubset to make sure that the source images can be mosaicked.

- Color Balancing During Mosaicking
- Parallel Compression, Copying, and Subsetting

6.17.1 Color Balancing During Mosaicking

The source images of the mosaicking operation can have different luminance or colors due to the differences in the lighting conditions, time of day, or other factors when the images were captured. Color balancing minimizes the color differences between the neighboring images and makes the resulting mosaic look more seamless.

SDO_GEOR_AGGR.mosaicSubset and SDO_GEOR_AGGR.getMosaicSubset provide some basic color balancing methods during the mosaicking process. Several color balancing methods are provided. They are identified by the keyword `colorbalance` in the `mosaicParam` parameter:

- `LINEARSTRETCHING`: Perform the min-max stretch on each band of the source images to a reference minimum and maximum range.

- `STATISTICSMATCHING`: Perform the image stretching so that the mean and standard deviation of each band of the source images is stretched and matched to the reference mean and standard deviation values.

- `HISTOGRAMMATCHING`: Perform the image stretching so that the histograms of the resulting images match the reference histograms.

There are several ways to specify the reference values for the color balancing methods. They are identified by the keyword `cbreference` in the `mosaicParam` parameter:

- VALUE: The reference values are provided through `referenceValue1`, `referenceValue2`, or `refHistograms` parameters directly.

- IMAGE: The reference values are derived from the image specified by the `referenceImage` parameter. The reference image must have the same number of bands as the source image.

- OVERLAP: The reference values are determined by the neighboring image through the overlapped area with the neighboring image. This option requires the source images have large enough overlaps so that the reference values can be derived from the overlapped area. Note that because linear stretching method does not provide a good result for this option, overlapped area reference is not supported for the linear stretching method.

Example 6-20   LINEARSTRETCHING Color Balancing

This example shows how to use the `LINEARSTRETCHING` color balancing method on the source images in the SDO_GEOR_AGGR.mosaicSubset procedure. The reference minimum and maximum values are specified in the `referenceValue1` and `referenceValue2` parameters. In the example, each band has different reference value.

```
DECLARE
gr sdo_georaster;
resolutions  sdo_number_array;
ref_min     sdo_number_arrray;
ref_max    sdo_number_array;
```

```
BEGIN
    -- create the new GeoRaster object for mosaic
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
          returning georaster into gr;

    -- set the output resolution
     resolutions := sdo_number_array(30, 30);



    -- Set the reference values, there are 3 values, one for each band
    ref_min := sdo_number_array(10, 10, 10);
    ref_max := sdo_number_array(200, 255, 230);

    -- Mosaic
    sdo_geor_aggr.MosaicSubset('georaster_table_1',
                    'georaster', null, 32610, null, null, null, null,
                     null, null, null, resolutions, null,
                    'colorBalance=linearstretching, cbreference=value',
                    'blocking=optimalpadding, blocksize=(512,512,3)',
                     gr, null, 'parallel=4', referenceValue1=>ref_min,
                    referenceValue2=>ref_max);

     update georaster_table set georaster = gr where georid=10;
     commit;
END;
/
```

Example 6-21   HISTOGRAMMATCHING Color Balancing

This example shows how to use the HISTOGRAMMATCHING color balancing method on the source images in the SDO_GEOR_AGGR.mosaicSubset procedure. The reference histograms are derived from the reference image. The reference image must have the same number of bands as the source images.

```
DECLARE
gr sdo_georaster;
resolutions   sdo_number_array;
ref_gr     sdo_georaster;
BEGIN
    -- create the new GeoRaster object for mosaic
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
          returning georaster into gr;

    -- set the output resolution
    resolutions := sdo_number_array(30, 30);

-- retrieve the reference image
Select georaster into ref_gr from georaster_table where georid = 1;

    -- Mosaic
    sdo_geor_aggr.MosaicSubset('georaster_table_1',
                    'georaster', null, 32610, null, null, null, null,
                     null, null, null, resolutions, null,
                    'colorBalance=histogramMatching, cbreference=image',
                    'blocking=optimalpadding, blocksize=(512,512,3)',
                     gr, null, 'parallel=4', refereneImage=>ref_gr);

     update georaster_table set georaster = gr where georid=10;
     commit;
```

```
END;
/
```

Example 6-22   STATISTICSMATCHING Color Balancing

This example shows how to use the STATISTICSMATCHING color balancing method on the source images in the SDO_GEOR_AGGR.mosaicSubsetprocedure. The reference statistics values are calculated from the overlapped area of the neighboring images. This requires that the source images have significant overlaps so that the statistics of the overlapped area can reflect the color difference between neighboring images.

```
DECLARE
gr sdo_georaster;
resolutions  sdo_number_array;
BEGIN
    -- create the new GeoRaster object for mosaic
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
         returning georaster into gr;

    -- set the output resolution
    resolutions := sdo_number_array(30, 30);

    -- Mosaic
    sdo_geor_aggr.MosaicSubset('georaster_table_1',
                    'georaster', null, 32610, null, null, null, null,
                     null, null, null, resolutions, null,
                    'colorBalance=statisticsMatching, cbreference=overlap',
                    'blocking=optimalpadding, blocksize=(512,512,3)',
                     gr, null, 'parallel=4');

    update georaster_table set georaster = gr where georid=10;
    commit;
END;
/
```

6.17.2 Parallel Compression, Copying, and Subsetting

To parallelize rectification, orthorectification and reprojecting, use SDO_GEOR.rectify. To parallelize warping, call SDO_GEOR.warp. All raster algebra operations are parallelized too.

You can use the SDO_GEOR_AGGR.mosaicSubset procedure to conduct several types of parallel operations, including parallel compression and decompression, parallel copying or change format copying, parallel subsetting, parallel reprojection, and parallel rectification. The copying and subsetting operations are not parallelized directly. For JPEG and DEFLATE, the SDO_GEOR.changeFormatCopy procedure can be called to do parallel compression and decompression if reformatting is not required. This topic gives some examples for parallelized compressing, copying, and subsetting operations. In all these cases, the SDO_GEOR_AGGR.mosaicSubset procedure works on single GeoRaster objects.

To illustrate the parallelized operations, the examples in this section use a null value for most parameters. In your applications, you can apply all other parameters of the SDO_GEOR_AGGR.mosaicSubset procedure; however, the mosaicParam parameter has no effect when the input is a single GeoRaster object.

Example 6-23   Parallel Compression

Example 6-23 shows parallel compression using the
SDO_GEOR_AGGR.mosaicSubset procedure. This applies to both DEFLATE and
JPEG compression and decompression.

```
DECLARE
  gr sdo_georaster;
  cur sys_refcursor;
  crop_area sdo_geometry := null;
BEGIN
  -- create a new georaster object with georid = 2
  -- to hold the compressed image
  delete from georaster_table where georid = 2;
  insert into georaster_table(georid, georaster) values (2,
      sdo_geor.init('RDT2', 2)) returning georaster into gr;


  -- reblock and compress the image with georid = 1 into JPEG using parallel degree
of 8
  open cur for 'select georaster from georaster_table where georid = 1';
  sdo_geor_aggr.mosaicSubset(cur, 0, null, null, null, crop_area,
                             null, null, null, null, null, null,
                             'compression=JPEG-F, blocking=optimalpadding,
blocksize=(512,512,3)',
                             gr, null, 'parallel=8');

  update georaster_table set georaster = gr where georid = 2;
  commit;
END;
/
```

In the preceding example, if you adjust the storageParam parameter, it works as a
parallelized SDO_GEOR.changeFormatCopy operation, including compression and
decompression.

Example 6-24   Parallel Subsetting and Copying

Example 6-24 shows parallel subsetting and copying using
theSDO_GEOR_AGGR.mosaicSubset procedure.

```
DECLARE
  gr sdo_georaster;
  cur sys_refcursor;
  crop_area sdo_geometry := null;
BEGIN
  -- create a new georaster object with georid = 2 to hold the copy
  delete from georaster_table where georid = 2;
  insert into georaster_table(georid, georaster) values (2,
      sdo_geor.init('RDT2', 2)) returning georaster into gr;


  -- set the crop_area for subsetting.
  crop_area := 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));

    -- subset from the image with georid = 1 using parallel degree of 8
    -- and do polygon clipping
    -- If the crop_area is set to null, the same call will do a simple parallelized
copying without subsetting.
    open cur for 'select georaster from georaster_table where georid = 1';
    sdo_geor_aggr.mosaicSubset(cur, 0, null, null, null, crop_area,
                               'true', null, null, null, null, null,
                               'pyramid=true', gr, null, 'parallel=8');
    update georaster_table set georaster = gr where georid = 2;
    commit;
END;
/
```

In Example 6-24, if you adjust the `storageParam` parameter, it works as a parallelized copy or SDO_GEOR.changeFormatCopy operation, including compression and decompression.

6.18 Virtual Mosaic

A virtual mosaic treats a set of GeoRaster images as one large virtually mosaicked image.

For some applications, mosaicking a collection of images into a single physical mosaic is not necessary or desirable. For example, you might not have enough disk space for storing the mosaic separately or you simply want to save disk space. Another example is if you do not want to keep two identical copies of the same data set but prefer to have the original data set stored as is, such as a DEM data set, yet you want to query over this data set seamlessly. Yet another example is if you want to apply different processing and mosaicking rules for the same region when mosaicking the source images -- a physical mosaic has no such flexibility.

In such cases, instead of mosaicking a set of GeoRaster images into one large GeoRaster image and storing it in a GeoRaster table, you can create a virtual mosaic. A virtual mosaic treats a set of GeoRaster images as one large virtually mosaicked image, without storing it in a GeoRaster table.

In GeoRaster, a **virtual mosaic** is defined as any large collection of georeferenced GeoRaster objects, rectified or unrectified, from one or more GeoRaster tables or views that is treated as if it is a single GeoRaster object. Pyramids of virtual mosaic are supported. A virtual mosaic can contain unlimited number of images, and a whole GeoRaster database can be treated as a virtual mosaic. You issue a single call to query the virtual mosaic based on area-of-interest (that is, subsetting or cropping), and you can request the cropped images to be in different coordinate system with different resolutions. On-the-fly transformations with resampling and mosaicking with common point rules, based on user requests, are done internally and automatically during the query processes.

The following are ways to define a virtual mosaic:

- As a GeoRaster table or a list of GeoRaster tables (see Virtual Mosaic as One or a List of GeoRaster Tables)

- As a database view with a GeoRaster column (see Virtual Mosaic as a View with a GeoRaster Column)

- As a SQL query statement (a cursor) that results in a collection of GeoRaster objects (see Virtual Mosaic as a SQL Query Statement or a Cursor)

Regardless of how the virtual mosaic is defined, the GeoRaster objects in the GeoRaster tables must have the spatialExtent attribute generated or set; otherwise, the SDO_GEOR_AGGR.getMosaicSubset and SDO_GEOR_AGGR.mosaicSubset procedures return an empty lob locator or empty GeoRaster object. For general use cases and best query performance, you should always create a spatial index beforehand on the spatialExtent attribute.

After a virtual mosaic is defined, you can use these procedures to query or process it:

- SDO_GEOR_AGGR.getMosaicSubset to perform on-the-fly queries over the virtual mosaic

  In spatial query of any portion of that virtually mosaicked image, the SDO_GEOR_AGGR.getMosaicSubset procedure performs the mosaic operation dynamically for the queried area and returns the required result in a BLOB on-the-fly, as if it were subsetting a physically stored mosaicked image.

- SDO_GEOR_AGGR.mosaicSubset to store the mosaicked subset in the database as a GeoRaster object

  The SDO_GEOR_AGGR.mosaicSubset procedure performs the mosaic operation for the queried area and stores the required result in another GeoRaster object persistently

For a typical workflow of using virtual mosaic, see Using Virtual Mosaic in Applications, and Special Considerations for Large-Scale Virtual Mosaic and its related topic Improving Query Performance Using MIN_X_RES$ and MAX_X_RES$.

- Virtual Mosaic as One or a List of GeoRaster Tables
- Virtual Mosaic as a View with a GeoRaster Column
- Virtual Mosaic as a SQL Query Statement or a Cursor
- Using Virtual Mosaic in Applications
- Special Considerations for Large-Scale Virtual Mosaic

6.18.1 Virtual Mosaic as One or a List of GeoRaster Tables

A virtual mosaic can be defined as one GeoRaster table or a list of GeoRaster tables. Applications specify each table and its GeoRaster column. In this approach, all GeoRaster objects in the specified GeoRaster columns of those GeoRaster tables are part of the virtual mosaic.

Example 6-25 specifies the source images for virtual mosaicking in a list of GeoRaster tables (GRTAB1, GRTAB2, and GRTAB3, which have the same definitions as GRTAB in Large-Scale Image Mosaicking).

Example 6-25   Virtual Mosaic as a List of GeoRaster Tables

```
DECLARE
  lb blob;
  cropArea sdo_geometry;
  outArea  sdo_geometry := null;
  outWin   sdo_number_array:=null;
  resolutions sdo_number_array;
BEGIN
    dbms_lob.createTemporary(lb, TRUE);
```

```
            cropArea :=  sdo_geometry(2003, 32610, null,
                          sdo_elem_info_array(1, 1003, 3),
                          sdo_ordinate_array(399180, 4247820,
                                              496140,4353900) );
         resolutions := sdo_number_array(30, 30);
        sdo_geor_aggr.getMosaicSubset('grtab1, grtab2, grtab3',
                    'grobj, grobj, grobj',
                    0, 32610, null, null, cropArea,
                    null, null, null, resolutions, null,
                    'commonPointRule = end, nodata=true',
                    lb, outArea, outWin);
    dbms_lob.freeTemporary(lb);
    if outWin is not null then
        dbms_output.put_line('output window: (' || outWin(1) || ',' || outWin(2)
||', ' || outWin(3) || ', ' || outWin(4) || ')');
    end if;
END;
/
```

6.18.2 Virtual Mosaic as a View with a GeoRaster Column

A virtual mosaic can be defined as one database view with a GeoRaster column. Applications specify the view name and its GeoRaster column. In this approach, all GeoRaster objects in the specified GeoRaster column of the view are part of the virtual mosaic. This approach allows you to select the images for the virtual mosaic in complex ways from any number of GeoRaster tables, taking advantage of the spatial index and any other relevant indexes.

You can also define a virtual mosaic as a list combining GeoRaster views and GeoRaster tables.

When a virtual mosaic is defined as a database view, the view can be specified in the georasterTableNames parameter when you query it. Example 6-26 queries the virtual mosaic defined as a view. Note that in this example, the queries sort the images based on their creation time and pick the latest (newest) image for the resulting mosaic in the overlapping areas.

Example 6-26   Using a View on GeoRaster Tables for Virtual Mosaic

```
Create or replace view grview as select * from (
        Select grobj, last_update from grtab1 where cloud_cover=0 union all
        Select grobj, last_update from grtab2 where cloud_cover=0 union all
        Select grobj, last_update from grtab3 ) order by last_update;

DECLARE
  lb blob;
  cropArea sdo_geometry;
  outArea  sdo_geometry := null;
  outWin   sdo_number_array:=null;
  resolutions sdo_number_array;
BEGIN
    dbms_lob.createTemporary(lb, TRUE);

    cropArea :=  sdo_geometry(2003, 32610, null,
                    sdo_elem_info_array(1, 1003, 3),
                    sdo_ordinate_array(399180, 4247820,
                                        496140,4353900) );
     resolutions := sdo_number_array(30, 30);
    sdo_geor_aggr.getMosaicSubset('grview', 'grobj',
```

```
                    0, 32610, null, null, cropArea,
                    null, null, null, resolutions, null,
                    'commonPointRule = end, nodata=true',
                    lb, outArea, outWin);
    dbms_lob.freeTemporary(lb);
    if outWin is not null then
        dbms_output.put_line('output window: (' || outWin(1) || ',' || outWin(2)
||', ' || outWin(3) || ', ' || outWin(4) || ')');
    end if;
END;
/
```

6.18.3 Virtual Mosaic as a SQL Query Statement or a Cursor

Instead of creating a view, you can define a virtual mosaic as a SQL statement or a runtime database cursor, which selects a collection of GeoRaster objects from the database. Applications create the cursor from the SQL statement and use the cursor as the virtual mosaic. In this definition, all GeoRaster objects in the cursor are part of the virtual mosaic. This approach allows you to select the images for the virtual mosaic in complex ways from any number of GeoRaster tables. However, the spatial indexes are not automatically used in queries over this type of virtual mosaic. To take advantage of spatial indexes, dynamically add a spatial query condition directly using the query window to the SQL statement, so that all images in that query window can be more quickly located.

The SDO_GEOR_AGGR.getMosaicSubset and SDO_GEOR_AGGR.mosaicSubsetprocedures accept a cursor of GeoRaster objects as the virtual mosaic, as shown in Example 6-27. Note that in this example, the queries sort the images based on their creation time and pick the latest (newest) image for the resulting mosaic in the overlapping areas. For best performance when there are many GeoRaster objects in the table, the query of the cursor should use the spatial query window to filter out the unrelated GeoRaster objects, as described in the preceding paragraph.

Example 6-27   Using a Cursor for Virtual Mosaic

```
DECLARE
  lb blob;
  outArea  sdo_geometry := null;
  outWin   sdo_number_array:=null;
  resolutions sdo_number_array;
  mosaic_stmt  varchar2(1000);
  condition        varchar2(1000);
BEGIN
    dbms_lob.createTemporary(lb, TRUE);

    resolutions := sdo_number_array(30, 30);

  -- Define the query window (cropArea)
   cropArea := sdo_geometry(2003, 32610, null,
                         sdo_elem_info_array(1, 1003, 3),
                         sdo_ordinate_array(399180, 4247820, 496140,4353900) );

    -- Define the virtual mosaic
    mosaic_stmt := 'select grobj from (select grobj, last_update from grtab1 ' ||
                   'where cloud_cover=0  union all select grobj, last_update from
grtab2 ' ||
                   'where cloud_cover=0) t ';
```

```
    -- Apply filtering using the query window (cropArea) to speed up query performance
     condition := 'where sdo_anyinteract(t.grobj.spatialExtent,:1) = ''true'' ' ||
                  ' order by last_update';

    -- Open the virtual mosaic for query
     open cur for mosaic_stmt || condition using cropArea;

    -- Query the virtual mosaic (make sure the cropArea used here is the same
    -- as the one used at opening the cursor)
     sdo_geor_aggr.getMosaicSubset(cur,
                  0, 32610, null, null, cropArea,
                  null, null, null, resolutions, null,
                  'commonPointRule=end, nodata=true',
                  lb, outArea, outWin);
    dbms_lob.freeTemporary(lb);
    close cur;
    if outWin is not null then
        dbms_output.put_line('output window: (' || outWin(1) || ',' || outWin(2)
||', ' || outWin(3) || ', ' || outWin(4) || ')');
    end if;
END;
/
```

6.18.4 Using Virtual Mosaic in Applications

Virtual mosaic can be used as an image serving engine and in a variety of other
application scenarios. The definitions of virtual mosaics can be stored by applications
separately as strings or other forms. Besides the major query procedures
SDO_GEOR_AGGR.getMosaicSubset and SDO_GEOR_AGGR.mosaicSubset,
GeoRaster provides other subprograms in the SDO_GEOR_AGGR package to
facilitate application development:

- SDO_GEOR_AGGR.validateForMosaicSubset

- SDO_GEOR_AGGR.getMosaicExtent

- SDO_GEOR_AGGR.getMosaicResolutions

SDO_GEOR_AGGR.validateForMosaicSubset, SDO_GEOR_AGGR.getMosaicExtent,
and SDO_GEOR_AGGR.getMosaicResolutions can be called in an application to
make sure that the virtual mosaic is valid and that the spatial query falls inside the
virtual mosaic. The following steps describe a possible workflow for virtual mosaic in
an application:

1. Define a virtual mosaic. For example:

   ```
   Create or replace view grview as select * from (
   Select grobj, last_update from grtab1 where cloud_cover=0 union all
   Select grobj, last_update from grtab2 where cloud_cover=0 union all
   Select grobj, last_update from grtab3 ) order by last_update;
   ```

   Note that tables GRTAB1, GRTAB2, and GRTAB3 were created using the same
   definition as GRTAB in Large-Scale Image Mosaicking, and Oracle Spatial and
   Graph spatial indexes have been created on the spatialExtent attribute of the
   GeoRaster object in these tables.

2. Validate the virtual mosaic data set. For example:

   ```
   EXECUTE SDO_GEOR_AGGR.validateForMosaicSubset('grview', 'grobj', OUTSRID,
   OUTResolutions);
   ```

A validation error table can be created and passed to the call if more detailed validation information is needed. See the SDO_GEOR_AGGR.validateForMosaicSubset reference section for details.

3. Get the spatial extent of the virtual mosaic. For example:

```
SELECT SDO_GEOR_AGGR.getMosaicExtent('grview', 'grobj', OUTSRID) from dual;
```

4. Get the resolution range of the existing source images. For example:

```
SELECT SDO_GEOR_AGGR.getMosaicResolutions('grview', 'grobj', 'unit=meter') from
dual;
```

The resolution range reflects the minimum and maximum resolutions of the source images, including all pyramid levels.

5. Based on the information acquired in the preceding two steps, pass in the spatial query window `cropArea` and `OUTResolutions` according to the application requests to get a subset of the virtual mosaic and optionally to apply different resampling methods, different common point rules, special nodata handling, and color balancing. For example:

```
SDO_GEOR_AGGR.getMosaicSubset('grview', 'grobj', null, OUTSRID, null, null,
  cropArea, null, null, null, OUTResolutions, null,
  'commonPointRule=end, nodata=true', lb, outArea, outWin);
```

Note that `OUTResolutions` must be within the source image resolution range. If `OUTResolutions` is the same as the resolutions of the source image at a specified pyramid level, the pyramid data is used in the output mosaic; otherwise, the source image is scaled to the target resolution.

A typical application repeatedly applies this step to query different areas of interest over the same virtual mosaic for image display, image distribution, or other purposes.

6.18.5 Special Considerations for Large-Scale Virtual Mosaic

A virtual mosaic can contain just several images, but it can also contain tens of thousands or millions of images. Both SDO_GEOR_AGGR.getMosaicSubset and SDO_GEOR_AGGR.mosaicSubset automatically search (using native spatial indexes) the virtual mosaic for all images touching or inside the `cropArea` and check the resolutions of those images and their pyramids. Only those images or their appropriate pyramid levels touching or inside the `cropArea` and with their resolutions close to the requested resolution will be used in the mosaicking process. So, the configuration of the source images and their pyramids is critical for the quality of the results and the overall query performance.

The guideline is to avoid too many small images from either different source images or their pyramids in the requested crop areas at the requested resolution.

For a smaller virtual mosaic with only a limited number of images, simply generate full pyramids for each source image, and the query performance will be good for most applications.

For a large area with a larger number of images (more than a few hundred images), the application can generate only a certain number of pyramid levels for each source image, mosaic their top pyramids into new GeoRaster objects, and then generate pyramids for those mosaics, and so forth. For large-scale web visualization projects, all images at source resolutions and at lower resolution levels might be stored as GeoRaster objects without any pyramids built for them.

In these cases (large number of images and large-scale web visualization), if each source image is small and there are many resolution levels in the virtual mosaic, a query on the lower resolution levels would involve metadata resolution queries on many unnecessary images at the higher resolution levels, which slows the query. To improve performance, applications can define many virtual mosaics, each of which includes only all the images at a specific resolution or a few resolution levels. Then, the application finds the right virtual mosaic or mosaics based on the requested resolution as the first step, and then only spatially queries those selected virtual mosaics. This approach can significantly improve performance.

In addition to the preceding considerations, see Improving Query Performance Using MIN_X_RES$ and MAX_X_RES$ for queries where many different resolution levels are involved for the same area.

- Improving Query Performance Using MIN_X_RES$ and MAX_X_RES$

6.18.5.1 Improving Query Performance Using MIN_X_RES$ and MAX_X_RES$

A more general solution (instead of defining multiple virtual mosaics) for speeding virtual mosaic queries if there are many different resolution levels involved for the same area is to use the MIN_X_RES$ and MAX_X_RES$ columns in the GeoRaster tables or views. You must define these columns (NUMBER data type) in the GeoRaster tables of a virtual mosaic, where they specify the minimum and maximum spatial resolution values, respectively, of the source GeoRaster object. After these columns are added and populated with correct resolution data, the SDO_GEOR_AGGR.getMosaicSubset procedure will (if you use the format with the `georasterTableNames` parameter) use the resolution range stored in these columns to filter out the source GeoRaster objects that are not at the requested resolutions as specified in the `outResolutions` parameter. This avoids parsing the metadata of each GeoRaster objects in the `cropArea`, thus significantly improving performance.

To use this approach, follow these steps:

1. Add the columns MIN_X_RES$ and MAX_X_RES$ to the GeoRaster tables. For example:

```
ALTER TABLE georaster_table ADD (MIN_X_RES$ number, MAX_X_RES$ number);
```

2. Populate the MIN_X_RES$ column. For example:

```
UPDATE georaster_table t
  SET min_x_res$ = (select column_value from the
    (select sdo_geor.generateSpatialResolutions(t.georaster, null,
    t.georaster.spatialextent.sdo_srid) from dual)
  WHERE rownum=1);
```

3. Populate the MAX_X_RES$ column. For example:

```
UPDATE georaster_table t
  max_x_res$ = min_x_res$ * power(2, sdo_geor.getPyramidMaxLevel(t.georaster));
```

If the virtual mosaic is defined as a view, the view should also have both columns. For example, the view definition in Example 6-26 must be changed to the following:

```
Create or replace view grview as select * from (
    Select grobj, min_x_res$, max_x_res$, last_update from grtab1 where
cloud_cover=0 union all
    Select grobj, min_x_res$, max_x_res$, last_update from grtab2 where
cloud_cover=0 union all
```

```
Select grobj, min_x_res$, max_x_res$, last_update from grtab3 )
order by last_update;
```

After a virtual mosaic is defined as described in this section, applications can query and use it in the same ways as with all other virtual mosaics, but with better performance for large-scale virtual mosaics that involve many resolution levels. For more information, see the SDO_GEOR_AGGR.getMosaicSubset and SDO_GEOR_AGGR.mosaicSubset reference sections.

6.19 Image Serving

Serving of image and raster data to clients or applications is supported through many features of the GeoRaster PL/SQL and Java APIs.

Direct image serving includes searching and then subsetting or cropping the rasters (SDO_GEOR.getRasterSubset), applying reprojection and rectification on-the-fly while cropping the images (SDO_GEOR.reproject and SDO_GEOR.rectify), and directly exporting to files (SDO_GEOR.exportTo).

Virtual mosaic is used mainly, and effectively, to serve an image database to various applications, particularly when you do not want to create large physical mosaics. Virtual mosaic does not require the source images to be preprocessed or mosaicked beforehand. Instead, all images are stored as is, and the whole image data set can be served based on small areas of interest using single calls (SDO_GEOR_AGGR.getMosaicSubset) to the server.

Often, one or a series of preprocessing operations are applied to multiple GeoRaster objects to create the resulting GeoRaster object, and then the features described in this section are used on the resulting GeoRaster object to serve the raster data directly to applications. Thus, a rich set of GeoRaster image manipulation and raster algebra capabilities (described in GeoRaster Data Query and Manipulation, Raster Algebra and Analytics, and this chapter) can be incorporated into the workflow to meet complex image serving requirements.

7

SDO_GEOR Package Reference

The 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 GeoRaster PL/SQL API. Many of the subprograms are also discussed in GeoRaster Database Creation and Management and GeoRaster Data Query and Manipulation.

Many examples in this chapter refer to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.

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

Topics:

- SDO_GEOR.addNODATA
- SDO_GEOR.addSourceInfo
- SDO_GEOR.affineTransform
- SDO_GEOR.calcCompressionRatio
- SDO_GEOR.changeCellValue
- SDO_GEOR.changeCellValues
- SDO_GEOR.changeFormatCopy
- SDO_GEOR.compressJP2
- SDO_GEOR.copy
- SDO_GEOR.createBlank
- SDO_GEOR.createTemplate
- SDO_GEOR.decompressJP2
- SDO_GEOR.deleteControlPoint
- SDO_GEOR.deleteNODATA
- SDO_GEOR.deletePyramid
- SDO_GEOR.evaluateDouble
- SDO_GEOR.evaluateDoubles
- SDO_GEOR.exportTo
- SDO_GEOR.generateAreaWeightedMean
- SDO_GEOR.generateBitmapPyramid
- SDO_GEOR.generateBlockMBR
- SDO_GEOR.generatePyramid

- SDO_GEOR.hasBitmapMask
- SDO_GEOR.hasGrayScale
- SDO_GEOR.hasNODATAMask
- SDO_GEOR.hasPseudoColor
- SDO_GEOR.importFrom
- SDO_GEOR.init
- SDO_GEOR.isBlank
- SDO_GEOR.isOrthoRectified
- SDO_GEOR.isRectified
- SDO_GEOR.isSpatialReferenced
- SDO_GEOR.mask
- SDO_GEOR.mergeLayers
- SDO_GEOR.mosaic
- SDO_GEOR.rectify
- SDO_GEOR.reproject
- SDO_GEOR.scaleCopy
- SDO_GEOR.schemaValidate
- SDO_GEOR.setBeginDateTime
- SDO_GEOR.setBinFunction
- SDO_GEOR.setBinTable
- SDO_GEOR.setBitmapMask
- SDO_GEOR.setBlankCellValue
- SDO_GEOR.setColorMap
- SDO_GEOR.setColorMapTable
- SDO_GEOR.setControlPoint
- SDO_GEOR.setDefaultAlpha
- SDO_GEOR.setDefaultBlue
- SDO_GEOR.setDefaultColorLayer
- SDO_GEOR.setDefaultGreen
- SDO_GEOR.setDefaultPyramidLevel
- SDO_GEOR.setDefaultRed
- SDO_GEOR.setEndDateTime
- SDO_GEOR.setGCPGeorefMethod
- SDO_GEOR.setGCPGeorefModel
- SDO_GEOR.setGrayScale
- SDO_GEOR.setGrayScaleTable
- SDO_GEOR.setHistogramTable

- SDO_GEOR.setID
- SDO_GEOR.setLayerID
- SDO_GEOR.setLayerOrdinate
- SDO_GEOR.setModelCoordLocation
- SDO_GEOR.setModelSRID
- SDO_GEOR.setNODATAMask
- SDO_GEOR.setOrthoRectified
- SDO_GEOR.setRasterType
- SDO_GEOR.setRectified
- SDO_GEOR.setScaling
- SDO_GEOR.setSourceInfo
- SDO_GEOR.setSpatialReferenced
- SDO_GEOR.setSpatialResolutions
- SDO_GEOR.setSpectralResolution
- SDO_GEOR.setSpectralUnit
- SDO_GEOR.setSRS
- SDO_GEOR.setStatistics
- SDO_GEOR.setULTCoordinate
- SDO_GEOR.setVAT
- SDO_GEOR.setVersion
- SDO_GEOR.subset
- SDO_GEOR.updateRaster
- SDO_GEOR.validateBlockMBR
- SDO_GEOR.validateGeoRaster
- SDO_GEOR.warp

7.1 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 NODATA Values and Value Ranges

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 NODATA Values and Value Ranges.

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 Bands_ Layers_ and Metadata.

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;
/
```

7.2 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 GeoRaster Metadata XML Schema). 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.
```

7.3 SDO_GEOR.affineTransform

Format

```
SDO_GEOR.affineTransform(
     inGeoRaster    IN SDO_GEORASTER,
     translation    IN SDO_NUMBER_ARRAY DEFAULT NULL,
     scales         IN SDO_NUMBER_ARRAY DEFAULT NULL,
     rotatePt       IN SDO_NUMBER_ARRAY DEFAULT NULL,
     rotateAngle    IN NUMBER DEFAULT NULL,
     shear          IN SDO_NUMBER_ARRAY DEFAULT NULL,
     reflection     IN NUMBER DEFAULT NULL,
     storageParam   IN VARCHAR2 DEFAULT NULL,
     outGeoRaster   IN OUT SDO_GEORASTER,
     bgValues       IN SDO_NUMBER_ARRAY DEFAULT NULL,
     parallelParam  IN VARCHAR2 DEFAULT NULL);
```

or

```
     inGeoRaster    IN SDO_GEORASTER,
     translation    IN SDO_NUMBER_ARRAY DEFAULT NULL,
     scales         IN SDO_NUMBER_ARRAY DEFAULT NULL,
     rotatePt       IN SDO_NUMBER_ARRAY DEFAULT NULL,
     rotateAngle    IN NUMBER DEFAULT NULL,
     shear          IN SDO_NUMBER_ARRAY DEFAULT NULL,
     reflection     IN NUMBER DEFAULT NULL,
     storageParam   IN VARCHAR DEFAULT2 DEFAULT NULL,
     rasterBlob     IN OUT NOCOPY_BLOB,
     outArea        OUT SDO_GEOMTRY,
     outWindow      OUT SDO_NUMBER_ARRAY,
     bgValues       IN SDO_NUMBER_ARRAY DEFAULT NULL,
     parallelParam  IN VARCHAR2 DEFAULT NULL);
```

Description

Performs affine transformation on the input GeoRaster image to produce an output GeoRaster image based on the values of the parameters `translation`, `scales`, `rotatePt`. `rotateAngle`, `shear`, and `reflection`.

Parameters

inGeoRaster
GeoRaster object on which to perform the operation. It does **not** need to be georeferenced. (Georeferencing is explained in Georeferencing GeoRaster Objects and Advanced Georeferencing.)

translation
When specified, should contain two integer numeric values with the number of rows and columns to be applied to the translation transformation. The values for row and columns translation are independent of each other, but positive values will translate the image to the right and to the bottom, and negative values will translate the image to the left and to the top. If this parameter is omitted, no translation is performed.

scales
When specified, should contain two numeric values with the scale factor to be applied to the rows and columns to be applied to the scale transformation. The values for row

and columns scaling are independent from each other but values between 0 and 1 will reduce the size of the image in rows and/or columns while values greater than 1 will enlarge the size of image is rows and/or columns. If this parameter is omitted, no scaling is performed.

rotatePt
When specified, should contain two numeric value representing the cell space coordinate (row and columns) to be used as the center of the rotation operation. In practical terms, the image feature associated with rotatePt will be the center of the new output image. If this parameter is omitted, the center of the image is assumed.

rotateAngle
When specified, should contain a numeric value between -180 to 180 identifying the angle to be applied to the rotation transformation. A positive value indicates that the rotation will turn to the right and negative value indicates rotation to the left. See usage notes for more information. If this parameter is omitted, no rotation is performed.

shear
When specified, should contain two numeric value between the shear factor to be applied to the x and y coordinates respectively in a shear transformation. The values for row and columns shear are independent from each other. If this parameter is omitted, no shearing is performed.

reflection
When specified, should contain the numeric values 1 or 2, representing vertical or horizontal reflection, respectively. If this parameter is omitted, no reflection is performed.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
GeoRaster object to hold the result of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`

rasterBlob
BLOB to hold the output reflecting the rectification. It must exist or have been initialized before the 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, the procedure performs an internal commit operation. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

This procedure has two formats:

- The first format generates a GeoRaster object for persistent storage in the database.

- The second format generates a BLOB for temporary storage or immediate use, such as to display data on the screen.

This procedure performs the specified simple affine transformation operations individually or in combination.

For all the possible operations and combinations of operations, this procedure will transform the physical representation of the stored image and build new georeferencing information that preserves the original location of features in the image. Thus, the image might look the same when projected by a visualization tool.

Examples

In the following example, the output GeoRaster object will be generated from rotating the source image by -90 degrees (90 degrees to the left).

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  select georaster into gr1 from georaster_table where georid = 1;

  insert into georaster_table values(2, 'Rotated 90 left',
          sdo_geor.init('rdt0',2)) returning georaster into gr2;

  sdo_geor.affineTransform(inGeoRaster   => gr1,
                           translation   => null,
                           scales        => null,
                           rotatePt      => null,
                           rotateAngle   => -90,
                           shear         => null,
                           reflection    => null,
                           storageParam  => 'pyramid=true',
                           outGeoraster  => gr2);

  update georaster_table set georaster = gr2 where georid = 2;
  commit;
END;
```

In the following example, the output GeoRaster object will be generated from enlarging the source image two times bigger while rotating it by 15 degrees to the right.

```
DECLARE
  gr1 sdo_georaster;
  gr3 sdo_georaster;
BEGIN
  select georaster into gr2 from georaster_table where georid = 1;

  insert into georaster_table values(3, 'Scaled x 2 Rotated 15',
        sdo_geor.init('rdt0',3)) returning georaster into gr3;

  sdo_geor.affineTransform(inGeoRaster   => gr1,
                           translation   => null,
                           scales        => sdo_number_array(2,2),
                           rotatePt      => null,
                           rotateAngle   => 15,
                           shear         => null,
                           reflection    => null,
                           storageParam  => 'blocksize=(512,512,3)',
                           outGeoraster  => gr3,
                           parallelParam => 'parallel=4');

  update georaster_table set georaster = gr3 where georid = 3;
  commit;
END;
```

In the following example, the output GeoRaster object will be generated from shearing the source image by a factor of 5 in both rows and columns:

```
DECLARE
  gr1 sdo_georaster;
  gr4 sdo_georaster;
BEGIN
  select georaster into gr2 from georaster_table where georid = 1;

  insert into georaster_table values(4, 'Shear 5,5',
        sdo_geor.init('rdt0',4)) returning georaster into gr4;

  sdo_geor.affineTransform(inGeoRaster   => gr1,
                           translation   => null,
                           scales        => null,
                           rotatePt      => null,
                           rotateAngle   => null,
                           shear         => sdo_number_array(5,5),
                           reflection    => null,
                           storageParam  => 'pyramid=true',
                           outGeoraster  => gr4,
                           parallelParam => 'parallel=4');

  update georaster_table set georaster = gr4 where georid = 4;
  commit;
END;
```

In the following example, the output GeoRaster object will be generated from the vertical reflection of the source image.

```
DECLARE
  gr1 sdo_georaster;
  gr5 sdo_georaster;
BEGIN
```

ORACLE®

```
      select georaster into gr2 from georaster_table where georid = 1;

      insert into georaster_table values(5, 'Vertical reflection',
              sdo_geor.init('rdt0',5)) returning georaster into gr5;

    sdo_geor.affineTransform(inGeoRaster   => gr1,
                             translation   => null,
                             scales        => null,
                             rotatePt      => null,
                             rotateAngle   => null,
                             shear         => null,
                             reflection    => 1,
                             storageParam  => 'pyramid=true',
                             outGeoraster  => gr5,
                             parallelParam => 'parallel=4');

    update georaster_table set georaster = gr5 where georid = 5;
    commit;
END;
```

7.4 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 Compression and Decompression.

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;
```

7.5 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). 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 GeoRaster Data Model.)

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 and Graph 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 Storage Parameters.)

```
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;
/
```

7.6 SDO_GEOR.changeCellValues

Format

```
SDO_GEOR.changeCellValues(
      georaster      IN OUT SDO_GEORASTER,
      rowNumbers     IN SDO_NUMBER_ARRAY,
      colNumbers     IN SDO_NUMBER_ARRAY,
      bandNumber     IN NUMBER,
      newCellValues  IN SDO_NUMBER_ARRAY,
      bgValues       IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR.changeCellValues(
      georaster      IN OUT SDO_GEORASTER,
      ptGeom         IN SDO_GEOMETRY,
      layerNumber    IN NUMBER,
      newCellValues  IN SDO_NUMBER_ARRAY,
      bgValues       IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Changes the value of raster cells specified by row/column arrays or by a multipoint geometry to new values.

Parameters

georaster
GeoRaster object.

rowNumbers
Numbers of the rows that contains the cells whose values are to be changed.

colNumbers
Numbers of the columns that contains the cells whose values are to be changed.

bandNumber
Number of the physical band that contains the cells whose value is to be changed.

ptGeom
Multipoint geometry that identifies the cells whose values are to be changed.

layerNumber
Number of the logical layer that contains the cells whose value is to be changed. (As mentioned in Bands_ Layers_ and Metadata, the logical layer number is the physical band number plus 1.)

newCellValues
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 Empty Raster Blocks). 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 default `bgValues` are zero (0). 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.

In the `ptGeom` SDO_GEOMETRY object, 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 `ptGeom` parameter geometry and the model space are different, the `ptGeom` 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 GeoRaster Data Model.)

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 values of cells located anywhere in the GeoRaster object, use the SDO_GEOR.getCellValues function.

Examples

The following example changes the value of two cells to 151 and 152 in band number
1. (It refers to a table named GEORASTER_TABLE, whose definition is presented
after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr sdo_georaster;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=110 FOR UPDATE;
  sdo_geor.changeCellValues(gr, sdo_number_array(100,67),sdo_number_array(134,113),
1,
     sdo_number_array(151,152));
  UPDATE georaster_table SET georaster=gr WHERE georid=110;
  COMMIT;
END;
/
```

7.7 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 Storage Parameters.

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 Blank and Empty GeoRaster Objects.) 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). 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 Storage Parameters.)

```
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;
/
```

The following example creates a GeoRaster object that is the same as the input object
except that raster data is compressed to deflate format and the compression process
is running in parallel. (It refers to a table named GEORASTER_TABLE, whose
definition is presented after Example 1-1 in Storage Parameters.)

```
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, 'compression=deflate parallel=4', gr2);
    UPDATE georaster_table SET georaster=gr2 WHERE georid=11;
    COMMIT;
END;
/
```

7.8 SDO_GEOR.compressJP2

Format

```
SDO_GEOR.compressJP2(
    inGeoRaster     IN SDO_GEORASTER,
    compressParam   IN VARCHAR2,
    outGeoRaster    IN OUT SDO_GEORASTER);
```

Description

Compresses the image in a GeoRaster object using JPEG 2000 compression.

Parameters

inGeoRaster
The SDO_GEORASTER object whose data is to be compressed.

compressParam
A string specifying one or more keywords for the compression parameter. For an
explanation of the available keywords, see the table in the Usage Notes.

outGeoRaster
The SDO_GEORASTER object to hold the result of the compression. Must be either a
valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster

objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`.

Usage Notes

The output compressed image is in JPEG 2000 (JP2) file format and stored in one raster block of the `outGeoRaster` object. There are no pyramid raster blocks stored in the raster data table, because the pyramids are stored in the JP2 file as part of the compression.

If not specified in `rlevel` keyword of `compressParam`, the maximum number of pyramid level is calculated as: `floor(log2(tsize))`, where `tsize` is the minimal value of the `tilesize` parameter values for rows and columns. If the `tiling` parameter value is`false`, `tsize` is the minimal value of the image height and width.

If neither `ratio` nor `psnr` is specified, the compression is loss-less

This procedure supports 8–bit and 16–bit source GeoRaster objects. The maximum of number of tiles allowed is 65535.

The following table lists the available `compressParam` keywords for JPEG 2000 (JP2) compression.

Table 7-1   compressParam Keywords for JPEG 2000 (JP2) Compression

Keyword	Explanation
codeBlockSize=(cbrow, cbcol)	Specifies the code block row and column size, where `cbrow` and `cbcol` are the size of the code block in rows and columns, respectively. It must be in the range of [4, 1024] and cbrow * cbcol <= 4096. By default, it is 64 x 64.
dwt=reversible | irreversible	Specifies the discrete wavelet transform, where `reversible` means to use the DWT 5–3 transform, and `irreversible` means to use the DWT 9–7 transform. Irreversible transforms always result in lossy compression.
mct=true | false	Specifies whether to use multiple component transform. By default, RGB->YCC conversion is used if there are 3 bands or more.
precinctSize=(pcrow, pccol)	Specifies the precinct size, where `pcrow` and `pccol` are the size of the precinct in rows and columns, respectively. By default it is 512 x 512 on each resolution.
progressOrder=LRCP|RLCP|RPCL|PCRL|CPRL	Specifies the progression order: `LRCP` (layer-resolution-component-position progressive, or rate scalable), `RLCP` (resolution-layer-component-position progressive, or resolution scalable), `RPCL` (resolution-position-component-layer progressive), `PCRL` (position-component-resolution-layer progressive), or `CPRL` (component-position-resolution-layer) progressive. By default, it is `LRCP`.

Table 7-1   (Cont.) compressParam Keywords for JPEG 2000 (JP2) Compression

Keyword	Explanation
psnr=(p1, p2, p3, ...)	Specifies the peak signal-to-noise ratio (PSNR), where p1, p2, p3, ... are the compression PSNR for layer 1, 2, 3, and so on of the JP2 code stream. It should be in increasing order. Example: psnr=(30, 40, 50). By default, the compression is loss-less. This parameter cannot be specified together with the `ratio` parameter.
ratio=(r1, r2, r3, ...)	Specifies the compression ratio, where p1, p2, p3, ... are the compression ratios for layers 1, 2, 3, and so on of the JP2 code stream. It should be in decreasing order. Example: ratio=(30, 20, 10). By default, the compression is loss-less. This parameter cannot be specified together with the `psnr` parameter.
rlevel=n	Specifies the number of decompositions of the wavelet transform, and thus the number of pyramids of the image. By default, the level of decomposition is floor(log2(tileSize)).
tileSize=(trow, tcol)	`trow` and `tcol` specify the row and column size of the tile. If the tile size is greater than the image size, no tiling is applied.
tiling=true | false	Specifies whether to use tiling in the JPEG2000 compression. By default, tiling is `true`. If `tiling` is `true` and if `tileSize` is not set, the default tile size is 512 x 512.

> **Note:**
>
> For any numbers in string (VARCHAR2) parameters to GeoRaster subprograms, the period (.) must be used for any decimal points regardless of the locale.

Examples

The following example creates a JPEG 2000 compressed GeoRaster object from the original object. The JP2 file internal tile size is 512 by 512 and the compression ratio values for JP2 layers 1, 2, and 3 are 30, 20, and 10, respectively. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
    VALUES (4, sdo_geor.init('RDT_1'))
    RETURNING georaster INTO gr2;
```

```
      SELECT georaster INTO gr1 FROM georaster_table WHERE georid=1;

      sdo_geor.compressJP2(gr1,'tilesize=(512, 512), ratio=(30, 20, 10)', gr2);
      UPDATE georaster_table SET georaster=gr2 WHERE georid=4;
      COMMIT;
END;
/
```

7.9 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 Blank and Empty GeoRaster Objects.) 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 Storage Parameters.)

```
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;
/
```

7.10 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 rasterType Attribute.

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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA). GeoRaster operations are described in GeoRaster Database Creation and Management and GeoRaster Data Query and Manipulation.)

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 Storage Parameters.)

```
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')
);
```

7.11 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 rasterType Attribute.

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 keyword-value pairs. The following keywords are supported for this parameter:

- `blocking` (for example, `blocking=TRUE`). For an explanation of this keyword, see Table 1-1 in Storage Parameters.

- `blocksize` (for example, `blocksize=(512,512,3)`). For an explanation of this keyword, see Table 1-1 in Storage Parameters.

ORACLE®

- cellDepth (for example, `cellDepth=8BIT_S`). For an explanation of this keyword, see Table 1-1 in Storage Parameters.

- compression (for example, `compression=JPEG-F`). For an explanation of this keyword, see Table 1-1 in Storage Parameters.

- dimSize (for example, `dimSize=(512,512,3)`): Specifies the row, column, and band dimension sizes. This keyword must be specified and must be consistent with the `rasterType` parameter.

- interleaving (for example, `interleaving=BIP`). For an explanation of this keyword, see Table 1-1 in Storage Parameters.

- quality (for example, `quality=75`). For an explanation of this keyword, see Table 1-1 in Storage Parameters.

- resampling (for example, `resampling=NN`): Specifies the resampling method. Must be one of the following: `NN`, `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, `AVERAGE16`. For more information, see Resampling and Interpolation.

  The `resampling` keyword is ignored if `rLevel` is not set.

- 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`.

- 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.

(Note that the following keywords in Table 1-1 in Storage Parameters are *not* supported for the `rasterSpec` parameter: `bitmapmask` and `pyramid`.)

If the `compression` keyword in the `rasterSpec` parameter is set as `JP2-F`, the blocking defaults to non-blocking no matter what the `rasterSpec` parameter specifies. If the `rlevel` keyword in the `rasterSpec` parameter is not set and `compression` is `JP2-F`, the pyramid maximum level in the GeoRaster metadata defaults to floor(log2(imageSize)), where `imageSize` is the minimum of the image row and column size. The pyramid maximum level should be adjusted later if the actual data has different levels of resolutions.

For more information about using this function in developing GeoRaster applications, see Using Template-Related Subprograms to Develop GeoRaster Applications.

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=(512,512,3) blocking=false rlevel=2',
                       null, 'TRUE');
  UPDATE georaster_table set georaster=gr where georid=1;
  COMMIT;
END;
/
```

7.12 SDO_GEOR.decompressJP2

Format

```
SDO_GEOR.decompressJP2(
     inGeoRaster   IN SDO_GEORASTER,
     pyramidLevel  IN NUMBER,
     cropArea      IN SDO_NUMBER_ARRAY,
     bandNumbers   IN VARCHAR2,
     storageParam  IN VARCHAR2,
     outGeoraster  IN OUT SDO_GEORASTER);
```

or

```
SDO_GEOR.decompressJP2(
     inGeoRaster   IN SDO_GEORASTER,
     pyramidLevel  IN NUMBER,
     cropArea      IN SDO_GEOMETRY,
     layerNumbers  IN VARCHAR2,
     storageParam  IN VARCHAR2,
     outGeoraster  IN OUT SDO_GEORASTER);
```

Description

Decompress the JPEG 2000 compressed GeoRaster image into a GeoRaster object.

Parameters

inGeoRaster
The SDO_GEORASTER object to be decompressed.

pyramidLevel
A number specifying the pyramid level to be decompressed in 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.

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).

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`.

Usage Notes

In the `storageParam` parameter, any `bitmapmask`, `compression`, `quality`, and `pyramid` keywords are ignored.

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 GeoRaster Data Model.)

Examples

The following example creates an uncompressed GeoRaster object that contains only specified bands from a specified window from the original object. The original object's raster data is compressed in JPEG 2000 compression. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
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.decompressJP2(gr1, 0, sdo_geometry(2003, NULL, NULL,
                            sdo_elem_info_array(1, 1003, 3),
                            sdo_ordinate_array(0,256,255,511)),
                  '3,1-2','blocksize=(512, 512, 3)', gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

7.13 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 Ground Control Point (GCP) Georeferencing Model.

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;
/
```

7.14 SDO_GEOR.deleteNODATA

Format

```
SDO_GEOR.deleteNODATA(title
     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 NODATA Values and Value Ranges

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 NODATA Values and Value Ranges.

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;
/
```

7.15 SDO_GEOR.deletePyramid

Format

```
SDO_GEOR.deletePyramid(
      georaster      IN OUT SDO_GEORASTER,
      pyramidLevel   IN NUMBER DEFAULT NULL);
```

Description

Deletes the pyramid data of a GeoRaster object from the given pyramid level and above.

Parameters

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

pyramidLevel
The level of pyramid (and above) for which to delete pyramid data. By default, all the pyramid data is deleted.

Usage Notes

For information about pyramid data, see Pyramids.

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

An exception is raised if `georaster` is invalid or if the value of `pyramidLevel` is less than 1.

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 Storage Parameters.)

```
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;
/
```

The following example deletes the pyramid data for a GeoRaster object where the pyramid level is greater than or equal to 3. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

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

  sdo_geor.deletePyramid(gr1, 3);
  UPDATE georaster_table SET georaster=gr1 WHERE georid=21;
  COMMIT;
END;
/
```

ORACLE®

7.16 SDO_GEOR.evaluateDouble

Format

```
SDO_GEOR.evaluateDouble(
      georaster           IN SDO_GEORASTER,
      pyramidLevel        IN NUMBER,
      row                 IN NUMBER,
      column              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 Bands_ Layers_ and Metadata, 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`, `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, `AVERAGE16`. For more information, see Resampling and Interpolation.

- `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`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, 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 Storage Parameters.

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.
```

7.17 SDO_GEOR.evaluateDoubles

Format

```
SDO_GEOR.evaluateDoubles(
      georaster           IN SDO_GEORASTER,
      pyramidLevel        IN NUMBER,
      rows                IN SDO_NUMBER_ARRAY,
      cols                IN SDO_NUMBER_ARRAY,
      band                IN NUMBER,
      interpolationMethod IN VARCHAR2
      ) RETURN SDO_NUMBER_ARRAY;
```

or

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

Description

Evaluates multiple locations using a specified interpolation method, and returns the raster values (double precision numbers) for the specified band or layer for those locations.

Parameters

georaster
GeoRaster object.

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

row
The row coordinates of the locations whose raster values are to be returned.

column

The column coordinates of the locations whose raster values are to be returned.

band

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

ptGeom

Multipoint geometry that identifies the cells whose values are to be returned.

layers

Number of the logical layer that contains the cell whose value is to be returned. (As mentioned in Bands_ Layers_ and Metadata, 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.

`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`, `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, `AVERAGE16`. For more information, see Resampling and Interpolation.

- `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`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, 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.getCellValues function.

Examples

The following example returns the raster values for specified locations 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 Storage Parameters.

```
SELECT SDO_GEOR.evaluateDoubles(a.georaster,
0,SDO_NUMBER_ARRAY(10.2,11.5),SDO_NUMBER_ARRAY(9.2,8.3),0,
    'interpolationMethod=BILINEAR')
  FROM georaster_table a WHERE georid=21;

SDO_GEOR.EVALUATEDOUBLES(A.GEORASTER,
0,SDO_NUMBER_ARRAY(10.2,11.5),SDO_NUMBER_ARRAY(9.2,8.3),
```

```
0,'interpolationMethod=BILINEAR')
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(86.68, 135.68)

1 row selected.
```

7.18 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 Storage Parameters, 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.

- `layerNumber`: Layer numbers of the layers to be exported. For example, `layerNumber=(3-5)` exports layers 3, 4, and 5; and `layerNumber=(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 GeoRaster Tools: Viewer_ Loader_ Exporter, 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 Georeferencing GeoRaster Objects.

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;
/
```

7.19 SDO_GEOR.generateAreaWeightedMean

Format

```
SDO_GEOR.generateAreaWeightedMean(
     georaster    IN SDO_GEORASTER,
     window       IN SDO_GEOMETRY,
     bandNumber   IN NUMBER,
     nodata       IN VARCHAR2 DEFAULT 'FALSE'
     ) RETURN NUMBER;
```

Description

Computes the area weighted statistical mean value on a specified band of the input
GeoRaster object.

Parameters

georaster
GeoRaster object.

window
A polygon geometry object from which to generate statistics. If it is in the model
space, it is transformed into raster space. See also the Usage Notes for the
SDO_GEOR.generateStatisticsfunction for SDO_SRID requirements.

bandNumber
Band ordinate number of the layer for which to perform the operation.

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 NODATA Values and Value Ranges.

Usage Notes

This function computes area weighted statistical mean value for the cells within a specific window of the input GeoRaster object. It uses the area of the cell as the weight for each cell value when computing the statistical mean value, so that all subcells cut by the window boundary are processed appropriately.

Examples

The following example computes area weighted statistical mean value for the first band of the input GeoRaster object.

```
SELECT sdo_geor.generateAreaWeightedMean(tmimage,sdo_geometry(2003,
        null,NULL,
        MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
        MDSYS.SDO_ORDINATE_ARRAY(9.5,9.5,12.5,12.5)),0)
    FROM landsat WHERE id=11;
```

7.20 SDO_GEOR.generateBitmapPyramid

Format

```
SDO_GEOR.generateBitmapPyramid(
    src_geor      IN SDO_GEORASTER,
    tmp_geor      IN OUT SDO_GEORASTER,
    target_geor   IN OUT SDO_GEORASTER,
    pyramidParams IN VARCHAR2,
    bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
    parallelParam IN VARCHAR2 DEFAULT NULL,
    addColorMap   IN NUMBER DEFAULT 1);
```

Description

Generates pyramid data for a bitmap GeoRaster object.

Parameters

src_geor
Source GeoRaster object for which pyramid data is to be generated.

tmp_geor
Temporary GeoRaster object used to store temporary data

target_geor
Target GeoRaster object used to store the resulting GeoRaster data.

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 Empty Raster Blocks). 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 default bgValues are zero (0). The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form parallel=*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)

addColorMap

A number to specify whether to add a colormap to the target GeoRaster object to display roads as white. 0 (zero) does not add such a colormap; 1 (the default) or any other nonzero value adds such a colormap.

Usage Notes

For bitmap raster of points, lines, or polylines, which are typically stored in 1-bit cell depth, the pyramiding approach described in Image Pyramiding: Parallel Generation and Partial Update may not create high-quality pyramids. Distorted point patterns and dashed lines are commonly seen in those pyramids. To solve such problems, you can instead use the GeoRaster raster algebra to perform bitmap pyramiding, as explained in Bitmap Pyramiding.

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, BILINEAR, BIQUADRATIC, CUBIC, AVERAGE4, AVERAGE16. For more information, see Resampling and Interpolation.

  Note that for this procedure, BILINEAR and AVERAGE4 have the same effect.

- 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, BIQUADRATIC, CUBIC, AVERAGE4, or AVERAGE16, 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 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.

Examples

The following example generates a bitmap pyramid.

```
DECLARE
  src_gr sdo_georaster;
  tmp_gr sdo_georaster;
  target_gr sdo_georaster;
BEGIN

  SELECT georaster INTO src_gr
    FROM georaster_table WHERE georid = 6;
  SELECT georaster INTO tmp_gr
    FROM georaster_table WHERE georid = 7 FOR UPDATE;
  SELECT georaster INTO target_gr
    FROM georaster_table WHERE georid = 8 FOR UPDATE;

 -- Generate bitmap pyramids.
  sdo_geor.generateBitmapPyramid(src_gr,tmp_gr,target_gr 'rLevel=5, resampling=NN');

  -- Update the target GeoRaster object.
  UPDATE georaster_table SET georaster =target_gr WHERE georid = 8;

END;
/
```

7.21 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 blockMBR Attribute) in each row of the raster data table associated with the GeoRaster object.

If you created the GeoRaster object as described in Creating New GeoRaster Objects, 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;
/
```

7.22 SDO_GEOR.generatePyramid

Format

```
SDO_GEOR.generatePyramid(
      georaster      IN OUT SDO_GEORASTER,
      pyramidParams IN VARCHAR2,
      bgValues       IN SDO_NUMBER_ARRAY DEFAULT NULL,
      parallelParam IN VARCHAR2 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

For information about pyramid data, see Pyramids.

`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`, `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, `AVERAGE16`. For more information, see Resampling and Interpolation.

  Note that for this procedure, `BILINEAR` and `AVERAGE4` have the same effect.

- `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`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, 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 a different `resampling` value from that specified in `pyramidParams`, the old pyramid data is deleted and new pyramid data is generated. However, a different `nodata` specification in `pyramidParams` does *not* cause the pyramid data to be regenerated. To cause a new `nodata` value to take effect, you must delete the old pyramid data and then regenerate it.

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;
/
```

7.23 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 and Graph 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 and Graph Developer's Guide*.

This function does not set the spatial extent of the GeoRaster object (`spatialExtent` attribute, described in spatialExtent Attribute). For information about setting the spatial extent, see Generating and Setting Spatial Extents.

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 Storage Parameters.)

```
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))
```

7.24 SDO_GEOR.generateSpatialResolutions

Format (Procedure)

```
SDO_GEOR.generateSpatialResolutions(
     georaster      IN OUT SDO_GEORASTER,
     outResolution  OUT SDO_NUMBER_ARRAY);
```

Format (Function)

```
SDO_GEOR.generateSpatialResolutions(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER DEFAULT NULL,
     SRID           IN NUMBER DEFAULT NULL,
     resolutionUnit IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Generates the spatial resolution value along each spatial dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

outResolution

Generated spatial resolutions of the GeoRaster object. It is an array of two numbers that represent spatial resolutions on the X axis and Y axis, respectively.

pyramidLevel

Pyramid level of the returned resolution values. The default is pyramid level 0.

SRID

Coordinate system. Must be a value from the SRID column of the MDSYS.CS_SRS table. The srid value cannot be 0 (zero). If not specified, the default is the SRID associated with `georaster`.

resolutionUnit

Unit of measurement: a quoted string with `unit=`. If not specified, the unit associated with `SRID` is used.

Usage Notes

SDO_GEOR.generateSpatialResolutions has two formats:

- The procedure format sets the generated spatial resolutions in the metadata for the GeoRaster object. The metadata for the GeoRaster object is updated. The GeoRaster object must be georeferenced in order to get the spatial resolution generated.

- The function format generates and returns the spatial resolutions based on the georeferencing information in the metadata for the GeoRaster object. The metadata for the GeoRaster object is *not* updated.

The GeoRaster object must be georeferenced. The generated spatial resolution is the resolution at the center cell of the GeoRaster object.

In the returned array of numeric values, 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 GeoRaster object is automatically validated after the operation completes.

See also the Usage Notes for the SDO_GEOR.getSpatialResolutions function.

Examples

The following example generates the spatial resolution value along each spatial dimension for pyramid level 2 of a specified GeoRaster object

```
SELECT sdo_geor.generateSpatialResolutions(georaster, 2, 4326, null) FROM
georaster_table WHERE georid=2;

SDO_GEOR.GENERATESPATIALRESOLUTIONS(GEORASTER,2,4326,NULL)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(.000024266, .000018006)
```

7.25 SDO_GEOR.generateStatistics

Format (Storage in Metadata)

```
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;
```

Format (No Storage in Metadata)

```
SDO_GEOR.generateStatistics(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatistics(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers, or with one or more layers and pyramid levels. The two sets of function formats have significant usage differences:

- **Storage in Metadata** formats also set statistical data in the GeoRaster object metadata for each specified layer, and optionally for the whole raster. These formats return the string TRUE or FALSE, indicating success or failure of the operation.

- **No Storage in Metadata** formats do not set any GeoRaster object metadata, and they calculate statistics for a single layer or for the aggregation of specified layers. These formats return an SDO_NUMBER_ARRAY object where the six numbers reflect the aggregated minimum, maximum, mean, median, mode, and standard deviation values.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level on which to perform the operation.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if polygonClip is TRUE, then the samplingWindow geometry object will be used for the operation instead of the MBR of the sampling window, in which case only cells within the samplingWindow geometry are counted.

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 SDO_GEOR_HISTOGRAM Object Type. The XML definitions of the <histogram> element and the histogramType complex type are included in GeoRaster Metadata XML Schema.

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.

bandNumbers

Band ordinate numbers of the layers for which values are used in computing 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, `'0,1,3-5,7'` specifies layers 1, 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

polygonClip

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

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes and can set the statistical data described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema.

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

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 GeoRaster Metadata XML Schema.) 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 optionally 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;
        /
```

The following example generates the statistical data for all bands on pyramid level 1; however, it does not create a histogram or modify the GeoRaster object metadata.

```
DECLARE
  gr sdo_georaster;
  window  sdo_geometry;
  stat    sdo_number_array;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=2021;
  stat:=sdo_geor.generateStatistics(gr,1,'samplingFactor=7',window,null,'false');
  dbms_output.put_line('min='||stat(1));
  dbms_output.put_line('max='||stat(2));
  dbms_output.put_line('mean='||stat(3));
  dbms_output.put_line('median='||stat(4));
  dbms_output.put_line('mode='||stat(5));
  dbms_output.put_line('std='||stat(6));
END;
/
```

7.26 SDO_GEOR.generateStatisticsMax

Format

```
SDO_GEOR.generateStatisticsMax(
     georaster       IN  SDO_GEORASTER,
     pyramidLevel    IN  NUMBER,
     samplingFactor  IN  VARCHAR2,
     samplingWindow  IN  SDO_NUMBER_ARRAY,
     bandNumbers     IN  VARCHAR2 DEFAULT NULL,
     nodata          IN  VARCHAR2 DEFAULT 'FALSE'
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMax(
     georaster       IN  SDO_GEORASTER,
     pyramidLevel    IN  NUMBER,
     samplingFactor  IN  VARCHAR2,
     samplingWindow  IN  SDO_GEOMETRY,
     bandNumbers     IN  VARCHAR2 DEFAULT NULL,
     nodata          IN  VARCHAR2 DEFAULT 'FALSE',
     polygonClip     IN  VARCHAR2 DEFAULT NULL
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMax(
     georaster       IN OUT SDO_GEORASTER,
     pyramidLevel    IN  NUMBER,
     samplingFactor  IN  VARCHAR2,
     samplingWindow  IN  SDO_NUMBER_ARRAY,
```

```
        bandNumbers    IN VARCHAR2 DEFAULT NULL,
        nodata         IN VARCHAR2 DEFAULT 'FALSE',
        parallelParam  IN VARCHAR2 DEFAULT NULL,
        ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatisticsMax(
        georaster      IN OUT SDO_GEORASTER,
        pyramidLevel   IN NUMBER,
        samplingFactor IN VARCHAR2,
        samplingWindow IN SDO_GEOMETRY,
        bandNumbers    IN VARCHAR2 DEFAULT NULL,
        nodata         IN VARCHAR2 DEFAULT 'FALSE',
        polygonClip    IN VARCHAR2 DEFAULT NULL',
        parallelParam  IN VARCHAR2 DEFAULT NULL,
        ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers on a specified pyramid level, and returns the maximum value. (It does not modify metadata in the GeoRaster object.)

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the returned resolution values. The default is pyramid level 0.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if `polygonClip` is TRUE, then the `samplingWindow` geometry object will be used for the operation instead of the MBR of the sampling window, in which case only cells within the `samplingWindow` geometry are counted.
If the data type is SDO_GEOMETRY, see also the Usage Notes for the SDO_GEOR.generateStatistics function for SDO_SRID requirements.

bandNumbers

Band ordinate 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 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

polygonClip

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

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes statistical data and returns the `MAX` value. (The statistical data is described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema; however, this function does not update any metadata.)

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

See also the SDO_GEOR.generateStatistics function.

Examples

The following example computes statistical data for all bands on pyramid level 1, and returns the maximum value.

```
DECLARE
 gr sdo_georaster;
 window sdo_geometry;
 max number;
BEGIN
 SELECT tmimage INTO gr FROM landsat WHERE id=2021;
 max:=sdo_geor.generateStatisticsMax(gr,1,'samplingFactor=7',window,null,'false');
END;
/
```

7.27 SDO_GEOR.generateStatisticsMean

Format

```
SDO_GEOR.generateStatisticsMean(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE'
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMean(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMean(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatisticsMean(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers on a specified pyramid level, and returns the mean (average) value. (It does not modify metadata in the GeoRaster object.)

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the returned resolution values. The default is pyramid level 0.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if `polygonClip` is `TRUE`, then the `samplingWindow` geometry object will be used for the operation instead of the MBR of the sampling window, in which case only cells within the `samplingWindow` geometry are counted.
If the data type is SDO_GEOMETRY, see also the Usage Notes for the SDO_GEOR.generateStatistics function for SDO_SRID requirements.

bandNumbers
Band ordinate 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 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

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

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=n`, where *n* is greater than 1. The database optimizer uses the degree of

parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes statistical data and returns the MEAN value. (The statistical data is described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema; however, this function does not update any metadata.)

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

See also the SDO_GEOR.generateStatistics function.

Examples

The following example computes statistical data for all bands on pyramid level 1, and returns the mean value.

```
DECLARE
  gr sdo_georaster;
  window sdo_geometry;
  mean number;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=2021;
  mean:=sdo_geor.generateStatisticsMean(gr,1,'samplingFactor=7',window,null,'false');
END;
/
```

7.28 SDO_GEOR.generateStatisticsMedian

Format

```
SDO_GEOR.generateStatisticsMedian(
     georaster       IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers     IN VARCHAR2 DEFAULT NULL,
     nodata          IN VARCHAR2 DEFAULT 'FALSE'
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMedian(
     georaster       IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers     IN VARCHAR2 DEFAULT NULL,
     nodata          IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMedian(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatisticsMedian(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers on a specified pyramid level, and returns the median value. (It does not modify metadata in the GeoRaster object.)

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the returned resolution values. The default is pyramid level 0.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if `polygonClip` is `TRUE`, then the `samplingWindow` geometry object will be used

for the operation instead of the MBR of the sampling window, in which case only cells within the `samplingWindow` geometry are counted.
If the data type is SDO_GEOMETRY, see also the Usage Notes for the SDO_GEOR.generateStatistics function for SDO_SRID requirements.

bandNumbers
Band ordinate 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 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

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

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes statistical data and returns the `MEDIAN` value. (The statistical data is described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema; however, this function does not update any metadata.)

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

See also the SDO_GEOR.generateStatistics function.

Examples

The following example computes statistical data for all bands on pyramid level 1, and returns the median value.

```
DECLARE
  gr sdo_georaster;
  window sdo_geometry;
  median number;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=2021;
  median:=sdo_geor.generateStatisticsMedian(gr,
1,'samplingFactor=7',window,null,'false');
```

```
END;
/
```

7.29 SDO_GEOR.generateStatisticsMin

Format

```
SDO_GEOR.generateStatisticsMin(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE'
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMin(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMin(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatisticsMin(
     georaster      IN OUT SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL',
     parallelParam  IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers on a specified pyramid level, and returns the minimum value. (It does not modify metadata in the GeoRaster object.)

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the returned resolution values. The default is pyramid level 0.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if `polygonClip` is `TRUE`, then the `samplingWindow` geometry object will be used for the operation instead of the MBR of the sampling window, in which case only cells within the `samplingWindow` geometry are counted.
If the data type is SDO_GEOMETRY, see also the Usage Notes for the SDO_GEOR.generateStatistics function for SDO_SRID requirements.

bandNumbers
Band ordinate 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 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

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

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=n`, where *n* is greater than 1. The database optimizer uses the degree of

parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes statistical data and returns the MIN value. (The statistical data is described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema; however, this function does not update any metadata.)

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

See also the SDO_GEOR.generateStatistics function.

Examples

The following example computes statistical data for all bands on pyramid level 1, and returns the minimum value.

```
DECLARE
 gr sdo_georaster;
 window sdo_geometry;
 min number;
BEGIN
 SELECT tmimage INTO gr FROM landsat WHERE id=2021;
 min:=sdo_geor.generateStatisticsMin(gr,1,'samplingFactor=7',window,null,'false');
END;
/
```

7.30 SDO_GEOR.generateStatisticsMode

Format

```
SDO_GEOR.generateStatisticsMode(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_NUMBER_ARRAY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE'
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMode(
     georaster      IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     samplingFactor IN VARCHAR2,
     samplingWindow IN SDO_GEOMETRY,
     bandNumbers    IN VARCHAR2 DEFAULT NULL,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip    IN VARCHAR2 DEFAULT NULL
     ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsMode(
     georaster       IN OUT SDO_GEORASTER,
     pyramidLevel    IN NUMBER,
     samplingFactor  IN VARCHAR2,
     samplingWindow  IN SDO_NUMBER_ARRAY,
     bandNumbers     IN VARCHAR2 DEFAULT NULL,
     nodata          IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam   IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatisticsMode(
     georaster       IN OUT SDO_GEORASTER,
     pyramidLevel    IN NUMBER,
     samplingFactor  IN VARCHAR2,
     samplingWindow  IN SDO_GEOMETRY,
     bandNumbers     IN VARCHAR2 DEFAULT NULL,
     nodata          IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip     IN VARCHAR2 DEFAULT NULL',
     parallelParam   IN VARCHAR2 DEFAULT NULL,
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers on a specified pyramid level, and returns the mode value (the value that occurs most frequently). (It does not modify metadata in the GeoRaster object.)

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the returned resolution values. The default is pyramid level 0.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if `polygonClip` is `TRUE`, then the `samplingWindow` geometry object will be used

for the operation instead of the MBR of the sampling window, in which case only cells within the `samplingWindow` geometry are counted.
If the data type is SDO_GEOMETRY, see also the Usage Notes for the SDO_GEOR.generateStatistics function for SDO_SRID requirements.

bandNumbers
Band ordinate 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 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

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

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`_n_, where _n_ is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes statistical data and returns the `MODEVALUE` value. (The statistical data is described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema; however, this function does not update any metadata.)

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

See also the SDO_GEOR.generateStatistics function.

Examples

The following example computes statistical data for all bands on pyramid level 1, and returns the mode value.

```
DECLARE
  gr sdo_georaster;
  window sdo_geometry;
  mode number;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=2021;
  mode:=sdo_geor.generateStatisticsMode(gr,1,'samplingFactor=7',window,null,'false');
```

```
END;
/
```

7.31 SDO_GEOR.generateStatisticsSTD

Format

```
SDO_GEOR.generateStatisticsSTD(
      georaster       IN SDO_GEORASTER,
      pyramidLevel    IN NUMBER,
      samplingFactor  IN VARCHAR2,
      samplingWindow  IN SDO_NUMBER_ARRAY,
      bandNumbers     IN VARCHAR2 DEFAULT NULL,
      nodata          IN VARCHAR2 DEFAULT 'FALSE'
      ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsSTD(
      georaster       IN SDO_GEORASTER,
      pyramidLevel    IN NUMBER,
      samplingFactor  IN VARCHAR2,
      samplingWindow  IN SDO_GEOMETRY,
      bandNumbers     IN VARCHAR2 DEFAULT NULL,
      nodata          IN VARCHAR2 DEFAULT 'FALSE',
      polygonClip     IN VARCHAR2 DEFAULT NULL
      ) RETURN NUMBER;
```

or

```
SDO_GEOR.generateStatisticsSTD(
      georaster       IN OUT SDO_GEORASTER,
      pyramidLevel    IN NUMBER,
      samplingFactor  IN VARCHAR2,
      samplingWindow  IN SDO_NUMBER_ARRAY,
      bandNumbers     IN VARCHAR2 DEFAULT NULL,
      nodata          IN VARCHAR2 DEFAULT 'FALSE',
      parallelParam   IN VARCHAR2 DEFAULT NULL,
      ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.generateStatisticsSTD(
      georaster       IN OUT SDO_GEORASTER,
      pyramidLevel    IN NUMBER,
      samplingFactor  IN VARCHAR2,
      samplingWindow  IN SDO_GEOMETRY,
      bandNumbers     IN VARCHAR2 DEFAULT NULL,
      nodata          IN VARCHAR2 DEFAULT 'FALSE',
      polygonClip     IN VARCHAR2 DEFAULT NULL',
      parallelParam   IN VARCHAR2 DEFAULT NULL,
      ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes statistical data associated with one or more layers on a specified pyramid level, and returns the standard deviation value. (It does not modify metadata in the GeoRaster object.)

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the returned resolution values. The default is pyramid level 0.

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
A sampling window for which to generate statistics, specified either as a numeric array or as an 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. However, if `polygonClip` is `TRUE`, then the `samplingWindow` geometry object will be used for the operation instead of the MBR of the sampling window, in which case only cells within the `samplingWindow` geometry are counted.
If the data type is SDO_GEOMETRY, see also the Usage Notes for the SDO_GEOR.generateStatistics function for SDO_SRID requirements.

bandNumbers
Band ordinate 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 2, 4, 5, 6, and 8. If `bandNumbers` is null, all bands are used in computing the statistics.

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 NODATA Values and Value Ranges.

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

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=n`, where n is greater than 1. The database optimizer uses the degree of

parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, you cannot roll back the results of this function.

Usage Notes

This function computes statistical data and returns the STD value. (The statistical data is described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in GeoRaster Metadata XML Schema; however, this function does not update any metadata.)

If `samplingWindow` is outside the GeoRaster object or if it contains only NODATA values, the following error is raised:

```
ORA-13393: null or invalid samplingWindow parameter
```

See also the SDO_GEOR.generateStatistics function.

Examples

The following example computes statistical data for all bands on pyramid level 1, and returns the standard deviation value.

```
DECLARE
 gr sdo_georaster;
 window sdo_geometry;
 std number;
BEGIN
 SELECT tmimage INTO gr FROM landsat WHERE id=2021;
 std:=sdo_geor.generateStatisticsSTD(gr,1,'samplingFactor=7',window,null,'false');
END;
/
```

7.32 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,
     setResolution           IN VARCHAR2 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,
     setResolution          IN VARCHAR2 DEFAULT NULL
     ) RETURN SDO_NUMBER_ARRAY;
```

or

```
SDO_GEOR.georeference(
     georaster              IN OUT SDO_GEORASTER,
     FFMethodType           IN VARCHAR2 DEFAULT NULL,
     srid                   IN NUMBER DEFAULT NULL,
     modelCoordinateLocation IN NUMBER DEFAULT NULL,
     setResolution          IN VARCHAR2 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 Graph, 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 SDO_GEOR_GCP_ COLLECTION Collection Type).

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.

setResolution

A flag indicating whether the spatial resolution is calculated and stored in the GeoRaster metadata. The string FALSE or a null value does not store the spatial resolution in the GeoRaster metadata; the string TRUE stores the spatial resolution 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 Georeferencing and Georeferencing GeoRaster Objects.

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 GeoRaster Tools: Viewer_ Loader_ Exporter) 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 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 Ground Control Point (GCP) Georeferencing Model.

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 Storage Parameters.)

```
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;
/
```

7.33 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 Bands_ Layers_ and Metadata.

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 Storage Parameters. 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
```

7.34 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 Storage Parameters.)

```
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
```

7.35 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)
```

7.36 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 Storage Parameters.

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

7.37 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 Storage Parameters. 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
```

7.38 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 Storage Parameters.

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 Blank and Empty GeoRaster Objects.) 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 Empty Raster Blocks). 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 Bitmap Masks.

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;
/
```

7.39 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 Storage Parameters.
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 GeoRaster Data Model.)

If the `window` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial and Graph 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 Empty Raster Blocks). 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 Compression and Decompression.

For an explanation of bitmap masks, see Bitmap Masks.

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;
/
```

7.40 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 Bitmap Masks.

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;
```

7.41 SDO_GEOR.getBitmapMaskValues

Format

```
SDO_GEOR.getBitmapMaskValues(
     georaster    IN SDO_GEORASTER,
     layerNumber  IN NUMBER,
     pyramidLevel IN VARCHAR2,
     rowNumbers   IN SDO_NUMBER_ARRAY,
     colNumbers   IN SDO_NUMBER_ARRAY
     ) RETURN SDO_NUMBER_ARRAY;
```

or

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

Description

Gets the values of multiple cells 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.

rowNumbers
Numbers of the rows that contain the cells whose values are to be returned.

colNumbers
Numbers of the columns that contain the cells whose values are to be returned.

ptGeom
Multipoint geometry that identifies the cells whose values are to be returned.

Usage Notes

You can specify the cells by an array of row and column numbers or by a multipoint 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 Bitmap Masks.

Examples

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

```
SELECT sdo_geor.getBitmapMaskValues(georaster,
0,0,sdo_number_array(0,9,9,10),sdo_number_array(0,9,10,9))
  FROM georaster_table WHERE georid=0;
```

7.42 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
```

7.43 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 Storage Parameters.

```
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
```

7.44 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 Storage Parameters.

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

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

7.45 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 Storage Parameters.)

```
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))
```

7.46 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 GeoRaster Metadata XML Schema. 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 Storage Parameters.

```
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
```

7.47 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 Bands_ Layers_ and Metadata, 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 Bands_ Layers_ and Metadata, 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 using 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 Storage Parameters.

```
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 Storage Parameters.

```
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)
```

7.48 SDO_GEOR.getCellValues

Format

```
SDO_GEOR.getCellValues(
     georaster     IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     rowNumbers    IN SDO_NUMBER_ARRAY,
     colNumbers    IN SDO_NUMBER_ARRAY,
     bandNumber    IN NUMBER
     ) RETURN SDO_NUMBER_ARRAY;
```

or

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

Description

Returns the values of one or more cells located anywhere in the GeoRaster object by specifying its row/column/band numbers in its cell coordinate space, or by specifying a multipoint geometry in either model coordinate space or cell coordinate space and its logical layer number.

Parameters

georaster
GeoRaster object.

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

rowNumbers
Numbers of the rows that contains the cells whose values are to be returned.

colNumbers
Numbers of the columns that contains the cells whose values are to be returned.

bandNumber
Number of the physical band that contains the cells whose values are to be returned.

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

layerNumber
Number of the logical layer that contains the cells whose values are to be returned. (As mentioned in Bands, Layers, and Metadata, the logical layer number is the physical band number plus 1.)

Usage Notes

This function returns the original cell values stored in the raster object. It does not do any interpolation using cell values. (To evaluate a point location using an interpolation method, use the SDO_GEOR.evaluateDoubles 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.

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 Storage Parameters.

```
SELECT sdo_geor.getCellValues(georaster,
0,SDO_NUMBER_ARRAY(383,47,128,320),SDO_NUMBER_ARRAY(47,383,192,256),0)
  FROM georaster_table WHERE georid=21;

SDO_GEOR.GETCELLVALUES(A.GEORASTER,
0,SDO_NUMBER_ARRAY(383,47,128,320),SDO_NUMBER_ARRAY(47,383,192,256),0)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(48,55,52,53)
```

7.49 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. SDO_GEOR_COLORMAP Object Type 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), 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), 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))
```

7.50 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 SDO_GEOR_COLORMAP Object Type.

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
Storage Parameters.

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

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

1 row selected.
```

7.51 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-F`, or `NONE` (the latter value meaning that the
GeoRaster object is not compressed). For information about GeoRaster compression,
see Compression and Decompression.

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 Storage Parameters.

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

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

7.52 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 Ground Control Point (GCP) Georeferencing Model.

This function returns an object of type SDO_GEOR_GCP, which is described in SDO_GEOR_GCP Object Type.

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)
```

7.53 SDO_GEOR.getDefaultAlpha

Format

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

Description

Returns the number of the layer to be used for the alpha color component (in the RGBA color space) for displaying a GeoRaster object. If this value is not set in the metadata, a null value is returned.

Parameters

georaster
GeoRaster object.

Usage Notes

The default red, green, blue, and alpha 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 four color components (RGBA) by using the SDO_GEOR.getDefaultColorLayer function.

Examples

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

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

    GEORID       RED      GREEN       BLUE      ALPHA
---------- ---------- ---------- ---------- ----------
         1          1          2          3          4
         2
         3         31         20         13         10
```

7.54 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 Storage Parameters.

```
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
```

7.55 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, blue, and alpha 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 (RGBA) by using the SDO_GEOR.getDefaultRed, SDO_GEOR.getDefaultGreen, SDO_GEOR.getDefaultBlue, and SDO_GEOR.getDefaultAlpha functions.

The alpha color component is optional. If the default alpha color component exists in the metadata, this functions returns an array of four numbers identifying the red, green, blue, and alpha color components, respectively. If only the default red, green, and blue color components exist in the metadata, this functions returns an array of three numbers identifying the red, green, and blue color components respectively.

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 Storage Parameters.)

```
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);
  sdo_geor.setDefaultAlpha(grobj, 4);
  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)
SDO_NUMBER_ARRAY(2, 3, 1, 4)

1 row selected.
```

7.56 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 Storage Parameters.

```
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
```

7.57 SDO_GEOR.getDefaultPyramidLevel

Format

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

Description

Returns the number of the default pyramid level for displaying a GeoRaster object. If this value is not set in the metadata, a null value is returned.

Parameters

georaster
GeoRaster object.

Usage Notes

Pyramid levels represent reduced or increased resolution images that require less or more storage space, respectively. For information about pyramids and pyramid levels, see Pyramids.

You can set the default pyramid level by using the
SDO_GEOR.setDefaultPyramidLevel procedure.

Examples

The following example returns the default pyramid level for displaying a specified
GeoRaster object in the table named GEORASTER_TABLE, whose definition is
presented after Example 1-1 in Storage Parameters.

```
SELECT georid, sdo_geor.getDefaultPyramidLevel(georaster) plevel,
FROM georaster_table WHERE georid = 6;

    GEORID      PLEVEL
---------- ----------
         6          3
```

7.58 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 Storage
Parameters.

```
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
```

7.59 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 Storage Parameters.)

```
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
```

7.60 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 Ground Control Point (GCP) Georeferencing Model.

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
```

7.61 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 Ground Control Point (GCP) Georeferencing Model.

The SDO_GEOR_GCPGEOREFTYPE object type is defined in
SDO_GEOR_GCPGEOREFTYPE Object Type.

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)
```

7.62 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 Georeferencing.

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 Storage Parameters.)

```
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
```

7.63 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. SDO_GEOR_GRAYSCALE Object Type 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 Storage Parameters.

```
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))
```

7.64 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. SDO_GEOR_GRAYSCALE Object Type 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 Storage Parameters. 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
```

7.65 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. SDO_GEOR_HISTOGRAM Object Type 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 Storage Parameters.

```
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))
```

7.66 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. SDO_GEOR_HISTOGRAM
Object Type 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 Storage
Parameters. 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
```

7.67 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 Storage Parameters.

```
SELECT georid, substr(sdo_geor.getID(georaster),1,50) GEOR_ID
  FROM georaster_table;

    GEORID GEOR_ID
---------- --------------------------------------------------
         2 TM_102
         4 TM_104
```

7.68 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 Storage Parameters.

```
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
```

7.69 SDO_GEOR.getJP2TileSize

Format

```
SDO_GEOR.getJP2TileSize(
     georaster    IN SDO_GEORASTER
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns an array showing the size of tiles in the JPEG2000 compressed GeoRaster image, in row and column order.

Parameters

georaster
GeoRaster object.

Usage Notes

If there is no tiling in the JPEG2000 compressed GeoRaster image, null is returned.

Examples

The following example returns the tile size in the JPEG2000 compressed 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 Storage Parameters.

```
SELECT sdo_geor.getJP2TileSize(georaster) JP2TileSize
   FROM georaster_table WHERE georid=21;

JP2TILESIZE
```

```
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(350, 512)
```

7.70 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 Bands_ Layers_ and Metadata. 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 Storage Parameters. (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')
```

7.71 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 Storage Parameters.

```
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
```

7.72 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 Bands_ Layers_ and Metadata: 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 Storage Parameters.

```
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
```

7.73 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 Storage Parameters.)

```
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)
```

7.74 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 Ground Control Point (GCP) Georeferencing Model.

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;
```

7.75 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 Storage Parameters.

```
SELECT georid, sdo_geor.getModelSRID(georaster) SRID FROM georaster_table;

    GEORID       SRID
---------- ----------
         2      82394
         4      82394
```

7.76 SDO_GEOR.getNODATA

Format

```
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

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 NODATA Values and Value Ranges.

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 NODATA Values and Value Ranges.

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 Storage Parameters.

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

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

7.77 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 Pyramids.

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 Storage Parameters.

```
SELECT substr(sdo_geor.getPyramidType(georaster),1,10) pyramidType,
       sdo_geor.getPyramidMaxLevel(georaster) maxLevel
  FROM georaster_table WHERE georid=21;

PYRAMIDTYP    MAXLEVEL
---------- ----------
DECREASE            3
```

7.78 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 Pyramids.

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 Storage Parameters.

```
SELECT substr(sdo_geor.getPyramidType(georaster),1,10) pyramidType,
       sdo_geor.getPyramidMaxLevel(georaster) maxLevel
  FROM georaster_table WHERE georid=21;

PYRAMIDTYP    MAXLEVEL
---------- ----------
DECREASE            3
```

7.79 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);
```

or

```
SDO_GEOR.getRasterBlockLocator(
      georaster      IN SDO_GEORASTER,
      pyramidLevel   IN NUMBER,
      rowNumber      IN NUMBER,
      colNumber      IN NUMBER,
      bandNumber     IN NUMBER,
      offset         OUT NUMBER,
      loc            IN OUT NOCOPY BLOB,
      isBitmapMask   IN VARCHAR2 DEFAULT NULL,
      lock_for_write IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.getRasterBlockLocator(
      georaster      IN SDO_GEORASTER,
      pyramidLevel   IN NUMBER,
      ptGeom         IN SDO_GEOMETRY,
      layerNumber    IN NUMBER,
      offset         OUT NUMBER,
      loc            IN OUT NOCOPY BLOB,
      isBitmapMask   IN VARCHAR2 DEFAULT NULL,
      lock_for_write IN VARCHAR2 DEFAULT NULL);
```

Description

This procedure has three formats:

- The first listed format returns the LOB locator of a raster block by specifying the `pyramidLevel`, `bandBlockNumber`, `rowBlockNumber`, and `columnBlockNumber` parameters.

- The second and third listed formats return the LOB locator of a raster block that contains a specific single cell and the offset of the cell within the raster block. The specific single cell is identified by the `pyramidLevel`, `rowNumber`, `columnNumber`, and `bandNumber` parameters or by a point geometry parameter (`ptGeom`) in either the cell coordinate space or the model coordinate space.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level of the block.

bandBlockNumber
Band number of the block.

bandNumber
Band number of the cell.

rowBlockNumber
Row number of the block.

rowNumber
Row number of the cell.

columnBlockNumber
Column number of the block.

columnNumber
Column number of the cell.

ptGeom
Point geometry that locates the cell.

layerNumber
Number of the logical layer that contains the cell whose value is to be returned. (As mentioned in Bands_ Layers_ and Metadata, the logical layer number is the physical band ordinate number plus 1.

offset
Output parameter to contain the offset (in bytes) of the cell inside the raster block that is located. If the raster block is compressed, it always refers to the offset of the cell in the decompressed version 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 Bitmap Masks.

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 (and for some formats, the offset) 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 GeoRaster Physical Storage), 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;
    offset number;
BEGIN
    select georaster into gr from georaster_table where georid=1;
    sdo_geor.getRasterBlockLocator(gr, 0, 0, 0, 0, offset,lb, null, 'TRUE');
    sdo_geor.getRasterBlockLocator(gr, 0, 0, 0, 0, offset,lb, 'TRUE', 'TRUE');
END;
/
```

7.80 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 SDO_RASTERSET Collection Type.

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 GeoRaster Data Model.)

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 Storage Parameters.)

```
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;
/
```

7.81 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 Storage Parameters.

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. The default `bgValues` are zero (0). This parameter is useful when the source has empty raster blocks (see Empty Raster Blocks). 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 Compression and Decompression.

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 Storage Parameters.

```
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;
/
```

7.82 SDO_GEOR.getRasterRange

Format

```
SDO_GEOR.getRasterRange(
     georaster    IN SDO_GEORASTER,
     layerNumber  IN NUMBER DEFAULT 0
     ) RETURN SDO_NUMBER_ARRAY;
```

Description

Computes and returns the minimum and maximum cell values of a specified layer of a GeoRaster object or of the GeoRaster object itself (the object layer).

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the range of cell values. A value of 0 (zero, the default) indicates the object layer. (As mentioned in Bands_ Layers_ and Metadata, the logical layer number is the physical band number plus 1.)

Usage Notes

This function goes through all cell values of a specified layer of a GeoRaster object, and returns an SDO_NUMBER_ARRAY object with two values: the minimum and maximum cell values.

If the GeoRaster object already has statistical metadata generated, you can call the SDO_GEOR.getStatistics function to quickly retrieve the minimum and maximum values directly for the raster range.

For information about bands and layers, see Bands_ Layers_ and Metadata.

Examples

The following example computes and returns the raster range for the whole GeoRaster object.

```
DECLARE
  gr sdo_georaster;
  range sdo_number_array;
BEGIN
  select georaster into gr from georaster_table where georid=1;
  range:=sdo_geor.getRasterRange(gr, 0);
END;
/
```

7.83 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);
```

or

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

ORACLE

Description

Creates a single BLOB object or a single SDO_NUMBER_ARRAY 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.

rasterData
SDO_NUMBER_ARRAY object to hold the result (the mosaicked raster subset) of the operation.
(Note: The upper limit of element numbers in an SDO_NUMBER_ARRAY object is 1048576.)

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 Storage Parameters.
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. The default `bgValues` are zero (0). This parameter is useful when the source has empty raster blocks and the output window intersects any empty raster blocks (see Empty Raster Blocks). 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 eight 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, whether the result of the operation is a BLOB or SDO_NUMEBR_ARRAY object, 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 GeoRaster Data Model.)

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 and Graph 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 Compression and Decompression.

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 two examples retrieve raster data of a specified pyramid level inside a specified window into a BLOB object and an SDO_NUMBER_ARRAY object. (They refer to the GEORASTER_TABLE table, whose definition is presented after Example 1-1 in Storage Parameters.). (It refers to the GEORASTER_TABLE table, whose definition is presented after Example 1-1 in Storage Parameters.)

```
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;
/

DECLARE
  gr sdo_georaster;
  data sdo_number_array;
  win sdo_number_array;
BEGIN
  SELECT georaster INTO gr FROM georaster_table WHERE georid=4;
  win := sdo_number_array(-21,100,100,200);
  sdo_geor.getRasterSubset(gr, 0, win, null, data);
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 Storage Parameters.)

```
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;
/
```

7.84 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 = (a₀ + a₁ * cellvalue) / (b₀ + b₁ * 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 Storage Parameters. 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)
```

7.85 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 GeoRaster Metadata XML Schema).

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.
```

7.86 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 Storage Parameters. (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)
```

7.87 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 Storage Parameters. 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
```

7.88 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 Storage Parameters.)

```
SELECT sdo_geor.getSpatialResolutions(georaster) spatialResolution
  FROM georaster_table WHERE georid=42;

SPATIALRESOLUTION
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(28.5, 28.5)
```

7.89 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 Storage Parameters.

```
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
```

7.90 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 Storage Parameters.

```
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
```

7.91 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 SDO_GEOR_SRS Object Type.

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 Storage Parameters.

```
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)
```

7.92 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 GeoRaster Metadata XML Schema. 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 Storage Parameters.)

```
SELECT sdo_geor.getStatistics(georaster, 1) layer1
FROM georaster_table WHERE georid=4;

LAYER1
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0, 255, 100, 127, 95, 25)
```

7.93 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 Bands_ Layers_ and Metadata.

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 Storage Parameters.

```
SELECT georid, sdo_geor.getTotalLayerNumber(georaster) totalLayerNumber
  FROM georaster_table;

    GEORID TOTALLAYERNUMBER
---------- ----------------
         2                1
         4                3
```

7.94 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 Storage Parameters.)

```
SELECT sdo_geor.getULTCoordinate(georaster) FROM georaster_table WHERE georid=23;

SDO_GEOR.GETULTCOORDINATE(GEORASTER)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(256, 0, 0)
```

7.95 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 Geographic Information Systems.

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 Storage Parameters. 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
```

7.96 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 Storage Parameters. (The output is reformatted for
readability.)

```
SELECT georid, sdo_geor.getVersion(georaster) version FROM georaster_table;

    GEORID VERSION
---------- -------------------------------------------------------------------
         2  10.1
         4  9i.2
```

7.97 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 Bitmap Masks.

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;
```

7.98 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. SDO_GEOR_GRAYSCALE Object Type 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 Storage Parameters.)

```
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
```

7.99 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 Bitmap Masks.

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;
```

7.100 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). SDO_GEOR_COLORMAP Object Type 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 Storage Parameters.)

```
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
```

7.101 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
Storage Parameters. Currently, the keywords supported for this operation are:

- `blocking`: (See the explanation in Table 1-1 in Storage Parameters.)

- `blocksize`: (See the explanation in Table 1-1 in Storage Parameters.)

- `compression`: (See the explanation in Table 1-1 in Storage Parameters.) 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 Storage Parameters.)

- `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 and Graph.

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 GeoRaster Tools: Viewer_ Loader_ Exporter, 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 Loading Raster Data.

If you receive an "insufficient memory" error when loading a very large image, see Reformatting the Source Raster Before Loading.

When loading an image into a GeoRaster database, you should always specify a block size, and it should generally be 512x512 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 GeoRaster Tools: Viewer_ Loader_ Exporter, 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 Georeferencing GeoRaster Objects.

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 this procedure is called, the calling user and the MDSYS user 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 users `HERMAN` and `MDSYS`:

```
call dbms_java.grant_permission('HERMAN','SYS:java.io.FilePermission',
  '/mydirectory/myimages/img1.tif', 'read' );
call dbms_java.grant_permission('MDSYS','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. The example grants the necessary permissions at the beginning and revokes them at the end.

```
connect / as sysdba

call dbms_java.grant_permission('HERMAN','SYS:java.io.FilePermission',
  '/mydirectory/myimages/img1.tif', 'read' );
call dbms_java.grant_permission('MDSYS','SYS:java.io.FilePermission',
  '/mydirectory/myimages/img1.tif', 'read' );

connect herman/<password>

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, 'blocking=OPTIMALPADDING,blocksize=(512,512,3)', 'TIFF',
'file',
   '/mydirectory/myimages/img1.tif');
UPDATE georaster_table SET georaster = geor WHERE georid = 1;
COMMIT;
END;/

connect / as sysdba

call dbms_java.revoke_permission('HERMAN','SYS:java.io.FilePermission',
  '/mydirectory/myimages/img1.tif', 'read' );
call dbms_java.revoke_permission('MDSYS','SYS:java.io.FilePermission',
  '/mydirectory/myimages/img1.tif', 'read' );
```

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;
/
```

7.102 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA). GeoRaster operations are described in GeoRaster Database Creation and Management and GeoRaster Data Query and Manipulation.)

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 Storage Parameters.)

```
INSERT INTO georaster_table (georid, georaster)
  VALUES (1, sdo_geor.init('RDT_1'));
```

7.103 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 Storage Parameters.)

```
SELECT georid, substr(sdo_geor.isBlank(georaster),1,7) isBlank
  FROM georaster_table;

    GEORID ISBLANK
---------- -------
         2 FALSE
         4 FALSE
```

7.104 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 Storage Parameters.)

```
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
```

7.105 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 Storage Parameters.)

```
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
```

7.106 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 Storage Parameters.)

```
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
```

The following example searches for all empty and nongeoreferenced GeoRaster objects.

```
SELECT georid FROM georaster_table a
    WHERE sdo_geor.isSpatialReferenced(a.georaster) IS NULL OR
          sdo_geor.isSpatialReferenced(a.georaster) = 'FALSE';
```

7.107 SDO_GEOR.mask

Format

```
SDO_GEOR.mask(
     inGeoRaster   IN SDO_GEORASTER,
     bandNumbers   IN VARCHAR2,
     mask          IN SDO_GEORASTER,
     storageParam  IN VARCHAR2,
     outGeoraster  IN OUT SDO_GEORASTER,
     zeroMapping   IN NUMBER DEFAULT 0,
     oneMapping    IN NUMBER DEFAULT 1,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Applies a mask to specified layers of an existing (input) GeoRaster object. The mask GeoRaster object and the input GeoRaster object can have the same storage format or different storage formats, and you can specify storage format options for the output GeoRaster object (for example, to change the blocking, cell depth, or interleaving).

For information about how to determine the mask value to use, see the Usage Notes.

Parameters

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

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 the second, third, and forth layers).

mask
The SDO_GEORASTER object to be used as a mask on the input GeoRaster object for generating the output GeoRaster object. If this parameter is specified as null, then available attached masks of the input GeoRaster object are applied to the specified layers.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The new SDO_GEORASTER object that reflects the results of the mask operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object.

(Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.)
Cannot be the same GeoRaster object as `inGeoRaster`.
If the output GeoRaster object has any existing raster data, it is deleted before the
mask operation is performed. The output GeoRaster object is overwritten as a result
of this function.

zeroMapping
Value used for mask cell value 0 (zero). The default value is 0.

oneMapping
Value used for mask cell value 1 (one). The default value is 1.

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 Empty Raster Blocks). The number of elements in
the SDO_NUMBER_ARRAY object must be either one (same filling value used for all
layers) or the band dimension size (a different filling value for each layer,
respectively). For example, `SDO_NUMBER_ARRAY(1,5,10)` fills the first layer with 1, the
second layer with 5, and the third layer with 10. The default `bgValues` are zero (0).
The filling values must be valid cell values as specified by the target cell depth
background values for filling sparse data.

Usage Notes

To determine the mask value to use with the `mask` parameter, apply the following logic:

```
If(cellValue_mask==0)
   cellValue_target=cellValue_source * zeroMapping;
else
   cellValue_target=cellValue_source * oneMapping;
```

where:

- `cellValue_source` is the cell value of `inGeoraster` at coordinate (x,y).

- `cellValue_target` is the cell value of `outGeoraster` at coordinate (x,y).

- `cellValue_mask` is the cell value of `mask` at coordinate (x,y).

If `inGeoRaster` is null, no operation is performed.

If pyramid data exists for `inGeoRaster`, then the `mask` GeoRaster object must have at
least the same number of pyramid levels as `inGeoRaster`.

If `mask` is not null, its dimension (row and column) size must be equal to that of
`inGeoRaster`, and `mask` must overlap on `inGeoRaster`. (You can check for overlap using
the SDO_GEOR_RA.isOverlap function.)

If `mask` is null and if no attached mask is available for the specified layers, then
`inGeoRaster` is copied to `outGeoRaster`, which is also modified as specified by any
`storageParam` specifications.

Contrast this function with the SDO_GEOR.setBitmapMask function:
SDO_GEOR.mask calculates cell values in layers and stores them in the target
GeoRaster object, whereas SDO_GEOR.setBitmapMask associates mask data with
specified layers of the source GeoRaster object.

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

- `inGeoRaster` is invalid.

- `outGeoRaster` has not been initialized.

- A raster data table for `outGeoRaster` does not exist and `outGeoRaster` is not a blank GeoRaster object.

Examples

The following example applies mask GeoRaster object `gr2` to the source GeoRaster object `gr1`. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  gr3 sdo_georaster;
BEGIN
  select tmimage into gr1 from landsat where id=103;
  select tmimage into gr3 from landsat where id=1015;
  select grobj into gr2 from grtab where id=1;
  SDO_GEOR.mask(gr1,null,gr2,'blocksize=(100,100,3)',gr3,0.1,0.9,null);
  update landsat set tmimage=gr3 where id=1007;
END;
/
```

The following example applies the attached mask of the source GeoRaster object `gr1` to its second layer.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  gr3 sdo_georaster;
BEGIN
  select tmimage into gr1 from landsat where id=103;
  select tmimage into gr3 from landsat where id=1015;
  gr2:=null;
  SDO_GEOR.mask(gr1,'1',gr2,'blocksize=(100,100,3)',gr3,0.1,0.9,null);
  update landsat set tmimage=gr3 where id=1007;
END;
/
```

7.108 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,
pyramidLevel        IN NUMBER 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 Storage Parameters.

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 Blank and Empty GeoRaster Objects.) 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

pyramidLevel

A number specifying the pyramid level at which the source GeoRaster objects are merged. If not specified, pyramid level 0 is used.

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;
/
```

For an example of using SDO_GEOR.mergLayers to append several layers to an existing GeoRaster object., see the example in Band Merging.

7.109 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 a set of source GeoRaster objects that are rectified, are geospatially aligned under the same SRID, and have the same resolution.

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 Storage Parameters. 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 Empty Raster Blocks), 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 default `bgValues` are zero (0). 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 limited mosaicking capabilities, and works well for preprocessed and perfectly aligned source GeoRaster objects only. It does not work on unrectified rasters and does not support parallel processing. For advanced mosaicking

capabilities, including parallel processing, use the SDO_GEOR_AGGR.mosaicSubset procedure. See Large-Scale Image Mosaicking for more information.

For this procedure, 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 GeoRaster Data Model.)

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 Empty Raster Blocks). 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).

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 Maintaining GeoRaster Objects and System Data in the Database.

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 Storage Parameters. 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, 'blocking=optimalpadding
blocksize=(512,512,1)');
  UPDATE georaster_table SET georaster=gr WHERE id=12;
END;
/
```

7.110 SDO_GEOR.rectify

Format

```
SDO_GEOR.rectify(
     inGeoRaster      IN SDO_GEORASTER,
     pyramidLevel     IN NUMBER,
     elevationParam   IN VARCHAR2,
     DEM              IN SDO_GEORASTER,
     outSRID          IN NUMBER,
     outModelCoordLoc IN NUMBER,
     cropArea         IN SDO_GEOMETRY,
     polygonClip      IN VARCHAR2,
     layerNumbers     IN VARCHAR2,
     outResolutions   IN SDO_NUMBER_ARRAY,
     resolutionUnit   IN VARCHAR2,
     referencePoint   IN SDO_GEOMETRY,
     resampleParam    IN VARCHAR2,
     storageParam     IN VARCHAR2,
     outGeoRaster     IN OUT SDO_GEORASTER,
     bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL,
     parallelParam    IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.rectify(
     inGeoRaster      IN SDO_GEORASTER,
     pyramidLevel     IN NUMBER,
     elevationParam   IN VARCHAR2,
     DEM              IN SDO_GEORASTER,
     outSRID          IN NUMBER,
     outModelCoordLoc IN NUMBER,
     cropArea         IN SDO_GEOMETRY,
     polygonClip      IN VARCHAR2,
```

```
layerNumbers      IN VARCHAR2,
outResolutions    IN SDO_NUMBER_ARRAY,
resolutionUnit    IN VARCHAR2,
referencePoint    IN SDO_GEOMETRY,
resampleParam     IN VARCHAR2,
storageParam      IN VARCHAR2,
rasterBlob        IN OUT NOCOPY BLOB,
outArea           OUT SDO_GEOMETRY,
outWindow         OUT SDO_NUMBER_ARRAY,
bgValues          IN SDO_NUMBER_ARRAY DEFAULT NULL,
parallelParam     IN VARCHAR2 DEFAULT NULL);
```

Description

Perform rectification on all or part of a georeferenced GeoRaster object. The resulting object can be a new GeoRaster object (for persistent storage) or a BLOB (for temporary use). If the input model coordinate system (SRID) is three-dimensional, the average elevation or a Digital Elevation Model (DEM) can be used to perform the orthorectification.

Parameters

inGeoRaster
GeoRaster object on which to perform the operation. It must be georeferenced (see the SDO_GEOR.georeference subprogram).

pyramidLevel
Pyramid level of the source GeoRaster object for the operation.

- For BLOB output, this parameter is required.

- For SDO_GEORASTER output, if this parameter is null and the `storageParam` `pyramid` value is `FALSE`, only the pyramid level 0 is rectified and the output will have only level 0. If this parameter is null and the `storageParam pyramid` value is `TRUE`, all the pyramid levels from the input are rectified.

- If the number 0 or greater is specified, only that pyramid level is used for the rectification, producing a result in scale based on that pyramid level image.

elevationParam
A string containing one or more of the elevation parameters `average` (average surface height), `scale` (scale value applied to all `DEM` values), and `offset` (offset applied to all `DEM` values), where the new value is `(value + offset) * scale`. This parameter must be a quoted string that contains one or more keyword=value pairs (for example, `'average=800 scale=3.2808399 offset=10'`). If this parameter is null, 0 is assumed for `average` and `offset`, and 1 is used for `scale`. Any `scale` and `offset` values are ignored if `DEM` is not specified.
The use of the `elevationParam` parameter requires that the input GeoRaster object have a 3D model SRID.
When the input GeoRaster object has a 3D model SRID, the average elevation is important for defining the extents of the output image. If that information is available, it should be specified even if `DEM` is also specified. If the average elevation is not specified, the procedure will calculate an approximate value for the average elevation.

> **✏️ Note:**
>
> For any numbers in string (VARCHAR2) parameters to GeoRaster subprograms, the period (.) must be used for any decimal points regardless of the locale.

DEM

GeoRaster object with a digital elevation model (DEM); used to perform orthorectification, as explained in the Usage Notes. Must have the same SRID as outGeoRaster. The DEM area should cover the entire `outGeoRaster` area, or the `cropArea` if used. The elevation data is assumed to be on the first layer of the `DEM` GeoRaster object. If this parameter is null, the `elevationParam` value is used. For best results, the resolution of the `DEM` GeoRaster object should be close to the resolution of the input GeoRaster object.

The use of the `DEM` parameter requires that the input GeoRaster object have a 3D model SRID.

When the input GeoRaster object has a 3D model SRID, the average elevation is important for defining the extents of the output image. If that information is available, it should be specified in the `elevationParam` parameter even if `DEM` is also specified. If the average elevation is not specified, the procedure will calculate an approximate value for the average elevation.

outSRID

Coordinate system for the output GeoRaster object. Must be either null or a value from the SRID column of the MDSYS.CS_SRS table. If it is null, the output GeoRaster object will have the same SRID as the input GeoRaster object.

outModelCoordLoc

A value specifying the model location of the base of the area represented by a cell: `0` for `CENTER` or `1` for `UPPERLEFT`. If null, `CENTER` is used.

cropArea

Crop area definition. If null, no cropping is performed, and `polygonClip` is ignored. If `polygonClip` is `FALSE`, the MBR of the `cropArea` is used to crop the output image. If `polygonClip` is `TRUE`, the geometry of the `cropArea` is used to crop the output image. Areas outside the crop area are filled with the background value

polygonClip

Ignored if `cropArea` is null. Otherwise, the string `TRUE` causes the `cropArea` value to be used to crop the mosaicked data; the string `FALSE` or a null value causes the MBR of `cropArea` to be used to crop the output image.

layerNumbers

A string identifying the logical layer numbers for the rectification and the output to `outGeoRaster`. Use commas to delimit the values, and a hyphen to indicate a range (for example, `2-4` for layers 2, 3, and 4).

outResolutions

An array of numeric values, one for each spatial dimension. Each value indicates the number of units of measure 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 measure for the ground data is meters, each pixel represents an area of 10

meters by 10 meters. If null, the default is the resolution of the source data at the specified pyramid level.

resolutionUnit
The unit of the `outResolutions` parameter. If `resolutionUnit` is different from the `outGeoRaster` SRID unit, an appropriate conversion is computed (the value of the output resolution is calculated by converting the `outResolutions` value in `resolutionUnit` to the unit of the output SRID) . If `resolutionUnit` is null, the default is the unit of the output SRID. If specified, it must be a quoted string in the format "unit=*value*" where *value* is the unit name. This parameter is ignored if `outResolutions` is null.

referencePoint
A point of type SDO_GEOMETRY indicating a reference to where the `outGeoRaster` object should be aligned so that the distance between the `referencePoint` and the upper-left corner of the output will have an integer number of pixels.

resampleParam
A comma-separated quoted string of *keyword=value* pairs for specifying resampling parameters. See the Usage Notes for more information.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
GeoRaster object to hold the result of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`

rasterBlob
BLOB to hold the output reflecting the rectification. It must exist or have been initialized before the 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of

parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, the procedure performs an internal commit operation. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

This procedure has two formats:

- One format generates a GeoRaster object for persistent storage in the database.

- The other format generates a BLOB for temporary storage or immediate use, such as to display data on the screen.

This procedure uses a non-parametric rectification method that takes the georeferencing polynomials from the input GeoRaster object to transform the original image space into the target image space. Therefore, the input GeoRaster object must be georeferenced (see the SDO_GEOR.georeference subprogram).

Orthorectification can be performed if the input GeoRaster object has a 3D model SRID. A digital elevation model (DEM) GeoRaster object can be used to improve the accuracy of the orthorectification. If the DEM parameter is not specified, the elevationParam average value is used as the height for the whole target area. If the elevationParam average value is not specified, it is estimated based on the SRS and DEM information (see Image Orthorectification).

resampleParam, if specified, 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, BILINEAR, BIQUADRATIC, CUBIC, AVERAGE4, or AVERAGE16. For more information, see Resampling and Interpolation.

- 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, BIQUADRATIC, CUBIC, AVERAGE4, or AVERAGE16, 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

In the following example, the input GeoRaster object is rectified to the same SRID. The input GeoRaster object was georeferenced using GCPs with SRID 32619.

The layerNumbers parameter indicates the order of selection of three bands of the seven bands from the input GeoRaster object, producing a three-band output GeoRaster object.

```
DECLARE
  gr1    sdo_georaster;
  gr2    sdo_georaster;
BEGIN
  select raster into gr1 from georaster_table where georid = 1;
  insert into georaster_table values(2, 'Rectified image',
         sdo_geor.init('georaster_rdt')) returning raster into gr2;
```

```
        sdo_geor.rectify(inGeoRaster      => gr1,
                         pyramidLevel     => null,
                         elevationParam   => null,
                         dem              => null,
                         outSRID          => null,
                         outModelCoordLoc => null,
                         cropArea         => null,
                         polygonClip      => null,
                         layerNumbers     => '2,4,5',
                         outResolutions   => null,
                         resolutionUnit   => null,
                         referencePoint   => null,
                         resampleParam    => null,
                         storageParam     => null,
                         outGeoraster     => gr2);
  update georaster_table set georaster = gr2 where georid = 2;
  commit;
END;
```

In the following example, the input GeoRaster object was georeferenced using 3D
GCPs with SRID 32619, and the function produces an orthorectified GeoRaster with
SRID 4326.

- The `dem` parameter specifies the GeoRaster object `gr3` as the digital elevation
  model for providing height values for each pixel for the orthorectification.

- `resampleParam` specifies the resampling method as `BILINEAR`.

- The `storageParam` parameter specifies the `interleaving` as `BSQ` and the `compression`
  as `DEFLATE`.

- The specified `outResolutions` value has the same unit of measurement as the
  output GeoRaster SRID because `resolutionUnit` is null.

- The point geometry specified by `referencePoint` causes the output image upper-
  left corner to be aligned with that coordinate, with a integer number of pixel (rows
  and columns resolution) in between them.

```
DECLARE
  gr1    sdo_georaster;
  gr2    sdo_georaster;
  gr3    sdo_georaster;
  pto    sdo_geometry;
BEGIN
  pto := sdo_geometry(2001, 4326, sdo_point_type(-71.50,42.0, null));
  select raster into gr1 from georaster_table where georid = 1;
  select raster into gr1 from georaster_table where georid = 3;
  insert into georaster_table values(2, 'Rectified image',
        sdo_geor.init('georaster_rdt')) returning raster into gr2;
  sdo_geor.rectify(inGeoRaster      => gr1,
                   pyramidLevel     => 1,
                   elevationParam   => null,
                   dem              => gr3,
                   outSRID          => 4326,
                   outModelCoordLoc => null,
                   cropArea         => null,
                   polygonClip      => null,
                   layerNumbers     => null,
                   outResolutions   => sdo_number_array(0.0025,0.0025),
                   resolutionUnit   => null,
                   referencePoint   => pto,
                   resampleParam    => 'resampling=BILINEAR',
```

```
                    storageParam      => 'interleaving=BSQ compress=DEFLATE',
                    outGeoraster      => gr2);
  update georaster_table set georaster = gr2 where georid = 2;
  commit;
END;
```

In the following example, the input GeoRaster object was georeferenced using GCPs with SRID 32619, and the output GeoRaster object is projected to SRID 4326.

- resampleParam specifies the resampling method as CUBIC.

- The storageParam parameter specifies blockSize as (512,512,3). Because interleaving is not specified, the interleaving method for inGeoRaster is used.

- outResolutions and resolutionUnit are specified in meters, which is a different unit from outSRID 4316. In this case, the SDO_NUMBER_ARRAY values (30,30) are converted to degrees.

- The SDO_GEOMETRY polygon specified for cropArea is used to crop the output area to the extents of that polygon; and because polygonClip is TRUE, the area of the image outside of the polygon is set to background values.

- The model coordinate location of the output object is UpperLeft because outModelCoordLoc is specified as 1.

```
DECLARE
  gr1    sdo_georaster;
  gr2    sdo_georaster;
  pol    sdo_geometry;
BEGIN
  pol := sdo_geometry(2003,4326,NULL,sdo_elem_info_array(1,1003,1),
          sdo_ordinate_array(-70.869495075803073, 42.349420282160885,
                             -70.468523716196913, 42.813138293441916,
                             -70.957334345349082, 43.218053058782452,
                             -71.350984405166344, 42.736563729419181,
                             -70.869495075803073, 42.349420282160885));
  select raster into gr1 from georaster_table where georid = 1;
  insert into georaster_table values(2, 'Rectified image',
        sdo_geor.init('georaster_rdt')) returning raster into gr2;
  sdo_geor.rectify(inGeoRaster       => gr1,
                   pyramidLevel      => 0,
                   elevationParam    => null,
                   dem               => null,
                   outSRID           => 4326,
                   outModelCoordLoc  => 1,
                   cropArea          => pol,
                   polygonClip       => 'true',
                   layerNumbers      => null,
                   outResolutions    => sdo_number_array(30,30),
                   resolutionUnit    => 'unit=meter',
                   referencePoint    => null,
                   resampleParam     => 'resampling=CUBIC',
                   storageParam      => 'blocking=optimalpadding
blockSize=(512,512,3)',
                   outGeoraster      => gr2);
  update georaster_table set georaster = gr2 where georid = 2;
  commit;
END;
```

7.111 SDO_GEOR.reproject

Format

```
SDO_GEOR.reproject(
     inGeoRaster     IN SDO_GEORASTER,
     resampleParam   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,
     resampleParam   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,
     resampleParam   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,
     resampleParam   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,
resampleParam    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 and Graph 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 Storage Parameters.

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 Blank and Empty GeoRaster Objects.) 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). 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 GeoRaster Data Model.)

If the `cropArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian. (For information about geodetic MBRs, see *Oracle Spatial and Graph 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`, `BILINEAR`, `CUBIC`, `AVERAGE4`, `AVERAGE16`. For more information, see Resampling and Interpolation.

- `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`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, 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, 0, 0, SDO_NUMBER_ARRAY(0, 0, 517, 517),
                           null, null, 'blocking=optimalpadding,
blocksize=(512,512,3),
                           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, 0, SDO_NUMBER_ARRAY(0, 0, 120, 300),
                           '0',  null, 'compression = JPEG-F', 32618,
                           lob1, outArea, outWindow);
```

```
      dbms_lob.freeTemporary(lob1);
      COMMIT;
END;
/
```

7.112 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:

> **Note:**
>
> For any numbers in string (VARCHAR2) parameters to GeoRaster subprograms, the period (.) must be used for any decimal points regardless of the locale.

- `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.

- `rowMaxDimSize` and `columnMaxDimSize`, to specify sizes in terms of the maximum number of cells for row and column dimensions. This pair of keywords must have numeric values for each dimension (for example, `'rowMaxDimSize=512,columnMaxDimSize=256'`). The aspect ratio can be changed, and the two keywords must be specified together.

- `rowScaleFactor` and `columnScaleFactor`, to reduce or enlarge as a multiple of the original size. This pair of keywords must have numeric values greater than 0 (zero). 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 row dimension is the original number multiplied by `rowScaleFactor`. The number of cells along column dimension is the original number multiplied by `columnScaleFactor`. `rowScaleFactor` and `columnScaleFactor` can be different numbers, but must be specified together.

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 Storage Parameters.

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 Blank and Empty GeoRaster Objects.) 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). 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`, `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, `AVERAGE16`. For more information, see Resampling and Interpolation.

- `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`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, 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 an optimized block size around 512 for each dimension in the storage parameters. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
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',
                     'blocking=optimalpadding blocksize=(512,512)', gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=21;
  COMMIT;
END;
/
```

7.113 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;
```

7.114 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 Storage Parameters.)

```
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;
/
```

7.115 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;
/
```

7.116 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 Storage Parameters.

```
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;
/
```

7.117 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 whether `mask` is to be interpreted as a NODATA mask. The string `TRUE` causes `mask` to be interpreted as a NODATA mask; any other value (including null) causes `mask` *not* 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 Bitmap Masks.

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;
/
```

7.118 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 Storage Parameters.)

```
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;
/
```

7.119 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 SDO_GEOR_COLORMAP Object Type.

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 Storage Parameters.)

```
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;
/
```

7.120 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. SDO_GEOR_COLORMAP Object Type
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 Storage Parameters.)

```
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;
/
```

7.121 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 SDO_GEOR_GCP Object Type.

Usage Notes

For an explanation of georeferencing using GCPs, see Ground Control Point (GCP) Georeferencing Model.

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;
/
```

7.122 SDO_GEOR.setDefaultAlpha

Format

```
SDO_GEOR.setDefaultAlpha(
     georaster      IN OUT SDO_GEORASTER,
     defaultAlpha  IN NUMBER);
```

Description

Sets the number of the layer to be used for the alpha color component (in the RGBA color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultAlpha` parameter.

Parameters

georaster
GeoRaster object.

defaultAlpha
Number of the layer to be used for the alpha color component (in the RGBA 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, blue, and alpha 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 the number of the layer to be used for the alpha color component only, or if `defaultAlpha` is not a valid layer number for the GeoRaster object.

Examples

The following example sets the default red, green, blue, and alpha 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, blue, and alpha color components for displaying these GeoRaster objects. (The GEORASTER_TABLE table definition is presented after Example 1-1 in Storage Parameters.)

```
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);
  sdo_geor.setDefaultAlpha(grobj, 2);
  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, 2)

1 row selected.
```

7.123 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 Storage Parameters.)

```
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.
```

7.124 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, blue, and alpha 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 or four numbers identifying the red, green, blue, and alpha 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 RGBA 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, SDO_GEOR.setDefaultBlue, and SDO_GEOR.setDefaultAlpha procedures.

The default RGBA layer numbers must be set or removed at the same time. The default alpha layer number is optional. If the default red, green, and blue layer numbers are set, the default alpha layer number can be set. If the default red, green, and blue layer numbers are removed, the default alpha layer number must also be removed.

Because the default alpha layer is optional, you can either (A) set the default red, green, and blue color components only by providing three numbers to this procedure or (B) set the default red, green, blue, and alpha color components by providing four numbers to this procedure. If `defaultRGB` is an array of three numbers, it identifies the red, green and blue color components only. If `defaultRGB` is an array of four numbers, it identifies the red, green, blue, and alpha components.

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, blue. and alpha 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 Storage Parameters.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=2 FOR UPDATE;
  sdo_geor.setDefaultColorLayer(grobj, sdo_number_array(1,1,1,1));
  UPDATE georaster_table SET georaster = grobj WHERE georid=2;
```

```
    COMMIT;
END;
/
```

7.125 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 Storage Parameters.)

```
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.
```

7.126 SDO_GEOR.setDefaultPyramidLevel

Format

```
SDO_GEOR.setDefaultPyramidLevel(
     georaster          IN OUT SDO_GEORASTER,
     defaultPyramidLevel  IN NUMBER);
```

Description

Sets the number of the layer to be used for the default pyramid level for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultPyramidLevel` parameter.

Parameters

georaster
GeoRaster object.

defaultPyramidLevel
The default pyramid level for displaying the specified GeoRaster object. Must be greater than or equal to 0 (zero) and less than or equal to the maximum pyramid level in the GeoRaster object.

Usage Notes

Pyramid levels represent reduced or increased resolution images that require less or more storage space, respectively. For information about pyramids and pyramid levels, see Pyramids.

Specifying a default pyramid level is an optional operation, and is intended for use only when visualizing GeoRaster objects with pyramids generated.

The GeoRaster object is automatically validated after the operation completes.

When pyramids are removed from a GeoRaster object by any other operation (such as SDO_GEOR.deletePyramid or SDO_GEOR.subset), the default pyramid level for the object is also removed from the metadata.

An exception is raised if there are no pyramids generated for the GeoRaster object, or if `defaultPyramidLevel` is not a valid pyramid level number for the GeoRaster object.

You can get the default pyramid level by using the SDO_GEOR.getDefaultPyramidLevel function.

Examples

The following example generates the pyramids and sets the default pyramid level for a specified GeoRaster object (GEORASTER column) in the GEORASTER_TABLE table, and it returns the default pyramid level set for the GeoRaster object. (The GEORASTER_TABLE table definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT georaster INTO grobj FROM georaster_table WHERE georid=6 FOR UPDATE;
  sdo_geor.generatePyramid(grobj, 'rLevel=5, resampling=NN');
  sdo_geor.setDefaultPyramidLevel(grobj, 3);
  UPDATE georaster_table SET georaster = grobj WHERE georid=6;
  COMMIT;
END;
/

SELECT georid, sdo_geor.getDefaultPyramidLevel(georaster) FROM georaster_table
  WHERE georid=6;

       GEORID    PLEVEL
-------------- ----------
            6         3
```

7.127 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 Storage Parameters.)

```
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.
```

7.128 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 Storage Parameters.)

```
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;
/
```

7.129 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 Ground Control Point (GCP) Georeferencing Model.

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;
/
```

7.130 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 Ground Control Point (GCP) Georeferencing Model.

The SDO_GEOR_GCPGEOREFTYPE object type is defined in SDO_GEOR_GCPGEOREFTYPE Object Type.

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;
         /
```

7.131 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 SDO_GEOR_GRAYSCALE Object Type.

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 Storage Parameters.)

```
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;
/
```

7.132 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

SDO_GEOR_GRAYSCALE Object Type 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 Storage Parameters.)

```
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;
        /
```

7.133 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. SDO_GEOR_HISTOGRAM Object Type 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 Storage Parameters.)

```
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;
/
```

7.134 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 Storage Parameters.)

```
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;
/
```

7.135 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 Storage Parameters.)

```
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;
/
```

7.136 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 Bands_ Layers_ and Metadata: 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 Storage Parameters.)

```
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;
/
```

7.137 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 Ground Control Point (GCP) Georeferencing Model.

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;
/
```

7.138 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 and it does not adjust the georeferencing coefficients accordingly. In other words, this procedure does not cause any reprojection or resampling on the cell data of the GeoRaster object, and you must specify the correct SRID.

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 Storage Parameters.)

```
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;
/
```

7.139 SDO_GEOR.setNODATAMask

Format

```
SDO_GEOR.setNODATAMask(
     georaster     IN OUT SDO_GEORASTER,
     layerNumber   IN NUMBER,
     isNODATAMask  IN VARCHAR2);
```

Description

Specifies whether a bitmap mask of the specified GeoRaster layer is a NODATA mask, and updates the GeoRaster metadata accordingly.

Parameters

georaster
GeoRaster object.

layerNumber
Layer with the relevant 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.

isNODATAMask
Specifies whether the bitmap mask of the `layerNumber` layer is to be interpreted as a NODATA mask. The string `TRUE` causes that bitmap mask to be interpreted as a NODATA mask; the string `FALSE` causes that bitmap mask *not* to be interpreted as a NODATA mask.

Usage Notes

For information about a bitmap mask being treated as a special type of NODATA, that is, a NODATA mask specifying one or more irregular areas as NODATA areas, see NODATA Values and Value Ranges.

Examples

The following example sets the bitmap mask of sublayer 1 to be a NODATA mask.

```
declare
    gr sdo_georaster;
begin
    select georaster into gr from georaster_table where georid=1 for update;
    sdo_geor.setNODATAMask(gr, 1, 'true');
     update georaster_table set georster=gr where georid=1;
     commit;
end;
```

7.140 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 Storage Parameters.)

```
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;
/
```

7.141 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 rasterType Attribute.

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 Storage Parameters.)

```
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;
/
```

7.142 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. (To rectify or orthorectify a GeoRaster object, you can use the SDO_GEOR.rectify procedure.)

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 Storage Parameters.)

```
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;
/
```

7.143 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 Storage Parameters.)

```
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;
/
```

7.144 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 GeoRaster Metadata XML Schema).

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.
```

7.145 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 Storage Parameters.)

```
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;
/
```

7.146 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 Storage Parameters.)

```
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;
/
```

7.147 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 Storage Parameters.)

```
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;
/
```

7.148 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 Storage Parameters.)

```
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;
/
```

7.149 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 SDO_GEOR_SRS Object Type.
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 Storage Parameters.) 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,
                    0,0,0,0,0,1,1,1,1,1,1,0,0,0,
                    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 Functional Fitting Georeferencing Model, 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;
/
```

7.150 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. SDO_GEOR_HISTOGRAM Object Type 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 GeoRaster Metadata XML Schema.

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 Storage Parameters.)

```
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;
/
```

7.151 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 GeoRaster Data Model.

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 Storage Parameters.)

```
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;
/
```

ORACLE®

7.152 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 Geographic Information Systems.

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 Storage Parameters.)

```
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;
/
```

7.153 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 Storage Parameters.)

```
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;
/
```

ORACLE

7.154 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 Storage Parameters.

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 Blank and Empty GeoRaster Objects.) 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 Empty Raster Blocks). 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 default `bgValues` are zero (0). 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 GeoRaster Data Model.)

If the `cropArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial and Graph 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 and Graph 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 Storage Parameters.)

```
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 Storage Parameters.)

```
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;
/
```

7.155 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.

For more information about using this parameter, see Image Pyramiding: Parallel Generation and Partial Update.

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.) For more information about using this parameter, see Image Pyramiding: Parallel Generation and Partial Update.

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 Empty Raster Blocks). 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 default `bgValues` are zero (0).

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 GeoRaster Data Model.)

If the `targetArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian. For information about geodetic MBRs, see *Oracle Spatial and Graph 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;
/
```

7.156 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 blockMBR Attribute) 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 Creating New GeoRaster Objects, 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
```

7.157 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 Pyramids.)
- 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 Functional Fitting Georeferencing Model. The limitations on offsets, scales, RMS values, pType, nVars, and number of coefficients of the polynomials are described in Functional Fitting Georeferencing Model and Table 2-4 in SDO_GEOR_SRS Object Type.

- 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 and Graph 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 SDO_GEOR_GCPGEOREFTYPE Object Type, and each GCP should follow the specification of the SDO_GEOR_GCP type as described in SDO_GEOR_GCP Object Type. 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 and Graph 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 blockMBR Attribute.) 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)), 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
```

7.158 SDO_GEOR.warp

Format

```
SDO_GEOR.warp(
     inGeoRaster      IN SDO_GEORASTER,
     pyramidLevel     IN NUMBER,
     outSRS           IN NUMBER,
```

```
        cropArea        IN SDO_GEOMETRY,
        dimensionSize   IN SDO_NUMBER_ARRAY,
        layerNumbers    IN VARCHAR2,
        elevationParam  IN VARCHAR2,
        resampleParam   IN VARCHAR2,
        storageParam    IN VARCHAR2,
        outGeoRaster    IN OUT SDO_GEORASTER,
        bgValues        IN SDO_NUMBER_ARRAY DEFAULT NULL,
        parallelParam   IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.warp(
        inGeoRaster     IN SDO_GEORASTER,
        pyramidLevel    IN NUMBER,
        outSRS          IN NUMBER,
        cropArea        IN SDO_GEOMETRY,
        dimensionSize   IN SDO_NUMBER_ARRAY,
        layerNumbers    IN VARCHAR2,
        elevationParam  IN VARCHAR2,
        resampleParam   IN VARCHAR2,
        storageParam    IN VARCHAR2,
        rasterBlob      IN OUT NOCOPY BLOB,
        outArea         OUT SDO_GEOMETRY,
        outWindow       OUT SDO_NUMBER_ARRAY,
        bgValues        IN SDO_NUMBER_ARRAY DEFAULT NULL,
        parallelParam   IN VARCHAR2 DEFAULT NULL);
```

Description

Perform geometric transformation on the input GeoRaster object to produce an output GeoRaster object with the specified output spatial reference system.

Parameters

inGeoRaster
GeoRaster object on which to perform the operation. It must be georeferenced (see Georeferencing GeoRaster Objects and Advanced Georeferencing).

pyramidLevel
Pyramid level of the source GeoRaster object for the operation.

- For BLOB output, this parameter is required.

- For SDO_GEORASTER output, if this parameter is null, all pyramid levels from the input GeoRaster object are processed.

- If the number 0 or greater is specified, only that pyramid level is used for the rectification, producing a result in scale based on that pyramid level image.

outSRS
Coordinate system (spatial reference system) for the output GeoRaster object. Must be either null or a value from the SRID column of the MDSYS.CS_SRS table. If it is null, the output GeoRaster object will have the same SRID as the input GeoRaster object.

cropArea

Defines the shape of the area to be covered by the output image. Areas outside this polygon will be filled with the background value. If this parameter is not specified, no cropping is performed.

dimensionSize

Dimension size array of the GeoRaster object. Defines the extent of the image in cell space.

layerNumbers

String specifying one or more numbers of layers from `inGeoRaster` to be transferred to `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. If this parameter is null (the default), all the layers will be processed.

elevationParam

A string containing one or more of the elevation parameters `average` (average surface height), `scale` (scale value applied to all `DEM` values), and `offset` (offset applied to all `DEM` values), where the new value is `(value + offset) * scale`. This parameter must be a quoted string that contains one or more keyword=value pairs (for example, `'average=800 scale=3.2808399 offset=10'`). If this parameter is null, 0 is assumed for `average` and `offset`, and 1 is used for `scale`. Any `scale` and `offset` values are ignored if `DEM` is not specified.

The use of the `elevationParam` parameter requires that the input GeoRaster object have a 3D model SRID.

When the input GeoRaster object has a 3D model SRID, the average elevation is important for defining the extents of the output image. If that information is available, it should be specified even if `DEM` is also specified. If the average elevation is not specified, the procedure will calculate an approximate value for the average elevation.

> **✎ Note:**
>
> For any numbers in string (VARCHAR2) parameters to GeoRaster subprograms, the period (.) must be used for any decimal points regardless of the locale.

resampleParam

A comma-separated quoted string of *keyword=value* pairs for specifying resampling parameters. See the Usage Notes for more information.

storaageParam

A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster

GeoRaster object to hold the result of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`

rasterBlob

BLOB to hold the output reflecting the rectification. It must exist or have been initialized before the 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 Empty Raster Blocks). 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 default bgValues are zero (0). The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form parallel=*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, the procedure performs an internal commit operation. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

This procedure has two formats:

- One format generates a GeoRaster object for persistent storage in the database.

- The other format generates a BLOB for temporary storage or immediate use, such as to display data on the screen.

This procedure uses a non-parametric rectification method that takes the georeferencing polynomials from the input GeoRaster object to transform the original image space into the georeferencing polynomials given by the outSRS parameter. Therefore, the input GeoRaster object must be georeferenced (see the SDO_GEOR.georeference subprogram).

For more information, see Image Warping.

Examples

In the following example, the output (generated) GeoRaster object will have the same spatial reference system (coordinate system) as outSRS. The input GeoRaster object is a fully georeferenced image.

```
DECLARE
  srs sdo_geor_srs;
  gr2 sdo_georaster;
  gr3 sdo_georaster;
  gr4 sdo_georaster;
BEGIN
  select georaster into gr2 from georaster_table where georid = 2;
  srs := sdo_geor.getSRS(gr2);
```

```
      select georaster into gr3 from georaster_table where georid = 3;

      insert into georaster_table values(4, 'Warped',
            sdo_geor.init('warp_rdt',4)) returning raster into gr4;

      sdo_geor.warp( inGeoRaster      => gr3,
                     pyramidLevel     => null,
                     outSRS           => srs,
                     cropArea         => null,
                     dimensionSize    => sdo_number_array(518,518),
                     layerNumbers     => '4,5,3',
                     elevationParam   => `average=300`,
                     resampleParam    => 'resampling=AVERAGE4',
                     storageParam     => 'pyramid=true',
                     outGeoRaster     => gr4,
                     bgValues         => sdo_number_array(0,0,0),
                     parallelParam    => 'parellel=4' );

   update georaster_table set georaster = gr4 where georid = 4;
   commit;
END;
```

8

SDO_GEOR_ADMIN Package Reference

The 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.

Topics:

8.1 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA).

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!
```

8.2 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
```

8.3 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
```

8.4 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')
```

8.5 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 in the SYSDATA table) or unregistered (the GeoRaster object has not been registered in 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')
```

8.6 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')
```

8.7 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')
```

8.8 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')
```

8.9 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')
```

8.10 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')
```

8.11 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 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA).

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.
```

8.12 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;
```

8.13 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;
```

8.14 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.

```
DECLARE
  ret SDO_STRING2_ARRAY;
BEGIN
  ret:=sdo_geor_admin.upgradeGeoraster;
  for i in 1..ret.count loop
    dbms_output.put_line(ret(i));
  end loop;
END;
/
```

9

SDO_GEOR_AGGR Package Reference

The SDO_GEOR_AGGR package contains subprograms (functions and procedures) for performing advanced large-scale mosaicking, appending, and virtual mosaic operations on GeoRaster objects. This chapter presents reference information, with one or more examples, for each subprogram.

Topics:

- SDO_GEOR_AGGR.append
- SDO_GEOR_AGGR.getMosaicExtent
- SDO_GEOR_AGGR.getMosaicResolutions
- SDO_GEOR_AGGR.getMosaicSubset
- SDO_GEOR_AGGR.mosaicSubset
- SDO_GEOR_AGGR.validateForMosaicSubset

9.1 SDO_GEOR_AGGR.append

Format

```
SDO_GEOR_AGGR.append(
     targetGeoRaster    IN OUT SDO_GEORASTER,
     sourceGeoRaster    IN SDO_GEORASTER,
     sourcePyramidLevel IN NUMBER,
     appendParam        IN VARCHAR2,
     bgValues           IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Appends the source GeoRaster object to the target GeoRaster object. Internal rectification, common point rules, gap filling, and color balancing are performed whenever necessary.

Parameters

targetGeoRaster
GeoRaster object to be updated. Cannot be the same as `sourceGeoRaster`. (Be sure to make a copy of this object before you update it.)

sourceGeoRaster
GeoRaster object to be appended to `targetGeoRaster`.

sourcePyramidLevel
Pyramid level of the source GeoRaster object to be appended at the target GeoRaster object pyramid Level 0. If NULL, pyramid level 0 is used.

appendParam

A comma-separated quoted string of *keyword=value* pairs for specifying parameters for the operation. It can contain one or more of the keywords in Table 9-1 in the SDO_GEOR_AGGR.mosaicSubset reference section, unless specified in that table that the keyword is always ignored for SDO_GEOR_AGGR.append.

If a nondefault value for `colorBalance` is specified, it is performed on the source GeoRaster object using the target GeoRaster object's statistics as the reference, and the following keywords (if specified) are ignored: `maxVal`, `minVal`, `std`, and `min`.

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 Empty Raster Blocks). 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 default `bgValues` are zero (0). The filling values must be valid cell values as specified by the target cell depth.

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 source and target GeoRaster objects must have the same number of bands or layers. If the cell depths of the source and target GeoRaster objects are not the same, the cell value of the source GeoRaster object is either expanded or truncated to the cell depth of the target GeoRaster object.

The target GeoRaster object's raster data must be blocked.

There is no change on metadata of target GeoRaster object, except that the extent and the number of blocks are updated and the statistics are removed.

The source GeoRaster object is appended to the target GeoRaster object pyramid level 0. The pyramids of the target GeoRaster object are also updated.

The overlapping areas and gaps of the source and target GeoRaster objects are resolved according to the rules defined in the `appendParam` parameter.

The source GeoRaster object can be located on or touching any side of the target GeoRaster object (that is, it does not have to be on the right or bottom side). The target GeoRaster object will be automatically expanded accordingly.

Examples

The following example appends the GeoRaster object with `georid` = 2 to the GeoRaster object with at `georid` = 1.

```
declare
    gr1 sdo_georaster;
    gr1 sdo_georaster;
```

```
begin
      select georaster into gr1 from georaster_table where georid = 1 for update;
      select georaster into gr2 from georaster_table where georid = 2;
      sdo_geor_aggr.append(gr1, gr2, 0, null);
      update georaster_table set georaster = gr1 where georid= 1;
      commit;
end;
/
```

9.2 SDO_GEOR_AGGR.getMosaicExtent

Format

```
SDO_GEOR_AGGR.getMosaicExtent(
      inGeoRasters IN SYS_REFCURSOR,
      outSRID       IN NUMBER DEFAULT NULL
      ) RETURN SDO_GEOMETRY;
```

or

```
SDO_GEOR_AGGR.getMosaicExtent(
      georasterTableNames   IN VARCHAR2,
      georasterColumnNames  IN VARCHAR2,
      outSRID               IN NUMBER DEFAULT NULL
      ) RETURN SDO_GEOMETRY;
```

Description

Calculates and returns the minimum bounding rectangle (MBR) of the spatial extent of
a virtual mosaic or a collection of GeoRaster objects.

Parameters

inGeoRasters
Source GeoRaster objects in a cursor.

georasterTableNames
Names (comma-separated) of the tables containing the source GeoRaster objects.

georasterColumnNames
Names (comma-separated) of the columns of type SDO_GEORASTER in tables
corresponding to the table names in georasterTableNames.

outSRID
Coordinate system for the output GeoRaster object. Must be either null or a value
from the SRID column of the MDSYS.CS_SRS table. If this parameter is null (the
default), 4326 (EPSG SRID value for the WGS 84 (longitude/latitude) coordinate
system) is used.

Usage Notes

In calculating the spatial extent of a virtual mosaic or a collection of GeoRaster
objects, this function tries to use the spatialExtent attribute of each GeoRaster object.
If the spatialExtent attribute is null, the extent of the GeoRaster object is calculated
based on the object's metadata.

Examples

The following example shows how to get the spatial extent when the virtual mosaic or the collection of GeoRaster objects is a cursor.

```
declare
    cur sys_refcursor,
begin
    open cur for select georaster from georaster_table_1 union all select georaster
from georaster_table_2;
    extent := sdo_geor_aggr.getMosaicExtent(cur, 26986);
    close cur;
end;
/
```

The following example shows how to get the mosaic extent by specifying the table column names.

```
select sdo_geor_aggr.getMosaicExtent('georaster_table_1, georaster_table_2',
'georaster, georaster', 26986) from dual;
```

9.3 SDO_GEOR_AGGR.getMosaicResolutions

Format

```
SDO_GEOR_AGGR.getMosaicResolutions(
    inGeoRasters     IN SYS_REFCURSOR,
     resolutionUnit IN VARCHAR2 DEFAULT NULL
     ) RETURN  SDO_RANGE_ARRAY;;
```

or

```
SDO_GEOR_AGGR.getMosaicResolutions(
    georasterTableNames  IN VARCHAR2,
    georasterColumnNames IN VARCHAR2,
     resolutionUnit      IN VARCHAR2 DEFAULT NULL
     ) RETURN  SDO_RANGE_ARRAY;
```

Description

Returns the resolution range of a virtual mosaic or a collection of GeoRaster objects in a specified unit.

Parameters

inGeoRasters
Source GeoRaster objects in a cursor.

georasterTableNames
Names (comma-separated) of the tables containing the source GeoRaster objects.

georasterColumnNames
Names (comma-separated) of the columns of type SDO_GEORASTER in tables corresponding to the table names in georasterTableNames.

resolutionUnit

Unit of measure for the returned resolution range. If specified, it must be a quoted string in the format `'unit=value'` where *value* is the unit name value (a valid UNIT_OF_MEAS_NAME value from the SDO_UNITS_OF_MEASURE table). If not specified or null, the returned resolution range is in the unit of `meter`.

Usage Notes

The returned resolution range is in the format of SDO_RANGE_ARRAY(SDO_RANGE(`min_x`, `max_x`), SDO_RANGE(`min_y`, `max_y`)). where `min_x`, `max_x` are the minimum and maximum resolution on the x dimension, and `min_y`, `max_y` are the minimum and maximum resolution on the y dimension.

The SDO_RANGE_ARRAY type is defined as `VARRAY(1048576) OF SDO_RANGE`. The SDO_RANGE type is defined as:

```
Name                                     Null?    Type
---------------------------------------- -------- -------
LB                                                NUMBER
UB                                                NUMBER
```

Examples

The following example gets the spatial resolution of a virtual mosaic or a collection of GeoRaster objects with the returned values in meters.

```
SELECT sdo_geor_aggr.getMosaicResolutions('georatser_table_1, georaster_table_2',
'georaster, georaster', 'unit=meter') FROM DUAL;
```

9.4 SDO_GEOR_AGGR.getMosaicSubset

Format

```
SDO_GEOR_AGGR.getMosaicSubset(
     inGeoRasters     IN SYS_REFCURSOR,
     pyramidLevel     IN NUMBER,
     outSRID          IN NUMBER,
     outModelCoordLoc IN NUMBER,
     referencePoint   IN SDO_GEOMETRY,
     cropArea         IN SDO_GEOMETRY,
     polygonClip      IN VARCHAR2,
     boundaryClip     IN VARCHAR2,
     layerNumbers     IN VARCHAR2,
     outResolutions   IN SDO_NUMBER_ARRAY,
     resolutionUnit   IN VARCHAR2,
     mosaicParam      IN VARCHAR2,
     rasterBlob       IN OUT NOCOPY BLOB,
     outArea          OUT SDO_GEOMETRY,
     outWindow        OUT SDO_NUMBER_ARRAY,
     storageParam     IN VARCHAR2 DEFAULT NULL,
     bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.getMosaicSubset(
     georasterTableNames  IN VARCHAR2,
     georasterColumnNames IN VARCHAR2,
     pyramidLevel         IN NUMBER,
     outSRID              IN NUMBER,
```

```
    outModelCoordLoc    IN NUMBER,
    referencePoint      IN SDO_GEOMETRY,
    cropArea            IN SDO_GEOMETRY,
    polygonClip         IN VARCHAR2,
    boundaryClip        IN VARCHAR2,
    layerNumbers        IN VARCHAR2,
    outResolutions      IN SDO_NUMBER_ARRAY,
    resolutionUnit      IN VARCHAR2,
    mosaicParam         IN VARCHAR2,
    rasterBlob          IN OUT NOCOPY BLOB,
    outArea             OUT SDO_GEOMETRY,
    outWindow           OUT SDO_NUMBER_ARRAY,
    storageParam        IN VARCHAR2 DEFAULT NULL,
    bgValues            IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.getMosaicSubset(
    inGeoRasters     IN SYS_REFCURSOR,
    pyramidLevel     IN NUMBER,
    elevationParam   IN VARCHAR2,
    outSRID          IN NUMBER,
    outModelCoordLoc IN NUMBER,
    referencePoint   IN SDO_GEOMETRY,
    cropArea         IN SDO_GEOMETRY,
    polygonClip      IN VARCHAR2,
    boundaryClip     IN VARCHAR2,
    layerNumbers     IN VARCHAR2,
    outResolutions   IN SDO_NUMBER_ARRAY,
    resolutionUnit   IN VARCHAR2,
    mosaicParam      IN VARCHAR2,
    rasterBlob       IN OUT NOCOPY BLOB,
    outArea          OUT SDO_GEOMETRY,
    outWindow        OUT SDO_NUMBER_ARRAY,
    storageParam     IN VARCHAR2 DEFAULT NULL,
    bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL,
    referenceImage   IN SDO_GEORASTER DEFAULT NULL,
    referenceValue1  IN SDO_NUMBER_ARRAY DEFAULT NULL,
    referenceValue2  IN SDO_NUMBER_ARRAY DEFAULT NULL,
    refHistograms    IN SDO_GEOR_HISTOGRAM_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.getMosaicSubset(
    georasterTableNames  IN VARCHAR2,
    georasterColumnNames IN VARCHAR2,
    pyramidLevel        IN NUMBER,
    elevationParam      IN VARCHAR2,
    outSRID             IN NUMBER,
    outModelCoordLoc    IN NUMBER,
    referencePoint      IN SDO_GEOMETRY,
    cropArea            IN SDO_GEOMETRY,
    polygonClip         IN VARCHAR2,
    boundaryClip        IN VARCHAR2,
    layerNumbers        IN VARCHAR2,
    outResolutions      IN SDO_NUMBER_ARRAY,
    resolutionUnit      IN VARCHAR2,
    mosaicParam         IN VARCHAR2,
    rasterBlob          IN OUT NOCOPY BLOB,
    outArea             OUT SDO_GEOMETRY,
    outWindow           OUT SDO_NUMBER_ARRAY,
```

```
storageParam          IN VARCHAR2 DEFAULT NULL,
bgValues              IN SDO_NUMBER_ARRAY DEFAULT NULL,
referenceImage        IN SDO_GEORASTER DEFAULT NULL,
referenceValue1       IN SDO_NUMBER_ARRAY DEFAULT NULL,
referenceValue2       IN SDO_NUMBER_ARRAY DEFAULT NULL,
refHistograms         IN SDO_GEOR_HISTOGRAM_ARRAY DEFAULT NULL);
```

Description

Performs subsetting over a virtual mosaic or a collection of GeoRaster objects. It performs mosaic operations dynamically for the queried area and returns the required result in a BLOB on-the-fly. Internal rectification, common point rules, gap filling, and color balancing are performed whenever necessary.

Parameters

inGeoRasters
Source GeoRaster objects in a cursor.

georasterTableNames
Names (comma-separated) of the tables containing the source GeoRaster objects. For information about defining and using MIN_X_RES$ and MAX_X_RES$ columns in these tables, see the Usage Notes and Improving Query Performance Using MIN_X_RES$ and MAX_X_RES$.

georasterColumnNames
Names (comma-separated) of the columns of type SDO_GEORASTER in tables corresponding to the table names in georasterTableNames.

pyramidLevel
Pyramid level of the source GeoRaster objects for the operation. This parameter is used when the outResolutions parameter is not specified; otherwise, the pyramid level of the source GeoRaster objects is determined by the outResolutions parameter.

elevationParam
A string containing the elevation parameter average (average surface height). This parameter must be a quoted string that contains a keyword=value pair (for example, 'average=800'). This parameter specifies the elevation of the output GeoRaster object. If this parameter is null, 0 is assumed for average.
The use of the elevationParam parameter requires that the input GeoRaster objects have a 3D model SRID and a nonzero average height.

outSRID
Coordinate system for the output GeoRaster object. Must be either null or a value from the SRID column of the MDSYS.CS_SRS table.

outModelCoordLoc
A value specifying the model location of the base of the area represented by a cell: 0 for CENTER or 1 for UPPERLEFT. If null, CENTER is used.

referencePoint
A point of type SDO_GEOMETRY used as the reference point of mosaic. If a point is specified, the mosaicked image's upper left corner aligns with the reference point, that is, the distance between the referencePoint and the upper-left corner of the output will have an integer number of pixels. If this parameter is null, the reference point implicitly

uses the upper-left corner of the `cropArea`, or when the `cropArea` is null, the upper-left corner of the output extent.

cropArea
Crop area definition. If the SDO_GEOMETRY object has a non-null SRID, the coordinates specified in the SDO_GEOMETRY object are in model space and the source GeoRaster objects must be georeferenced; otherwise, the coordinates specified in the SDO_GEOMETRY object are in cell space and the source GeoRaster objects can be georeferenced or non-georeferenced. If `polygonClip` is `FALSE`, the MBR of the `cropArea` is used to crop the mosaicked data. If `polygonClip` is `TRUE`, the geometry of the `cropArea` is used to crop the mosaicked data.

polygonClip
The string `TRUE` causes the `cropArea` value to be used to crop the mosaicked data; the string `FALSE` or a null value causes the MBR of the `cropArea` to be used to crop the mosaicked data.

boundaryClip
The string `TRUE` or a null value causes the boundary of the virtual mosaic to be used to clip the `cropArea`; the string `FALSE` causes the area that is outside the virtual mosaic but within the `cropArea` to be filled with the background value.

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 not specified, the mosaic result contains the same number of bands as the source GeoRaster objects.

outResolutions
Resolution requested for the output GeoRaster data. If null, the default is the resolution of the first encountered GeoRaster object. See the Usage Notes for details.

resolutionUnit
The unit of the `outResolutions` parameter. If null, the default is the unit of the output SRID. If specified, it must be a quoted string in the format "unit=*value*" where *value* is the unit name value (a valid UNIT_OF_MEAS_NAME value from the SDO_UNITS_OF_MEASURE table). This parameter is ignored if `outResolutions` is null.

mosaicParam
A comma-separated quoted string of *keyword=value* pairs for specifying mosaic parameters. It can contain one or more of the keywords in Table 9-1 in the SDO_GEOR_AGGR.mosaicSubset reference section.

rasterBlob
BLOB (binary large object) to hold the result of the operation. It must exist or have been initialized before the operation is performed. It is usually a temporary BLOB.

outArea
Geometry object that describes the extent of the output data.

outWindow
An SDO_NUMBER_ARRAY object that identifies the coordinates of the upper-left and lower-right corners of the output window in the cell space. For this function, the upper-left corner of the output window is always (0,0).

storageParam
A string specifying storage parameters, as explained in Storage Parameters. If this parameter is null, the resulting GeoRaster object has the same storage parameters (`cellDepth`, `interleaving`, and `compression`) as the first encountered source GeoRaster object in the model space (if applicable) or cell space.
If `pyramid=true` is specified, it is ignored for this procedure, but not for SDO_GEOR_AGGR.mosaicSubset.
If `bitmapmask=true` is specified, it is ignored for this procedure.

bgValues
Background cell values for filling partially empty raster blocks (full empty raster blocks are left as empty without filling). They are used to fill the empty areas resulted from the mosaicking operation, such as the areas that are outside of the clipping polygon or the gap areas not covered by any source images. (See alsoEmpty Raster Blocks.) 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 default `bgValues` are zero (0).
The filling values must be valid cell values as specified by the target cell depth.

referenceImage
Specifies the reference image used during the color balance operation. If null, it defaults to the source image that is closest to the center of the output mosaic.

referenceValue1
Specifies the reference value or values used during the color balance operation. The meaning of the values is determined by the color balance method.

referenceValue2
Specifies the reference value or values used during the color balance operation. The meaning of the values is determined by the color balance method.

referenceHistograms
Specifies the reference histograms used during the color balance operation.

Usage Notes

The first two formats of the procedure provide the basic color balancing method and assume that the elevation of the output GeoRaster object is 0. The last two formats of the procedure provide more advanced color balancing method using the reference values or reference images, and they let you specify the elevation value for the output GeoRaster object.

The source GeoRaster objects must be prepared images or raster data so that they can be mosaicked. That is, the GeoRaster objects in the virtual mosaic 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 number of layers or bands. There is no restriction on the row and column dimension sizes of the source objects.

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 GeoRaster Data Model.)

The resulting GeoRaster object's spatial reference metadata information is determined by the `outSRID` and `outResolutions` parameters. If `outSRID` is not specified, the SRID of the first encountered source GeoRaster object is used. If `outResolutions` is not specified, the spatial resolution of the first encountered source GeoRaster object at thespecified pyramid level (`pyramidLevel` parameter) is used. The spatial resolution must be set in the metadata of all the source images.

If any source GeoRaster object has a different SRID from `outSRID` or is not rectified, it is dynamically reprojected or rectified so that the mosaicked GeoRaster object has uniform SRID and spatial resolution values.

If the source GeoRaster objects overlap, data of the overlapping area follows the rules specified in the `mosaicParam` parameter. By default, the cell value of the last encountered source GeoRaster object is used.

The source GeoRaster objects that contributes to the output mosaic are selected by the `cropArea` parameter spatially and `outResolutions` parameter in resolution wise (resolution selection).

- The `cropArea` parameter is used to query the source GeoRaster objects' spatial extents to determine the GeoRaster objects that are covered by the `cropArea` geometry; thus, a patial index should be built on the `spatialExtent` attribute of the GeoRaster objects.

- The `outResolutions` parameter is used to find the source GeoRaster objects so that the `outResolutions` value is within the resolution range of the source GeoRaster objects. A GeoRaster object's spatial resolution range is determined by the resolution at pyramid level 0 and the resolution at the top pyramid level of the GeoRaster object. It is a half pyramid lower than the pyramid level 0 and a half pyramid level higher than the top pyramid level. For example, if a GeoRaster object has a resolution of 30 meters at pyramid level 0 and a resolution of 960 meters at the top pyramid level 5, this GeoRaster object's spatial resolution range is between 22.5 meters ((30 + 30/2)/2) and 1440 meters ((960 + 960*2)/2).

The resolution selection using the `outResolutions` parameter can be speeded up by defining the columns MIN_X_RES$ and MAX_X_RES$ (both of type NUMBER) in the tables listed in the `georasterTableNames` parameter, where MIN_X_RES$ and MAX_X_RES$ specify the minimum and maximum spatial resolution values, respectively, of the source GeoRaster object. (A B-tree index should be created on the MIN_X_RES$ and MAX_X_RES$ columns if there are a large number of source GeoRaster objects.) To take advantage of this feature, the procedure format with the `georasterTableNames` parameter must be used. It also requires that the resolution values stored in the MIN_X_RES$ and MAX_X_RES$ columns have the same unit as the unit of the SRID that is stored in the source GeoRaster object's `spatialExtent` attribute.

The resolution selection can be turned off by setting `resFilter=false` in the `mosaicParam` parameter. When the resolution selection is turned off, the source GeoRaster objects will only be filtered spatially.

If the source GeoRaster objects have empty raster blocks or do not cover the whole area, the empty areas are filled with the values specified in the `bgValues` parameter, or with 0 if the `bgValues` parameter is not specified.

In order to use the `colorBalance` option in `mosaicParam`, you should call SDO_GEOR.generateStatistics on the source images to generate the image's statistics and store them in the metadata for the source image.

If all source GeoRaster objects are blank and have the same `blankCellValue` value, the resulting `rasterBlob` is filled with the `blankCellValue` value.

For more information, see Virtual Mosaic.

Examples

The following example gets the subset of a virtual mosaic (defined as two GeoRaster tables) at SRID 32610 with resolution of 30 meters by specifying a `cropArea` window. NODATA is considered at the resampling process (if there is one) and at the overlapping area of the source images.

```
declare
  lb blob;
  cropArea sdo_geometry;
  outArea  sdo_geometry := null;
  outWin    sdo_number_array:=null;
  resolutions sdo_number_array;
begin
    dbms_lob.createTemporary(lb, TRUE);

     cropArea :=  sdo_geometry(2003, 26986, null,
                   sdo_elem_info_array(1, 1003, 3),
                   sdo_ordinate_array(399180, 4247820,
                                      496140,4353900) );
      resolutions := sdo_number_array(30, 30);
     sdo_geor_aggr.getMosaicSubset('georaster_table_1, georaster_table_2',
                 'georaster, georaster',
                 0, 32610, null, null, cropArea,
                 null, null, null, resolutions, null,
                 'nodata=true',
                 lb, outArea, outWin);
    dbms_lob.freeTemporary(lb);
    if outWin is not null then
        dbms_output.put_line('output window: (' || outWin(1) || ',' || outWin(2)
||', ' || outWin(3) || ', ' || outWin(4) || ')');
    end if;
end;
```

9.5 SDO_GEOR_AGGR.mosaicSubset

Format

```
SDO_GEOR_AGGR.mosaicSubset(
     inGeoRasters       IN SYS_REFCURSOR,
     pyramidLevel      IN NUMBER,
     outSRID           IN NUMBER,
     outModelCoordLoc  IN NUMBER,
     referencePoint    IN SDO_GEOMETRY,
     cropArea          IN SDO_GEOMETRY,
     polygonClip       IN VARCHAR2,
     boundaryClip      IN VARCHAR2,
     layerNumbers      IN VARCHAR2,
     outResolutions    IN SDO_NUMBER_ARRAY,
     resolutionUnit    IN VARCHAR2,
     mosaicParam       IN VARCHAR2,
     storageParam      IN VARCHAR2,
     outGeoRaster      IN OUT SDO_GEORASTER,
     bgValues          IN SDO_NUMBER_ARRAY DEFAULT NULL,
     parallelParam     IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.mosaicSubset(
     georasterTableNames  IN VARCHAR2,
     georasterColumnNames IN VARCHAR2,
     pyramidLevel          IN NUMBER,
     outSRID              IN NUMBER,
     outModelCoordLoc     IN NUMBER,
     referencePoint       IN SDO_GEOMETRY,
     cropArea             IN SDO_GEOMETRY,
     polygonClip          IN VARCHAR2,
     boundaryClip         IN VARCHAR2,
     layerNumbers         IN VARCHAR2,
     outResolutions       IN SDO_NUMBER_ARRAY,
     resolutionUnit       IN VARCHAR2,
     mosaicParam          IN VARCHAR2,
     storageParam         IN VARCHAR2,
     outGeoRaster         IN OUT SDO_GEORASTER,
     bgValues             IN SDO_NUMBER_ARRAY DEFAULT NULL,
     parallelParam        IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.mosaicSubset(
     inGeoRasters     IN SYS_REFCURSOR,
     pyramidLevel     IN NUMBER,
     elevationParam   IN VARCHAR2,
     outSRID          IN NUMBER,
     outModelCoordLoc IN NUMBER,
     referencePoint   IN SDO_GEOMETRY,
     cropArea         IN SDO_GEOMETRY,
     polygonClip      IN VARCHAR2,
     boundaryClip     IN VARCHAR2,
     layerNumbers     IN VARCHAR2,
     outResolutions   IN SDO_NUMBER_ARRAY,
     resolutionUnit   IN VARCHAR2,
     mosaicParam      IN VARCHAR2,
     storageParam     IN VARCHAR2,
     outGeoRaster     IN OUT SDO_GEORASTER,
     bgValues         IN SDO_NUMBER_ARRAY DEFAULT NULL,
     parallelParam    IN VARCHAR2 DEFAULT NULL,
     referenceImage   IN SDO_GEORASTER DEFAULT NULL,
     referenceValue1  IN SDO_NUMBER_ARRAY DEFAULT NULL,
     referenceValue2  IN SDO_NUMBER_ARRAY DEFAULT NULL,
     refHistograms    IN SDO_GEOR_HISTOGRAM_ARRAY DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.mosaicSubset(
     georasterTableNames  IN VARCHAR2,
     georasterColumnNames IN VARCHAR2,
     pyramidLevel          IN NUMBER,
     elevationParam       IN VARCHAR2,
     outSRID              IN NUMBER,
     outModelCoordLoc     IN NUMBER,
     referencePoint       IN SDO_GEOMETRY,
     cropArea             IN SDO_GEOMETRY,
     polygonClip          IN VARCHAR2,
     boundaryClip         IN VARCHAR2,
     layerNumbers         IN VARCHAR2,
     outResolutions       IN SDO_NUMBER_ARRAY,
     resolutionUnit       IN VARCHAR2,
```

```
mosaicParam          IN VARCHAR2,
storageParam         IN VARCHAR2,
outGeoRaster         IN OUT SDO_GEORASTER,
bgValues             IN SDO_NUMBER_ARRAY DEFAULT NULL,
parallelParam        IN VARCHAR2 DEFAULT NULL,
referenceImage       IN SDO_GEORASTER DEFAULT NULL,
referenceValue1      IN SDO_NUMBER_ARRAY DEFAULT NULL,
referenceValue2      IN SDO_NUMBER_ARRAY DEFAULT NULL,
refHistograms        IN SDO_GEOR_HISTOGRAM_ARRAY DEFAULT NULL);
```

Description

Performs advanced large-scale mosaicking or subsetting from a virtual mosaic or a collection of GeoRaster objects. The output data is written into a GeoRaster object for persistent storage or other processing. Internal rectification, common point rules, gap filling, and color balancing are performed whenever necessary.

Parameters

inGeoRasters
Source GeoRaster objects in a cursor.

georasterTableNames
Names (comma-separated) of the tables containing the source GeoRaster objects. For information about defining and using MIN_X_RES$ and MAX_X_RES$ columns in these tables, see the Usage Notes and Improving Query Performance Using MIN_X_RES$ and MAX_X_RES$.

georasterColumnNames
Names (comma-separated) of the columns of type SDO_GEORASTER in tables corresponding to the table names in `georasterTableNames`.

pyramidLevel
Pyramid level of the source GeoRaster objects for the operation. This parameter is used when the `outResolutions` parameter is not specified; otherwise, the pyramid level of the source GeoRaster objects involved in the mosaic is determined by the `outResolutions` parameter.
The `pyramidLevel` parameter and the option `pyramid=true` in the `storageParam` parameter are valid only when the source GeoRaster objects have the same resolution on pyramid level 0.
If the `outResolutions` parameter is not null, the `pyramidLevel` parameter is ignored.

elevationParam
A string containing the elevation parameter `average` (average surface height). This parameter must be a quoted string that contains a keyword=value pair (for example, `'average=800'`). This parameter specifies the elevation of the output GeoRaster object. If this parameter is null, 0 is assumed for `average`.
The use of the `elevationParam` parameter requires that the input GeoRaster objects have a 3D model SRID and a nonzero average surface height.

outSRID
Coordinate system for the output GeoRaster object. Must be either null or a value from the SRID column of the MDSYS.CS_SRS table.

outModelCoordLoc

A value specifying the model location of the base of the area represented by a cell: `0` for `CENTER` or `1` for `UPPERLEFT`. If null, `CENTER` is used.

referencePoint

A point of type SDO_GEOMETRY used as the reference point of mosaic. If a point is specified, the mosaicked image's upper left corner aligns with the reference point, that is, the distance between the `referencePoint` and the upper-left corner of the output will have an integer number of pixels. If this parameter is null, the reference point implicitly uses the upper-left corner of the `cropArea`, or when the `cropArea` is null, the upper-left corner of the output extent.

cropArea

Crop area definition. If the SDO_GEOMETRY object has a non-null SRID, the coordinates specified in the SDO_GEOMETRY object are in model space and the source GeoRaster objects must be georeferenced; otherwise, the coordinates specified in the SDO_GEOMETRY object are in cell space and the source GeoRaster objects can be georeferenced or non-georeferenced. If `polygonClip` is `FALSE`, the MBR of the `cropArea` is used to crop the mosaicked data. If `polygonClip` is `TRUE`, the geometry of the `cropArea` is used to crop the mosaicked data.

polygonClip

The string `TRUE` causes the `cropArea` value to be used to crop the mosaicked data; the string `FALSE` or a null value causes the MBR of the `cropArea` to be used to crop the mosaicked data.

boundaryClip

The string `TRUE` or a null value causes the boundary of the virtual mosaic to be used to clip the `cropArea`; the string `FALSE` causes the area that is outside the virtual mosaic but within the `cropArea` to be filled with the background value

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 not specified, the mosaic result contains the same number of bands as the source GeoRaster objects.

outResolutions

Resolution of the output GeoRaster data. If null, the default is the resolution of the first encountered GeoRaster object. See the Usage Notes for details.

resolutionUnit

The unit of the `outResolutions` parameter. If null, the default is the unit of the output SRID. If specified, it must be a quoted string in the format "unit=*value*" where *value* is the unit name value (a valid UNIT_OF_MEAS_NAME value from the SDO_UNITS_OF_MEASURE table). This parameter is ignored if `outResolutions` is null.

mosaicParam

A comma-separated quoted string of *keyword=value* pairs for specifying mosaic parameters. It can contain one or more of the keywords in Table 9-1.

> **✎ Note:**
>
> For any numbers in string (VARCHAR2) parameters to GeoRaster subprograms, the period (.) must be used for any decimal points regardless of the locale.

Keyword	Explanation
cbreference	(Supported with the procedure formats that include the `referenceImage`, `referenceValue1`, / `referenceValue2`, and `refHistograms` parameters.) Specifies the source of the reference value used in `colorBalance`. Can have one of the following values: • `VALUE`: The reference value is specified in the `referenceValue1` and `referenceValue2` parameters. • `IMAGE`: The reference value is retrieved from the `referenceImage` parameter. • `OVERLAP`: the reference value is derived from the overlapping area of the source images. (Always ignored if specified in `appendParam` for SDO_GEOR_AGGR.append.)

Keyword	Explanation
colorBalance	Specifies the method for color balancing. Can have one of the following values: • `NONE` (the default): No color balancing is performed. • `LINEARSTRETCHING`: Use the linear stretching method to stretch the source image's cell values to the reference minimum and maximum value. If the `cbreference` parameter is `VALUE`, then parameter `referenceValue1` has the reference minimum values and parameter `referenceValue2` has the reference maximum values. If the `referenceValue1` or `referenceValue2` is null, default values 0 and 255 are used, respectively. If the `referenceValue1` or `referenceValue2` has only one value, it is applied to all the bands; otherwise, the reference values are corresponding to each band. Thus, the number of values in the SDO_NUMBER_ARRAY must be the same as the number of bands in the source images. If the `cbreference` parameter is `IMAGE`, then the minimum and maximum values of `referenceImage` are used as the reference. If the `cbreference` parameter is not specified, the `minVal` and `maxVal` parameters are used to specify the minimum and maximum reference values, respectively. • `STATISTICMATCHING`: Stretch the source images to match the reference statistics. If the `cbreference` parameter is `VALUE`, then parameter `referenceValue1` has the reference mean value and parameter `referenceValue2` has the reference standard deviation value. If the `referenceValue1` or `referenceValue2` is null, default

Keyword	Explanation
	values 128 and 100 are used, respectively. If the `referenceValue1` or `referenceValue2` has only one value, it is applied to all the bands; otherwise, the reference values are corresponding to each band. Thus, the number of values in the SDO_NUMBER_ARRAY must be the same as the number of bands in the source images. If the `cbreference` parameter is `IMAGE`, then the reference image's mean and standard deviation values are used for the matching. If the `cbreference` parameter is `OVERLAP`, then the mean and standard deviation values are derived from the overlapping area. If the `cbreference` parameter is not specified, the `std` and `mean` parameters are used to specify the reference statistics. • `HISTOGRAMMATCHING`: Use the reference histogram as the source image's histogram. If the `cbreference` parameter is `VALUE`, then parameter `referenceHistogram` defines the reference histograms for each band. If only one reference histogram is specified, it is applied to all the bands. If the `cbreference` parameter is `IMAGE`, then the referencce image's histograms for each band are used as the reference for matching If the `cbreference` parameter is `OVERLAP`, then histograms at the overlapping area are used to derive the matching.

Keyword	Explanation
commonPointRule	Specifies the method for getting the cell value at the overlapping area. Can have one of the following values: • `START`: The value from the first encountered GeoRaster object is used. • `END`: The value from the last encountered GeoRaster object is used. • `LATEST`: The value from the GeoRaster object that has the most recent `EndDateTime` in the metadata is used. • `OLDEST`: The value from the GeoRaster object that has the oldest `EndDateTime` in the metadata is used. • `CTC`: The call value from the GeoRaster object that is closest to the center of the output window is used. • `HIGH`: The maximum cell value of all the overlapping GeoRaster objects is used. • `LOW`: The minimum cell value of all the overlapping GeoRaster objects is used. • `AVERAGE`: The average of all cell values from the overlapping GeoRaster objects is used. • `HIGHRES`: The value from the GeoRaster object that has the highest spatial resolution is used.
fillGap	Specifies whether or not to fill the narrow gap between source images. `TRUE` causes any gap that is less than or equal to 2 pixels wide to be filled with the nearest-neighbor pixel value. `FALSE` causes any gaps to be filled with zero or `bgValue`.
maxVal	Ignored if `colorBalance` is not `LINEARSTRETCHING` or if the `cbreference` keyword is specified; otherwise, specifies the highest value in the range of the linear stretching method. Defaults to 255. (Always ignored if specified in `appendParam` for SDO_GEOR_AGGR.append.)

Keyword	Explanation
mean	Ignored if `colorBalance` is not `STATISTICMATCHING` or if the `cbreference` keyword is specified; otherwise, specifies the reference mean of the statistic matching method. (Always ignored if specified in `appendParam` for SDO_GEOR_AGGR.append.)
minVal	Ignored if `colorBalance` is not `LINEARSTRETCHING` or if the `cbreference` keyword is specified; otherwise, specifies the lowest value in the range of the linear stretching method. Defaults to 0. (Always ignored if specified in `appendParam` for SDO_GEOR_AGGR.append.)
nodata	Specifies whether or not to consider NODATA (NODATA value or NODATA bitmap mask) when handling the overlap area or when resampling is performed. The default value is `FALSE`. When handling the overlap area, `nodata=TRUE` causes any NODATA values cells not to be involved in the overlap area calculation; `nodata=FALSE` causes all the overlapped cells to be involved in the overlap area calculation. When resampling is performed and the resampling method is `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`, if any of the cells involved in the resampling has a NODATA value, then `nodata=TRUE` causes the result of the resampling to be a NODATA value.
resampling	Specifies the resampling method (if resampling is involved or rectification is needed) to be used during the mosaic operation. Can have one of the following values: `NN` (the default), `BILINEAR`, `BIQUADRATIC`, `CUBIC`, `AVERAGE4`, or `AVERAGE16`. For more information, see Resampling and Interpolation.

Keyword	Explanation
resamplingTolerance	Specifies the tolerance for not doing resampling when the source GeoRaster objects are not perfectly aligned. The value should be between 0 and 0.5, where the unit is pixel or cell (for example, `0.5` meaning one-half pixel or cell). If not specified, `0.5` is used, which means no resampling will occur.
resFilter	Specifies whether or not to filter the source GeoRaster objects to select only the GeoRaster object that is in the range of the output resolution (`outResolution` parameter). `TRUE` (the default) enables resolution filtering. `FALSE` disables resolution filtering, causing all the source GeoRaster objects to be involved in the mosaicking process. (Always ignored if specified in `appendParam` for SDO_GEOR_AGGR.append.)
std	Ignored if `colorBalance` is not `STATISTICMATCHING` or if the `cbreference` keyword is specified; otherwise, specifies the reference standard deviation for the statistic matching method. (Always ignored if specified in `appendParam` for SDO_GEOR_AGGR.append.)

storageParam

A string specifying storage parameters, as explained in Storage Parameters. If this parameter is null, the resulting GeoRaster object has the same storage parameters (`blockSize`, `cellDepth`, `interleaving`, and `compression`) as the first encountered 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 first encountered GeoRaster object to be mosaicked.

If `pyramid=true` is specified, the pyramids of the source GeoRaster objects are mosaicked when the `outResolutions` parameter is null and the `pyramidLevel` parameter is not null. The maximum pyramid level of the result GeoRaster object is the minimal value of the maximum pyramid level of the source GeoRaster objects. By default, `pyramid=false`, and thus the pyramids are not mosaicked.

If `bitmapmask=true` is specified, the bitmap masks of the source GeoRaster objects are mosaicked also.

outGeoRaster

GeoRaster object to hold the result of the operation. Cannot be the same as any source GeoRaster object.

bgValues

Background cell values for filling partially empty raster blocks (full empty raster blocks are left as empty without filling). They are used to fill the empty areas resulted from the mosaicking operation, such as the areas that are outside of the clipping polygon or the gap areas not covered by any source images. (See also Empty Raster Blocks.) 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 default `bgValues` are zero (0).

The filling values must be valid cell values as specified by the target cell depth.

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)

If parallelism is specified, the procedure performs an internal commit operation. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting mosaicked GeoRaster object explicitly in order to roll back the operation.

referenceImage

Specifies the reference image used during the color balance operation. If null, it defaults to the first image from the query.

referenceValue1

Specifies the reference value or values used during the color balance operation. The meaning of the values is determined by the color balance method.

referenceValue2

Specifies the reference value or values used during the color balance operation. The meaning of the values is determined by the color balance method.

referenceHistograms

Specifies the reference histograms used during the color balance operation.

Usage Notes

The first two formats of the procedure provide the basic color balancing method and assume that the elevation of the output GeoRaster object is 0. The last two formats of the procedure provide more advanced color balancing method using the reference values or reference images, and they let you specify the elevation value for the output GeoRaster object.

The source GeoRaster objects must be prepared images or raster data so that they can be mosaicked. That is, 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 number of layers or bands. There is no restriction on the row and column dimension sizes of the source objects.

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 GeoRaster Data Model.)

The resulting GeoRaster object's spatial reference metadata information is determined by the `outSRID` and `outResolutions` parameters. If `outSRID` is not specified, the SRID of the first encountered source GeoRaster object is used. If `outResolutions` is not specified, the spatial resolution of the first encountered source GeoRaster object at specified pyramid level (`pyramidLevel` parameter) is used. The spatial resolution must be set in the metadata of all the source images.

If any source GeoRaster object has a different SRID from `outSRID` or is not rectified, it is dynamically reprojected or rectified so that the mosaicked GeoRaster object has uniform SRID and spatial resolution values.

If the source GeoRaster objects overlap, data of the overlapping area follows the rules specified in the `mosaicParam` parameter. By default, the cell value of the last encountered source GeoRaster object is used.

The source GeoRaster objects that contributes to the output mosaic are selected by the `cropArea` parameter spatially and `outResolutions` parameter in resolution wise (resolution selection).

- The `cropArea` parameter is used to query the source GeoRaster objects' spatial extents to determine the GeoRaster objects that are covered by the `cropArea` geometry,. Thus, a spatial index should be built on the `spatialExtent` attribute of the GeoRaster objects.

- The `outResolutions` parameter is used to find the source GeoRaster objects so that the `outResolutions` value is within the resolution range of the source GeoRaster objects. A GeoRaster object's spatial resolution range is determined by the resolution at pyramid level 0 and the resolution at the top pyramid level of the GeoRaster object. It is a half pyramid lower than the pyramid level 0 and a half pyramid level higher than the top pyramid level. For example, if a GeoRaster object has a resolution of 30 meters at pyramid level 0 and a resolution of 960 meters at the top pyramid level 5, this GeoRaster object's spatial resolution range is between 22.5 meters ((30 + 30/2)/2) and 1440 meters ((960 + 960*2)/2).

The resolution selection using the `outResolutions` parameter can be speeded up by defining the columns MIN_X_RES$ and MAX_X_RES$ (both of type NUMBER) in the tables listed in `georasterTableNames` parameter, where MIN_X_RES$ and MAX_X_RES$ specify the minimum and maximum spatial resolution values, respectively, of the source GeoRaster object. (A B-tree index should be created on the MIN_X_RES$ and MAX_X_RES$ columns if there are a large number of source GeoRaster objects.) To take advantage of this feature, the procedure format with the `georasterTableNames` parameter must be used. It also requires that the resolution values stored in the MIN_X_RES$ and MAX_X_RES$ columns have the same unit as the unit of the SRID that is stored in the source GeoRaster object spatialExtent attribute.

The resolution selection can be turned off by setting `resFilter=false` in the `mosaicParam` parameter. When the resolution selection is turned off, the source GeoRaster objects will only be filtered spatially.

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 Empty Raster Blocks). 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 `bitmapmask` parameter is set to `true` in the `storageParam` string, the bitmap masks are mosaicked. By default, the bitmap masks are not mosaiked. Note that the bitmap mask may also be considered as NODATA; and if they are, see the NODATA keyword in Table 9-1.

In order to use the `colorBalance` option in `mosaicParam`, you should you should call SDO_GEOR.generateStatistics on the source images to generate the image's statistics and store them in the metadata for the source image.

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).

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 as explained in Maintaining GeoRaster Objects and System Data in the Database.

For more information, see Large-Scale Image Mosaicking and Virtual Mosaic.

Examples

The following example shows how to color balance the source images using the linear stretching method (`colorBalance=linearstretching`). It stretches the source image cell values (`cbreference=value`) to the minimum and maximum reference values. The reference values are specified in the `ref_min` and `ref_max` variables.

```
DECLARE
gr sdo_georaster;
resolutions   sdo_number_array;
ref_min      sdo_number_arrray;
ref_max    sdo_number_array;
BEGIN
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
         returning georaster into gr;

    resolutions := sdo_number_array(30, 30);
    ref_min := sdo_number_array(10, 10, 10);
    ref_max := sdo_number_array(200, 255, 230);

    sdo_geor_aggr.mosaicSubset('georaster_table_1, georaster_table_2',
                  'georaster, georaster',
                  0, null, 32610, null, null, null,
                  null, null, null, resolutions, null,
                  'commonPointRule=average, colorBalance=linearstretching,
cbreference=value',
                  'blocking=optimalpadding, blocksize=(512,512,3)', gr, null,
'parallel=4', null, ref_min, ref_max);

    update georaster_table set georaster = gr where georid=10;
    commit;
END;
/
```

The following example shows how to color balance the source images using the histogram matching method (`colorBalance=histogramMatching`). It matches the histogram of each source image (`cbreference=image`) to the reference image's histogram.

```
DECLARE
gr sdo_georaster;
resolutions  sdo_number_array;
refgr sdo_georaster;
BEGIN
    -- get the reference image
select georaster into refgr from georaster_table where georid = 1;

-- insert new georaster for the mosaic
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
         returning georaster into gr;

     resolutions := sdo_number_array(30, 30);

    -- mosaic
     sdo_geor_aggr.mosaicSubset('georaster_table_1, georaster_table_2',
                    'georaster, georaster',
                    0, 32610, null, null, null, null,
                    null, null, null, resolutions, null,
                    'commonPointRule=average, colorBalance=histogramMatching,
cbreference=image',
                    'blocking=optimalpadding, blocksize=(512,512,3)', gr, null,
'parallel=4', refgr);

      update georaster_table set georaster = gr where georid=10;
      commit;
END;
/
```

The following example shows how to color balance the source images using the statistic matching method (`colorBalance=statisticMatching`). It adjusts the statistics of the source images based on the overlapping areas (`cbreference=overlap`).

```
DECLARE
gr sdo_georaster;
resolutions  sdo_number_array;
BEGIN
    -- get the reference image
select georaster into refgr from georaster_table where georid = 1;

-- insert new georaster for the mosaic
    insert into georaster_table (georid, georaster)
        values (10, sdo_geor.init('RDT_1',10))
         returning georaster into gr;

     resolutions := sdo_number_array(30, 30);

    -- mosaic
     sdo_geor_aggr.mosaicSubset('georaster_table_1, georaster_table_2',
                    'georaster, georaster',
                    0, null, 32610, null, null, null,
                    null, null, null, resolutions, null,
                    'commonPointRule=average, colorBalance=statisticMatching,
cbreference=overlap',
                    'blocking=optimalpadding, blocksize=(512,512,3)', gr, null,
```

```
                     'parallel=4');

                           update georaster_table set georaster = gr where georid=10;
                           commit;
                     END;
                     /
```

9.6 SDO_GEOR_AGGR.validateForMosaicSubset

Format

```
SDO_GEOR_AGGR.validateForMosaicSubset(
       inGeoRasters     IN SYS_REFCURSOR,
       outSRID          IN NUMBER,
       outResolutions   IN SDO_NUMBER_ARRAY,
       resolutionUnit   IN VARCHAR2,
       resultTableName  IN VARCHAR2);
```

or

```
SDO_GEOR_AGGR.validateForMosaicSubset(
       georasterTableNames   IN VARCHAR2,
       georasterColumnNames  IN VARCHAR2,
       outSRID               IN NUMBER,
       outResolutions        IN SDO_NUMBER_ARRAY,
       resolutionUnit        IN VARCHAR2,
       resultTableName       IN VARCHAR2);
```

Description

Checks if it is feasible to do mosaicking or subset query operations over a virtual mosaic or a large collection of GeoRaster objects. Any validation errors and notes are stored in a user-created result table.

Parameters

inGeoRasters
Source GeoRaster objects in a cursor.

georasterTableNames
Names (comma-separated) of the tables containing the source GeoRaster objects.

georasterColumnNames
Names (comma-separated) of the columns of type SDO_GEORASTER in tables corresponding to the table names in `georasterTableNames`.

outSRID
Coordinate system for the output GeoRaster object. Must be either null or a value from the SRID column of the MDSYS.CS_SRS table.

outResolutions
Resolution of the output GeoRaster data. If null, the default is the resolution of the first encountered source GeoRaster object.

resolutionUnit
The unit of the `outResolutions` parameter. If null, the default is the unit of the output SRID. If specified, it must be a quoted string in the format "unit=*value*" where *value* is

the unit name value (a valid UNIT_OF_MEAS_NAME value from the SDO_UNITS_OF_MEASURE table). This parameter is ignored if `outResolutions` is null.

resultTableName
Name of the validation result table. This table must already exist, and it must have the following column definitions:

```
time timestamp,
type varchar2(16),
description varchar2(512),
table_name varchar2(32),
column_name varchar2(1024),
rdt_table_name varchar2(32),
raster_id number
```

Usage Notes

The following considerations apply to the `resultTableName` parameter value:

- If the specified table does not exist, an error is generated.

- If the parameter is not specified or is specified as null, the procedure throws an error at the first validation error found; otherwise, the procedure puts all the validation errors in the table and completes without error.

- If the specified table is not empty, the procedure appends rows to the existing data in the table; and if there is a unique constraint on any column and if the newly appended data has the same value as existing data in that constrained column, an error is generated.

- The TYPE column of the table contains a string indicating the type of issue, such as `ERROR` (something that must be fixed) or `NOTE` (information that may or may not require some action). The DESCRIPTION column provides details about the issue.

This procedure performs the following validation checks:

- The source GeoRaster objects must have the same band dimension size.

- The source GeoRaster objects must have consistent georeference status, that is, either all are georeferenced or all are not georeferenced.

- If reprojection or rectification to be performed when doing the mosaic, the operation must be feasible.

Examples

The following example checks if a mosaic operation is possible. Any validation errors are stored in the predefined table MOSAIC_ERROR.

```
EXECUTE sdo_geor_aggr.validateForMosaicSubset('georaster_table_1,
georaster_table_2', 'georaster, georaster', 26986, sdo_number_array(30, 30),
'unit=meter', 'mosaic_error');

-- Check the validation results:
SELECT table_name table, column_name column, rdt_table_name rdt, raster_id rid,
type, description FROM mosaic_error ORDER BY time;

TABLE       COLUMN    RDT   RID TYPE  DESCRIPTION
----------  --------  ----- --- ----- --------------------------------------
GEORASTER_1 GEORASTER RDT_1   2 ERROR The source georaster object is
                                        not georeferenced.
```

10

SDO_GEOR_IP Package Reference

The SDO_GEOR_IP package contains subprograms (functions and procedures) for performing image processing operations on GeoRaster objects. This chapter presents reference information, with one or more examples, for each subprogram.

Many examples in this chapter refer to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.

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

Topics:

10.1 SDO_GEOR_IP.dodge

Format

```
SDO_GEOR_IP.dodge(
     inGeoRaster        IN SDO_GEORASTER,
     gridsize           IN SDO_NUMBER_ARRAY,
     samplingFactor     IN VARCHAR2 DEFAULT NULL,
     means              IN SDO_NUMBER_ARRAY DEFAULT NULL,
     standardDeviations IN SDO_NUMBER_ARRAY DEFAULT NULL,
     storageParam       IN VARCHAR2 DEFAULT NULL,
     outGeoraster       IN OUT SDO_GEORASTER,
     parallelParam      IN VARCHAR2 DEFAULT NULL);
```

Description

Apply a dodging algorithm on the input GeoRaster object to color balance the image.

Parameters

inGeoRaster
The SDO_GEORASTER object to be processed.

gridSize
The size of each grid in x and y direction, respectively. It is an array of one or two numbers. If only one number is specified, then it is for both x and y direction.

samplingFactor

Sampling factor, used to control the calculation of the statistics, 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. If not specified (null), the default is 1.

`samplingFactor` cannot be greater than or equal to one-half (0.5) of `gridSize`.

means

The target mean values for each output bands. If only one value is specified, it is applied to all the output bands; otherwise, it must have the same number of values as the number bands of the output GeoRaster object. If null, it is calculated as the average mean value over all the grids.

standardDeviations

The target standard deviation values for each output bands. If only one value is specified, it is applied to all the output bands; otherwise, it must have the same number of values as the number of bands of the output GeoRaster object. If null, it is calculated as the average standard deviation over all the grids.

storageParam

A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster

The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`.

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)

If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

This dodging operation uses an adaptive image enhancement method to make the image tone more balanced, that is, the darker area becomes brighter and the bright area becomes darker. The statistics for the grid defined by the `gridsize` parameter are collected on the fly and adjusted to the target mean and standard deviation values. The grid size should be smaller than the imbalanced area in order to remove the imbalance. Adjust the `gridSize` parameter to achieve the best result.

The input GeoRaster image must have a `cellDepth` value of `8BIT_U`. Any `celldepth` value in the storage parameters is ignored. The cell depth of the `outGeoRaster` object is always `8BIT_U`.

Color map in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

Examples

The following example creates a GeoRaster object that is the result of dodging on the input GeoRaster object. The desired mean and standard deviation are set as 125 and 80, respectively. The grid size is 512 on x and y direction. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
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_ip.dodge(gr1, sdo_number_array(512, 512), 'samplingFactor=3',
sdo_number_array(125), sdo_number_array(80), null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

10.2 SDO_GEOR_IP.equalize

Format

```
SDO_GEOR_IP.equalize(
     inGeoRaster   IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea      IN SDO_NUMBER_ARRAY,
     bandNumbers   IN VARCHAR2,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.equalize(
     inGeoRaster   IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea      IN SDO_GEOMETRY,
     layerNumbers IN VARCHAR2,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Process the input GeoRaster object by using histogram equalization method. The processed image is stored in the output GeoRaster object.

Parameters

inGeoRaster
The SDO_GEORASTER object to be equalized.

pyramidLevel
A number specifying the pyramid level to be equalized in the source GeoRaster
object. If null, the default is 0.

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.

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).

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The output SDO_GEORASTER object that reflects the results of the operation. Must
be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty
GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be
the same GeoRaster object as `inGeoRaster`.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form
`parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of
parallelism specified by this parameter. If not specified, then by default there is no
parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, the procedure performs an internal commit while the
process is running. Therefore, you cannot roll back the results of this procedure. If an
error occurs (even if it is raised by the Oracle parallel server), you must delete the
resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

The input GeoRaster object must have histogram generated for each output band.

The equalization is performed based on the histograms stored in the metadata of the
input GeoRaster object.

Color map stored in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

The cell depth of the output GeoRaster object is always `8BIT_U`.

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 `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 GeoRaster Data Model.)

If the `cropArea` parameter specifies a geodetic MBR, it cannot cross the date line meridian

Examples

The following example creates a GeoRaster object that has each input band equalized based on the histogram stored in the metadata. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_number_array := null;
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_ip.equalize(gr1, cropArea, null, null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

10.3 SDO_GEOR_IP.filter

Format

```
SDO_GEOR_IP.filter(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea     IN SDO_NUMBER_ARRAY,
     bandNumbers  IN VARCHAR2,
     filterParam  IN VARCHAR2,
     filterKernel IN SDO_NUMBER_ARRAY,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.filter(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea     IN SDO_GEOMETRY,
     layerNumbers IN VARCHAR2,
     filterParam  IN VARCHAR2,
     filterKernel IN SDO_NUMBER_ARRAY,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Applies the convolution filter on the input GeoRaster object. The processed image is stored in the output GeoRaster object.

Parameters

inGeoRaster
The SDO_GEORASTER object for the filter operation.

pyramidLevel
A number specifying the pyramid level for the filter operation in the source GeoRaster object. If null, the default is 0.

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.

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).

filterParam
The type of the filter to be applied on the input GeoRaster. It is in the format `'filterType=`*value*`'` where *value* can be one of the following: `LPF` (low-pass filter, the default), `HPF` (high-pass filter), `HBF` (high-boost filter), `MIN` (minimum filter), `MAX` (maximum filter), `MEDIAN` (median filter), `MODE` (mode filter), or `CUSTOM` (user-provided filter kernel).
`filterParam` can also include:

- `'kernelSize=(`*kx, ky*`)'` where *kx, ky* are the size of the kernel on the x and y direction. For a filter type other than `CUSTOM`, the `kx` and `ky` should be odd numbers greater than or equal to 3.

- `'p1 = `*value*`'`, used by the `HBF` filter to indicate the degree of boost.

filterKernel
Required only when `filterType=CUSTOM`. It is the `kx * ky` numbers in an SDO_NUMBER_ARRAY object.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

For an introduction to image filtering, see Image Filtering.

The following are `filterKernel` values for some of the predefined 3x3 filters:

- LPF: (1, 1, 1, 1, 1, 1, 1, 1, 1)*1/9

- HPF: (-1, -1, -1, -1, 8, -1, -1, -1, -1)*1/9

- HBF: (-k, -k, -k, -k, 8k, -k, -k, -k, -k)*1/9, where *k* is the boost factor specified by parameter `p1`

For `kernelSize`, *kx * ky* must be less than 10000.

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 `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 GeoRaster Data Model.)

Color map in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

Examples

The following example creates a GeoRaster object that is the result of using the default low-pass filter on the input GeoRaster object The filter kernel size is 3 by 3. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_geometry := null;
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_ip.filter(gr1, 0, cropArea, null, 'filtertype=LPF, kernelsize=(3, 3)',
null, null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

The following example applies a custom (user–provided) 3 by 3 filter on the input GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_geometry := null;
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_ip.filter(gr1, 0, cropArea, null, 'filtertype=CUSTOM, kernelsize=(3,3)',
sdo_number_array(1/4, 1/2, 1/4, 1/2, 1, 1/2, 1/4, 1.2, 1/4 ), null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

10.4 SDO_GEOR_IP.histogramMatch

Format

```
SDO_GEOR_IP.histogramMatch(
     inGeoRaster     IN  SDO_GEORASTER,
     pyramidLevel    IN  NUMBER,
     cropArea        IN  SDO_NUMBER_ARRAY,
     bandNumbers     IN  VARCHAR2,
     refHistograms   IN  SDO_GEOR_HISTOGRAM_ARRAY,
     storageParam    IN  VARCHAR2,
```

```
      outGeoraster    IN OUT SDO_GEORASTER,
      parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.histogramMatch(
      inGeoRaster     IN SDO_GEORASTER,
      pyramidLevel    IN NUMBER,
      cropArea        IN SDO_GEOMETRY,
      layerNumbers    IN VARCHAR2,
      refHistograms   IN SDO_GEOR_HISTOGRAM_ARRAY,
      storageParam    IN VARCHAR2,
      outGeoraster    IN OUT SDO_GEORASTER,
      parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.histogramMatch(
      inGeoRaster  IN SDO_GEORASTER,
      pyramidLevel IN NUMBER,
      cropArea     IN SDO_NUMBER_ARRAY,
      bandNumbers  IN VARCHAR2,
      refGeoRaster IN SDO_GEORASTER,
      storageParam IN VARCHAR2,
      outGeoraster IN OUT SDO_GEORASTER,
      parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.histogramMatch(
      inGeoRaster  IN SDO_GEORASTER,
      pyramidLevel IN NUMBER,
      cropArea     IN SDO_GEOMETRY,
      layerNumbers IN VARCHAR2,
      refGeoRaster IN SDO_GEORASTER,
      storageParam IN VARCHAR2,
      outGeoraster IN OUT SDO_GEORASTER,
      parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Processes the input GeoRaster object so that the histograms of the output GeoRaster object matches the histogram of a reference GeoRaster object (refGeoRaster) or a reference histogram (refHistograms).

Parameters

inGeoRaster
The SDO_GEORASTER object for the histogram matching operation.

pyramidLevel
A number specifying the pyramid level for the histogram matching operation. If null, the default is 0.

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.

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).

refHistograms
An array of reference histograms. If there is only one element in the array, all the output bands match to this histogram; otherwise, the length of the array must be the same as the number of the bands in the output GeoRaster object.

refGeoRaster
The reference GeoRaster object. The reference GeoRaster object must have a histogram stored in the metadata, and must have the same number of bands as the input GeoRaster object.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as inGeoRaster.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form parallel=*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

The input GeoRaster object must have histograms generated for each band and stored in the metadata.

The reference GeoRaster object must have the same number of bands as the input GeoRaster object. The cell depth of the reference GeoRaster cannot be greater than the cell depth of the outGeoRaster object. The reference GeoRaster object must have histograms set in the metadata.

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 `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 GeoRaster Data Model.)

Color map stored in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

Examples

The following example creates a GeoRaster object that is the result of histogram matching the input GeoRaster object to the reference histograms. In the example, the reference histograms are stored in a table `histogram_table(histogram sdo_geor_histogram, band number)`. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_number_array := null;
  refHists sdo_geor_histogram_array;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
    VALUES (41, sdo_geor.init('RDT_1'))
    RETURNING georaster INTO gr2;

  -- get the source GeoRaster object
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=4;

  -- get the reference histogram
  SELECT histogram bulk collect into refHists from histogram_table order by band;

  sdo_geor_ip.histogramMatch(gr1, 0, cropArea, null, refHists, null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

The following example creates a GeoRaster object that is the result of histogram matching the input GeoRaster object to the reference GeoRaster object. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  refgr sdo_georaster;
  cropArea sdo_number_array := null;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
    VALUES (41, sdo_geor.init('RDT_1'))
    RETURNING georaster INTO gr2;

  -- get the source GeoRaster object
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=4;
```

```
       -- get the reference GeoRaster object
       SELECT georaster INTO refgr FROM georaster_table WHERE georid=5;

       sdo_geor_ip.histogramMatch(gr1, 0, cropArea, null, refgr, null, gr2);
       UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
       COMMIT;
     END;
     /
```

10.5 SDO_GEOR_IP.normalize

Format

```
SDO_GEOR_IP.normalize(
     inGeoRaster        IN SDO_GEORASTER,
     pyramidLevel       IN NUMBER,
     cropArea           IN SDO_NUMBER_ARRAY,
     bandNumbers        IN VARCHAR2,
     means              IN SDO_NUMBER_ARRAY,
     standardDeviations IN SDO_NUMBER_ARRAY,
     storageParam       IN VARCHAR2,
     outGeoraster       IN OUT SDO_GEORASTER,
     parallelParam      IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.normalize(
     inGeoRaster        IN SDO_GEORASTER,
     pyramidLevel       IN NUMBER,
     cropArea           IN SDO_GEOMETRY,
     layerNumbers       IN VARCHAR2,
     means              IN SDO_NUMBER_ARRAY,
     standardDeviations IN SDO_NUMBER_ARRAY,
     storageParam       IN VARCHAR2,
     outGeoraster       IN OUT SDO_GEORASTER,
     parallelParam      IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.normalize(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea     IN SDO_NUMBER_ARRAY,
     bandNumbers  IN VARCHAR2,
     refGeoRaster IN SDO_GEORASTER,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.normalize(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea     IN SDO_GEOMETRY,
     layerNumbers IN VARCHAR2,
     refGeoRaster IN SDO_GEORASTER,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Normalizes the input GeoRaster object using the specified mean and standard deviation.

Parameters

inGeoRaster
The SDO_GEORASTER object to be stretched.

pyramidLevel
A number specifying the pyramid level to be stretched in the source GeoRaster object. If null, the default is 0.

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.

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).

means
The target mean values for each output band. If only one value is specified, it is applied to all the output bands; otherwise, it must have the same number of values as the number of bands of the output GeoRaster object. The target mean values must be in the range of the cell depth of the `outGeoRaster` object. If null, it defaults to 128.

standardDeviations
The target standard deviation values for each output band. If only one value is specified, it is applied to all the output bands; otherwise, it must have the same number of values as the number of bands of the output GeoRaster object. The target standard deviation values must be in the range of the cell depth of the `outGeoRaster` object. If null, it defaults to 100.

refGeoRaster
The reference GeoRaster object. Instead of giving the target mean and standard deviation, the mean and standard deviation of `refGeoRaster` are used as the target. The reference GeoRaster object must have the same number of bands as the input GeoRaster object.

storageParam

A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster

The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as inGeoRaster.

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form parallel=n, where n is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

The input GeoRaster object must have statistics generated for each output band.

The reference GeoRaster object must have the same number of bands as the input GeoRaster object. The cell depth of the refGeoRaster object cannot be greater than the cell depth of the outGeoRaster object. The reference GeoRaster object must have statistics set in the metadata.

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 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 GeoRaster Data Model.)

Color map in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

Examples

The following example creates a GeoRaster object and performs normalization on the bands of the input GeoRaster object based on the given means and standard deviations. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_number_array := null;
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_ip.normalize(gr1, 0, cropArea, null, sdo_number_array(50, 80, 100),
sdo_number_array(30, 20, 50), null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

The following example creates a GeoRaster object and performs normalization on the
bands of the input GeoRaster object based on the given reference GeoRaster object .
(It refers to a table named GEORASTER_TABLE, whose definition is presented after
Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  refgr sdo_georaster;
  cropArea sdo_number_array := null;
BEGIN
  INSERT INTO georaster_table (georid, georaster)
    VALUES (41, sdo_geor.init('RDT_1'))
    RETURNING georaster INTO gr2;

  -- get the source GeoRaster object
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=4;

  -- get the reference GeoRaster object
  SELECT georaster INTO refgr FROM georaster_table WHERE georid=5;

  sdo_geor_ip.normalize(gr1, 0, cropArea, null, refgr, null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

10.6 SDO_GEOR_IP.piecewiseStretch

Format

```
SDO_GEOR_IP.piecewiseStretch(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea     IN SDO_NUMBER_ARRAY,
     bandNumbers  IN VARCHAR2,
     inValues     IN SDO_NUMBER_ARRAYSET DEFAULT NULL,
     outValues    IN SDO_NUMBER_ARRAYSET DEFAULT NULL,
     storageParam IN VARCHAR2,
     outGeoraster IN OUT SDO_GEORASTER,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.piecewiseStretch(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel IN NUMBER,
     cropArea     IN SDO_GEOMETRY,
     layerNumbers IN VARCHAR2,
     inValues     IN SDO_NUMBER_ARRAYSET DEFAULT NULL,
     outValues    IN SDO_NUMBER_ARRAYSET DEFAULT NULL,
```

```
storageParam IN VARCHAR2,
outGeoraster IN OUT SDO_GEORASTER,
parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Performs a linear stretch on the input GeoRaster object using the minimum and maximum values. The processed image is stored in the output GeoRaster object.

Parameters

inGeoRaster
The SDO_GEORASTER object to be stretched.

pyramidLevel
A number specifying the pyramid level to be stretched in the source GeoRaster object. If null, the default is 0.

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.

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).

inValues
An array of SDO_NUMBER_ARRAY objects corresponding to each input band. If only one element is specified, all the output bands are stretched in the same way. Each SDO_NUMBER_ARRAY specifies the input GeoRaster object cell value ranges used in the stretch. For example, sdo_number_array(0, 50, 51, 255) specifies two value ranges, from 0 to 50 and 51 to 255.

outValues
An array of SDO_NUMBER_ARRAY objects corresponding to each output band. If only one element is specified, all the output bands are stretched in the same way. Each SDO_NUMBER_ARRAY specifies the target values ranges corresponding to the value ranges of the `inValues` parameter. For example, sdo_number_array(0, 200, 200, 255), specifies two value ranges from 0 to 200 and 200 to 255.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as inGeoRaster.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form parallel=*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

The inValues and outValues parameters must have the same number of values, and the range specified in the inValues parameter must be in ascending order. The values in the inValues parameter must be in the range of the cell depth of the inGeoRaster object. The values in the outValues parameter must be in the range of the cell depth of the outGeoRaster object.

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

- Null, to specify raster space

- A value from the SRID column of the MDSYS.CS_SRS table

    If the SDO_SRID values for the cropArea parameter geometry and the model space are different, the 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 GeoRaster Data Model.)

Color map stored in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

Examples

The following example creates a GeoRaster object that piecewise stretches all bands of the input GeoRaster object into different cell depths: a linear stretch of input values between 0 and 255 to the range 0 to 2000. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_number_array := null;
  inValues sdo_number_arrayset;
  outValues sdo_number_arrayset;
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;
```

```
    -- define the input and output value ranges
    inValues := sdo_number_arrayset(sdo_number_array(0, 255));
    outValues := sdo_number_arrayset(sdo_number_array(0, 2000));

    sdo_geor_ip.piecewisestretch(gr1, 0, cropArea, null,
                  inValues, outValues, 'celldepth=16bit_u', gr2);
    UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
    COMMIT;
END;
/
```

The following example piecewise stretches each band for different ranges. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_number_array := null;
  inValues sdo_number_arrayset;
  outValues sdo_number_arrayset;
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;

  -- define the input and output values ranges
  inValues := sdo_number_arrayset(sdo_number_array(0, 30, 30, 80, 80, 255),
                                  sdo_number_array(0, 10, 10, 50, 50, 255),
                                  sdo_number_array(0, 50, 50, 150, 150, 255));
  outValues := sdo_number_arrayset(
                  sdo_number_array(0, 80, 80, 200, 200, 255),
                  sdo_number_array(0, 60, 50, 150, 150, 255),
                  sdo_number_array(0, 100, 100, 250, 250,255) );

  sdo_geor_ip.piecewisestretch(gr1, 0, cropArea, null, inValues, outValues,
                                      null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

10.7 SDO_GEOR_IP.stretch

Format

```
SDO_GEOR_IP.stretch(
     inGeoRaster  IN SDO_GEORASTER,
     pyramidLevel  IN NUMBER,
     cropArea      IN SDO_NUMBER_ARRAY,
     bandNumbers   IN VARCHAR2,
     minValues     IN SDO_NUMBER_ARRAY DEFAULT NULL,
     maxValues     IN SDO_NUMBER_ARRAY DEFAULT NULL,
     storageParam  IN VARCHAR2,
     outGeoraster  IN OUT SDO_GEORASTER),
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_IP.stretch(
     inGeoRaster    IN SDO_GEORASTER,
     pyramidLevel   IN NUMBER,
     cropArea       IN SDO_GEOMETRY,
     layerNumbers   IN VARCHAR2,
     minValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     max_values     IN SDO_NUMBER_ARRAY DEFAULT NULL,
     storageParam   IN VARCHAR2,
     outGeoraster   IN OUT SDO_GEORASTER),
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Performs a min-max linear stretch on the input GeoRaster object using the minimum and maximum values. The processed image is stored in the output GeoRaster object.

Parameters

inGeoRaster
The SDO_GEORASTER object to be stretched.

pyramidLevel
A number specifying the pyramid level to be stretched in the source GeoRaster object. If null, the default is 0.

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.

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).

minValues
The minimum values of the bands to be stretched. If it is null, the minimum value of the input GeoRaster object that is stored in the layer metadata is used. If there is only one value in the SDO_NUMBER_ARRAY, it is used as the minimum value of all the input bands; otherwise, the number of values in the SDO_NUMBER_ARRAY must be the same as the number of bands of the output GeoRaster object.

maxValues
The maximum values of the bands to be stretched. If it is null, the maximum value of the input GeoRaster object that is stored in the layer metadata is used. If there is only one value in the SDO_NUMBER_ARRAY, it is used as the maximum value of all the input bands; otherwise, the number of values in the SDO_NUMBER_ARRAY must be the same as the number of bands of the output GeoRaster object.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
The output SDO_GEORASTER object that reflects the results of the operation. Must be either a valid existing GeoRaster object or an empty GeoRaster object. (Empty GeoRaster objects are explained in Blank and Empty GeoRaster Objects.) Cannot be the same GeoRaster object as `inGeoRaster`.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
If parallelism is specified, the procedure performs an internal commit while the process is running. Therefore, you cannot roll back the results of this procedure. If an error occurs (even if it is raised by the Oracle parallel server), you must delete the resulting output GeoRaster object explicitly in order to roll back the operation.

Usage Notes

The input GeoRaster object is stretched to 0-255 using the values in `minValues` and `maxValues` parameters, or the minimum and maximum values stored in the statistics metadata of the image. If the `minValues` and `maxValues` parameters are null, the input GeoRaster object must have the statistics set in its metadata.

The cell depth of the output GeoRaster object is always `8BIT_U`. When a GeoRaster image needs to be stretched to an image with a cell depth other than `8BIT_U`, you can use SDO_GEOR_IP.piecewiseStretch.

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 `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 GeoRaster Data Model.)

Color map stored in the input GeoRaster object is not supported.

The output GeoRaster object has no pyramid or mask.

Examples

The following example creates a GeoRaster object that stretches band 2 of the input GeoRaster object from value range of (30, 150) to (0, 255). (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
  cropArea sdo_number_array := null;
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_ip.stretch(gr1, 0, cropArea, '2', sdo_number_array(30),
sdo_number_array(150), null, gr2);
  UPDATE georaster_table SET georaster=gr2 WHERE georid=41;
  COMMIT;
END;
/
```

11

SDO_GEOR_RA Package Reference

The SDO_GEOR_RA package contains subprograms (functions and procedures) for raster algebra and analytic operations related to GeoRaster. This chapter presents reference information, with one or more examples, for each subprogram.

Topics:

11.1 SDO_GEOR_RA.classify

Format

```
SDO_GEOR_RA.classify(
     inGeoRaster    IN SDO_GEORASTER,
     expression     IN VARCHAR2,
     rangeArray     IN SDO_NUMBER_ARRAY,
     valueArray     IN SDO_NUMBER_ARRAY,
     storageParam   IN VARCHAR2,
     outGeoRaster   IN OUT SDO_GEORASTER,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue    IN NUMBER DEFAULT 0,
     parallelParam  IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
     inGeoRaster    IN SDO_GEORASTER,
     cropArea       IN SDO_NUMBER_ARRAY,
     expression     IN VARCHAR2,
     rangeArray     IN SDO_NUMBER_ARRAY,
     valueArray     IN SDO_NUMBER_ARRAY,
     storageParam   IN VARCHAR2,
     outGeoRaster   IN OUT SDO_GEORASTER,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue    IN NUMBER DEFAULT 0,
     parallelParam  IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
     inGeoRaster   IN SDO_GEORASTER,
     cropArea      IN SDO_GEOMETRY,
     expression    IN VARCHAR2,
     rangeArray    IN SDO_NUMBER_ARRAY,
     valueArray    IN SDO_NUMBER_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
     georArray     IN SDO_GEORASTER_ARRAY,
     expression    IN VARCHAR2,
     rangeArray    IN SDO_NUMBER_ARRAY,
     valueArray    IN SDO_NUMBER_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
     georArray     IN SDO_GEORASTER_ARRAY,
     cropArea      IN SDO_NUMBER_ARRAY,
     expression    IN VARCHAR2,
     rangeArray    IN SDO_NUMBER_ARRAY,
     valueArray    IN SDO_NUMBER_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
     georArray     IN SDO_GEORASTER_ARRAY,
     cropArea      IN SDO_GEOMETRY,
     expression    IN VARCHAR2,
     rangeArray    IN SDO_NUMBER_ARRAY,
     valueArray    IN SDO_NUMBER_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
     inGeoRasters  IN SYS_REFCURSOR,
     expression    IN VARCHAR2,
     rangeArray    IN SDO_NUMBER_ARRAY,
     valueArray    IN SDO_NUMBER_ARRAY,
```

```
        storageParam  IN VARCHAR2,
        outGeoRaster  IN OUT SDO_GEORASTER,
        nodata        IN VARCHAR2 DEFAULT 'FALSE',
        nodataValue   IN NUMBER DEFAULT 0,
        parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
        inGeoRasters  IN SYS_REFCURSOR,
        cropArea      IN SDO_NUMBER_ARRAY,
        expression    IN VARCHAR2,
        rangeArray    IN SDO_NUMBER_ARRAY,
        valueArray    IN SDO_NUMBER_ARRAY,
        storageParam  IN VARCHAR2,
        outGeoRaster  IN OUT SDO_GEORASTER,
        nodata        IN VARCHAR2 DEFAULT 'FALSE',
        nodataValue   IN NUMBER DEFAULT 0,
        parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.classify(
        inGeoRasters  IN SYS_REFCURSOR,
        cropArea      IN SDO_GEOMETRY,
        expression    IN VARCHAR2,
        rangeArray    IN SDO_NUMBER_ARRAY,
        valueArray    IN SDO_NUMBER_ARRAY,
        storageParam  IN VARCHAR2,
        outGeoRaster  IN OUT SDO_GEORASTER,
        nodata        IN VARCHAR2 DEFAULT 'FALSE',
        nodataValue   IN NUMBER DEFAULT 0,
        polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
        parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Generates a new GeoRaster object after applying the specified classification operation on the input GeoRaster object or objects.

There are three formats for each of three input GeoRaster object or objects specification options (SDO_GEORASTER, SDO_GEORASTER_ARRAY, SYS_REEFCURSOR). Within each group of three, you can specify no crop area or a crop area of SDO_NUMBER_ARRAY or SDO_GEOMETRY; and for an SDO_GEOMETRY crop area, you can specify a polygon clip option.

Parameters

inGeoRaster
Input GeoRaster object.

georArray
An array of GeoRaster objects. The data type is SDO_GEOR_ARRAY, which is defined as `VARRAY(10485760) OF SDO_GEORASTER`.

inGeoRasters
Cursor (SYS_REFCURSOR type) for the input GeoRaster objects.

cropArea
Crop area definition. If the data type is SDO_GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as a rectangular crop area to generate the output GeoRaster object. If the parameter `polygonClip` is `TRUE`, then only cells within the crop area geometry are processed, and all cells outside the crop area geometry are set to zero (0). If the parameter `polygonClip` is `FALSE`, then all cells within the minimum bounding rectangle are processed.
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.

expression
An arithmetic expression used to classify cell values. See the Usage Notes for more information about specifying this parameter.

rangeArray
A number array that defines ranges for classifying cell values. The array must contain at least one element.

valueArray
A number array that defines the target cell value for each range. The number of elements must be 1 greater than the elements in `rangeArray` (that is, its length must be `rangeArray`+1).

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
Output GeoRaster object.

nodata
The string `TRUE` specifies that for any NODATA cells in an input GeoRaster object, the corresponding cells in the output GeoRaster object are to be set to the value specified for the `nodataValue` parameter. The string `FALSE` (the default) causes cells with NODATA values to be considered as regular data. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

nodataValue
The value used to set NODATA cells if the `nodata` parameter value is the string `TRUE`.

polygonClip
Ignored if `cropArea` is null. Otherwise, the string `TRUE` causes the `cropArea` geometry to be used to process the data; the string `FALSE` or a null value causes the minimum bounding rectangle (MBR) of the `cropArea` geometry to be used to process the data.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

Usage Notes

This procedure generates a one-layer GeoRaster object based on the input GeoRaster object or objects and the `expression` parameter, which is an arithmetic expression

string. For each cell in the output GeoRaster object, `expression` is evaluated against corresponding cell values in the input GeoRaster object, and the following algorithm is used to calculate cell values of the output GeoRaster object:

```
if  (value of expression < rangeArray[0])
    cellValue=valueArray[0]
else if (value of expression >= rangeArray[n-1])
    cellValue=valueArray[n]
else if rangeArray[m-1] <= value of expression < rangeArray[m]
     cellValue=valueArray[m]
```

In the `expression` calculation:

- Length of `rangeArray` is n
- Length of `valueArray` is $n+1$
- $0 < m < n-1$

For more information, see Raster Algebra Language.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

Examples

The following example classifies cell values based on the cell values of the first layer.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  rangeArray SDO_NUMBER_ARRAY;
  valueArray SDO_NUMBER_ARRAY;
BEGIN
  rangeArray:=sdo_number_array(70,80,90,100,110,120,130,140,150,160,170,180);
  valueArray:=sdo_number_array(70,80,90,100,110,120,130,140,150,160,170,180,190);
  select georaster into geor from georaster_table where georid = 1;
  insert into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
georaster into geor1;
  sdo_geor_ra.classify(geor,'{0}',rangeArray,valueArray,null,geor1);
  update georaster_table set georaster = geor1 where georid = 5;
  commit;
END;
/
```

11.2 SDO_GEOR_RA.diff

Format

```
SDO_GEOR_RA.diff(
     georaster1    IN SDO_GEORASTER,
     georaster2    IN SDO_GEORASTER,
     cropArea      IN SDO_NUMBER_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.diff(
     georaster1    IN SDO_GEORASTER,
     georaster2    IN SDO_GEORASTER,
     cropArea      IN SDO_GEOMETRY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     ploygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Generates a new GeoRaster object by performing the diff operation (explained in the Usage Notes).

Parameters

georaster1
First input GeoRaster object.

georaster2
Second input GeoRaster object.

cropArea
Crop area definition. If the data type is SDO_GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as a rectangular crop area to generate the output GeoRaster object. If the parameter `polygonClip` is `TRUE`, then only cells within the crop area geometry are processed, and all cells outside the crop area geometry are set to zero (0). See also the Usage Notes for SDO_GEOR.reproject for SDO_SRID requirements.
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.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
Output GeoRaster object.

nodata
The string `TRUE` specifies that for any NODATA cells in an input GeoRaster object, the corresponding cells in the output GeoRaster object are to be set to the value specified for the `nodataValue` parameter. The string `FALSE` (the default) causes cells with NODATA values to be considered as regular data. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

nodataValue
The value used to set NODATA cells if the `nodata` parameter value is the string `TRUE`.

polygonClip
Ignored if `cropArea` is null. Otherwise, the string `TRUE` causes the `cropArea` value to be used to crop the mosaicked data; the string `FALSE` or a null value causes the MBR of `cropArea` to be used to crop the output image.

parallelParam

Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=n`, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

Usage Notes

This procedure generates a new GeoRaster object by taking two input GeoRaster objects and applying the over operation: specifically, for each cell value in each layer, if the values in the two input objects are different, the value on the first input is output, but if the values in the two input objects are the same, the output is zero.

The two input GeoRaster objects must have same dimension size and same number of bands; otherwise, the ORA-13397 error is generated.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

For more information, see Logical Operations.

Examples

The following example performs the diff operation on the two input GeoRaster objects.

```
declare
  geor        MDSYS.SDO_GEORASTER;
  geor1       MDSYS.SDO_GEORASTER;
  geor2       MDSYS.SDO_GEORASTER;
  geom        mdsys.sdo_geometry;
begin
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101;
  select georaster into geor2 from georaster_table where georid = 102 for update;
  geom:=null;
  mdsys.sdo_geor_ra.diff(geor,geor1,geom,null,geor2);
  update georaster_table set georaster = geor2 where georid = 102;
end;
/
```

11.3 SDO_GEOR_RA.findCells

Format

```
SDO_GEOR_AGGR.findCells(
     inGeoRaster   IN  SDO_GEORASTER,
     condition     IN  VARCHAR2,
     storageParam  IN  VARCHAR2
     outGeoRaster  OUT SDO_GEORASTER,
     bgValues      IN  SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN  VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN  VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.findCells(
     inGeoRaster   IN SDO_GEORASTER,
     cropArea      IN SDO_NUMBER_ARRAY,
     condition     IN VARCHAR2,
     storageParam  IN VARCHAR2
     outGeoRaster  OUT SDO_GEORASTER,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_AGGR.findCells(
     inGeoRaster   IN SDO_GEORASTER,
     cropArea      IN SDO_GEOMETRY,
     condition     IN VARCHAR2,
     storageParam  IN VARCHAR2
     outGeoRaster  OUT SDO_GEORASTER,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Generates a new GeoRaster object based on the input GeoRaster object, but masking all cells that do not satisfy the `condition` parameter specification.

Parameters

inGeoRaster
Input GeoRaster object.

cropArea
Crop area definition. If the data type is SDO_GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as a rectangular crop area to generate the output GeoRaster object; see also the Usage Notes for SDO_GEOR.reproject for SDO_SRID requirements.. If the parameter `polygonClip` is `TRUE`, then only cells within the crop area geometry are processed, and all cells outside the crop area geometry are set to zero (0). If the parameter `polygonClip` is `FALSE`, then all cells within the minimum bounding rectangle are processed.
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.

condition
An expression string used to filter out cells. (See the Usage Notes for more information.).

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
Output GeoRaster object.

bgValues
Background values to represent values of cells in the empty raster blocks of the input GeoRaster object. 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 default `bgValues` are zero (0).
The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

nodata
The string `TRUE` specifies to keep the original values for any NODATA cells in the GeoRaster object. The string `FALSE` (the default) causes cells with NODATA values to be considered as regular data. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

polygonClip
Ignored if `cropArea` is null. Otherwise, the string `TRUE` causes the `cropArea` geometry to be used to process the data; the string `FALSE` or a null value causes the minimum bounding rectangle (MBR) of the `cropArea` geometry to be used to process the data.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

Usage Notes

This procedure generates a new GeoRaster object based on the input GeoRaster object and the `condition` parameter, which is `booleanExpr`, a Boolean expression string. For each cell in the output GeoRaster object, condition is evaluated against corresponding cell values in the input GeoRaster object. If `condition` is true for a cell, the original cell value is kept in the output GeoRaster object; otherwise, `bgValues` are filled for the cell in the output GeoRaster object

For more information, see Raster Algebra Language.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

Examples

The following example changes cell values to default background values 0, if cell value of the second layer is less than or equal to 200.

```
DECLARE
  geor  SDO_GEORASTER;
  geor1 SDO_GEORASTER;
BEGIN
  select georaster into geor from georaster_table where georid = 1;
  insert into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
georaster into geor1;
  sdo_geor_ra.findcells(geor, '{1}>200',null,geor1);
  update georaster_table set georaster = geor1 where georid = 5;
  commit;
```

```
END;
/
```

The following example uses a geometry object (`geom`) as the input `cropArea`.

```
DECLARE
  geor        SDO_GEORASTER;
  geor0       SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  geom        SDO_GEOMETRY;
BEGIN
  geom:= sdo_geometry(2003,82394, NULL,
                        sdo_elem_info_array(1, 1003, 1),
                        sdo_ordinate_array(21783.775, 1008687.9,
                                            18783.775, 966687.905,
                                            63783.775, 966687.905,
                                            81783.775, 990687.905,
                                            21783.775, 1008687.9));
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101 for update;
  sdo_geor_ra.findcells(geor,geom,'({1}=42)','blocking=true,
blocksize=(256,256,3)',geor1,null,'false');
  update georaster_table set georaster = geor1 where georid = 101;
END;
/
```

11.4 SDO_GEOR_RA.isOverlap

Format

```
SDO_GEOR.isOverlap(
     georaster1 IN SDO_GEORASTER,
     georaster2 IN SDO_GEORASTER,
     tolerance  IN NUMBER DEFAULT 0.5
     ) RETURN VARCHAR2;
```

or

```
SDO_GEOR_RA.isOverlap(
     georArray IN SDO_GEORASTER_ARRAY,
     tolerance IN NUMBER DEFAULT 0.5
     ) RETURN VARCHAR2;
```

or

```
SDO_GEOR_RA.isOverlap(
     geor_cur  IN SYS_REFCURSOR,
     tolerance IN NUMBER DEFAULT 0.5
     ) RETURN VARCHAR2;
```

Description

Returns the string TRUE if two or more GeoRaster objects overlap, or FALSE if two or more GeoRaster objects do not overlap. (See the Usage Notes for the logic used to determine of two GeoRaster objects, whether georeferenced or not, overlap.)

Parameters

georaster1
GeoRaster object.

georaster2
GeoRaster object.

georArray
An array of GeoRaster objects. The data type is SDO_GEOR_ARRAY, which is defined as `VARRAY(10485760) OF SDO_GEORASTER`.

geor_cur
Cursor (SYS_REFCURSOR type) for the input GeoRaster objects.

tolerance
Tolerance value used to determine if two cells in the cell space overlap in the model space. The value should be between 0 and 1, and the unit is cell. For example, 0.5 (the default) means one-half cell, namely, that two cells overlap if the distance between them in 0.5 cell or less.

Usage Notes

The GeoRaster objects being compared for overlap must be either all georeferenced or all non-georeferenced.

The following logic is applied to determine if two GeoRaster objects overlap:

1. If the row or column dimension size of two GeoRaster objects is different, then return `'FALSE'`. Otherwise, continue to the next step.

2. Check if both GeoRaster objects are georeferenced.

    a. If one is georeferenced and the other one is not, then return `'FALSE'`.

    b. If both are non-georeferenced, and if the `ultCoordinate` of both GeoRaster objects is the same, then return `'TRUE'`; else, return `'FALSE'`.

    c. If both are georeferenced, go to the next step.

3. Check the `pType`, `nVars`, `order`, and `nCoefficients` values (explained in Functional Fitting Georeferencing Model) of the `p`, `q`, `r`, and `s` polynomials. If any are different, then return 'FALSE'; else, go to the next step.

4. Calculate the upper-left, upper-right, lower-left, and lower-right four points from cell space to model space. If the distance of corresponding points of the two GeoRaster objects is within the tolerance value (converted from cell space to model space), then return `'TRUE'`; else, return `'FALSE'`.

The raster algebra functions of GeoRaster require the raster layers from different GeoRaster objects have the same size and completely overlap each other. Before you apply raster algebra operations over two or more GeoRaster objects or perform other operations, you can use the SDO_GEOR_RA.isOverlap function to determine if the GeoRaster objects are of the same size and cover the same ground area.

Examples

The following examples check if two GeoRaster objects overlap. (They use two different formats of the function.)

```
DECLARE
  geor       MDSYS.SDO_GEORASTER;
  geor1      MDSYS.SDO_GEORASTER;
BEGIN

  SELECT georaster INTO geor FROM georaster_table WHERE georid = 1;
  SELECT georaster INTO geor1 FROM georaster_table WHERE georid = 30;
  dbms_output.put_line(sdo_geor_ra.isOverlap(geor,geor1,0.5));
END;
/

DECLARE
  mycursor   sys_refcursor;
BEGIN
  OPEN mycursor FOR
    SELECT georaster FROM georaster_table WHERE georid = 1 or georid=30;
  dbms_output.put_line(sdo_geor_ra.isOverlap(mycursor,0.5));
END;
/
```

11.5 SDO_GEOR_RA.over

Format

```
SDO_GEOR_RA.over(
     georaster1    IN SDO_GEORASTER,
     georaster2    IN SDO_GEORASTER,
     cropArea      IN SDO_NUMBER_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.over(
     georaster1    IN SDO_GEORASTER,
     georaster2    IN SDO_GEORASTER,
     cropArea      IN SDO_GEOMETRY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     ploygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Generates a new GeoRaster object by performing the over operation (explained in the Usage Notes).

Parameters

georaster1
First input GeoRaster object.

georaster2
Second input GeoRaster object.

cropArea
Crop area definition. If the data type is SDO_GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as a rectangular crop area to generate the output GeoRaster object. If the parameter `polygonClip` is `TRUE`, then only cells within the crop area geometry are processed, and all cells outside the crop area geometry are set to zero (0). See also the Usage Notes for SDO_GEOR.reproject for SDO_SRID requirements.
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.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
Output GeoRaster object.

nodata
The string `TRUE` specifies that for any NODATA cells in an input GeoRaster object, the corresponding cells in the output GeoRaster object are to be set to the value specified for the `nodataValue` parameter. The string `FALSE` (the default) causes cells with NODATA values to be considered as regular data. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

nodataValue
The value used to set NODATA cells if the `nodata` parameter value is the string `TRUE`.

polygonClip
Ignored if `cropArea` is null. Otherwise, the string `TRUE` causes the `cropArea` value to be used to crop the mosaicked data; the string `FALSE` or a null value causes the MBR of `cropArea` to be used to crop the output image.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

Usage Notes

This procedure generates a new GeoRaster object by taking two input GeoRaster objects and applying the over operation: specifically, each cell in each layer in the output GeoRaster object will contain the value from the corresponding cell in first input GeoRaster object if that value is not zero; otherwise, it will contain the value of the corresponding cell in the second GeoRaster object.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

For more information, see Logical Operations.

Examples

The following example performs the over operation on the two input GeoRaster objects.

```
declare
  geor      MDSYS.SDO_GEORASTER;
  geor1     MDSYS.SDO_GEORASTER;
  geor2     MDSYS.SDO_GEORASTER;
  geom      mdsys.sdo_geometry;
begin
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101;
  select georaster into geor2 from georaster_table where georid = 102 for update;
  geom:=null;
  mdsys.sdo_geor_ra.over(geor,geor1,geom,null,geor2);
  update georaster_table set georaster = geor2 where georid = 102;
end;
/
```

11.6 SDO_GEOR_RA.rasterMathOp

Format

```
SDO_GEOR_RA.rasterMathOp(
     inGeoRaster   IN SDO_GEORASTER,
     operation     IN SDO_STRING2_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);


SDO_GEOR_RA.rasterMathOp(
     inGeoRaster   IN SDO_GEORASTER,
     cropArea      IN SDO_NUMBER_ARRAY,
     operation     IN SDO_STRING2_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);


SDO_GEOR_RA.rasterMathOp(
     inGeoRaster   IN SDO_GEORASTER,
     cropArea      IN SDO_GEOMETRY,
     operation     IN SDO_STRING2_ARRAY,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.rasterMathOp(
     georArray     IN SDO_GEORASTER_ARRAY,
     operation     IN SDO_STRING2_ARRAY,
```

```
      storageParam  IN VARCHAR2,
      outGeoRaster  IN OUT SDO_GEORASTER,
      nodata        IN VARCHAR2 DEFAULT 'FALSE',
      nodataValue   IN NUMBER DEFAULT 0,
      parallelParam IN VARCHAR2 DEFAULT NULL);

SDO_GEOR_RA.rasterMathOp(
      georArray     IN SDO_GEORASTER_ARRAY,
      cropArea      IN SDO_NUMBER_ARRAY,
      operation     IN SDO_STRING2_ARRAY,
      storageParam  IN VARCHAR2,
      outGeoRaster  IN OUT SDO_GEORASTER,
      nodata        IN VARCHAR2 DEFAULT 'FALSE',
      nodataValue   IN NUMBER DEFAULT 0,
      parallelParam IN VARCHAR2 DEFAULT NULL);

SDO_GEOR_RA.rasterMathOp(
      georArray     IN SDO_GEORASTER_ARRAY,
      cropArea      IN SDO_GEOMETRY,
      operation     IN SDO_STRING2_ARRAY,
      storageParam  IN VARCHAR2,
      outGeoRaster  IN OUT SDO_GEORASTER,
      nodata        IN VARCHAR2 DEFAULT 'FALSE',
      nodataValue   IN NUMBER DEFAULT 0,
      polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
      parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.rasterMathOp(
      inGeoRasters  IN SYS_REFCURSOR,
      operation     IN SDO_STRING2_ARRAY,
      storageParam  IN VARCHAR2,
      outGeoRaster  IN OUT SDO_GEORASTER,
      nodata        IN VARCHAR2 DEFAULT 'FALSE',
      nodataValue   IN NUMBER DEFAULT 0,
      parallelParam IN VARCHAR2 DEFAULT NULL);

SDO_GEOR_RA.rasterMathOp(
      inGeoRasters  IN SYS_REFCURSOR,
      cropArea      IN SDO_NUMBER_ARRAY,
      operation     IN SDO_STRING2_ARRAY,
      storageParam  IN VARCHAR2,
      outGeoRaster  IN OUT SDO_GEORASTER,
      nodata        IN VARCHAR2 DEFAULT 'FALSE',
      nodataValue   IN NUMBER DEFAULT 0,
      parallelParam IN VARCHAR2 DEFAULT NULL);

SDO_GEOR_RA.rasterMathOp(
      inGeoRasters  IN SYS_REFCURSOR,
      cropArea      IN SDO_GEOMETRY,
      operation     IN SDO_STRING2_ARRAY,
      storageParam  IN VARCHAR2,
      outGeoRaster  IN OUT SDO_GEORASTER,
      nodata        IN VARCHAR2 DEFAULT 'FALSE',
      nodataValue   IN NUMBER DEFAULT 0,
      polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
      parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

ORACLE

```
SDO_GEOR_RA.rasterMathOp(
     georaster0    IN SDO_GEORASTER,
     georaster1    IN SDO_GEORASTER,
     constant      IN NUMBER,
     operator      IN PLS_INTEGER,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);

SDO_GEOR_RA.rasterMathOp(
     georaster0    IN SDO_GEORASTER,
     georaster1    IN SDO_GEORASTER,
     cropArea      IN SDO_NUMBER_ARRAY,
     constant      IN NUMBER,
     operator      IN PLS_INTEGER,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     parallelParam IN VARCHAR2 DEFAULT NULL);

SDO_GEOR_RA.rasterMathOp(
     georaster0    IN SDO_GEORASTER,
     georaster1    IN SDO_GEORASTER,
     cropArea      IN SDO_GHEOMETRY,
     constant      IN NUMBER,
     operator      IN PLS_INTEGER,
     storageParam  IN VARCHAR2,
     outGeoRaster  IN OUT SDO_GEORASTER,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue   IN NUMBER DEFAULT 0,
     polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Performs a raster mathematical operation on one or more GeoRaster objects.

Parameters

inGeoRaster
Input GeoRaster object.

georArray
An array of GeoRaster objects. The data type is SDO_GEOR_ARRAY, which is defined as `VARRAY(10485760) OF SDO_GEORASTER`.

inGeoRasters
Cursor (SYS_REFCURSOR type) for the input GeoRaster objects.

georaster0
The left operand.

georaster1

The right operand.

cropArea

Crop area definition. If the data type is SDO_GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as a rectangular crop area to generate the output GeoRaster object. If the parameter `polygonClip` is `TRUE`, then only cells within the crop area geometry are processed, and all cells outside the crop area geometry are set to zero (0). See also the Usage Notes for SDO_GEOR.reproject for SDO_SRID requirements.

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.

operation

An array of `arithmeticExpr` expression strings used to calculate raster cell values in the output GeoRaster object. Each element of the array corresponds to a layer in the output GeoRaster object. The data type is SDO_STRING2_ARRAY, which is defined as `VARRAY(2147483647) OF VARCHAR2(4096)`.

The syntax for the `arithmeticExpr` expressions is explained in Raster Algebra Language.

constant

Constant value for some operators (see the `operator` parameter), such as `addConst` and `divConstant`.

operator

One of the following math operators, which are defined in the SDO_GEOR_RA package:

```
OPERATOR_ABSOLUTE
OPERATOR_ADD
OPERATOR_ADDCONST
OPERATOR_DIVIDE
OPERATOR_DIVIDECONST
OPERATOR_EXP
OPERATOR_INVERT
OPERATOR_LOG
OPERATOR_MULTIPLY
OPERATOR_MULTIPLYCONST
OPERATOR_SUBTRACT
OPERATOR_SUBTRACTCONST
```

For the definitions of these operators, see the Usage Notes.

storageParam

A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster

Output GeoRaster object.

bgValues

Background values to represent values of cells in the empty raster blocks of the input GeoRaster object. 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 default `bgValues` are zero (0).
The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

nodata
The string `TRUE` specifies that for any NODATA cells in an input GeoRaster object, the corresponding cells in the output GeoRaster object are to be set to the value specified for the `nodataValue` parameter. The string `FALSE` (the default) causes cells with NODATA values to be considered as regular data. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

nodataValue
The value used to set NODATA cells if the `nodata` parameter value is the string `TRUE`.

polygonClip
Ignored if `cropArea` is null. Otherwise, the string `TRUE` causes the `cropArea` value to be used to crop the mosaicked data; the string `FALSE` or a null value causes the MBR of `cropArea` to be used to crop the output image.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

Usage Notes

This procedure has 12 formats, which can be considered as 4 groups of 3 formats each:

- Operation on a single GeoRaster object

- Operation on an array of GeoRaster objects

- Operation on GeoRaster objects specified by a cursor (SYS_REFCURSOR type)

- Operation on all corresponding cells of each layer or one or two GeoRaster objjects

Within each group, the options include specifying a crop eras of type SDO_NUMBER_ARRAY or SDO_GEOMETRY, and for an SDO_GEOMETRY crop area, whether to use the crop area's geometry object or the MBR of that object.

The first three groups of formats are used to generate a new GeoRaster object from layers of one or more input GeoRaster objects based on the `operation` parameter. For example, the following example generates a new GeoRaster object that has three layers, and each layer's value is the cell value of the input GeoRaster object minus 10:

```
sdo_geor_ra.rasterMathOp(geor,SDO_STRING2_ARRAY('{0,0}-10','{0,1}-10','{0,2}-10'),nul
l,geor1);
```

The fourth group of formats applies a mathematical operation on all corresponding cells of each layer of input GeoRaster objects, and generates a new GeoRaster object with the same dimension size as the first input GeoRaster object (`geoRaster0`). The two input GeoRaster objects must have same row/column/band dimension size.

For the fourth group of formats, all pyramids are removed in the resulting GeoRaster object, but masks of the first input GeoRaster object are kept in the resulting GeoRaster object.

For the `operator` parameter, the operators have the following definitions:

```
OPERATOR_ABSOLUTE  :
 if (src[x][y][b] < 0) {
     dst[x][y][b] = -src[x][y][b];
 } else {
     dst[x][y][b] =  src[x][y][b];
 }

OPERATOR_ADD
   dst[x][y][b]=src1[x][y][b]+src2[x][y][b]
OPERATOR_ADDCONST
   dst[x][y][b]=src[x][y][b] +constant    --constant is the third parameter

OPERATOR_DIVIDE
   dst[x][y][b]=src1[x][y][b]/src2[x][y][b]

OPERATOR_DIVIDECONST
   dst[x][y][b]=src[x][y][b]/constant    --constant is the third parameter

OPERATOR_EXP
   dst[x][y][b]=exp(src[x][y][b])

OPERATOR_INVERT        :
   Inverts the cell values: dst[x][y][b]=-src[x][y][b]

OPERATOR_LOG           :
   dst[x][y][b]=log(src[x][y][b])

OPERATOR_MULTIPLY
   dst[x][y][b]=src1[x][y][b]*src2[x][y][b]

OPERATOR_MULTIPLYCONST
   dst[x][y][b]=src[x][y][b]*constant    --constant is the third parameter

OPERATOR_SUBTRACT
   dst[x][y][b]=src1[x][y][b]-src2[x][y][b]

OPERATOR_SUBTRACTCONST
   dst[x][y][b]=src[x][y][b]-constant    --constant is the third parameter
```

For more information about the raster algebra language, see Raster Algebra Language.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

Examples

The following example adds the constant 10 to all cell values of the input GeoRaster object.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
```

```
BEGIN

  select georaster into geor from georaster_table where georid = 1;
  insert into georaster_table values (5, sdo_geor.init('rdt_1', 5)) returning
georaster into geor1;
  sdo_geor_ra.rasterMathOp(geor,null,10,sdo_geor_ra.OPERATOR_ADDCONST,null,geor1);
  update georaster_table set georaster = geor1 where georid = 5;
  commit;
END;
/
```

The following example generates a new three-layer GeoRaster object from three
layers of the input GeoRaster object, and each cell value in the new GeoRaster object
is the value of the corresponding "old" cell divided by 2.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  geo_array   SDO_GEORASTER_ARRAY;
BEGIN
  select georaster into geor from georaster_table where georid = 2;
  insert into georaster_table values (20, sdo_geor.init('rdt_1', 20)) returning
georaster into geor1;
  geo_array:=SDO_GEORASTER_ARRAY(geor);
  sdo_geor_ra.rasterMathOp(geo_array,SDO_STRING2_ARRAY('{0,0}/2','{0,1}/2','{0,2}/
2'),null,geor1);
  update georaster_table set georaster = geor1 where georid = 20;
  commit;
END;
/
```

The following example performs a raster math operation with a geometry object as the
crop area.

```
DECLARE
  geor        SDO_GEORASTER;
  geor1       SDO_GEORASTER;
  geom        SDO_GEOMETRY;
  geo_array   SDO_GEORASTER_ARRAY;
BEGIN
  geom:= sdo_geometry(2003,82394, NULL,
                          sdo_elem_info_array(1, 1003, 1),
                          sdo_ordinate_array(21783.775, 1008687.9,
                                             18783.775, 966687.905,
                                             63783.775, 966687.905,
                                             81783.775, 990687.905,
                                             21783.775, 1008687.9));
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor1 from georaster_table where georid = 101 for update;
  geo_array:=SDO_GEORASTER_ARRAY(geor);
  mdsys.sdo_geor_ra.rasterMathOp(geo_array,geom,SDO_STRING2_ARRAY('{0,0}/2','{0,1}/
2','{0,2}/2'),null,geor1);
  update georaster_table set georaster = geor1 where georid = 101;
  commit;
END;
/
```

11.7 SDO_GEOR_RA.rasterUpdate

Format

```
SDO_GEOR_RA.rasterUpdate(
     geoRaster     IN OUT SDO_GEORASTER,
     pyramidLevel  IN NUMBER,
     conditions    IN SDO_STRING2_ARRAY,
     vals          IN SDO_STRING2_ARRAYSET,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.rasterUpdate(
     geoRaster     IN OUT SDO_GEORASTER,
     pyramidLevel  IN NUMBER,
     targetArea    IN SDO_NUMBER_ARRAY,
     conditions    IN SDO_STRING2_ARRAY,
     vals          IN SDO_STRING2_ARRAYSET,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.rasterUpdate(
     geoRaster     IN OUT SDO_GEORASTER,
     pyramidLevel  IN NUMBER,
     targetArea    IN SDO_GEOMETRY,
     conditions    IN SDO_STRING2_ARRAY,
     vals          IN SDO_STRING2_ARRAYSET,
     bgValues      IN SDO_NUMBER_ARRAY DEFAULT NULL,
     nodata        IN VARCHAR2 DEFAULT 'FALSE',
     polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
     parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Updates all cells for which the `conditions` specification is true, using values calculated from the `vals` specification.

Parameters

geoRaster
GeoRaster object that is used for input and for output (updating based on specified conditions).

pyramidLevel
Pyramid level to be updated. If this parameter is null, all pyramid levels are updated.

targetArea
Target area definition. If the data type is SDO_GEOMETRY, then if the parameter `polygonClip` is `TRUE`, only cells within the target area geometry are updated, and all cells outside the target area geometry keep original values; but if the

parameter `polygonClip` is `FALSE`, all cells in the MBR of the target area geometry are updated.
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.

conditions
An array of `booleanExpr` expression strings used to select cells. (See the Usage Notes for more information.) The data type is SDO_STRING2_ARRAY, which is defined as `VARRAY(2147483647) OF VARCHAR2(4096)`.

vals
An array or arrays of `arithmeticExpr` expressions, with the outer array corresponding to each condition and the inner array corresponding to each layer. The data type is SDO_STRING2_ARRAYSET, which is defined as `VARRAY(2147483647) OF SDO_STRING2_ARRAY`.

bgValues
Background values to represent values of cells in the empty raster blocks of the input GeoRaster object. 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 default `bgValues` are zero (0).
The filling values must be valid cell values as specified by the target cell depth background values for filling sparse data.

nodata
The string `TRUE` specifies that any NODATA cells in the GeoRaster object are not to be updated. The string `FALSE` (the default) causes cells with NODATA values to be considered as regular cells and thus eligible for updating. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

polygonClip
Ignored if `targetArea` is null. Otherwise, the string `TRUE` causes the `targetArea` geometry value to be used to update raster cell values; the string `FALSE` or a null value causes the MBR of `targetArea` geometry to be used to update raster cell values.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.)
Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

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 selects cells from the specified GeoRaster object based on `booleanExpr` strings specified in the `conditions` parameter, and updates corresponding cell values

by calculating `arithmeticExpr` expression strings specified in the **vals** parameter. For example, if:

```
conditions = SDO_STRING2_ARRAY('{0}=48','{0}=108')
vals =
SDO_STRING2_ARRAYSET(SDO_STRING2_ARRAY('123','54','89'),SDO_STRING2_ARRAY('98','56','
123'))
```

Then:

- For all cells whose first layer value equals 48, their first, second, and third layer values are set to 123,54,89, respectively.

- For all cells whose first layer value equals 108, their first, second, and third layer values are set to 98,56,123, respectively.

For more information, see Raster Algebra Language.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

Examples

The following example updates all cells for which the `conditions` specification is true, using values calculated from the `vals` specification.

```
DECLARE
  geor SDO_GEORASTER;
BEGIN
  select georaster into geor from georaster_table where georid = 1;
  sdo_geor_ra.rasterUpdate(geor,0,SDO_STRING2_ARRAY('(abs({0}-{1})=48)&({2}-
{1}=-101)','2*{0}-{1}/
3=108'),SDO_STRING2_ARRAYSET(SDO_STRING2_ARRAY('123','54','89'),SDO_STRING2_ARRAY('98
','56','123')));
END;
/
```

11.8 SDO_GEOR_RA.stack

Format

```
SDO_GEOR_RA.stack(
     georArray      IN SDO_GEORASTER_ARRAY,
     cropArea       IN SDO_NUMBER_ARRAY,
     layerList      IN SDO_NUMBER_ARRAY,
     method         IN VARCHARs,
     storageParam   IN VARCHAR2,
     outGeoRaster   IN OUT SDO_GEORASTER,
     nodata         IN VARCHAR2 DEFAULT 'FALSE',
     nodataValue    IN NUMBER DEFAULT 0,
     parallelParam  IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR_RA.stack(
     georArray      IN SDO_GEORASTER_ARRAY,
     cropArea       IN SDO_GEOMETRY,
     layerList      IN SDO_NUMBER_ARRAY,
```

```
method        IN VARCHARs,
storageParam  IN VARCHAR2,
outGeoRaster  IN OUT SDO_GEORASTER,
nodata        IN VARCHAR2 DEFAULT 'FALSE',
nodataValue   IN NUMBER DEFAULT 0,
polygonClip   IN VARCHAR2 DEFAULT 'FALSE',
parallelParam IN VARCHAR2 DEFAULT NULL);
```

Description

Generates a single-layer GeoRaster object whose cell values are a local statistics value of a list of layers of the input GeoRaster array. The input layers are specified by the `layerList` parameter, and the statistics method is specified by the `method` parameter.

Parameters

georArray
An array of GeoRaster objects. The data type is SDO_GEOR_ARRAY, which is defined as `VARRAY(10485760) OF SDO_GEORASTER`.

cropArea
Crop area definition. If the SDO_GEOMETRY object has a non-null SRID, the source GeoRaster objects must be georeferenced; otherwise, the source GeoRaster objects can be georeferenced or non-georeferenced. If `polygonClip` is `FALSE`, the MBR of the `cropArea` is used to crop the data. If `polygonClip` is `TRUE`, the geometry of the `cropArea` is used to crop the data.
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.

layerList
A number array to specify which bands of the input GeoRaster objects are used to compute statistics value for output. For example, if `georArray` specifies three GeoRaster objects `geor1,geor2,geor3`, which have 2,3,4 bands respectively, a layer list {0,3,7} is used to specify three bands as follows:

• The first band of the first GeoRaster object `geor1`

• The second band of the second GeoRaster object `geor2`

• The third band of the third GeoRaster object `geor3`

method
A string to specify what local statistics value should be returned. It should be one of the following values: `max`, `min`, `median`, `mean`, `std`, `sum`, `minority`, `majority`, `diversity`.

storageParam
A string specifying storage parameters, as explained in Storage Parameters.

outGeoRaster
Output GeoRaster object.

nodata
The string `TRUE` specifies that for any NODATA cells in an input GeoRaster object, the corresponding cells in the output GeoRaster object are to be set to the value specified for the `nodataValue` parameter. The string `FALSE` (the default) causes cells with

NODATA values to be considered as regular data. NODATA values and value ranges are discussed in NODATA Values and Value Ranges.

nodataValue
The value used to set NODATA cells if the `nodata` parameter value is the string `TRUE`.

parallelParam
Specifies the degree of parallelism for the operation. If specified, must be in the form `parallel=`*n*, where *n* is greater than 1. The database optimizer uses the degree of parallelism specified by this parameter. If not specified, then by default there is no parallel processing. (For more information, see Parallel Processing in GeoRaster.) Specifying `parallelParam` means that you cannot roll back the results of this procedure, as explained in the Usage Notes.

Usage Notes

All of the input GeoRaster objects must have same dimension size; otherwise, the ORA-13397 error is generated.

If you specify `parallelParam`, some execution units of the procedure run as autonomous transactions, which means that some changes are committed while the procedure is running and therefore you cannot roll back those changes. If you do not specify this parameter, you can roll back all changes.

Examples

The following example performs the stack operation on two input GeoRaster objects.

```
DECLARE
  geor        MDSYS.SDO_GEORASTER;
  geor1       MDSYS.SDO_GEORASTER;
  geor2       MDSYS.SDO_GEORASTER;
  geom        mdsys.sdo_geometry;
BEGIN
  geom:= sdo_geometry(2003,82394, NULL,
              sdo_elem_info_array(1, 1003, 1),
              sdo_ordinate_array(20283.775, 1011087.9,
                                 18783.775, 1008687.9,
                                 21783.775, 1008687.9,
                                 22683.775+0.001, 1009587.9+0.001,
                                 20283.775, 1011087.9));
  select georaster into geor from georaster_table where georid = 100;
  select georaster into geor2 from georaster_table where georid = 102;
  select georaster into geor1 from georaster_table where georid = 101 for update;

mdsys.sdo_geor_ra.stack(SDO_GEORASTER_ARRAY(geor,geor2),geom,SDO_NUMBER_ARRAY(3,5),'m
ax',null,geor1,'false',0,'TRUE');
  update georaster_table set georaster = geor1 where georid = 101;
END;
/
```

12

SDO_GEOR_UTL Package Reference

The 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.

Topics:

- SDO_GEOR_UTL.calcOptimizedBlockSize
- SDO_GEOR_UTL.calcRasterNominalSize
- SDO_GEOR_UTL.calcRasterStorageSize
- SDO_GEOR_UTL.calcSurfaceArea
- SDO_GEOR_UTL.clearReportTable
- SDO_GEOR_UTL.createDMLTrigger
- SDO_GEOR_UTL.createReportTable
- SDO_GEOR_UTL.disableReport
- SDO_GEOR_UTL.dropReportTable
- SDO_GEOR_UTL.emptyBlocks
- SDO_GEOR_UTL.enableReport
- SDO_GEOR_UTL.fillEmptyBlocks
- SDO_GEOR_UTL.generateColorRamp
- SDO_GEOR_UTL.generateGrayRamp
- SDO_GEOR_UTL.getAllStatusReport
- SDO_GEOR_UTL.getMaxMemSize
- SDO_GEOR_UTL.getReadBlockMemSize
- SDO_GEOR_UTL.getProgress
- SDO_GEOR_UTL.getStatusReport
- SDO_GEOR_UTL.getWriteBlockMemSize
- SDO_GEOR_UTL.isReporting
- SDO_GEOR_UTL.makeRDTNamesUnique
- SDO_GEOR_UTL.recreateDMLTriggers
- SDO_GEOR_UTL.renameRDT
- SDO_GEOR_UTL.setClientID
- SDO_GEOR_UTL.setMaxMemSize
- SDO_GEOR_UTL.setReadBlockMemSize
- SDO_GEOR_UTL.setSeqID

- • SDO_GEOR_UTL.setWriteBlockMemSize

12.1 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 in Storage Parameters.

In the `dimensionSize` and `blockSize` parameter values, specify the values for the dimensions in this order: row, column, band.

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);
```

ORACLE®

```
       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
```

12.2 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 Bitmap Masks.

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 Compression and Decompression.

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
```

12.3 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
```

12.4 SDO_GEOR_UTL.calcSurfaceArea

Format

```
SDO_GEOR_UTL.calcSurfaceArea(
     georaster  IN SDO_GEORASTER,
     window     IN SDO_GEOMETRY,
     parallel   IN NUMBER
  ) RETURN NUMBER;
```

Description

Calculates and returns the three–dimensional (3D) surface area represented by digital elevation model (DEM) data that is stored in a GeoRaster object.

Parameters

georaster
GeoRaster object in which DEM data is stored.

window
The 2D geometry object specifying the area of which the 3D surface area is to be calculated.

parallel
Degree of parallelism for the operation. It should be more than 0. The recommended value is 2 times the number of CPUs

Usage Notes

This function first finds out all cells within or touching a certain area specified by the `window` parameter, splits each of the cells into two 3D triangles, computes the 3D surface area of each triangle, and then returns the sum of these area values as the result. The areas of the triangles that intersect with the window boundary are computed based on the intersected geometries, so this function returns the surface area with a high degree of precision.

If the `parallel` parameter value is less than 1, then 1 is used (that is, no parallelism).

Examples

The following example calculates the surface area within `geom` using DEM data. (It refers to a table named GEORASTER_TABLE, whose definition is presented after Example 1-1 in Storage Parameters.)

```
DECLARE
  geor  SDO_GEORASTER;
  geom  SDO_GEOMETRY;
  area  number;
BEGIN
  geom:=sdo_geometry(2003,82394, NULL,
                     mdsys.sdo_elem_info_array(1, 1003, 1),
                     mdsys.sdo_ordinate_array(20283.775, 1011087.9,
```

```
                                                  18783.775, 1008687.9,
                                                  21783.775, 1008687.9,
                                                  22683.775+0.001, 1009587.9,
                                                  20283.775, 1011087.9));
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 3;
  area:=SDO_GEOR_UTL.calcSurfaceArea(geor,geom,2);
END;
/
```

12.5 SDO_GEOR_UTL.clearReportTable

Format

```
SDO_GEOR_UTL.clearReportTable(
     client_id   IN NUMBER DEFAULT NULL);
```

Description

Deletes records in the table that contains GeoRaster operation status information.

Parameters

client_id
ID of the client whose records are to be deleted. If this parameter is not specified, all records in the table are deleted.
(The client ID can be set by using the SDO_GEOR_UTL.setClientID procedure.)

Usage Notes

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example deletes all the records in the report table.

```
EXECUTE SDO_GEOR_UTL.clearReportTable;
```

12.6 SDO_GEOR_UTL.createDMLTrigger

Format

```
SDO_GEOR_UTL.createDMLTrigger(
     tableName    IN VARCHAR2,
     columnName   IN 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

> ✎ **Note:**
>
> A more convenient alternative may be to use the SDO_GEOR_UTL.recreateDMLTriggers procedure, where one call to the procedure re-creates or creates the DML triggers on all GeoRaster columns that the current user has privileges to access.

As explained in 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. 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');
```

12.7 SDO_GEOR_UTL.createReportTable

Format

```
SDO_GEOR_UTL.createReportTable;
```

Description

Creates the table to contain GeoRaster operation status information.

Parameters

None.

Usage Notes

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example creates the table to contain GeoRaster operation status information.

```
EXECUTE SDO_GEOR_UTL.createReportTable;
```

12.8 SDO_GEOR_UTL.disableReport

Format

```
SDO_GEOR_UTL.disableReport;
```

Description

Disables status reporting on GeoRaster operations in the current session.

Parameters

None.

Usage Notes

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example disables status reporting on GeoRaster operations in the current session.

```
EXECUTE SDO_GEOR_UTL.disableReport;
```

12.9 SDO_GEOR_UTL.dropReportTable

Format

```
SDO_GEOR_UTL.dropReportTable;
```

Description

Drops the table that contains GeoRaster operation status information.

Parameters

None.

Usage Notes

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example drops the table that contains GeoRaster operation status information.

```
EXECUTE SDO_GEOR_UTL.dropReportTable;
```

12.10 SDO_GEOR_UTL.emptyBlocks

Format

```
SDO_GEOR_UTL.emptyBlocks(
     georaster IN OUT SDO_GEORASTER,
     bgValues  IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Trims all blocks that contain only the specified background values to empty LOBs, thus making them empty blocks. Can be used to reduce disk space required for GeoRaster storage.

Parameters

georaster
GeoRaster object.

bgValues
Background values for determining if a block can be made an empty raster block. The number of elements in the SDO_NUMBER_ARRAY object must be either one (same filling value used for all layers) or the layer dimension size (a different filling value for each layer, respectively). For example, SDO_NUMBER_ARRAY(1,5,10) means that a block with the first layer with 1, the second layer with 5, and the third layer with 10 are made empty blocks. If this parameter is null, then `bgValues` will be the default value (a single element SDO_NUMBER_ARRAY(0)).

Usage Notes

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

Contrast this procedure with SDO_GEOR_UTL.fillEmptyBlocks, which uses specified background values to fill in all empty blocks.

Examples

The following example empties blocks whose cell values are background values (255,0,0).

```
DECLARE
  geor  SDO_GEORASTER;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 3 FOR UPDATE;
```

```
      SDO_GEOR_UTL.emptyBlocks(geor, SDO_NUMBER_ARRAY(255,0,0));
      UPDATE georaster_table SET georaster = geor WHERE georid = 3;
      COMMIT;
END;
/
```

12.11 SDO_GEOR_UTL.enableReport

Format

```
SDO_GEOR_UTL.enableReport;
```

Description

Enables status reporting on GeoRaster operations in the current session.

Parameters

None.

Usage Notes

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example enables status reporting on GeoRaster operations in the current session.

```
EXECUTE SDO_GEOR_UTL.enableReport;
```

12.12 SDO_GEOR_UTL.fillEmptyBlocks

Format

```
SDO_GEOR_UTL.fillEmptyBlocks(
     georaster IN OUT SDO_GEORASTER,
     bgValues  IN SDO_NUMBER_ARRAY DEFAULT NULL);
```

Description

Fills in all empty blocks with specified background values.

Parameters

georaster
GeoRaster object in which to fill empty blocks.

bgValues
Background values for filling empty raster blocks. 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. If this parameter is null, then `bgValues` will be 0 (zero).

Usage Notes

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

If pyramid data exists for `georaster`, the pyramid is regenerated based on pyramid information stored in the metadata.

Contrast this procedure with SDO_GEOR_UTL.emptyBlocks, which turns blocks with specified background values into empty blocks.

Examples

The following example empties blocks that have background values (255,0,0).

```
DECLARE
  geor  SDO_GEORASTER;
BEGIN
  SELECT georaster INTO geor FROM georaster_table WHERE georid = 3 FOR UPDATE;
  SDO_GEOR_UTL.emptyBlocks(geor, SDO_NUMBER_ARRAY(255,0,0));
  UPDATE georaster_table SET georaster = geor WHERE georid = 3;
  COMMIT;
END;
/
```

12.13 SDO_GEOR_UTL.generateColorRamp

Format

```
SDO_GEOR_UTL.generateColorRamp(
     colorSeeds      IN  SDO_GEOR_COLORMAP,
     rampSteps       IN  SDO_NUMBER_ARRAY,
     cellValueType   IN  VARCHAR2 DEFAULT 'integer',
     interpoParam    IN  VARCHAR2 DEFAULT 'method=linear'
     ) RETURN SDO_GEOR_COLORMAP;
```

Description

Generates an SDO_GEOR_COLORMAP object with a color ramp (gradient) by interpolating the red, green, blue, and alpha values entered as seed.

Parameters

colorSeeds
An SDO_GEOR_COLORMAP object with the seed values to be used as the beginning and end values for the interpolation. Should contain at least two groups, each containing a pixel value and set of RGBA values. (RGBA = red, green, blue, alpha)

rampSteps
An array where each item indicates how many interpolated values will be generated for each section of the color ramp. Should contain at least one integer value indicating how many values will be generated for each section.

cellValueType

A string indicating whether the cell values should be creates as integer or real values. The possible values for this parameter are `integer` (the default) or `real`.

interpoParam

A string specifying the interpolation method. The possible values are `method=linear` (the default) and `method=cosine`.

Usage Notes

This function returns an object of type SDO_GEOR_COLORMAP, which is described in SDO_GEOR_COLORMAP Object Type.

A color ramp can have several different sections with different lengths for each section. If `colorSeeds` specifies three RGBA values, the function will generate two sections on the same color ramp. The number of color levels for each section is controlled by `rampSteps` array. To generate independent sections, insert a new item in the `colorSeeds` parameter, and a new item in `rampSetps`. (Any value of 0 in `rampSetps` indicates that no interpolated values are needed to generate in that section.)

Examples

The following example generates and applies a color ramp containing 256 colors interpolated linearly from green to red. (It assumes the existence of a table named RAMP_TEST containing at least columns named ID and RASTER.)

```
DECLARE
  gr sdo_georaster;
  cmp sdo_geor_colormap;
BEGIN
  select raster into gr from ramp_test where id = 1 for update;

  cmp := sdo_geor_utl.generateColorRamp(
           colorSeeds => sdo_geor_colormap(
             sdo_number_array(   1,1000),
             sdo_number_array(   0, 255),
             sdo_number_array( 255,   0),
             sdo_number_array(   0,   0),
             sdo_number_array( 255, 255)),
           rampSteps  => sdo_number_array( 256 ),
           cellValueType => 'integer',
           interpoParam  => 'method=linear' );

  sdo_geor.setColorMap( gr, 0, cmp );

  update ramp_test set raster = gr where id = 1;
  commit;
END;
```

The following example generates and applies a color ramp that goes from shades of green turning into blue and from blue turning into red. (It assumes the existence of a table named RAMP_TEST containing at least columns named ID and RASTER.) The two sections will share the second item from the `colorSeeds` parameter value, the blue color.

```
DECLARE
  gr sdo_georaster;
  cmp sdo_geor_colormap;
BEGIN
  select raster into gr from ramp_test where id = 1 for update;
```

```
      cmp := sdo_geor_utl.generateColorRamp(
                colorSeeds => sdo_geor_colormap(
                  sdo_number_array(   1, 500, 1000),
                  sdo_number_array(   0,   0,  255),
                  sdo_number_array( 255,   0,    0),
                  sdo_number_array(   0, 255,    0),
                  sdo_number_array( 255, 255,  255)),
                rampSteps  => sdo_number_array( 128, 128 ),
                cellValueType => 'integer',
                interpoParam  => 'method=linear' );

  sdo_geor.setColorMap( gr, 0, cmp );

  update ramp_test set raster = gr where id = 1;
  commit;
END;
```

The following example generates a color ramp of 256 items with a section that goes from green to blue and a section that goes from white to red. (It assumes the existence of a table named RAMP_TEST containing at least columns named ID and RASTER.)

```
DECLARE
  gr sdo_georaster;
  cmp sdo_geor_colormap;
BEGIN
  select raster into gr from ramp_test where id = 1 for update;

  cmp := sdo_geor_utl.generateColorRamp(
                colorSeeds => sdo_geor_colormap(
                  sdo_number_array(   1, 500,   500, 1000),
                  sdo_number_array(   0,   0,  255,  255),
                  sdo_number_array( 255,   0,  255,    0),
                  sdo_number_array(   0, 255,  255,    0),
                  sdo_number_array( 255, 255,  255,  255)),
                rampSteps  => sdo_number_array( 128, 0, 128 ),
                cellValueType => 'integer',
                interpoParam  => 'method=linear' );

  sdo_geor.setColorMap( gr, 0, cmp );

  update ramp_test set raster = gr where id = 1;
  commit;
END;
```

The value of 0 in the `rampSteps` array (`rampSteps => sdo_number_array( 128, 0, 128 );`) indicates that no interpolated values are needed for generation in the second section.

12.14 SDO_GEOR_UTL.generateGrayRamp

Format

```
SDO_GEOR_UTL.generateGrayRamp(
     graySeeds      IN  SDO_GEOR_GRAYSCALE,
     rampSteps      IN  SDO_NUMBER_ARRAY,
     cellValueType  IN  VARCHAR2 DEFAULT 'integer',
     interpoParam   IN  VARCHAR2 DEFAULT 'method=linear'
     ) RETURN SDO_GEOR_GRAYSCALE;
```

Description

Generates an SDO_GEOR_GRAYSCALE object with a grayscale ramp (gradient) by interpolating the values entered as seed.

Parameters

colorSeeds
An SDO_GEOR_GRAYSCALE object with the seed values to be used as the beginning and end values for the interpolation. Should should contain at least one pixel-value/gray-value pair.

rampSteps
An array where each item indicates how many interpolated values will be generated for each section of the gray ramp. Should contain at least one integer value indicating how many values will be generated for each section of the gray ramp.

cellValueType
A string indicating whether the cell values should be creates as integer or real values. The possible values for this parameter are `integer` (the default) or `real`.

interpoParam
A string specifying the interpolation method. The possible values are `method=linear` (the default) and `method=cosine`.

Usage Notes

This function returns an object of type SDO_GEOR_GRAYSCALE, which is described in SDO_GEOR_GRAYSCALE Object Type.

A gray ramp can have several different sections with different lengths for each section. If `graySeeds` specifies three values, the function will generate two sections on the same gray ramp. The number of gray levels for each section is controlled by `rampSteps` array. To generate independent sections, insert a new item in the `graySeeds` parameter, and a new item in `rampSetps`. (Any value of 0 in `rampSetps` indicates that no interpolated values are needed to generate in that section.)

Examples

The following example generates and applies a gray ramp containing 256 colors interpolated linearly from green to red. (It assumes the existence of a table named RAMP_TEST containing at least columns named ID and RASTER.)

```
DECLARE
  gr sdo_georaster;
  gs sdo_geor_grayscale;
BEGIN
  select raster into gr from ramp_test where id = 3 for update;

  gs := sdo_geor_utl.generateGrayRamp(
          graySeeds  => sdo_geor_grayscale(
            sdo_number_array( 0, 1000),
            sdo_number_array( 0,  255)),
          rampSteps  => sdo_number_array( 100 ),
          cellValueType => 'integer',
          interpoParam  => 'method=linear' );

  sdo_geor.setGrayScale( gr, 0, gs );
```

```
  update ramp_test set raster = gr where id = 3;
  commit;
END;
```

The following example generates and applies a gray ramp 256 items interpolating values from -100 to 0 and from 0 to 1000. (It assumes the existence of a table named RAMP_TEST containing at least columns named ID and RASTER.).

```
DECLARE
  gr sdo_georaster;
  gs sdo_geor_grayscale;
BEGIN
  select raster into gr from ramp_test where id = 3 for update;

  gs := sdo_geor_utl.generateGrayRamp(
          graySeeds   => sdo_geor_grayscale(
            sdo_number_array( -100, 0, 1000),
            sdo_number_array(  255, 0,  255)),
          rampSteps   => sdo_number_array( 100, 100 ),
          cellValueType => 'real',
          interpoParam  => 'method=linear' );

  sdo_geor.setGrayScale( gr, 0, gs );

  update ramp_test set raster = gr where id = 3;
  commit;
END;
```

The following example generates a gray ramp of 256 items with a section that goes from green to blue and a section that goes from white to red. (It assumes the existence of a table named RAMP_TEST containing at least columns named ID and RASTER.)

```
DECLARE
  gr sdo_georaster;
  gs sdo_geor_grayscale;
BEGIN
  select raster into gr from ramp_test where id = 3 for update;

  gs := sdo_geor_utl.generateGrayRamp(
          graySeeds   => sdo_geor_grayscale(
            sdo_number_array( -100,  0,   0, 1000),
            sdo_number_array(    0, 10, 100,  200)),
          rampSteps   => sdo_number_array( 100, 0, 100 ),
          cellValueType => 'real',
          interpoParam  => 'method=linear' );

  sdo_geor.setGrayScale( gr, 0, gs );

  update ramp_test set raster = gr where id = 3;
  commit;
END;
```

The value of 0 in the `rampSteps` array (`rampSteps => sdo_number_array( 128, 0, 128 );`) indicates that no interpolated values are needed for generation in the second section.

12.15 SDO_GEOR_UTL.getAllStatusReport

Format

SDO_GEOR_UTL.getAllStatusReport() RETURN SDO_STRING2_ARRAYSET;

Description

Returns the current status for all operations for all clients in the status table.

Parameters

None.

Usage Notes

This function returns an array with a comma-delimited list of status information: *<client_id>, <sequence_id>, <timestamp>, <operation name>, <RDT table name>, <Raster ID>, <progress>, <description>*. The data type is SDO_STRING2_ARRAYSET, which is defined as VARRAY(2147483647) OF SDO_STRING2_ARRAY.

If the status table has not been created, the function returns 'The report table does not exist.'

This function is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example returns the current status for all operations for all clients. It returns two SDO_STRING2_ARRAY objects.

```
SELECT * from the (SELECT SDO_GEOR_UTL.getAllStatusReport FROM DUAL);

COLUMN_VALUE
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('Client:23', 'Sequence:1', '24-SEP-12 11.10.42.030169 AM',
'Mosaic', 'RDT:LANDSAT_MOSAIC_RDT', 'RID:1', '100% complete', NULL)
SDO_STRING2_ARRAY('Client:1', 'Sequence:0', '24-SEP-12 11.10.42.379631 AM',
'GeneratePyramid', 'RDT:LANDSAT_MOSAIC_RDT', 'RID:1', '100% complete', 'operation
completed')

2 rows selected.
```

The following example also returns the current status for all operations for all clients. It uses a different SELECT statement format than the preceding example, and returns a single SDO_STRING2_ARRAYSET object that contains two SDO_STRING2_ARRAY objects.

```
set linesize 80
SELECT SDO_GEOR_UTL.getAllStatusReport FROM DUAL;

SDO_GEOR_UTL.GETALLSTATUSREPORT()
--------------------------------------------------------------------------------
SDO_STRING2_ARRAYSET(SDO_STRING2_ARRAY('Client:27', 'Sequence:1', '26-SEP-12 11.
31.03.473087 AM', 'Mosaic', 'RDT:LANDSAT_MOSAIC_RDT', 'RID:1', '100% complete',
```

```
NULL), SDO_STRING2_ARRAY('Client:-1', 'Sequence:0', '26-SEP-12 11.31.03.962948 A
M', 'GeneratePyramid', 'RDT:LANDSAT_MOSAIC_RDT', 'RID:1', '100% complete', 'oper
ation completed'))

1 row selected.
```

12.16 SDO_GEOR_UTL.getMaxMemSize

Format

```
SDO_GEOR_UTL.getMaxMemSize RETURN NUMBER;
```

Description

Returns the upper limit size of the memory that can be used for managing memory using the GeoRaster buffer system.

Parameters

(None.)

Usage Notes

Managing Memory for GeoRaster Buffering provides conceptual and usage information, including explanations of the relevant parameters.

Examples

The following example returns the current upper limit size of the memory that can be used for managing memory using the GeoRaster buffer system.

```
SELECT sdo_geor_utl.getMaxMemSize FROM DUAL;

GETMAXMEMSIZE
-------------
    16777216
```

12.17 SDO_GEOR_UTL.getReadBlockMemSize

Format

```
SDO_GEOR_UTL.getReadBlockMemSize RETURN NUMBER;
```

Description

Returns the size in bytes of the GeoRaster internal data block for read-only operations.

Parameters

(None.)

Usage Notes

Managing Memory for GeoRaster Buffering provides conceptual and usage information, including explanations of the relevant parameters.

Examples

The following example returns the current size in bytes of the GeoRaster internal data block for read-only operations.

```
SELECT sdo_geor_utl.getReadBlockMemSize FROM DUAL;

GETREADBLOCKMEMSIZE
-------------------
              32768
```

12.18 SDO_GEOR_UTL.getProgress

Format

```
SDO_GEOR_UTL.getProgress(
     client_id IN NUMBER,
     seq_id    IN NUMBER DEFAULT 0
     ) RETURN NUMBER;
```

Description

Returns the progress of the operation for a specified client (session) and optionally for a specified operation. The returned value is the percentage of completion as a floating point number between 0 and 1.

Parameters

client_id
Unique numeric value identifying the session.

seq_id
Unique numeric value (within the specified session) identifying the operation for which to return status information.

Usage Notes

This function returns a number that is the latest estimated progress of the operation identified by the `client_id` and `seq_id`. Make sure that the correct `client_id` and `seq_id` values are used.

If the status table has no record of the specified operation with the given `client_id` and `seq_id`, null is returned.

If the status table has not been created, the function returns `'The report table does not exist.'`

This function is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example returns the progress of the operation with client ID 5 and operation sequence ID 3.

```
SELECT sdo_geor_utl.getgetProgress(5, 3) progress FROM dual;
```

```
PROGRESS
--------
    .305
```

12.19 SDO_GEOR_UTL.getStatusReport

Format

```
SDO_GEOR_UTL.getStatusReport(
     client_id IN NUMBER,
     seq_id    IN NUMBER DEFAULT 0
     ) RETURN SDO_STRING2_ARRAY;
```

Description

Returns the current status of the operations in the status table for a specified client (session) and optionally for a specified operation.

Parameters

client_id
Unique numeric value identifying the session.

seq_id
Unique numeric value (within the specified session) identifying the operation for which to return status information.

Usage Notes

This function returns the current status of a specified session (`client_id`) in an array of comma-delimited lists of status information: *<client_id>, <sequence_id>, <timestamp>, <operation name>, <RDT table name>, <Raster ID>, <progress>, <description>*. The data type is SDO_STRING2_ARRAY, which is defined as `VARRAY(2147483647) OF VARCHAR2(4096).`

If the status table has not been created, the function returns `'The report table does not exist.'`

This function is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example returns the status of the operation with client ID 5 and operation sequence ID 3.

```
SELECT sdo_geor_utl.getStatusReport(5, 3) FROM dual;

SDO_GEOR_UTL.GETSTATUSREPORT(5,3)
--------------------------------------------------------------------------------
SDO_STRING2_ARRAY('24-SEP-12 11.10.43.477804 AM', 'Mosaic', 'RDT:LANDSAT_MOSAIC_
RDT', 'RID:2', '100% complete', 'operation completed')
```

12.20 SDO_GEOR_UTL.getWriteBlockMemSize

Format

SDO_GEOR_UTL.getWriteBlockMemSize RETURN NUMBER;

Description

Returns the size in bytes of the GeoRaster internal data block for read/write operations.

Parameters

(None.)

Usage Notes

Managing Memory for GeoRaster Buffering provides conceptual and usage information, including explanations of the relevant parameters.

Examples

The following example returns the current size in bytes of the GeoRaster internal data block for read/write operations.

```
SELECT sdo_geor_utl.getWriteBlockMemSize FROM DUAL;

GETWRITEBLOCKMEMSIZE
--------------------
               65536
```

12.21 SDO_GEOR_UTL.isReporting

Format

SDO_GEOR_UTL.isReporting() RETURN NUMBER;

Description

Checks if any session has status reporting enabled.

Parameters

None.

Usage Notes

This function returns 1 if one or more sessions have status reporting enabled; it returns 0 if no sessions have status reporting enabled.

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example checks if any session has status reporting enabled.

```
SELECT SDO_GEOR_UTL.isReporting FROM DUAL;

ISREPORTING
-----------
          1
```

12.22 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;
```

12.23 SDO_GEOR_UTL.recreateDMLTriggers

Format

```
SDO_GEOR_UTL.recreateDMLTriggers;
```

Description

Re-creates the required standard GeoRaster data manipulation language (DML) triggers on all GeoRaster columns that the current user has privileges to access, so that the appropriate operations are performed when the triggers are fired.

Parameters

(None)

Usage Notes

As explained in 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. If this happens you can either call the SDO_GEOR_UTL.createDMLTrigger procedure for those added GeoRaster columns or call the SDO_GEOR_UTL.recreateDMLTriggers procedure to recreate the DML triggers on all GeoRaster columns.

You usually do not need to call this procedure, but it is useful for re-creating the DML triggers in some scenarios, such as a database upgrade or a data migration.

Examples

The following example re-creates the standard GeoRaster DML triggers.

```
EXECUTE sdo_geor_utl.recreateDMLTriggers;
```

12.24 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');
```

12.25 SDO_GEOR_UTL.setClientID

Format

```
SDO_GEOR_UTL.setClientID(
     client_id   IN NUMBER);
```

Description

Sets the client ID for a session.

Parameters

client_id
Unique ID value to identify the session.

Usage Notes

This procedure can be used to identify different sessions under the same user. The client ID can be the database session ID or the client ID in the mid-tier environment.

If this procedure is not called, the client ID in the status report defaults to the database session ID.

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example sets the client ID to 1.

```
EXECUTE SDO_GEOR_UTL.setClientID(1);
```

12.26 SDO_GEOR_UTL.setMaxMemSize

Format

```
SDO_GEOR_UTL.setMaxMemSize(
     maxMemSize   IN NUMBER DEFAULT 16777216);
```

Description

Sets the upper limit size of the memory that can be used for managing memory using the GeoRaster buffer system.

Parameters

maxMemSize
Number of bytes that can be used for managing memory using the GeoRaster buffer system. The default value is 16777216 (16 MB).

Usage Notes

Managing Memory for GeoRaster Buffering provides conceptual and usage information, including explanations of the relevant parameters.

Examples

The following example sets the `maxMemSize` value to 100 million bytes (approximately 100 MB).

```
EXECUTE sdo_geor.setMaxMemSize(100000000);
```

The following example sets the `maxMemSize` value to the default size (16 MB).

```
EXECUTE sdo_geor_utl.setMaxMemSize();
```

12.27 SDO_GEOR_UTL.setReadBlockMemSize

Format

```
SDO_GEOR_UTL.setReadBlockMemSize(
     memBlockSize   IN NUMBER DEFAULT 32768);
```

Description

Sets the size of the GeoRaster internal data block for read-only operations.

Parameters

memBlockSize
Number of bytes that can be used for the GeoRaster internal data block for read-only operations. The default value is 32768 (32 KB).

Usage Notes

Managing Memory for GeoRaster Buffering provides conceptual and usage information, including explanations of the relevant parameters.

Examples

The following example sets the `memBlockSize` value to 1 million bytes (approximately 1 MB).

```
EXECUTE sdo_geor.setReadBlockMemSize(1000000);
```

The following example sets the `memBlockSize` value to the default value (32 KB).

```
EXECUTE sdo_geor_utl.setReadBlockMemSize();
```

12.28 SDO_GEOR_UTL.setSeqID

Format

```
SDO_GEOR_UTL.setSeqID(
     seq_id   IN NUMBER);
```

Description

Sets the sequence ID for a session.

Parameters

seq_id
Unique ID value to identify the operation in a session.

Usage Notes

This procedure can be used to identify different operations in the same session.

If this procedure is not called, the sequence ID in the status report defaults to 0.

This procedure is one of the subprograms available for monitoring and reporting the progress of GeoRaster operations. For an overview of this capability, see Reporting Operation Progress in GeoRaster.

Examples

The following example sets the sequence ID to 1.

```
EXECUTE SDO_GEOR_UTL.setSeqID(1);
```

12.29 SDO_GEOR_UTL.setWriteBlockMemSize

Format

```
SDO_GEOR_UTL.setWriteBlockMemSize(
     memBlockSize   IN NUMBER);
```

Description

Sets the size of the GeoRaster internal data block for read/write operations.

Parameters

memBlockSize
Number of bytes that can be used for the GeoRaster internal data block for read/write operations. The default value is 65536 (64 KB).

Usage Notes

Managing Memory for GeoRaster Buffering provides conceptual and usage information, including explanations of the relevant parameters.

Examples

The following example sets the `memBlockSize` value to 1 million bytes (approximately 1 MB).

```
EXECUTE sdo_geor.setWriteBlockMemSize(1000000);
```

The following example sets the `memBlockSize` value to the default value (64 KB).

```
EXECUTE sdo_geor_utl.setWriteBlockMemSize();
```

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 GeoRaster XML Schema.)

```
<?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="defaultBlue" type="xsd:positiveInteger" minOccurs="0"/>

    <xsd:element name="defaultAlpha" type="xsd:positiveInteger" minOccurs="0"/>

    <xsd:element name="defaultPyramidLevel" type="xsd:integer" 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="JPEG-F"/>
        <xsd:enumeration value="DEFLATE"/>
        <xsd:enumeration value="LT-MG2"/>
        <xsd:enumeration value="LT-MG3"/>
        <xsd:enumeration value="LT-JP2"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="compressionQuality">
      <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="compressionDescriptionType">
      <xsd:sequence>
        <xsd:element name="type" type="compressionType" default="NONE"/>
        <xsd:element name="quality" type="compressionQuality" minOccurs="0"/>
        <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="blockingType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NONE"/>
        <xsd:enumeration value="REGULAR"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="blockingDescriptionType">
      <xsd:sequence>
        <xsd:element name="type" type="blockingType" default="NONE"/>
        <xsd:element name="totalRowBlocks" type="xsd:positiveInteger" default="1"/>
        <xsd:element name="totalColumnBlocks" type="xsd:positiveInteger" default="1"/>
        <xsd:element name="totalBandBlocks" type="xsd:positiveInteger" default="1"
minOccurs="0"/>
        <xsd:element name="rowBlockSize" type="xsd:positiveInteger"/>
        <xsd:element name="columnBlockSize" type="xsd:positiveInteger"/>
        <xsd:element name="bandBlockSize" type="xsd:positiveInteger" minOccurs="0"/>
        <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:simpleType name="cellInterleavingType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="BSQ"/>
        <xsd:enumeration value="BIL"/>
        <xsd:enumeration value="BIP"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="pyramidType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NONE"/>
        <xsd:enumeration value="DECREASE"/>
        <xsd:enumeration value="INCREASE"/>
        <xsd:enumeration value="BIDIRECTION"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="resamplingType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NN"/>
        <xsd:enumeration value="BILINEAR"/>
        <xsd:enumeration value="CUBIC"/>
        <xsd:enumeration value="AVERAGE4"/>
        <xsd:enumeration value="AVERAGE16"/>
        <xsd:enumeration value="BIQUADRATIC"/>
        <xsd:enumeration value="OTHER"/>
      </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