Oracle® Spatial

GeoRaster

10*g* Release 1 (10.1)

Part No. B10827-01

December 2003

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

ORACLE®

Oracle Spatial GeoRaster, 10*g* Release 1 (10.1)

Part No. B10827-01

Primary Author: Chuck Murray

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

Contents

Send Us Your Comments xiii

Preface xv

Audience xvi
Documentation Accessibility xvi
Organization xvii
Related Documentation xvii
Conventions xviii

1 GeoRaster Overview and Concepts

1.1 Vector and Raster Data 1-2
1.2 GeoRaster Targeted Data Sources and Uses 1-2
1.2.1 Remote Sensing 1-3
1.2.2 Photogrammetry 1-3
1.2.3 Geographic Information Systems 1-4
1.2.4 Cartography 1-4
1.2.5 Digital Image Processing 1-5
1.2.6 Geology, Geophysics, and Geochemistry 1-5
1.3 GeoRaster Data Model 1-5
1.4 GeoRaster Physical Storage 1-10
1.4.1 Storage Parameters 1-14
1.4.2 Raster Data Table 1-16
1.4.3 Blank and Empty GeoRaster Objects 1-17

1.5 Bands, Layers, and Interleaving ... 1-18
1.6 Georeferencing ... 1-20
1.6.1 GeoRaster Georeferencing Method ... 1-21
1.7 Pyramids ... 1-22
1.8 GeoRaster PL/SQL Subprogram Categories ... 1-25
1.8.1 Subprograms to Create, Load, and Export GeoRaster Data ... 1-25
1.8.2 Subprograms to Validate and Process GeoRaster Objects ... 1-26
1.8.3 Subprogram for GeoRaster DML Triggers ... 1-26
1.8.4 Subprograms to Get and Set GeoRaster Metadata and Data ... 1-27
1.9 GeoRaster Tools: Loader, Viewer, Exporter ... 1-35
1.10 GeoRaster PL/SQL Demo Files ... 1-36

2 GeoRaster Data Types and Related Structures

2.1 SDO_GEORASTER Object Type ... 2-1
2.1.1 rasterType Attribute ... 2-2
2.1.2 spatialExtent Attribute ... 2-2
2.1.3 rasterDataTable Attribute ... 2-3
2.1.4 rasterID Attribute ... 2-3
2.1.5 metadata Attribute ... 2-3
2.2 SDO_RASTER Object Type and the Raster Data Table ... 2-4
2.2.1 rasterID Attribute ... 2-4
2.2.2 pyramidLevel Attribute ... 2-4
2.2.3 bandBlockNumber Attribute ... 2-5
2.2.4 rowBlockNumber Attribute ... 2-5
2.2.5 columnBlockNumber Attribute ... 2-5
2.2.6 blockMBR Attribute ... 2-5
2.2.7 rasterBlock Attribute ... 2-5
2.3 Other GeoRaster Types ... 2-5
2.3.1 SDO_GEOR_HISTOGRAM Object Type ... 2-5
2.3.2 SDO_GEOR_COLORMAP Object Type ... 2-6
2.3.3 SDO_GEOR_GRAYSCALE Object Type ... 2-7
2.3.4 SDO_RASTERSET Collection Type ... 2-8
2.3.5 SDO_GEOR_SRS Object Type ... 2-8
2.4 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA) ... 2-10
2.4.1 TABLE_NAME Column ... 2-11

2.4.2 COLUMN_NAME Column ... 2-11
2.4.3 METADATA_COLUMN_NAME Column ... 2-11
2.4.4 RDT_TABLE_NAME Column ... 2-12
2.4.5 RASTER_ID Column ... 2-12
2.4.6 OTHER_TABLE_NAMES Column ... 2-12
2.5 GeoRaster XML Schema Table ... 2-12

3 GeoRaster Operations

3.1 Creating the Standard GeoRaster DML Trigger ... 3-2
3.2 Creating New GeoRaster Objects ... 3-3
3.3 Loading GeoRaster Data ... 3-3
3.4 Validating GeoRaster Objects ... 3-4
3.5 Georeferencing GeoRaster Objects ... 3-4
3.6 Indexing GeoRaster Data ... 3-5
3.7 Changing Raster Storage ... 3-6
3.8 Querying and Updating GeoRaster Metadata ... 3-6
3.9 Querying and Updating Cell Data ... 3-7
3.10 Processing GeoRaster Objects ... 3-7
3.11 Viewing GeoRaster Objects ... 3-7
3.12 Exporting GeoRaster Objects ... 3-8
3.13 Transferring GeoRaster Data Between Databases ... 3-8
3.14 Dealing with Possible GeoRaster Data Problems ... 3-10

4 SDO_GEOR Package Reference

SDO_GEOR.changeCellValue ... 4-2
SDO_GEOR.changeFormat ... 4-5
SDO_GEOR.changeFormatCopy ... 4-7
SDO_GEOR.copy ... 4-9
SDO_GEOR.createBlank ... 4-11
SDO_GEOR.deletePyramid ... 4-13
SDO_GEOR.exportTo ... 4-14
SDO_GEOR.generatePyramid ... 4-19
SDO_GEOR.generateSpatialExtent ... 4-21

SDO_GEOR.georeference 4-23
SDO_GEOR.getBandDimSize 4-26
SDO_GEOR.getBeginDateTime 4-27
SDO_GEOR.getBinTable 4-28
SDO_GEOR.getBinType 4-30
SDO_GEOR.getBlankCellValue 4-32
SDO_GEOR.getBlockingType 4-34
SDO_GEOR.getBlockSize 4-35
SDO_GEOR.getCellCoordinate 4-36
SDO_GEOR.getCellDepth 4-38
SDO_GEOR.getCellValue 4-39
SDO_GEOR.getColorMap 4-41
SDO_GEOR.getColorMapTable 4-44
SDO_GEOR.getCompressionType 4-46
SDO_GEOR.getDefaultBlue 4-47
SDO_GEOR.getDefaultColorLayer 4-48
SDO_GEOR.getDefaultGreen 4-50
SDO_GEOR.getDefaultRed 4-51
SDO_GEOR.getEndDateTime 4-52
SDO_GEOR.getGrayScale 4-53
SDO_GEOR.getGrayScaleTable 4-55
SDO_GEOR.getHistogram 4-57
SDO_GEOR.getHistogramTable 4-58
SDO_GEOR.getID 4-60
SDO_GEOR.getInterleavingType 4-61
SDO_GEOR.getLayerDimension 4-62
SDO_GEOR.getLayerID 4-63
SDO_GEOR.getLayerOrdinate 4-65
SDO_GEOR.getModelCoordinate 4-67
SDO_GEOR.getModelSRID 4-69
SDO_GEOR.getNODATA 4-70

SDO_GEOR.getPyramidMaxLevel 4-71
SDO_GEOR.getPyramidType 4-72
SDO_GEOR.getRasterBlocks 4-73
SDO_GEOR.getRasterData 4-75
SDO_GEOR.getRasterSubset 4-77
SDO_GEOR.getScaling 4-80
SDO_GEOR.getSpatialDimNumber 4-81
SDO_GEOR.getSpatialDimSizes 4-83
SDO_GEOR.getSpatialResolutions 4-84
SDO_GEOR.getSpectralResolution 4-85
SDO_GEOR.getSpectralUnit 4-86
SDO_GEOR.getSRS 4-87
SDO_GEOR.getStatistics 4-88
SDO_GEOR.getTotalLayerNumber 4-90
SDO_GEOR.getULTCoordinate 4-91
SDO_GEOR.getVAT 4-92
SDO_GEOR.getVersion 4-94
SDO_GEOR.hasGrayScale 4-95
SDO_GEOR.hasPseudoColor 4-97
SDO_GEOR.importFrom 4-99
SDO_GEOR.init 4-103
SDO_GEOR.isBlank 4-105
SDO_GEOR.isOrthoRectified 4-106
SDO_GEOR.isRectified 4-108
SDO_GEOR.isSpatialReferenced 4-109
SDO_GEOR.mosaic 4-110
SDO_GEOR.scale 4-112
SDO_GEOR.scaleCopy 4-115
SDO_GEOR.schemaValidate 4-118
SDO_GEOR.setBeginDateTime 4-119
SDO_GEOR.setBinTable 4-121

SDO_GEOR.setBlankCellValue 4-123
SDO_GEOR.setColorMap 4-125
SDO_GEOR.setColorMapTable 4-127
SDO_GEOR.setDefaultBlue 4-129
SDO_GEOR.setDefaultColorLayer 4-131
SDO_GEOR.setDefaultGreen 4-133
SDO_GEOR.setDefaultRed 4-135
SDO_GEOR.setEndDateTime 4-137
SDO_GEOR.setGrayScale 4-139
SDO_GEOR.setGrayScaleTable 4-141
SDO_GEOR.setHistogramTable 4-143
SDO_GEOR.setID 4-145
SDO_GEOR.setLayerID 4-147
SDO_GEOR.setLayerOrdinate 4-149
SDO_GEOR.setModelSRID 4-151
SDO_GEOR.setOrthoRectified 4-153
SDO_GEOR.setRasterType 4-155
SDO_GEOR.setRectified 4-156
SDO_GEOR.setScaling 4-158
SDO_GEOR.setSpatialReferenced 4-160
SDO_GEOR.setSpatialResolutions 4-162
SDO_GEOR.setSpectralResolution 4-164
SDO_GEOR.setSpectralUnit 4-166
SDO_GEOR.setSRS 4-168
SDO_GEOR.setStatistics 4-170
SDO_GEOR.setULTCoordinate 4-172
SDO_GEOR.setVAT 4-174
SDO_GEOR.setVersion 4-176
SDO_GEOR.subset 4-178
SDO_GEOR.validateGeoraster 4-181

5 SDO_GEOR_UTL Package Reference

SDO_GEOR_UTL.createDMLTrigger 5-2

A GeoRaster Metadata XML Schema

Index

List of Examples

1–1 Using storageParam Keywords 1-16
1–2 Creating a Raster Data Table 1-16

List of Figures

1–1 Raster Space and Model Space 1-8
1–2 Physical Storage of GeoRaster Data 1-12
1–3 GeoRaster Data in an Oracle Database 1-13
1–4 Layers, Bands, and the Raster Data Table 1-19
1–5 Pyramid Levels 1-23

List of Tables

1–1	storageParam Keywords for Raster Data	1-14
1–2	Subprograms to Create, Load, and Export GeoRaster Data	1-25
1–3	Subprograms to Validate and Process GeoRaster Objects	1-26
1–4	Subprogram for GeoRaster Triggers	1-27
1–5	Subprograms to Get and Set Whole Object Metadata	1-28
1–6	Subprograms to Get and Set Cell Coordinates and Values	1-30
1–7	Subprograms to Get and Set Spatial Reference System Metadata	1-31
1–8	Subprograms to Get and Set Date and Time Metadata	1-32
1–9	Subprograms to Get and Set Imagery Band System Metadata	1-33
1–10	Subprograms to Get and Set Layer Metadata	1-33
1–11	Subprograms to Get Pyramid Metadata	1-35
1–12	GeoRaster PL/SQL Demo Files	1-37
2–1	SDO_GEOR_HISTOGRAM Object Type Attributes	2-6
2–2	SDO_GEOR_COLORMAP Object Type Attributes	2-7
2–3	SDO_GEOR_GRAYSCALE Object Type Attributes	2-7
2–4	SDO_GEOR_SRS Object Type Attributes	2-9
2–5	SDO_GEOR_XMLSCHEMA_TABLE Table Columns	2-12

Send Us Your Comments

Oracle Spatial GeoRaster, 10*g* Release 1 (10.1)

Part No. B10827-01

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

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

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: GeoRaster Documentation
- Postal service:
  Oracle Corporation
  GeoRaster Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  USA

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

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

Preface

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

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

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

This preface contains these topics:

- Audience
- Documentation Accessibility
- Organization
- Related Documentation
- Conventions

Audience

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

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

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

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

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

Organization

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

Chapter 1, "GeoRaster Overview and Concepts"

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

Chapter 2, "GeoRaster Data Types and Related Structures"

Explains the object-relational schema associated with GeoRaster.

Chapter 3, "GeoRaster Operations"

Explains the main operations that you can perform using GeoRaster.

Chapter 4, "SDO_GEOR Package Reference"

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

Chapter 5, "SDO_GEOR_UTL Package Reference"

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

Appendix A, "GeoRaster Metadata XML Schema"

Provides the XML schema definition of the GeoRaster metadata.

Related Documentation

For more information, see the following document:

- *Oracle Spatial User's Guide and Reference*

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

Printed documentation is available for sale in the Oracle Store at

http://oraclestore.oracle.com/

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

http://otn.oracle.com/membership

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

http://otn.oracle.com/documentation

Conventions

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

The following conventions are used in this guide:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface text indicates a term defined in the text.
`monospace text`	Monospace text is used for the names of parameters, files, and directory paths. It is also used for SQL and PL/SQL code examples.
italic text	Italic text is used for book titles, emphasis, and some special terms.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

GeoRaster Overview and Concepts

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

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

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

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

This chapter contains the following major sections:

- Section 1.1, "Vector and Raster Data"
- Section 1.2, "GeoRaster Targeted Data Sources and Uses"
- Section 1.3, "GeoRaster Data Model"
- Section 1.4, "GeoRaster Physical Storage"
- Section 1.5, "Bands, Layers, and Interleaving"
- Section 1.6, "Georeferencing"
- Section 1.7, "Pyramids"
- Section 1.8, "GeoRaster PL/SQL Subprogram Categories"
- Section 1.9, "GeoRaster Tools: Loader, Viewer, Exporter"
- Section 1.10, "GeoRaster PL/SQL Demo Files"

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

1.2 GeoRaster Targeted Data Sources and Uses

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

technologies. These technologies use several data formats and create a variety of products.

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

1.2.1 Remote Sensing

Remote sensing obtains information about an area or object through a device that is not physically connected to the area or object. For example, the sensor might be in a satellite, balloon, airplane, boat, or ground station. The sensor device can be any of a variety of devices, including a frame camera, pushbroom (swath) imager, synthetic aperture radar (SAR), hydrographic sonar, or paper or film scanner. Remote sensing applications include environmental assessment and monitoring, global change detection and monitoring, and natural resource surveying.

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

1.2.2 Photogrammetry

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

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

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

1.2.3 Geographic Information Systems

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

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

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

1.2.4 Cartography

Cartography is the science of creating maps, which are two-dimensional representations of the three-dimensional Earth (or of a non-Earth space using a local coordinate system). Today, maps are digitized or scanned into digital forms, and map production is largely automated. Maps stored on a computer can be queried, analyzed, and updated quickly.

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

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

1.2.5 Digital Image Processing

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

1.2.6 Geology, Geophysics, and Geochemistry

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

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

1.3 GeoRaster Data Model

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

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

GeoRaster uses a generic raster data model that is component-based, logically layered, and multidimensional. The core data in a raster is a multidimensional matrix of raster cells. Each cell is one element of the matrix, and its value is called the cell value. If the GeoRaster object represents an image, a cell can also be called a pixel, which has only one value. (In GeoRaster, the terms *cell* and *pixel* are interchangeable.) The matrix has a number of dimensions, a cell depth, and a size for each dimension. The cell depth is the data size of the value of each cell. The cell depth defines the range of all cell values, and it applies to each single cell, not to an array of cells. This core raster data set can be blocked for optimal storage and retrieval.

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

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

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

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

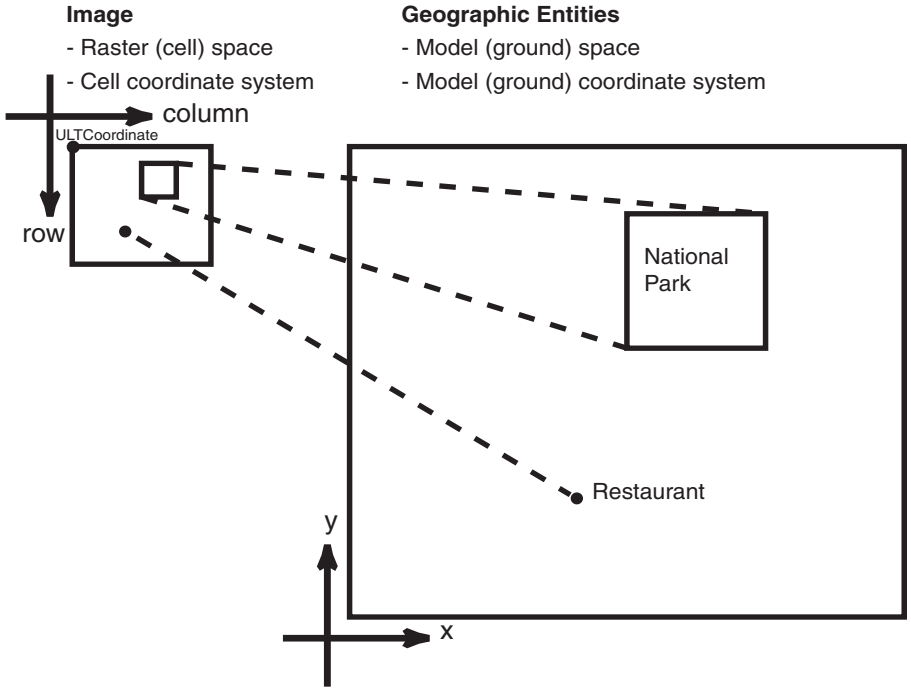
The GeoRaster object data types, described in Chapter 2, are based on the GeoRaster data model.

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

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

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

Figure 1–1 Raster Space and Model Space



In Figure 1–1:

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

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

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

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

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

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

- **Spatial reference system**, also called *GeoRaster SRS*, which maps cell coordinates (row,column,vertical) to model coordinates (X,Y,Z). Using the

spatial reference system with GeoRaster data is referred to as *georeferencing* the data. (Georeferencing is discussed in Section 1.6.)

- **Temporal reference system**, also called *GeoRaster TRS*, which maps cell coordinates (temporal) to model coordinates (T).
- **Band reference system**, also called *GeoRaster BRS*, which maps cell coordinates (band) to model coordinates (S, for Spectral).

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

1.4 GeoRaster Physical Storage

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

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

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

specification: band interleaved by line, blocking as (64,64,3), and 8 bands, each with 64 rows and 64 columns. In this case:

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

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

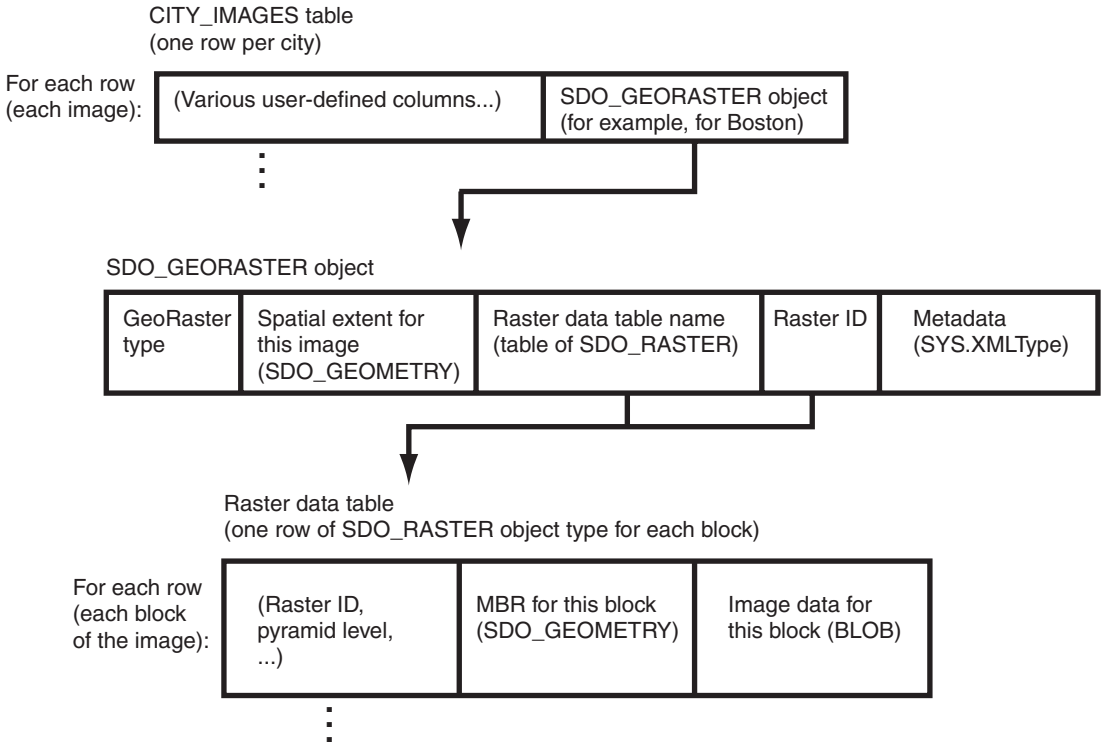
Based on this physical storage model, two object types are provided: SDO_GEORASTER for the raster data set and related metadata, and SDO_RASTER for each block in a raster image.

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

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

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

Figure 1–2 Physical Storage of GeoRaster Data



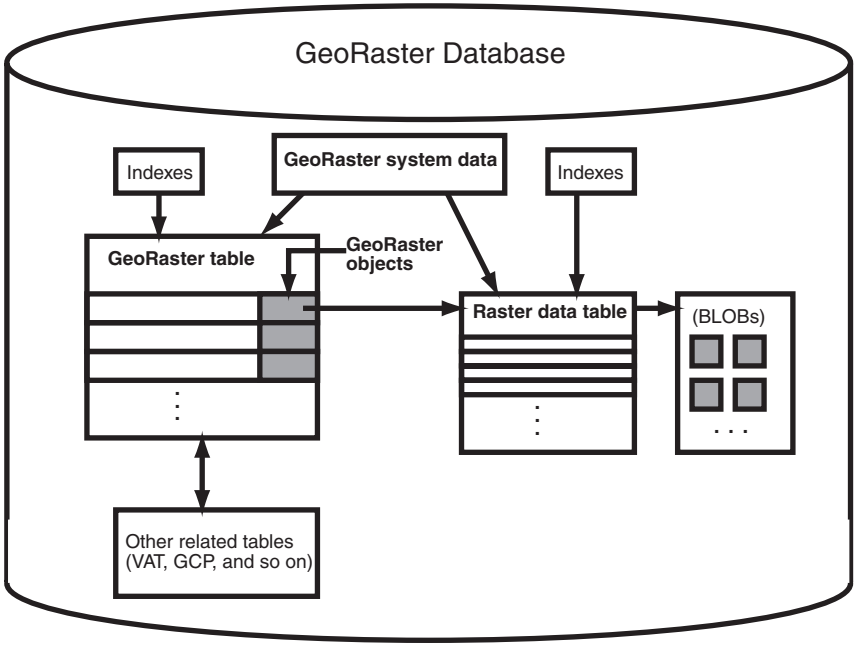
As shown in Figure 1–2:

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

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

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

Figure 1–3 GeoRaster Data in an Oracle Database



In Figure 1–3:

- Each GeoRaster object in the GeoRaster table has an associated raster data table, which has an entry for each block of the raster image.
- The BLOB with image data for each raster image block is stored separately from the raster table data. You can specify storage parameters (described in Section 1.4.1) for the BLOBs.
- Each GeoRaster object has a raster data table associated with it. However, a raster data table can store blocks of multiple GeoRaster objects, and GeoRaster objects in a GeoRaster table can be associated with one or multiple raster data tables.
- GeoRaster system data (described in Section 2.4) maintains the relationship between the GeoRaster tables and the raster data tables.

- Indexes (standard and Spatial) can be built on the GeoRaster table and raster data tables. For information about indexing GeoRaster data, see Section 3.6.
- Additional tables, such as ground control point (GCP) tables and value attribute tables (VATs), can be related to the GeoRaster objects. For more information about these additional related tables, see Section 2.4.6.

1.4.1 Storage Parameters

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

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

Table 1–1 storageParam Keywords for Raster Data

Keyword	Explanation
blocking	Specifies whether or not raster data is blocked. `TRUE` causes raster data to be blocked using the blocks of the specified or default `blockSize` value; `FALSE` causes raster data not to be blocked (that is, only one block will be used for the entire image). The default value for `blocking` is `TRUE` if you specify the `blockSize` keyword. If you specify `blocking=TRUE` but do not specify the `blockSize` keyword, the default `blockSize` is (256,256,*B*), where *B* is the number of bands in the output GeoRaster object. If you specify neither `blocking` nor `blockSize`, default values are derived from the source GeoRaster object: that is, if the original data is not blocked, the data in the output GeoRaster object is by default not blocked; and if the original data is blocked, the data in the output GeoRaster object is blocked with the same blocking scheme.

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=(256,64,3)` specifies 256 for the row dimension, 64 for the column dimension, and 3 for the band dimension; and `blocksize=(128,128)` specifies row and column block sizes of 128 for a GeoRaster object that has no band dimension. For the row and column dimensions, the value must be 1 or a positive power of 2 (2, 4, 8, 16, 32, 64, and so on). (These requirements apply only to the use of the `blocksize` keyword to perform reblocking; the dimension sizes of the original GeoRaster object can be any values.) See also the explanation of the `blocking` keyword. Only regular blocking is supported; that is, all blocks must be the same size and be aligned with each other, except for some top-level pyramids.
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`, and `64BIT_REAL`. If `cellDepth` is not specified, the value from the source GeoRaster object is used by default. Example: `celldepth=16BIT_U`
interleaving	Specifies the interleaving type. Must be one of the following values: `BSQ` (band sequential), `BIL` (band interleaved by line), or `BIP` (band interleaved by pixel). Example: `interleaving=BSQ`
pyramid	`TRUE` specifies to keep the original pyramid data; `FALSE` specifies not to keep the original pyramid data. The default value depends on the specific procedure: the default is `TRUE` for SDO_GEOR.copy, SDO_GEOR.changeFormat, and SDO_GEOR.changeFormatCopy; the default is `FALSE` for SDO_GEOR.scale, SDO_GEOR.scaleCopy, SDO_GEOR.mosaic, and SDO_GEOR.subset. (A value of `TRUE` is invalid and is ignored for SDO_GEOR.scale, SDO_GEOR.scaleCopy, SDO_GEOR.mosaic, or SDO_GEOR.subset.) You cannot generate pyramid data through the use of storage parameters; instead, you must use the SDO_GEOR.generatePyramid procedure after creating the GeoRaster object.

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 landsat values (21, '21', sdo_geor.init('T21'), null)
     returning tmimage into gr2;
  select tmimage into gr1 from landsat where id=2;
  sdo_geor.changeFormatCopy(gr1, 'blocksize=(128,128) pyramid=FALSE', gr2);
  update landsat set tmimage=gr2 where id=21;
  commit;
end;
/
```

In Example 1–1, the raster data table for GeoRaster object `gr2` is T21. If raster data is to be written into table T21, that table must exist before the PL/SQL block is run; otherwise, an error is generated by the SDO_GEOR.changeFormatCopy procedure.

1.4.2 Raster Data Table

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

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

Example 1–2 creates a raster data table named T21 (the same name specified in Example 1–1).

Example 1–2 Creating a Raster Data Table

```
CREATE TABLE t21 OF SDO_RASTER
  (PRIMARY KEY (rasterID, pyramidLevel, bandBlockNumber, rowBlockNumber,
    rowBlockNumber, columnBlockNumber))
  TABLESPACE tbs3 NOLOGGING
  LOB(rasterBlock) STORE AS lobseg
    (TABLESPACE  tbs3_2
     CHUNK 8192
```

```
 CACHE READS
 NOLOGGING
 PCTVERSION 0
 STORAGE (PCTINCREASE 0)
);
```

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

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

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

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

1.4.3 Blank and Empty GeoRaster Objects

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

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

1.5 Bands, Layers, and Interleaving

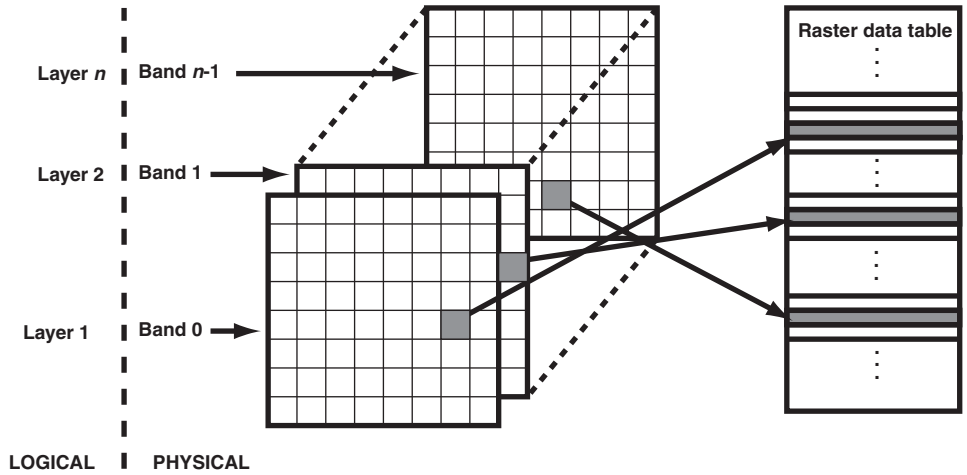
In GeoRaster, *band* and *layer* are different concepts. **Band** is a physical dimension of the multidimensional raster data set; that is, it is one ordinate in the cell space. For example, the cell space might have the ordinates row, column, and band. Bands are numbered from 0 to *n*-1, where *n* is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

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

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

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

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



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

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

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

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 GeoRaster object by calling the SDO_GEOR.changeFormat or SDO_GEOR.changeFormatCopy procedure, so that the data can be more efficiently processed and used.

1.6 Georeferencing

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

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

GeoRaster currently supports six-parameter affine transformation that georeferences two-dimensional raster data. The affine transformation is a special type of the Functional Fitting polynomial model. If an affine transformation is provided and is valid in the metadata, the GeoRaster object is considered georeferenced, and the isReferenced value in the SRS metadata will be TRUE; otherwise, it should be FALSE.

Rectification can be done with horizontal coordinates, so that cells of a GeoRaster data set can be mapped to a projection map coordinate system. After rectification, each cell is regularly sized in the map units and is aligned with the model coordinate system, that is, with the East-West dimension and the North-South dimension. If elevation data (DEM) is used in rectification, it is called **orthorectification**, a special form of rectification that corrects terrain displacement.

If a GeoRaster object is rectified, the `isRectified` value in its metadata will be `TRUE`; otherwise, it should be `FALSE`. If a GeoRaster object is orthorectified, the `isOrthoRectified` value in its metadata will be `TRUE`; otherwise, it should be `FALSE`.

To georeference a GeoRaster object, see Section 3.5.

1.6.1 GeoRaster Georeferencing Method

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

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

In these formulas:

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

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

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

This is the simplest case for georeferencing.

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

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

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

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

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

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

1.7 Pyramids

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

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

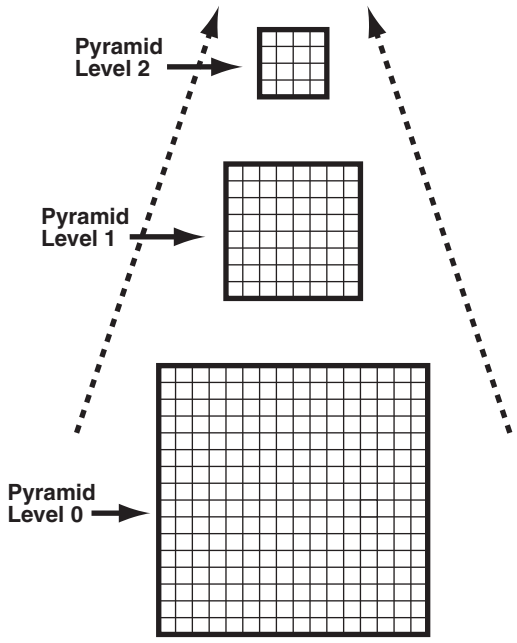
Pyramid type indicates the type of pyramid, and can be one of the following values:

- `DECREASE` means that pyramids decrease in size as the pyramid level increases.

- NONE means that there are no pyramids associated with the GeoRaster object.

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

Figure 1–5 Pyramid Levels



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

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

In the preceding formulas:

- r(0) and c(0) are the original row and column dimension size.
- r(n) and c(n) are the row and column dimension size of pyramid level *n*.

- `int` rounds off a number to the integer value that is less than but closest to that number.
- `2^n` means 2 to the power of *n*.

The smaller of the row and column dimension sizes of the smallest top-level pyramid is between 64 and 128. This decides the maximum reduced-resolution pyramid level, 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 and column dimension size.

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

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

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

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

- Average16 using 16 neighboring cells

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

- SDO_GEOR.generatePyramid generates pyramid data for a GeoRaster object.
- SDO_GEOR.deletePyramid deletes pyramid data for a GeoRaster object.
- SDO_GEOR.getPyramidMaxLevel returns the maximum pyramid level of a GeoRaster object.
- SDO_GEOR.getPyramidType returns the pyramid type for a GeoRaster object.

1.8 GeoRaster PL/SQL Subprogram Categories

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

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

1.8.1 Subprograms to Create, Load, and Export GeoRaster Data

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

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

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

1.8.2 Subprograms to Validate and Process GeoRaster Objects

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

Table 1–3 Subprograms to Validate and Process GeoRaster Objects

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

1.8.3 Subprogram for GeoRaster DML Triggers

Table 1–4 lists the subprogram for creating the standard GeoRaster trigger to fire after data manipulation language (DML) operations on a GeoRaster column in a GeoRaster table. (For information about using triggers with GeoRaster, see Section 3.1.) This subprogram is in the SDO_GEOR_UTL package.

Table 1–4 Subprogram for GeoRaster Triggers

Subprogram	Description
SDO_GEOR_UTL.createDMLTrigger	Creates the standard GeoRaster data manipulation language (DML) trigger on a GeoRaster column in a GeoRaster table, so that the appropriate operations are performed when its associated trigger is fired.

1.8.4 Subprograms to Get and Set GeoRaster Metadata and Data

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

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

For many of the *get* functions, there is a corresponding procedure, whose name starts with *set*, to set, modify, or delete the value of a metadata attribute. For most *set* procedures, to delete the value of a metadata attribute, 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.

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

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

Table 1–5 lists subprograms for getting and setting information about the whole GeoRaster object.

Table 1–5 Subprograms to Get and Set Whole Object Metadata

Subprogram	Description
SDO_GEOR.getID	Returns the user-defined identifier value associated with a GeoRaster object.
SDO_GEOR.setID	Sets a user-defined identifier to be associated with a GeoRaster object, or deletes the existing value if you specify a null `id` parameter
SDO_GEOR.getVersion	Returns the user-specified version of a GeoRaster object.
SDO_GEOR.setVersion	Sets the user-specified version of a GeoRaster object.
SDO_GEOR.setRasterType	Sets the raster type of a GeoRaster object.
SDO_GEOR.isSpatialReferenced	Returns `TRUE` if the GeoRaster object is spatially referenced, or `FALSE` if the GeoRaster object is not spatially referenced.
SDO_GEOR.setSpatialReferenced	Specifies whether or not a GeoRaster object is spatially referenced, or deletes the existing value if you specify a null `isReferenced` parameter.
SDO_GEOR.isBlank	Returns `TRUE` if the GeoRaster object is a blank GeoRaster object, or `FALSE` if the GeoRaster object is not a blank GeoRaster object.
SDO_GEOR.getBlankCellValue	Returns the cell value to be used for all cells if a specified GeoRaster image is a blank GeoRaster object.

Table 1–5 (Cont.) Subprograms to Get and Set Whole Object Metadata

Subprogram	Description
SDO_GEOR.setBlankCellValue	Sets the cell value to be used for all cells if a specified GeoRaster object is a blank GeoRaster object, or deletes the existing value if you specify a null `value` parameter.
SDO_GEOR.getDefaultColorLayer	Returns the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object.
SDO_GEOR.setDefaultColorLayer	Sets the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object, or deletes the existing values if you specify a null `defaultRGB` parameter.
SDO_GEOR.getDefaultRed	Returns the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object.
SDO_GEOR.setDefaultRed	Sets the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultRed` parameter.
SDO_GEOR.getDefaultGreen	Returns the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object.
SDO_GEOR.setDefaultGreen	Sets the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultGreen` parameter.
SDO_GEOR.getDefaultBlue	Returns the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object.
SDO_GEOR.setDefaultBlue	Sets the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultBlue` parameter.
SDO_GEOR.getInterleavingType	Returns the interleaving type for a GeoRaster object.
SDO_GEOR.getSpatialDimNumber	Returns the number of spatial dimensions of a GeoRaster object.
SDO_GEOR.getSpatialDimSizes	Returns the number of cells in each spatial dimension of a GeoRaster object.

Table 1–5 (Cont.) Subprograms to Get and Set Whole Object Metadata

Subprogram	Description
SDO_GEOR.getBandDimSize	Returns the number of bands in a GeoRaster object.
SDO_GEOR.getLayerDimension	Returns the dimension that is mapped as the logical layer dimension of a GeoRaster object.
SDO_GEOR.getTotalLayerNumber	Returns the total number of layers in a GeoRaster object.

Table 1–6 lists subprograms for getting and setting information about cell coordinates and values.

Table 1–6 Subprograms to Get and Set Cell Coordinates and Values

Subprogram	Description
SDO_GEOR.getModelCoordinate	Returns the coordinates in the model (ground) coordinate system associated with the point at the specified cell (raster) coordinates.
SDO_GEOR.getCellCoordinate	Returns the coordinates in the cell (raster) coordinate system associated with the point at the specified model (ground) coordinates.
SDO_GEOR.getCellValue	Returns the value of a single cell located anywhere in the GeoRaster object by specifying its row, column, and band number in its cell coordinate system, or by specifying a point geometry in its model coordinate system and its logical layer number.
SDO_GEOR.changeCellValue	Changes the value of raster data cells in a specified window of a GeoRaster object.
SDO_GEOR.getRasterSubset	Creates a single BLOB object containing all cells of a specified pyramid level that are inside or on the boundary of a specified window.
SDO_GEOR.getRasterBlocks	Returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that have any spatial interaction with a specified window.
SDO_GEOR.getRasterData	Creates a single BLOB object that contains all raster data of the input GeoRaster object at the specified pyramid level.
SDO_GEOR.getCellDepth	Returns the cell depth in bits.

Table 1–6 (Cont.) Subprograms to Get and Set Cell Coordinates and Values

Subprogram	Description
SDO_GEOR.getNODATA	Returns the value representing NODATA cells in a GeoRaster object.
SDO_GEOR.getULTCoordinate	Returns the cell coordinates of the upper-left corner of a GeoRaster object.
SDO_GEOR.setULTCoordinate	Sets the cell coordinate values of the upper-left corner of a GeoRaster object, or deletes the existing values if you specify a null `ultCoord` parameter.
SDO_GEOR.getCompressionType	Returns the compression type for a GeoRaster object.
SDO_GEOR.getBlockingType	Returns the blocking type for a GeoRaster object.
SDO_GEOR.getBlockSize	Returns the number of cells for each dimension in each block of a GeoRaster object in an array showing the number of cells for each row, column, and (if relevant) band.

Table 1–7 lists subprograms for getting and setting information about the GeoRaster spatial reference system.

Table 1–7 Subprograms to Get and Set Spatial Reference System Metadata

Subprogram	Description
SDO_GEOR.getSRS	Returns an object of type SDO_GEOR_SRS containing information related to the spatial referencing of a GeoRaster object.
SDO_GEOR.setSRS	Sets the spatial reference information of a GeoRaster object, or deletes the existing information if you specify a null `srs` parameter.
SDO_GEOR.isRectified	Returns `TRUE` if the GeoRaster object is identified as rectified, or `FALSE` if the GeoRaster object is not identified as rectified.
SDO_GEOR.setRectified	Specifies whether or not a GeoRaster object is rectified, or deletes the existing value if you specify a null `isRectified` parameter.
SDO_GEOR.isOrthoRectified	Returns `TRUE` if the GeoRaster object is identified as orthorectified, or `FALSE` if the GeoRaster object is not identified as orthorectified.

Table 1–7 (Cont.) Subprograms to Get and Set Spatial Reference System Metadata

Subprogram	Description
SDO_GEOR.setOrthoRectified	Specifies whether or not a GeoRaster object is orthorectified, or deletes the existing value if you specify a null `isOrthoRectified` parameter.
SDO_GEOR.getModelSRID	Returns the coordinate system (SDO_SRID value) associated with the model (ground) space for a GeoRaster object.
SDO_GEOR.setModelSRID	Sets the coordinate system (SDO_SRID value) for the model (ground) space for a GeoRaster object, or deletes the existing value if you specify a null `srid` parameter.
SDO_GEOR.getSpatialResolutions	Returns the spatial resolution value along each spatial dimension of a GeoRaster object.
SDO_GEOR.setSpatialResolutions	Sets the spatial resolution value along each spatial dimension of a GeoRaster object, or deletes the existing values if you specify a null `resolutions` parameter.

Table 1–8 lists subprograms for getting and setting information about the date and time that the GeoRaster data was collected.

Table 1–8 Subprograms to Get and Set Date and Time Metadata

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

Table 1–9 lists subprograms for getting and setting information about the imagery band reference system.

Table 1–9 Subprograms to Get and Set Imagery Band System Metadata

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

Table 1–10 lists subprograms for getting and setting information about layers (or bands).

Table 1–10 Subprograms to Get and Set Layer Metadata

Subprogram	Description
SDO_GEOR.getScaling	Returns the coefficients of the scaling function for a layer of a GeoRaster object.
SDO_GEOR.setScaling	Sets the scaling function associated with a layer, or deletes the existing value if you specify a null `scalingFunc` parameter.
SDO_GEOR.getBinTable	Returns the name of the bin table associated with a layer.
SDO_GEOR.setBinTable	Sets the name of the bin table associated with a layer, or deletes the existing value if you specify a null `tableName` parameter.
SDO_GEOR.getBinType	Returns the bin type associated with a layer.
SDO_GEOR.getStatistics	Returns statistical data associated with a layer.
SDO_GEOR.setStatistics	Sets statistical data associated with a layer.
SDO_GEOR.getHistogram	Returns the histogram for a layer.
SDO_GEOR.getHistogramTable	Returns the histogram table for a layer.
SDO_GEOR.setHistogramTable	Sets the histogram table for a layer.

Table 1–10 (Cont.) Subprograms to Get and Set Layer Metadata

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

Table 1–10 (Cont.) Subprograms to Get and Set Layer Metadata

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

Table 1–11 lists subprograms for getting information about pyramids.

Table 1–11 Subprograms to Get Pyramid Metadata

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

1.9 GeoRaster Tools: Loader, Viewer, Exporter

GeoRaster includes several client-side command line tools. If you selected the option to install demo files during the Oracle installation, these tools are in the following directory under the Spatial installation directory (which by default is *$ORACLE_HOME*/md):

```
/demos/georaster/java
```

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

- GeoRaster loader, which loads raster data into the GeoRaster objects. You can use this as an alternative to the SDO_GEOR.importFrom procedure, which is documented in Chapter 4. However, on non-Windows systems this loader tool does not support the BMP or GIF image formats, so you must use the SDO_GEOR.importFrom procedure with these formats on non-Windows systems. This tool does not support raster data that has a cell depth value of `2BIT` or source multiband raster data with BIL or BSQ interleaving types. The imported GeoRaster object has the BIP interleaving type. The loading operation of this tool cannot be rolled back.
- GeoRaster viewer, which displays GeoRaster objects and metadata in a Java viewer application. For more information about viewing GeoRaster objects, see Section 3.11.

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

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

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

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

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

1.10 GeoRaster PL/SQL Demo Files

GeoRaster includes several PL/SQL demo files that show common operations. If you selected the option to install demo files during the Oracle installation, these files are in the following directory under the Spatial installation directory (which by default is *`$ORACLE_HOME`*`/md`):

```
/demos/georaster/plsql
```

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

Table 1–12 GeoRaster PL/SQL Demo Files

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

2

GeoRaster Data Types and Related Structures

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

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

2.1 SDO_GEORASTER Object Type

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

The SDO_GEORASTER object type is defined as:

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

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

2.1.1 rasterType Attribute

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

- `[d]` identifies the number of spatial dimensions. Must be 2 for the current release.
- `[b]` indicates band or layer information: 0 means one band or layer; 1 means one or more than one band or layer. Note that you are *not* specifying the total number of bands or layers in this field. (For information about bands and layers, see Section 1.5.)
- `[t]` is reserved for future use and should be specified as 0 (zero).
- `[gt]` identifies the 2-digit GeoRaster type, and must be the following:

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

For example, a RasterType value of 20001 means:

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

2.1.2 spatialExtent Attribute

The `spatialExtent` attribute identifies the **spatial extent**, or *footprint*, associated with the raster data. The spatial extent is an Oracle Spatial geometry of type SDO_GEOMETRY. The spatial extent geometry can be in any coordinate system; however, it is in the model (ground) space of the GeoRaster object if the GeoRaster

object is georeferenced and if you call SDO_GEOR.generateSpatialExtent to generate the spatial extent geometry. The spatial extent geometry can also be in cell space that has a null SRID value. The SDO_GEOMETRY data type is described in *Oracle Spatial User's Guide and Reference*.

Because of the potential performance benefits of spatial indexing for GeoRaster applications, the geometry is associated with the `spatialExtent` attribute, rather than being included in the XML `metadata` attribute described in Section 2.1.5. For information about indexing GeoRaster data, see Section 3.6.

2.1.3 rasterDataTable Attribute

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

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

2.1.4 rasterID Attribute

The `rasterID` attribute value is stored in the rows of the raster data table to identify which rows belong to the GeoRaster object. The `rasterDataTable` attribute and `rasterID` attribute together uniquely identify the GeoRaster object in the database. That is, each GeoRaster object has a raster data table, although a raster data table can contain data from multiple GeoRaster objects.

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

2.1.5 metadata Attribute

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

2.2 SDO_RASTER Object Type and the Raster Data Table

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

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

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

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

2.2.1 rasterID Attribute

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

2.2.2 pyramidLevel Attribute

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

2.2.3 bandBlockNumber Attribute

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

2.2.4 rowBlockNumber Attribute

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

2.2.5 columnBlockNumber Attribute

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

2.2.6 blockMBR Attribute

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

2.2.7 rasterBlock Attribute

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

2.3 Other GeoRaster Types

In addition to SDO_GEORASTER (described in Section 2.1) and SDO_RASTER (described in Section 2.2), GeoRaster provides several other object and collection types, which are used for specific kinds of operations.

2.3.1 SDO_GEOR_HISTOGRAM Object Type

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

The SDO_GEOR_HISTOGRAM object type is defined as:

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

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

Table 2–1 SDO_GEOR_HISTOGRAM Object Type Attributes

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

2.3.2 SDO_GEOR_COLORMAP Object Type

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

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

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

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

The SDO_GEOR_COLORMAP object type is defined as:

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

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

cellValue value must be consistent with the cellDepth value of the GeoRaster object.

Table 2–2 SDO_GEOR_COLORMAP Object Type Attributes

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

2.3.3 SDO_GEOR_GRAYSCALE Object Type

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

The SDO_GEOR_GRAYSCALE object type is defined as:

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

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

Table 2–3 SDO_GEOR_GRAYSCALE Object Type Attributes

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

2.3.4 SDO_RASTERSET Collection Type

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

The SDO_RASTERSET collection type is defined as:

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

The SDO_RASTER type is described in Section 2.2.

2.3.5 SDO_GEOR_SRS Object Type

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

The SDO_GEOR_SRS object type is defined as:

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

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

Table 2–4 SDO_GEOR_SRS Object Type Attributes

Attribute	Description
isReferenced	`TRUE` if the GeoRaster object is georeferenced; `FALSE` if the GeoRaster object is not georeferenced.
isRectified	`TRUE` if the GeoRaster object is both georectified and georeferenced; `FALSE` if the GeoRaster object is not georectified.
isOrthoRectified	`TRUE` if the GeoRaster object is orthorectified, georectified, and georeferenced; `FALSE` if the GeoRaster object is not orthorectified.
srid	SRID value of the model (ground) coordinate system.
spatialResolution	Spatial resolution values: an array of numeric values, one for each spatial dimension. Each value indicates the number of units of measurement associated with the data area represented by that spatial dimension of a cell.
spatialTolerance	Tolerance value. (For an explanation of tolerance, see *Oracle Spatial User's Guide and Reference*.)
coordLocation	The model coordinate location representing either the upper-left corner or the center of each cell in the model space when cell coordinates (integer numbers) are converted to model coordinates (double numbers).
rowOff	Must be 0 (zero) for the current release.
columnOff	Must be 0 (zero) for the current release.
xOff	Must be 0 (zero) for the current release.
yOff	Must be 0 (zero) for the current release.
zOff	Must be 0 (zero) for the current release.
rowScale	Must be 1 for the current release.
columnScale	Must be 1 for the current release.
xScale	Must be 1 for the current release.
yScale	Must be 1 for the current release.
zScale	Must be 1 for the current release.
rowRMS	Must be NULL for the current release.
columnRMS	Must be NULL for the current release.
totalRMS	Must be NULL for the current release.

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

Attribute	Description
rowNumerator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the numerator of the row polynomial, where pType=1, nVars=2, order=1, and nCoefficients=3. The three coefficients are `a`, `b`, `c` in the formulas in Section 1.6.1.
rowDenominator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the denominator of the row polynomial, where pType=1, nVars=0, order=0, and nCoefficients=1. The value of the single coefficient must be 1 for the current release.
columnNumerator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the numerator of the column polynomial, where pType=1, nVars=2, order=1, and nCoefficients=3. The three coefficients are `d`, `e`, `f` in the formulas in Section 1.6.1.
columnDenominator	`pType`, `nVars`, `order`, `nCoefficients`, and all coefficients of the denominator of the column polynomial, where pType=1, nVars=0, order=0, and nCoefficients=1. The value of the single coefficient must be 1 for the current release.

2.4 GeoRaster System Data Views (xxx_SDO_GEOR_SYSDATA)

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

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

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

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

- USER_SDO_GEOR_SYSDATA contains system data for all GeoRaster objects owned by the user (schema).

- ALL_SDO_GEOR_SYSDATA contains system data for all GeoRaster objects on which the user has SELECT permission.

The GeoRaster system data table and the USER_SDO_GEOR_SYSDATA and ALL_SDO_GEOR_SYSDATA views should never be modified directly by users, although they are updated by the DML trigger that you must create on each SDO_GEORASTER column in each GeoRaster table. (For information about using GeoRaster triggers, See Section 3.1).

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 for VARCHAR2 data in any columns, names are stored in all uppercase characters.

2.4.1 TABLE_NAME Column

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

2.4.2 COLUMN_NAME Column

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

2.4.3 METADATA_COLUMN_NAME Column

The METADATA_COLUMN_NAME column is ignored for the current release.

2.4.4 RDT_TABLE_NAME Column

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

2.4.5 RASTER_ID Column

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

2.4.6 OTHER_TABLE_NAMES Column

The OTHER_TABLE_NAMES column is ignored for the current release.

2.5 GeoRaster XML Schema Table

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

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

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

Table 2–5 SDO_GEOR_XMLSCHEMA_TABLE Table Columns

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

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

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

3

GeoRaster Operations

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

1. Create the GeoRaster table, raster data table, and standard GeoRaster DML trigger (see Section 3.1 about the DML trigger).
2. Initialize or create GeoRaster objects (see Section 3.2).
3. Load raster imagery or grids (see Section 3.3).
4. Validate GeoRaster objects, if they have not already been validated (see Section 3.4).
5. Georeference the GeoRaster objects, if necessary (see Section 3.5).
6. Create spatial indexes or other indexes, or both (see Section 3.6).
7. Change the GeoRaster storage format, if necessary (see Section 3.7).
8. Query and update the GeoRaster metadata (see Section 3.8).
9. Query and update cell data (see Section 3.9).
10. Process GeoRaster objects (see Section 3.10).
11. View GeoRaster objects (see Section 3.11).
12. Export GeoRaster objects (see Section 3.12).
13. Transfer GeoRaster data between databases (see Section 3.13).
14. If necessary, deal with possible GeoRaster data problems (see Section 3.14).

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

Some operations can be performed using SQL, and some operations must be performed using PL/SQL blocks. For examples of these operations, see the demo files described in Section 1.10 and the examples in Chapter 4.

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

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

3.1 Creating the Standard GeoRaster DML Trigger

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

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

Each time you call the SDO_GEOR_UTL.createDMLTrigger procedure successfully, GeoRaster creates a trigger with a unique name. When you drop a GeoRaster table, all GeoRaster triggers associated with the table are automatically dropped also.

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

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

object is null), the trigger inserts a row into that view for the new GeoRaster object. If there is an entry in the USER_SDO_GEOR_SYSDATA view for the old GeoRaster object, the trigger updates the information to reflect the new GeoRaster object.

- After a delete operation, the trigger deletes raster data blocks for the GeoRaster object in its raster data table, and it deletes the row in the USER_SDO_GEOR_SYSDATA view for the GeoRaster object.

3.2 Creating New GeoRaster Objects

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

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

You cannot perform any GeoRaster operations if the object has not been properly created (that is, if the object is an atomic null). The SDO_GEOR.init and SDO_GEOR.createBlank functions initialize GeoRaster objects with their raster data table and raster ID values if these are not already specified, and 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, see Section 1.4.1, especially Example 1–2.

To avoid potential GeoRaster data problems (some of which are described in Section 3.14), always register an initialized GeoRaster object in the GeoRaster system views by inserting the GeoRaster object into a GeoRaster table, and do this before you perform any other operations on the GeoRaster object.

3.3 Loading GeoRaster Data

To load GeoRaster data, you have the following options:

- Call the SDO_GEOR.importFrom procedure to load images into GeoRaster objects.
- Use the GeoRaster loader tool, which is described in Section 1.9.

3.4 Validating GeoRaster Objects

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

If you used the GeoRaster loader tool described in Section 1.9, the GeoRaster objects were validated during the load operation.

GeoRaster provides the following validation subprograms:

- SDO_GEOR.validateGeoraster validates the GeoRaster object, including cell data and metadata. It returns `TRUE` if the object is valid; otherwise, it returns one of the following: an Oracle error code indicating why the GeoRaster object is invalid, `FALSE` if validation fails for an unknown reason, or `NULL` if the GeoRaster object is null. You should always use this function after you create a GeoRaster object.
- SDO_GEOR.schemaValidate validates the metadata against the GeoRaster XML schema. You can use this function to locate errors if the SDO_GEOR.validateGeoraster function returned the error code 13454. The SDO_GEOR.schemaValidate and SDO_GEOR.validateGeoraster functions do not validate the spatial extent geometry.

3.5 Georeferencing GeoRaster Objects

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

- If the original image is already georeferenced and if the georeferencing information is stored in an ESRI world file, you can use the SDO_GEOR.importFrom procedure to load an ESRI world file from a file or from a CLOB object, along with the image data itself (in either FILE or BLOB format). You can also use the GeoRaster client-side loader tool (described in Section 1.9) to load an ESRI world file from a file, along with the image file itself.

  Because an ESRI world file does not specify the model coordinate system, after loading a world file you can call the SDO_GEOR.setModelSRID procedure to set the model space of the georeferenced GeoRaster object using an Oracle

SRID. You can also call this procedure to change the model space of a georeferenced GeoRaster object.

- You can use the SDO_GEOR.setSRS procedure to add, modify, and delete georeferencing information. For example, you can create an SDO_GEOR_SRS object and assign the coefficients and related georeferencing information, and then call the setSRS procedure to add or update the spatial reference information of any GeoRaster object. If you know that one GeoRaster object has the same SRS information as another GeoRaster object, you can call the SDO_GEOR.getSRS function to retrieve an SDO_GEOR_SRS object from this GeoRaster object, and then call the SDO_GEOR.setSRS procedure to georeference the first GeoRaster object.
- You can call the SDO_GEOR.georeference procedure to georeference a GeoRaster object directly. This function takes the coefficients `A`, `B`, `C`, `D`, `E`, `F` (described in a formula in Section 1.6.1) and other information, converts them into the coefficients `a`, `b`, `c`, `d`, `e`, `f`, and stores them in the spatial reference information of a GeoRaster object. If the original raster data is rectified and if the model coordinate of its origin (upper-left corner) is (x0, y0) and its spatial resolution or scale is `s`, then the following are true: A = s, B = 0, C = x0, D = 0, E = -s, F = y0.

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

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

3.6 Indexing GeoRaster Data

GeoRaster data can be indexed in various ways. The most important index you can create on a GeoRaster object is a spatial index on the spatial extent (footprint) geometry of the GeoRaster object (`spatialExtent` attribute, described in

Section 2.1.2). For information about creating spatial indexes, see *Oracle Spatial User's Guide and Reference*.

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

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

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

3.7 Changing Raster Storage

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

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

3.8 Querying and Updating GeoRaster Metadata

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

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

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

See Section 1.8.4 for categories and lists of subprograms that get and set GeoRaster metadata and cell data.

3.9 Querying and Updating Cell Data

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

- SDO_GEOR.getCellValue returns the value of a single cell of the GeoRaster object.
- SDO_GEOR.getRasterSubset returns a single BLOB object containing all cells of a precise subset of the GeoRaster object (as specified by a window, layer or band numbers, and pyramid level). This BLOB object contains only raster cells and no related metadata.
- SDO_GEOR.getRasterData creates a single BLOB object containing all cells of the GeoRaster object at a specified pyramid level. This BLOB object contains only raster cells and no related metadata.
- SDO_GEOR.getRasterBlocks returns an object that includes all image data inside or touching a specified window. Specifically, it returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that are inside or touch a specified window.

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

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

See Section 1.8.4 for categories and lists of subprograms that get and set GeoRaster metadata and cell data.

3.10 Processing GeoRaster Objects

You can perform a variety of processing operations on GeoRaster data, including changing the format, subsetting (cropping), scaling, and generating pyramids. See the GeoRaster PL/SQL demo files, described in Section 1.10, for examples and explanatory comments.

3.11 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 provided with GeoRaster.

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

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

3.12 Exporting GeoRaster Objects

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

- Call the SDO_GEOR.exportTo procedure (which allows you to export either to a file or a BLOB object).
- Use the GeoRaster exporter tool, described in Section 1.9.

3.13 Transferring GeoRaster Data Between Databases

You can use either the Data Pump Export and Import utilities (new for Oracle Database 10*g* Release 1) 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 at the same time. After the transfer, you may also need to insert the GeoRaster system data for the imported GeoRaster objects into the USER_SDO_GEOR_SYSDATA view (described in Section 2.4) in the target schema, and you should use the SDO_GEOR.validateGeoraster function to check the validity of imported GeoRaster objects.

For small data sets or where performance is not a concern, you can follow these steps:

1. Export and import the GeoRaster table and raster data table definitions (if they do not exist in the target schema). If you use the Data Pump Export and Import

utilities, set CONTENT=METADATA_ONLY. If you use the original Export and Import utilities, set ROWS=N.

2. Export and import the GeoRaster object and raster data. If you use the Data Pump Export and Import utilities, set CONTENT=DATA_ONLY. If you use the original Export and Import utilities, set IGNORE=Y.

The preceding approach ensures that during the data import in Step 2, any GeoRaster-related DML triggers defined on a GeoRaster table are fired, thus automatically maintaining the GeoRaster system data in the target schema. However, Step 2 uses conventional loading, so performance is not optimal.

For large data sets where performance is a concern, you can use an approach that exports and imports table definitions and table data together. This approach requires you to maintain the GeoRaster system data in the target schema after the import operation. For the following steps, assume that Table mode is used for the export and import operations.

1. Create a table to hold all related GeoRaster system data. For example:

```
CREATE TABLE tmp_sysdata_table
  AS SELECT * FROM user_sdo_geor_sysdata
    WHERE table_name='<table_to_be_exported>';
```

   In the preceding example, replace <table_to_be_exported> with the name of the GeoRaster table to be exported.

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

   Perform the operation or operations in this step once for each source schema and target schema pair, because the USER_SDO_GEOR_SYSDATA view is schema-specific.

2. Export and import the table that you created in Step 1, together with GeoRaster tables and their related raster data tables.

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

```
INSERT INTO user_sdo_geor_sysdata SELECT * FROM tmp_sysdata_table;
```

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

In similar manner, if you export and import GeoRaster data using a mode other than Table mode (such as Full, Schema, Tablespace, or Transportable Tablespace),

you must also maintain the GeoRaster system data in the target schema after the import operation.

The success of importing GeoRaster data into a target schema depends on there being no conflicts in the target schema's GeoRaster system data. That is, the pairs of raster data table name and raster ID to be inserted into the target schema's USER_SDO_GEOR_SYSDATA view must be unique. If a conflict occurs, you must modify one of the GeoRaster objects involved in the conflict in either the source or the target schema. To avoid moving data around, fixing a conflict usually means changing the raster ID of one GeoRaster object to another number. For example:

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

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

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

If you import GeoRaster-related DML triggers with GeoRaster tables (as is usually the case), you should drop these triggers and re-create them using the SDO_GEOR_UTL.createDMLTrigger procedure after the import operation finishes. The GeoRaster-related DML triggers have names that start with `GRDMLTR_`.

3.14 Dealing with Possible GeoRaster Data Problems

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

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

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

If dangling raster blocks exist, they cause wasted disk space in the raster data table, although otherwise they do not present a problem as long as the necessary primary key is defined on the raster data table. If you want to remove the raster data table rows associated with dangling raster blocks, you can try to find rows associated with problems and to fix the problems. To find rows associated with problems, follow these steps:

1. To find out whether there is any dangling raster block data in a raster data table, issue a query of the following general form:

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

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

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

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

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

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

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

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

4

SDO_GEOR Package Reference

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

The subprograms are presented in alphabetical order in this chapter. They can be grouped into several logical categories, as explained in Section 1.8. Many of the subprograms are also discussed in Chapter 3, "GeoRaster Operations".

SDO_GEOR.changeCellValue

Format

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

or

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

Description

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

Parameters

georaster
GeoRaster object.

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

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

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

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

Usage Notes

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

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

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

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

If `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.

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

SDO_GEOR.changeFormat

Format

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

Description

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

Parameters

georaster
The SDO_GEORASTER object whose format is to be changed.

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

Usage Notes

This procedure changes the input GeoRaster object. To create a new GeoRaster object that has the specified changes, use the SDO_GEOR.changeFormatCopy procedure.

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

An exception is raised if `georaster` is invalid.

Examples

The following example changes the interleaving type of a GeoRaster object to `BIL`.

```
DECLARE
    gr1 sdo_georaster;
BEGIN
    SELECT grobj INTO gr1 from t0 WHERE id=11 FOR UPDATE;
    sdo_geor.changeFormat(gr1, 'interleaving=BIL');
    UPDATE t0 SET grobj=gr1 WHERE id=11;
    COMMIT;
```

```
END;
/
```

SDO_GEOR.changeFormatCopy

Format

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

Description

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

Parameters

inGeoraster
The SDO_GEORASTER object whose format is to be copied.

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

outGeoraster
The SDO_GEORASTER object to hold the copy.

Usage Notes

This procedure does not change the input GeoRaster object, but creates a new GeoRaster object that has the specified changes. To change the input GeoRaster object, use the SDO_GEOR.changeFormat procedure.

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

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

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

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

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

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

Examples

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

```
DECLARE
    gr1 sdo_georaster;
    gr2 sdo_georaster;
BEGIN
    SELECT grobj INTO gr2 from t0 WHERE id=11 FOR UPDATE;
    SELECT grobj INTO gr1 from t0 WHERE id=1;

    sdo_geor.changeFormatCopy(gr1, 'blocksize=(2048,2048)', gr2);
    UPDATE t0 SET grobj=gr2 WHERE id=11;
    COMMIT;
END;
/
```

SDO_GEOR.copy

Format

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

Description

Makes a copy of an existing GeoRaster object.

Parameters

inGeoraster
GeoRaster object to be copied.

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

Usage Notes

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

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

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

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

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

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

Examples

The following example inserts an initialized GeoRaster object (gr2) into the GROBJ column of table T0, makes gr2 an exact copy of another GeoRaster object (gr1), and updates the row that had been inserted using gr2 for the GROBJ column value.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO t0 VALUES (11, sdo_geor.init('RDT_11', 1)) returning grobj
    INTO gr2;
  SELECT grobj INTO gr1 from t0 WHERE id=1;

  sdo_geor.copy(gr1, gr2);
  UPDATE t0 SET grobj=gr2 WHERE id=11;
  COMMIT;
END;
/
```

SDO_GEOR.createBlank

Format

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

Description

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

Parameters

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

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

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

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

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

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

Usage Notes

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

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

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

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

Examples

The following example creates a table, and inserts a row containing a blank GeoRaster object into the table.

```
create table t0 (id number, grobj sdo_georaster);

INSERT INTO t0 VALUES (
  1,
  sdo_geor.createBlank(20002, SDO_NUMBER_ARRAY(0,0),
                       SDO_NUMBER_ARRAY(1024,1024), 255, 't_cb01', 10)
);
```

SDO_GEOR.deletePyramid

Format

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

Description

Deletes the pyramid data of a GeoRaster object.

Parameters

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

Usage Notes

For information about pyramid data, see Section 1.7.

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

An exception is raised if `georaster` is invalid.

Examples

The following example deletes the pyramid data for a GeoRaster object.

```
DECLARE
  gr1 sdo_georaster;
BEGIN
  SELECT tmimage INTO gr1 FROM landsat WHERE id=21;

  sdo_geor.deletePyramid(gr1);
  UPDATE landsat SET tmimage=gr1 WHERE id=21;
  COMMIT;
END;
/
```

SDO_GEOR.exportTo

Format

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

or

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

or

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

Description

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

Parameters

georaster
GeoRaster object that will be exported.

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

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

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

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

r_destName
Destination file name (with full path specification) if `destType` is `FILE`. Do not specify the file extension.

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

h_destFormat
Geoheader destination format. Must be `WORLDFILE`.

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

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

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

Usage Notes

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

This procedure does not support JPEG or GIF as a destination file format. You can use the client-side GeoRaster exporter tool, described in Section 1.9, to export to a JPEG file.

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

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

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

```
call dbms_java.grant_permission('HERMAN','SYS:java.io.FilePermission',
   'sdo/demos/georaster/data/img1.tif', 'write' );
```

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 MDSYS.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.
mdsys.sdo_geor.exportTo(geor, NULL, 'BMP', 'file',
  'sdo/demos/georaster/data/img1_export');

-- Export a subset to a file with a world file.
fileName := '/mydir/parrotExported';
tfwName := '/mydir/parrotWorldFile';
SELECT georaster INTO geor FROM georaster_table WHERE georid = 8;
mdsys.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 MDSYS.SDO_GEORASTER;
  geor2 MDSYS.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;
mdsys.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;
mdsys.sdo_geor.exportTo(geor2, 'cropArea=(0,0,500,500)', 'TIFF', lobd2,
  'WORLDFILE', lobd3);
UPDATE blob_table set blob_col = lobd2, clob_col = lobd3 where blobid = 4;
COMMIT;

END;
/
```

SDO_GEOR.generatePyramid

Format

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

Description

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

Parameters

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

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

Usage Notes

For information about pyramid data, see Section 1.7.

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

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

the 16 nearest cells in the original GeoRaster object), CUBIC (cubic convolution of the 16 nearest cells in the original GeoRaster object).

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

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

If you do not specify an rLevel value or 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 / 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.

An exception is raised if georaster is invalid.

Examples

The following example creates pyramid data for a GeoRaster object.

```
DECLARE
  gr sdo_georaster;
BEGIN

  SELECT georaster INTO gr
    from georaster_table where georid = 6 FOR UPDATE;

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

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

  COMMIT;
END;
/
```

SDO_GEOR.generateSpatialExtent

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

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

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

Examples

The following examples return the spatial extent geometry of GeoRaster objects in the TMIMAGE column of the LANDSAT table.

```
SELECT sdo_geor.generateSpatialExtent(tmimage) spatialExtent
  FROM landsat WHERE id=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(tmimage) spatialExtent
  FROM landsat WHERE id=4;

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

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

SDO_GEOR.georeference

Format

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

Description

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

Parameters

georaster
The SDO_GEORASTER object to be georeferenced.

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

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

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

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

Usage Notes

Use this procedure to georeference a GeoRaster object. Georeferencing is explained in Section 1.6 and Section 3.5.

This procedure sets the spatial resolutions if the georeferenced GeoRaster object is rectified; otherwise, it might not set the spatial resolutions. If necessary, call the SDO_GEOR.setSpatialResolutions procedure after calling this procedure.

The following also perform operations related to georeferencing:

- The SDO_GEOR.setSRS procedure adds, modifies, and deletes georeferencing information.
- The SDO_GEOR.importFrom procedure can load an ESRI world file from a file or from a CLOB object.
- The GeoRaster loader tool (described in Section 1.9) can load an ESRI world file from a file.

Examples

The following example georeferences a GeoRaster object, and it calls the SDO_GEOR.getSRS function to retrieve information related to the spatial referencing of the object.

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

PL/SQL procedure successfully completed.

SET NUMWIDTH 20
SELECT id, sdo_geor.getSRS(tmimage) SRS FROM landsat WHERE id = 1;

                  ID
--------------------
SRS(ISREFERENCED, ISRECTIFIED, ISORTHORECTIFIED, SRID,
SPATIALRESOLUTION, SPATIA
--------------------------------------------------------------------------------
```

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

SDO_GEOR.getBandDimSize

Format

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

Description

Returns the number of bands in a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For an explanation of bands, see Section 1.5.

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

Examples

The following example returns the spatial dimension sizes and the number of bands (one in this case) for a GeoRaster object. (The output is reformatted for readability.)

```
SELECT sdo_geor.getSpatialDimSizes(tmimage) spatialDimSizes,
       sdo_geor.getBandDimSize(tmimage) bandDimSize
  FROM landsat WHERE id=21;

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

SDO_GEOR.getBeginDateTime

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

Examples

The following example returns the beginning and ending dates and times for raster data collection in the metadata for the GeoRaster object in a table named LANDSAT where the ID column contains the value 4.

```
SELECT sdo_geor.getBeginDateTime(tmimage) beginDateTime,
  sdo_geor.getEndDateTime(tmimage) endDateTime
  FROM landsat WHERE id=4;

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

SDO_GEOR.getBinTable

Format

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

Description

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

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 LANDSAT.

```
SELECT sdo_geor.getBinTable(tmimage, 4) FROM landsat WHERE id=4;
```

SDO_GEOR.getBinType

Format

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

Description

Returns the bin type associated with a layer.

Parameters

georaster
GeoRaster object.

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

Usage Notes

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

The `LINEAR` bin type is defined as follows:

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

The `LOGARITHM` bin type is defined as follows:

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

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

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

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

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

Examples

The following example returns the bin types for layers 0 and 1 of a GeoRaster object. (The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getBinType(tmimage, 0),1,20) binType0,
       substr(sdo_geor.getBinType(tmimage, 1),1,20) binType1
  FROM landsat WHERE id=4;

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

SDO_GEOR.getBlankCellValue

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

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

Examples

The following example returns the blank cell values for all blank GeoRaster objects in the GROBJ column of table T0.

```
SELECT id, sdo_geor.getBlankCellValue(grobj) blankValue from t0
 where sdo_geor.isBlank(grobj)='TRUE';

        ID BLANKVALUE
---------- ----------
         1        255
```

```
2         155
```

SDO_GEOR.getBlockingType

Format

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

Description

Returns the blocking type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

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

Examples

The following example returns the cell depth, interleaving type, and blocking type of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 21 in the LANDSAT table.

```
SELECT sdo_geor.getCellDepth(tmimage) CellDepth,
       substr(sdo_geor.getInterleavingType(tmimage),1,8) interleavingType,
       substr(sdo_geor.getBlockingType(tmimage),1,8) blocking
  FROM landsat WHERE id=21;

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

SDO_GEOR.getBlockSize

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

Examples

The following example returns the number of cells (512 in each dimension) in each block of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 21 in the LANDSAT table.

```
SELECT sdo_geor.getBlockSize(tmimage) blockSize
   FROM landsat WHERE id=21;

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

SDO_GEOR.getCellCoordinate

Format

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

Description

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

Parameters

georaster
GeoRaster object.

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

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

Usage Notes

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

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

Examples

The following example returns the cell coordinates in the raster image associated with model coordinate values (32343.64,7489527.23) in a specified GeoRaster object.

```
SELECT sdo_geor.getCellCoordinate(tmimage, 0, sdo_geometry(2001,82394,
  sdo_point_type(32343.64,7489527.23,null), null,null)) coord
FROM landsat WHEREid=4;

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

SDO_GEOR.getCellDepth

Format

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

Description

Returns the cell depth in bits.

Parameters

georaster
GeoRaster object.

Usage Notes

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

Examples

The following example returns the cell depth, interleaving type, and blocking type of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 21 in the LANDSAT table.

```
SELECT sdo_geor.getCellDepth(tmimage) CellDepth,
       substr(sdo_geor.getInterleavingType(tmimage),1,8) interleavingType,
       substr(sdo_geor.getBlockingType(tmimage),1,8) blocking
  FROM landsat WHERE id=21;

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

SDO_GEOR.getCellValue

Format

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

or

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

Description

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

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

Parameters

georaster
GeoRaster object.

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

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

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

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.

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

Usage Notes

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

Examples

The following example returns the values of four cells of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 21 in the LANDSAT table.

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

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

SDO_GEOR.getColorMap

Format

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

Description

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

Parameters

georaster
GeoRaster object.

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

Usage Notes

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

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

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

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

Examples

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

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

```
SDO_GEOR.GETCOLORMAP(GEORASTER,1)(CELLVALUE, RED, GREEN, BLUE, ALPHA)
--------------------------------------------------------------------------------
SDO_GEOR_COLORMAP(SDO_NUMBER_ARRAY(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126,
127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158,
159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190,
191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206,
207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,
223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238,
239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254,
255), SDO_NUMBER_ARRAY(180, 180, 180, 180, 180, 180, 180, 180, 180, 180, 180, 18
0, 127, 127, 100, 50, 50, 127, 159, 191, 223, 255, 255, 255, 255, 218, 182, 145,
 109, 72, 36, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 36, 72, 109, 145, 182, 218, 255, 200, 206, 212, 218, 224, 230, 236, 242,
248, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255), SDO_NUMBER_ARRAY(127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 180, 127, 50, 100, 50, 127,
159, 191, 223, 255, 200, 150, 100, 122, 144, 166, 188, 210, 232, 255, 255, 255,
248, 241, 234, 227, 220, 213, 206, 200, 150, 100, 87, 75, 62, 50, 37, 25, 12, 0,
200, 127, 0, 0, 0, 0, 0, 0, 0, 0, 0, 28, 56, 85, 113, 141, 170, 198, 226, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
```

```
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255), SDO_NUMBER_ARRAY(127, 127, 127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 180, 50, 50, 100, 127, 95, 63, 31,
0, 0, 0, 0, 18, 36, 54, 72, 90, 108, 127, 100, 50, 43, 37, 31, 25, 18, 12, 6, 0,
0, 0, 31, 63, 95, 127, 159, 191, 223, 255, 255, 255, 127, 108, 90, 72, 54, 36,
18, 0, 0, 28, 56, 85, 113, 141, 170, 198, 226, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255), SDO_NUMBER_ARRAY(255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255))
```

SDO_GEOR.getColorMapTable

Format

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

Description

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

Parameters

georaster
GeoRaster object.

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

Usage Notes

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

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

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

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

Examples

The following example returns the colormap table for layer 2 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
SELECT sdo_geor.getColorMapTable(tmimage, 2) FROM landsat WHERE id=4;
```

```
SDO_GEOR.GETCOLORMAPTABLE(TMIMAGE,2)
--------------------------------------------------------------------------------
CMT1

1 row selected.
```

SDO_GEOR.getCompressionType

Format

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

Description

Returns the compression type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For this release, this function always returns `NONE`, which means that the GeoRaster object is not compressed.

Examples

The following example returns the compression type for the GeoRaster objects (TIMIAGE column) in the LANDSAT table.

```
SELECT id, substr(sdo_geor.getCompressionType(tmimage),1,20) compressionType
  FROM landsat;

        ID COMPRESSIONTYPE
---------- --------------------
         2 NONE
         4 NONE
```

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 LANDSAT.

```
SELECT id, sdo_geor.getDefaultRed(tmimage) red,
  sdo_geor.getDefaultGreen(tmimage) green,
  sdo_geor.getDefaultBlue(tmimage) blue
FROM landsat;

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

SDO_GEOR.getDefaultColorLayer

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GROBJ column) in table T0, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 2);
  sdo_geor.setDefaultGreen(grobj, 3);
  sdo_geor.setDefaultBlue(grobj, 1);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
```

```
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(tmimage) FROM landsat WHERE id=4;

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

1 row selected.
```

SDO_GEOR.getDefaultGreen

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

Examples

The following example returns the layer numbers for the red, blue, and green color components for displaying the GeoRaster objects in the table named LANDSAT.

```
SELECT id, sdo_geor.getDefaultRed(tmimage) red,
  sdo_geor.getDefaultGreen(tmimage) green,
  sdo_geor.getDefaultBlue(tmimage) blue
FROM landsat;

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

SDO_GEOR.getDefaultRed

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

Examples

The following example returns the layer numbers for the red, blue, and green color components for displaying the GeoRaster objects in the table named LANDSAT.

```
SELECT id, sdo_geor.getDefaultRed(tmimage) red,
  sdo_geor.getDefaultGreen(tmimage) green,
  sdo_geor.getDefaultBlue(tmimage) blue
FROM landsat;

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

SDO_GEOR.getEndDateTime

Format

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

Description

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

Parameters

georaster
GeoRaster object.

Usage Notes

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

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

Examples

The following example returns the beginning and ending dates and times for raster data collection in the metadata for the GeoRaster object in a table named LANDSAT where the ID column contains the value 4.

```
SELECT sdo_geor.getBeginDateTime(tmimage) beginDateTime,
  sdo_geor.getEndDateTime(tmimage) endDateTime
  FROM landsat WHERE id=4;

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

SDO_GEOR.getGrayScale

Format

```
SDO_GEOR.getGrayScale(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_GEOR_GRAYSCALE;
```

Description

Returns the grayscale mappings for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

This function returns an object of type SDO_GEOR_GRAYSCALE. Section 2.3.3 describes grayscale display and this object type.

To set the grayscale mappings for a layer in a GeoRaster object, use the SDO_GEOR.setGrayScale procedure.

Examples

The following example returns the grayscale mappings for layer 0 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 0 in the LANDSAT table.

```
SELECT sdo_geor.getGrayScale(tmimage, 0) FROM landsat WHERE id=0;

SDO_GEOR.GETGRAYSCALE(TMIMAGE,0)(CELLVALUE, GRAY)
--------------------------------------------------------------------------------
SDO_GEOR_GRAYSCALE(SDO_NUMBER_ARRAY(10, 20, 30, 255), SDO_NUMBER_ARRAY(180, 210,
```

```
                230, 250))
```

SDO_GEOR.getGrayScaleTable

Format

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

Description

Returns the grayscale mapping table for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

This function returns the name of a user-defined grayscale table. Section 2.3.3 describes grayscale display.

To set the grayscale mapping table for a layer in a GeoRaster object, use the SDO_GEOR.setGrayScaleTable procedure.

Examples

The following example returns the grayscale mapping tables for layers 0, 1, 2, and 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table. (The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getGrayScaleTable(tmimage, 0),1,20) grayScaleTable0,
       substr(sdo_geor.getGrayScaleTable(tmimage, 1),1,20) grayScaleTable1,
       substr(sdo_geor.getGrayScaleTable(tmimage, 2),1,20) grayScaleTable2,
       substr(sdo_geor.getGrayScaleTable(tmimage, 3),1,20) grayScaleTable3
  FROM landsat WHERE id=4;
```

```
GRAYSCALETABLE0      GRAYSCALETABLE1      GRAYSCALETABLE2      GRAYSCALETABLE3
-------------------- -------------------- -------------------- ----------------
SCL0                 SCL1                 SCL2                 SCL3
```

SDO_GEOR.getHistogram

Format

```
SDO_GEOR.getHistogram(
   georaster      IN SDO_GEORASTER,
   layerNumber  IN NUMBER
   ) RETURN SDO_GEOR_HISTOGRAM;
```

Description

Returns the histogram for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

This function returns an object of type SDO_GEOR_HISTOGRAM. Section 2.3.1 describes this object type and briefly discusses histograms.

Examples

The following example returns the histogram for layer 1 of a 4-bit GeoRaster object in the IMAGES table.

```
SELECT sdo_geor.getHistogram(tmimage, 1) layer1
 FROM images WHERE id=17;

LAYER1(CELLVALUE, COUNT)
--------------------------------------------------------------------------------
SDO_GEOR_HISTOGRAM(SDO_NUMBER_ARRAY(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12, 13,
 14, 15), SDO_NUMBER_ARRAY(10, 18, 10, 110, 200, 120, 130, 150, 160, 103, 106,
 190, 12, 17, 10, 5))
```

SDO_GEOR.getHistogramTable

Format

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

Description

Returns the histogram table for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

This function returns a user-defined histogram table. Section 2.3.1 briefly discusses histograms.

To set the name of the histogram table for a layer, use the SDO_GEOR.setHistogramTable procedure.

Examples

The following example returns the histogram tables for layers 0 (the whole object), 1, 2, and 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table. (The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getHistogramTable(tmimage, 0),1,20) histogramTable0,
       substr(sdo_geor.getHistogramTable(tmimage, 1),1,20) histogramTable1,
       substr(sdo_geor.getHistogramTable(tmimage, 2),1,20) histogramTable2,
       substr(sdo_geor.getHistogramTable(tmimage, 3),1,20) histogramTable3
```

```
  FROM landsat WHERE id=4;

HISTOGRAMTABLE0      HISTOGRAMTABLE1      HISTOGRAMTABLE2      HISTOGRAMTABLE3
-------------------- -------------------- -------------------- ----------------
HIST0                HIST1                HIST2                HIST3
```

SDO_GEOR.getID

Format

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

Description

Returns the user-defined identifier value associated with a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

To set a user-defined identifier value for a GeoRaster object, use the SDO_GEOR.setID procedure.

Examples

The following example returns the user-defined identifier values of the GeoRaster objects (TMIMAGE column) in the LANDSAT table.

```
SELECT id, substr(sdo_geor.getID(tmimage),1,50) GEOR_ID FROM landsat;

        ID GEOR_ID
---------- --------------------------------------------------
         2 TM_102
         4 TM_104
```

SDO_GEOR.getInterleavingType

Format

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

Description

Returns the interleaving type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns one of the following values: `BSQ` (band sequential), `BIL` (band interleaved by line), or `BIP` (band interleaved by pixel).

To change the interleaving type for a GeoRaster object, use the SDO_GEOR.changeFormat or 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 (TMIMAGE column) in the row with the ID column value of 21 in the LANDSAT table.

```
SELECT sdo_geor.getCellDepth(tmimage) CellDepth,
       substr(sdo_geor.getInterleavingType(tmimage),1,8) interleavingType,
       substr(sdo_geor.getBlockingType(tmimage),1,8) blocking
  FROM landsat WHERE id=21;

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

SDO_GEOR.getLayerDimension

Format

```
SDO_GEOR.getLayerDimension(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_STRING_ARRAY;
```

Description

Returns the dimension that is mapped as the logical layer dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The **layer dimension** refers to the physical entity associated with the logical term *layer*. For the current release, the only supported layer dimension is `BAND`: that is, the logical concept *layer* is associated with the physical term *band*, as shown in Figure 1–4 in Section 1.5. In this case, layers will be mapped to the `BAND` dimension, so that the first layer is band 0, the second layer is band 1, and so on.

Examples

The following example returns the layer dimension of each GeoRaster object (TMIMAGE column) in the LANDSAT table. (The output is reformatted for readability.)

```
SELECT id, sdo_geor.getLayerDimension(tmimage) FROM landsat;

        ID SDO_GEOR.GETLAYERDIMENSION(TMIMAGE)
---------- --------------------------------------------------------------
         2 SDO_STRING_ARRAY('BAND')
         4 SDO_STRING_ARRAY('BAND')
```

SDO_GEOR.getLayerID

Format

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

Description

Returns the user-defined identifier value associated with a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to return the user-defined identifier value. A value of 0 (zero) indicates the object layer.

Usage Notes

To set a user-defined identifier value for a layer in a GeoRaster object, use the SDO_GEOR.setLayerID procedure.

Examples

The following example returns the user-defined identifier values of layers 0, 1, 2, and 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
SELECT substr(sdo_geor.getLayerID(tmimage, 0),1,12) layerID0,
       substr(sdo_geor.getLayerID(tmimage, 1),1,12) layerID1,
       substr(sdo_geor.getLayerID(tmimage, 2),1,12) layerID2,
       substr(sdo_geor.getLayerID(tmimage, 3),1,12) layerID3
  FROM landsat WHERE id=4;

LAYERID0     LAYERID1     LAYERID2     LAYERID3
```

```
------------ ------------ ------------ ------------
TM543        TM3          TM4          TM5
```

SDO_GEOR.getLayerOrdinate

Format

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

Description

Returns the band ordinate for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

The returned number refers to the physical band that a layer (`layerNumber` parameter value) is associated with. For the current release, by default the associations are as shown in Figure 1–4 in Section 1.5: layer 1 is band 0, layer 2 is band 1, and so on.

To set the band ordinate value for a layer, use the SDO_GEOR.setLayerOrdinate procedure.

Examples

The following example returns the band numbers associated with layers 0, 1, 2, and 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
SELECT sdo_geor.getLayerOrdinate(tmimage, 0) layerOrdinate0,
       sdo_geor.getLayerOrdinate(tmimage, 1) layerOrdinate1,
       sdo_geor.getLayerOrdinate(tmimage, 2) layerOrdinate2,
```

```
       sdo_geor.getLayerOrdinate(tmimage, 3) layerOrdinate3
  FROM landsat WHERE id=4;

LAYERORDINATE0 LAYERORDINATE1 LAYERORDINATE2 LAYERORDINATE3
-------------- -------------- -------------- --------------
                             0              1              2
```

SDO_GEOR.getModelCoordinate

Format

```
SDO_GEOR.getModelCoordinate(
   georaster        IN SDO_GEORASTER,
   pyramidLevel     IN NUMBER,
   cellCoordinate   IN SDO_NUMBER_ARRAY
   ) RETURN SDO_GEOMETRY;
```

Description

Returns a point geometry object that contains the coordinates in the model (ground) coordinate system associated with the point at the specified cell (raster) coordinates.

Parameters

georaster
GeoRaster object.

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

cellCoordinate
Array of two coordinates identifying the point in the cell coordinate system. The two coordinates are the row number and column number of the point.

Usage Notes

Use this function to transform the location of a point on the GeoRaster object to the longitude and latitude coordinates of its associated point in the ground coordinate system.

The input GeoRaster data must be georeferenced. The resulting geometry has the same SDO_SRID value as the input GeoRaster object.

Contrast this function with the SDO_GEOR.getCellCoordinate function, which returns the coordinates in the cell (raster) coordinate system associated with the point at the specified model (ground) coordinates.

Examples

The following example returns a point geometry object containing the model coordinates associated with cell coordinates (100,100) in a specified GeoRaster object.

```
SET NUMWIDTH 20
SELECT sdo_geor.getModelCoordinate(tmimage, 0,
sdo_number_array(100,100)) mcoord
  FROM landsat WHERE id=4;

MCOORD(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------

SDO_GEOMETRY(2001, 82394, SDO_POINT_TYPE(347.666315789474, 43274.9052631579, NUL
L), NULL, NULL)
```

SDO_GEOR.getModelSRID

Format

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

Description

Returns the coordinate system (SDO_SRID value) associated with the model (ground) space for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns a null value if no coordinate system is associated with the model space.

To set the coordinate system (SDO_SRID value) associated with the model space, use the SDO_GEOR.setModelSRID procedure.

Examples

The following example returns the SDO_SRID values associated with the GeoRaster objects (TMIMAGE column) in the LANDSAT table.

```
SELECT id, sdo_geor.getModelSRID(tmimage) SRID FROM landsat;

        ID       SRID
---------- ----------
         2       82394
         4       8304
```

SDO_GEOR.getNODATA

Format

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

Description

Returns the value representing NODATA cells in a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

Some cells of a GeoRaster object may have no meaningful value assigned or collected. The NODATA value is the cell value for those cells, and it means that those cells are not semantically defined. The application is responsible for defining the meaning or significance of cells identified as NODATA cells.

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

Examples

The following example returns the value to be used for NODATA cells in the GeoRaster objects (GROBJ column) in table T0.

```
SELECT id, sdo_geor.getNODATA(grobj) NODATA from t0;

        ID     NODATA
---------- ----------
         1
         2    -9999.99
```

SDO_GEOR.getPyramidMaxLevel

Format

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

Description

Returns the level number of the top pyramid of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For information about pyramids, see Section 1.7.

Examples

The following example returns the pyramid type and level number of the top pyramid for the GeoRaster object (TMIMAGE column) in the row with an ID column value of 21 in the LANDSAT table.

```
SELECT substr(sdo_geor.getPyramidType(tmimage),1,10) pyramidType,
       sdo_geor.getPyramidMaxLevel(tmimage) maxLevel
  FROM landsat WHERE id=21;

PYRAMIDTYP   MAXLEVEL
---------- ----------
DECREASE            3
```

SDO_GEOR.getPyramidType

Format

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

Description

Returns the pyramid type for a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The pyramid type can be `NONE` (no pyramids) or `DECREASE`.

For information about pyramids, see Section 1.7.

Examples

The following example returns the pyramid type and level number of the top pyramid for the GeoRaster object (TMIMAGE column) in the row with an ID column value of 21 in the LANDSAT table.

```
SELECT substr(sdo_geor.getPyramidType(tmimage),1,10) pyramidType,
       sdo_geor.getPyramidMaxLevel(tmimage) maxLevel
  FROM landsat WHERE id=21;

PYRAMIDTYP   MAXLEVEL
---------- ----------
DECREASE            3
```

SDO_GEOR.getRasterBlocks

Format

```
SDO_GEOR.getRasterBlocks(
   georaster      IN SDO_GEORASTER,
   pyramidLevel   IN NUMBER,
   window         IN SDO_NUMBER_ARRAY
   ) RETURN SDO_RASTERSET;
```

or

```
SDO_GEOR.getRasterBlocks(
   georaster      IN SDO_GEORASTER,
   pyramidLevel   IN NUMBER,
   window         IN SDO_GEOMETRY
   ) RETURN SDO_RASTERSET;
```

Description

Returns an object of the SDO_RASTERSET collection type that identifies all blocks of a specified pyramid level that have any spatial interaction with a specified window.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level from which to return the blocks that have any spatial interaction with the specified window.

window
Window from which to return the blocks that are in `pyramidLevel`. The data type can be SDO_NUMBER_ARRAY or SDO_GEOMETRY. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and

lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_GEOMETRY, see the Usage Notes for SDO_SRID requirements.

Usage Notes

The SDO_RASTERSET collection type is described in Section 2.3.4.

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

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

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

Examples

The following example returns a collection set that identifies all raster blocks that have any spatial interaction with the specified window.

```
DECLARE
  gr sdo_georaster;
  ds sdo_rasterset;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=2;
  ds := sdo_geor.getRasterBlocks(gr, 0, sdo_number_array(11,65,192,244));
  COMMIT;
END;
/
```

SDO_GEOR.getRasterData

Format

```
SDO_GEOR.getRasterData(
   georaster      IN SDO_GEORASTER,
   pyramidLevel   IN NUMBER,
   rasterBlob     IN OUT BLOB);
```

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.

Usage Notes

If the GeoRaster object is blocked, the mosaic of all blocks of the specified pyramid level is returned.

Examples

The following example creates a BLOB object with all raster data from the GeoRaster object whose ID value is 2 in the LANDSAT table. It assumes that a BLOB with an ID value of 2 is already initialized in table T0.

```
DECLARE
  gr sdo_georaster;
  lb blob;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=2;
```

```
  SELECT data INTO lb FROM t0 WHERE id=2 FOR UPDATE;
  sdo_geor.getRasterData(gr, 0, lb);
  COMMIT;
END;
/
```

SDO_GEOR.getRasterSubset

Format

```
SDO_GEOR.getRasterSubset(
    georaster       IN SDO_GEORASTER,
    pyramidLevel    IN NUMBER,
    window          IN SDO_GEOMETRY,
    layerNumbers    IN VARCHAR2,
    rasterBlob      IN OUT BLOB);
```

or

```
SDO_GEOR.getRasterSubset(
    georaster       IN SDO_GEORASTER,
    pyramidLevel    IN NUMBER,
    window          IN SDO_NUMBER_ARRAY,
    bandNumbers     IN VARCHAR2,
    rasterBlob      IN OUT BLOB);
```

Description

Creates a single BLOB object containing all cells of a specified pyramid level that are inside or on the boundary of a specified window.

Parameters

georaster
GeoRaster object.

pyramidLevel
Pyramid level on which to perform the operation.

window
A rectangular window from which to crop the cells. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and

lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_GEOMETRY, the MBR of the geometry object is used as the window; see also the Usage Notes for SDO_SRID requirements.

If `window` is of type SDO_GEOMETRY, use the `layerNumbers` parameter to specify one or more layer numbers; if `window` 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). 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.

Usage Notes

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

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

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

Examples

The following example creates a BLOB object with all raster data from a specified window in pyramid level 0 of the GeoRaster object whose ID value is 4 in the LANDSAT table.

```
DECLARE
```

```
  gr sdo_georaster;
  lb blob;
BEGIN
  SELECT tmimage INTO gr FROM landsat WHERE id=4;
  SELECT data INTO lb from t0 WHERE id=4 FOR UPDATE;
  sdo_geor.getRasterSubset(gr, 0, sdo_number_array(-21,100,100,200), , lb);
END;
/
```

SDO_GEOR.getScaling

Format

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

Description

Returns the coefficients of the scaling function for a layer of a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

The scaling function is as follows:

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

The order of the coefficients is: a_0, a_1, b_0, b_1.

Examples

The following example returns the scaling coefficients for layer number 0 (the whole object) of a specified GeoRaster object in a table named LANDSAT. It scales original value range 0.0 to 1000.0 to be in the range 0.0 to 250.0.

```
SELECT sdo_geor.getScaling(tmimage, 0) FROM landsat WHERE id=0;

SDO_GEOR.GETSCALING(TMIMAGE,0)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(0.0, 0.25, 1, 0.0)
```

SDO_GEOR.getSpatialDimNumber

Format

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

Description

Returns the number of spatial dimensions of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For the current release, this function always returns 2.

To return the number of cells in each spatial dimension of a GeoRaster object, use the SDO_GEOR.getSpatialDimSizes function.

Examples

The following example returns the ID column value, the number of spatial dimensions, and the number of cells in each spatial dimension for the GeoRaster objects in the table named LANDSAT.

```
SELECT id, sdo_geor.getSpatialDimNumber(tmimage) spatialDim,
           sdo_geor.getSpatialDimSizes(tmimage) spatialDimSizes
   FROM landsat;

        ID SPATIALDIM
---------- ----------
SPATIALDIMSIZES
--------------------------------------------------------------------------------
         0          2
SDO_NUMBER_ARRAY(1024, 1024)

         1          2
```

```
SDO_NUMBER_ARRAY(384, 251)

         2          2
SDO_NUMBER_ARRAY(512, 512)

         4          2
SDO_NUMBER_ARRAY(512, 512)

        11          2
SDO_NUMBER_ARRAY(7957, 5828)
```

SDO_GEOR.getSpatialDimSizes

Format

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

Description

Returns the number of cells in each spatial dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

To return the number of spatial dimensions for a GeoRaster object, use the SDO_GEOR.getSpatialDimNumber function.

Examples

The following example returns the spatial dimension sizes and the number of bands for a GeoRaster object. (The output is reformatted for readability.)

```
SELECT sdo_geor.getSpatialDimSizes(tmimage) spatialDimSizes,
       sdo_geor.getBandDimSize(tmimage) bandDimSize
  FROM landsat WHERE id=21;

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

SDO_GEOR.getSpatialResolutions

Format

```
SDO_GEOR.getSpatialResolutions
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_NUMBER_ARRAY;
```

Description

Returns the spatial resolution value along each spatial dimension of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

Each value indicates the number of units of measurement associated with the data area represented by that spatial dimension of a pixel. For example, if the spatial resolution values are (10,10) and the unit of measurement for the ground data is meters, each pixel represents an area of 10 meters by 10 meters.

The spatial resolutions may be inconsistent with the georeferencing information, especially when the GeoRaster object is not georectified. You can use the SDO_GEOR.setSpatialResolutions procedure to set the spatial resolutions to be the average resolutions for an image or the resolutions when the data was collected. In this case, georeferencing information should be used for precise measurement.

Examples

The following example returns the spatial resolution values along the row and column dimensions of a GeoRaster object.

```
SELECT sdo_geor.getSpatialResolutions(tmimage) spatialResolution
  FROM landsat WHERE id=42;

SPATIALRESOLUTION
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(28.5, 28.5)
```

SDO_GEOR.getSpectralResolution

Format

```
SDO_GEOR.getSpectralResolution
   georaster  IN SDO_GEORASTER
   ) RETURN NUMBER;
```

Description

Returns the spectral resolution of a GeoRaster object if it is a hyperspectral or multiband image.

Parameters

georaster
GeoRaster object.

Usage Notes

Taken together, the spectral unit and spectral resolution identify the wavelength interval for a band. For example, if the spectral resolution value is 2 and the spectral unit value is `MILLIMETER`, the wavelength interval for a band is 2 millimeters.

To set the spectral resolution for a GeoRaster object, use the SDO_GEOR.setSpectralResolution procedure.

Examples

The following example returns the spectral unit and spectral resolution for all spatially referenced GeoRaster objects (TMIMAGE column) in the LANDSAT table.

```
SELECT id, substr(sdo_geor.getSpectralUnit(tmimage),1,20) spectralUnit,
           sdo_geor.getSpectralResolution(tmimage) spectralResolution
  FROM landsat
 where sdo_geor.isSpatialReferenced(tmimage)='TRUE';

        ID SPECTRALUNIT         SPECTRALRESOLUTION
---------- -------------------- ------------------
         4 MILLIMETER                        0.075
```

SDO_GEOR.getSpectralUnit

Format

```
SDO_GEOR.getSpectralUnit
   georaster  IN SDO_GEORASTER
   ) RETURN VARCHAR2;
```

Description

Returns the unit of measurement for identifying the wavelength interval for a band.

Parameters

georaster
GeoRaster object.

Usage Notes

This function can return one of the following values: `METER`, `MILLIMETER`, `MICROMETER`, `NANOMETER`.

Taken together, the spectral unit and spectral resolution identify the wavelength interval for a band. For example, if the spectral resolution value is 2 and the spectral unit value is `MILLIMETER`, the wavelength interval for a band is 2 millimeters.

To set the spectral unit for a GeoRaster object, use the SDO_GEOR.setSpectralUnit procedure.

Examples

The following example returns the spectral unit and spectral resolution for all spatially referenced GeoRaster objects (TMIMAGE column) in the LANDSAT table.

```
SELECT id, substr(sdo_geor.getSpectralUnit(tmimage),1,20) spectralUnit,
           sdo_geor.getSpectralResolution(tmimage) spectralResolution
  FROM landsat
 where sdo_geor.isSpatialReferenced(tmimage)='TRUE';

        ID SPECTRALUNIT         SPECTRALRESOLUTION
---------- -------------------- ------------------
         4 MILLIMETER                        0.075
```

SDO_GEOR.getSRS

Format

```
SDO_GEOR.getSRS(
   georaster  IN SDO_GEORASTER
   ) RETURN SDO_GEOR_SRS;
```

Description

Returns an object of type SDO_GEOR_SRS containing information related to the spatial referencing of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The SDO_GEOR_SRS object type is described in Section 2.3.5.

Examples

The following example returns information related to the spatial referencing of all spatially referenced GeoRaster objects (TMIMAGE column) in the LANDSAT table.

```
SELECT id, sdo_geor.getSRS(tmimage) SRS
  FROM landsat
  WHERE sdo_geor.isSpatialReferenced(tmimage)='TRUE';

        ID
----------
SRS(ISREFERENCED, ISRECTIFIED, ISORTHORECTIFIED, SRID, SPATIALRESOLUTION, SPATIA
--------------------------------------------------------------------------------
         4
SDO_GEOR_SRS('TRUE', 'TRUE', NULL, 82262, SDO_NUMBER_ARRAY(28.5, 28.5), NULL, NU
LL, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, NULL, NULL, NULL, SDO_NUMBER_ARRAY(1, 2, 1, 3,
 32631.5614, 0, -.03508772), SDO_NUMBER_ARRAY(1, 0, 0, 1, 1), SDO_NUMBER_ARRAY(1
, 2, 1, 3, -7894.7544, .035087719, 0), SDO_NUMBER_ARRAY(1, 0, 0, 1, 1))
```

SDO_GEOR.getStatistics

Format

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

Description

Returns statistical data associated with a layer.

Parameters

georaster
GeoRaster object.

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

Usage Notes

This function returns statistical data described by the `<statisticDatasetType>` element in the GeoRaster metadata XML schema, which is described in Appendix A. The function returns an array with the following values: `MIN`, `MAX`, `MEAN`, `MEDIAN`, `MODEVALUE`, and `STD`.

To set the statistical data associated with a layer, use the SDO_GEOR.setStatistics procedure.

Examples

The following example returns statistical data for layer 1 of a GeoRaster object.

```
SELECT sdo_geor.getStatistics(tmimage, 1) layer1
FROM landsat WHERE id=4;

LAYER1
--------------------------------------------------------------------------------
```

```
SDO_NUMBER_ARRAY(0, 255, 100, 127, 95, 25)
```

SDO_GEOR.getTotalLayerNumber

Format

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

Description

Returns the total number of layers in a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

For information about layers, see Section 1.5.

Examples

The following example returns the total number of layers in each GeoRaster object (TMIMAGE column) in the LANDSAT table.

```
SELECT id, sdo_geor.getTotalLayerNumber(tmimage) totalLayerNumber
  FROM landsat;

        ID TOTALLAYERNUMBER
---------- ----------------
         2                1
         4                3
```

SDO_GEOR.getULTCoordinate

Format

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

Description

Returns the cell coordinates of the upper-left corner of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns two or three numbers. If it returns two numbers, they are row and column ordinates. If it returns three numbers, they are row, column, and band ordinates.

Examples

The following example returns the row, column, and band ordinates for the upper-left corner of a GeoRaster object.

```
SELECT sdo_geor.getULTCoordinate(tmimage) FROM landsat WHERE id=23;

SDO_GEOR.GETULTCOORDINATE(TMIMAGE)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(256, 0, 0)
```

SDO_GEOR.getVAT

Format

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

Description

Returns the name of the value attribute table (VAT) associated with a layer of a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

Usage Notes

For more information about value attribute tables, see Section 1.2.3.

To set the name of the value attribute table to be associated with a layer of a GeoRaster object, use the SDO_GEOR.setVAT procedure.

Examples

The following example returns the value attribute tables for layers 0, 1, 2, and 3 of the GeoRaster objects (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table. (The output is reformatted for readability.)

```
SELECT substr(sdo_geor.getVAT(tmimage, 0),1,20) vatTable0,
       substr(sdo_geor.getVAT(tmimage, 1),1,20) vatTable1,
       substr(sdo_geor.getVAT(tmimage, 2),1,20) vatTable2,
       substr(sdo_geor.getVAT(tmimage, 3),1,20) vatTable3
  FROM landsat WHERE id=4;
```

```
VATTABLE0            VATTABLE1            VATTABLE2            VATTABLE3
-------------------- -------------------- -------------------- ----------------
VAT0                 VAT1                 VAT2                 VAT1
```

SDO_GEOR.getVersion

Format

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

Description

Returns the user-specified version of a GeoRaster object.

Parameters

georaster
GeoRaster object.

Usage Notes

The version returned is in the format *major-version.minor-version*.

To set the user-specified version of a GeoRaster object, use the SDO_GEOR.setVersion procedure.

Examples

The following example returns the user-specified version of the GeoRaster objects (TMIMAGE column) in the LANDSAT table. (The output is reformatted for readability.)

```
SELECT id, sdo_geor.getVersion(tmimage) version FROM landsat;

        ID VERSION
---------- ----------------------------------------------------------------
         2  10.1
         4  9i.2
```

SDO_GEOR.hasGrayScale

Format

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

Description

Checks if a layer of a GeoRaster object has grayscale information.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer to check. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns `TRUE` if the layer has grayscale information, or `FALSE` if the layer does not use grayscale representation. Section 2.3.3 describes grayscale display.

If the layer has grayscale information, you can get and set the grayscale mappings and the grayscale mapping table name. See the following: SDO_GEOR.getGrayScale and SDO_GEOR.getGrayScaleTable functions, and SDO_GEOR.setGrayScale and SDO_GEOR.setGrayScaleTable procedures.

Examples

The following example checks if layers 0 and 1 of a specified GeoRaster object (TMIMAGE column) have grayscale information.

```
SELECT substr(sdo_geor.hasGrayScale(tmimage, 0),1,15) hasGrayScale0,
       substr(sdo_geor.hasGrayScale(tmimage, 1),1,15) hasGrayScale1
  FROM landsat WHERE id=4;
```

```
HASGRAYSCALE0   HASGRAYSCALE1
--------------- ---------------
TRUE            FALSE
```

SDO_GEOR.hasPseudoColor

Format

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

Description

Checks if a layer of a GeoRaster object has pseudocolor information.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer to check. A value of 0 (zero) indicates the object layer.

Usage Notes

This function returns `TRUE` if the layer has pseudocolor information, or `FALSE` if the layer does not use pseudocolor representation. Section 2.3.2 describes colormaps and pseudocolor display.

If the layer has pseudocolor information, you can get and set the colormap and colormap table name. See the following: SDO_GEOR.getColorMap and SDO_GEOR.getColorMapTable functions, and SDO_GEOR.setColorMap and SDO_GEOR.setColorMapTable procedures.

Examples

The following example checks if layers 0 and 1 of a specified GeoRaster object (TMIMAGE column) have pseudocolor information.

```
SELECT substr(sdo_geor.hasPseudoColor(tmimage, 0),1,15) hasPseudoColor0,
       substr(sdo_geor.hasPseudoColor(tmimage, 1),1,15) hasPseudoColor1
  FROM landsat WHERE id=4;
```

```
HASPSEUDOCOLOR0 HASPSEUDOCOLOR1
--------------- ---------------
FALSE           TRUE
```

SDO_GEOR.importFrom

Format

```
SDO_GEOR.importFrom(
    georaster          IN OUT SDO_GEORASTER,
    storageParam       IN VARCHAR2,
    r_sourceFormat     IN VARCHAR2,
    r_sourceType       IN VARCHAR2,
    r_sourceName       IN VARCHAR2,
    h_sourceFormat     IN VARCHAR2 DEFAULT NULL,
    h_sourceType       IN VARCHAR2 DEFAULT NULL,
    h_sourceName       IN VARCHAR2 DEFAULT NULL);
```

or

```
SDO_GEOR.importFrom(
    georaster          IN OUT SDO_GEORASTER,
    storageParam       IN VARCHAR2,
    r_sourceFormat     IN VARCHAR2,
    r_sourceBLOB       IN BLOB,
    h_sourceFormat     IN VARCHAR2 DEFAULT NULL,
    h_sourceCLOB       IN CLOB DEFAULT NULL);
```

Description

Imports an image file or BLOB object into a GeoRaster object stored in the database.

Parameters

georaster
GeoRaster object to hold the result of the operation.

storageParam
String containing storage parameters. The format and usage are as explained in Section 1.4.1. Currently, the keywords supported for this operation are:

- `blocksize`: (See the explanation in Table 1–1 in Section 1.4.1.)
- `blocking`: `FALSE` causes the image to be stored as a single block. If the blocksize parameter is not specified, `TRUE` causes the image to be reblocked using the default reblocking parameter values: (256,256,*B*), where *B* is the total number of bands that the image contains. If the `blocksize` parameter is specified, blocking is automatically interpreted as `TRUE`.

r_sourceFormat
Raster source format. Must be one of the following: `TIFF`, `GIF`, `BMP`, or `PNG`. (`JPEG` is not supported for this procedure; however, you can use the client-side GeoRaster loader tool, described in Section 1.9, 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`.

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`.

h_sourceCLOB
Geoheader source as an object of type CLOB.

Usage Notes

Specify values for the parameters with names that start with *r_* and *h_* only if the raster image and the geoheader are in separate files or objects.

This procedure can load an ESRI world file from a file or from a CLOB object.

This procedure does not support JPEG as a source file format. You can use the client-side GeoRaster loader tool, described in Section 1.9, to import a JPEG file.

This procedure does not support raster data that has a cell depth value of `2BIT` or source multiband raster data with BIL and BSQ interleaving types.

The imported GeoRaster object has the BIP interleaving type.

The import operation cannot be rolled back.

Before you call this procedure, you must have read permission on the files to be imported or the directory that contains the files. The following example (run as user `SYSTEM`) grants read permission on a file to user `HERMAN`:

```
call dbms_java.grant_permission('HERMAN','SYS:java.io.FilePermission',
  'sdo/demos/georaster/data/img1.tif', 'read' );
```

Examples

The following example initializes an empty GeoRaster object into which an external image in TIFF format is to be imported, and then imports the image.

```
DECLARE
   geor SDO_GEORASTER;
BEGIN
-- Initialize an empty GeoRaster object into which the external image
-- is to be imported.
INSERT INTO georaster_table
   values( 1, 'TIFF', sdo_geor.init('rdt_1') );

-- Import the TIFF image.
SELECT georaster INTO geor from georaster_table
   where georid = 1 FOR UPDATE;
sdo_geor.importFrom(geor, NULL, 'TIFF', 'file',
   'sdo/demos/georaster/data/img1.tif');
UPDATE georaster_table SET georaster = geor where georid = 1;
COMMIT;
END;/
```

The following example imports images from a BLOB and an ESRI world file from a CLOB.

```
CREATE TABLE blob_table (blob_col BLOB, blobid NUMBER unique, clob_col CLOB);
INSERT INTO blob_table VALUES (empty_blob(), 1, null);
INSERT INTO blob_table VALUES (empty_blob(), 2, empty_clob());
```

```
COMMIT;

DECLARE
  geor1 MDSYS.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;
mdsys.sdo_geor.importFrom(geor1,'', 'TIFF', lobd1, 'WORLDFILE', lobd2);
sdo_geor.setModelSRID(geor1, 82394);
UPDATE georaster_table SET georaster = geor1 WHERE georid = 14;
COMMIT;
END;
/
```

SDO_GEOR.init

Format

```
SDO_GEOR.init(
   rasterDataTable  IN VARCHAR2 DEFAULT NULL,
   rasterID         IN NUMBER DEFAULT NULL
   ) RETURN SDO_GEORASTER;
```

Description

Initializes an empty GeoRaster object, which will be registered by GeoRaster in the xxx_SDO_GEOR_SYSDATA views (described in Section 2.4).

Parameters

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

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

Usage Notes

This function returns an empty SDO_GEORASTER object with its `rasterDataTable` and `rasterID` attributes set. All other attributes of the SDO_GEORASTER object are null.

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

Examples

The following example inserts an initialized GeoRaster object into the GR_TABLE table. The raster data table associated with the GeoRaster object is RDT_1.

```
INSERT INTO gr_table (id, georaster) VALUES (1, sdo_geor.init('RDT_1'));
```

SDO_GEOR.isBlank

Format

```
SDO_GEOR.isBlank(
   georaster  IN SDO_GEORASTER
   ) RETURN BOOLEAN;
```

Description

Returns `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 TMIMAGE column of the LANDSAT table is a blank GeoRaster object.

```
SELECT id, substr(sdo_geor.isBlank(tmimage),1,7) isBlank FROM landsat;

        ID ISBLANK
---------- -------
         2 FALSE
         4 FALSE
```

SDO_GEOR.isOrthoRectified

Format

```
SDO_GEOR.isOrthoRectified
   georaster  IN SDO_GEORASTER
   ) RETURN BOOLEAN;
```

Description

Returns `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 (TMIMAGE column) in the LANDSAT table are specified as spatially referenced, rectified, and orthorectified.

```
SELECT id, substr(sdo_geor.isSpatialReferenced(tmimage),1,20)
             isSpatialReferenced,
           substr(sdo_geor.isRectified(tmimage),1,20) isRectified,
           substr(sdo_geor.isOrthoRectified(tmimage),1,20) isOrthoRectified
  FROM landsat;

        ID ISSPATIALREFERENCED  ISRECTIFIED          ISORTHORECTIFIED
---------- -------------------- -------------------- --------------------
         2 TRUE                 TRUE                 TRUE
```

```
         4 TRUE                TRUE                FALSE
```

SDO_GEOR.isRectified

Format

```
SDO_GEOR.isRectified(
   georaster  IN SDO_GEORASTER
   ) RETURN BOOLEAN;
```

Description

Returns `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 (TMIMAGE column) in the LANDSAT table are specified as spatially referenced, rectified, and orthorectified.

```
SELECT id, substr(sdo_geor.isSpatialReferenced(tmimage),1,20)
             isSpatialReferenced,
           substr(sdo_geor.isRectified(tmimage),1,20) isRectified,
           substr(sdo_geor.isOrthoRectified(tmimage),1,20) isOrthoRectified
  FROM landsat;

        ID ISSPATIALREFERENCED  ISRECTIFIED          ISORTHORECTIFIED
---------- -------------------- -------------------- --------------------
         2 TRUE                 TRUE                 TRUE
         4 TRUE                 TRUE                 FALSE
```

SDO_GEOR.isSpatialReferenced

Format

```
SDO_GEOR.isSpatialReferenced(
   georaster  IN SDO_GEORASTER
   ) RETURN BOOLEAN;
```

Description

Returns `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 (TMIMAGE column) in the LANDSAT table are specified as spatially referenced, rectified, and orthorectified.

```
SELECT id, substr(sdo_geor.isSpatialReferenced(tmimage),1,20)
             isSpatialReferenced,
           substr(sdo_geor.isRectified(tmimage),1,20) isRectified,
           substr(sdo_geor.isOrthoRectified(tmimage),1,20) isOrthoRectified
  FROM landsat;

        ID ISSPATIALREFERENCED  ISRECTIFIED          ISORTHORECTIFIED
---------- -------------------- -------------------- --------------------
         2 TRUE                 TRUE                 TRUE
         4 TRUE                 TRUE                 FALSE
```

SDO_GEOR.mosaic

Format

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

Description

Mosaics GeoRaster objects into one GeoRaster object.

Parameters

georasterTableName
Name of the table containing all source GeoRaster objects.

georasterColumnName
Column of type SDO_GEORASTER in `georasterTableName`.

georaster
GeoRaster object to hold the result of the mosaic operation.

storageParam
A string specifying storage parameters, as explained in Section 1.4.1. If this parameter is null, the resulting GeoRaster object has the same storage parameters (`blockSize`, `cellDepth`, and `interleaving`) as the upper-left corner source GeoRaster object in the model space (if applicable) or cell space.

Usage Notes

The source GeoRaster objects must be prepared tiles or blocks so that they can be mosaicked directly and seamlessly. The GeoRaster objects to be mosaicked must:

- Be rectified, and have the same SRID value and spatial resolutions
- Completely cover the mosaic area, and not overlap or have any gaps

- Have the same number of layers or bands, and be spatially aligned along row and column dimensions
- Have the same mapping between band number and layers

If applicable, the resulting GeoRaster object takes the spatial reference metadata information from the upper-left corner source GeoRaster object in the model space.

If all source GeoRaster objects are blank and have the same `blankCellValue` value, the resulting GeoRaster object is blank and has that `blankCellValue` value; otherwise, the resulting GeoRaster object is not blank.

Any pyramid data for the source GeoRaster objects is not considered, and the `pyramid` parameter is ignored if it is specified in the `storageParam` string.

The mosaic operation performs internal commit operations at regular intervals, and thus it cannot be rolled back. If the operation is interrupted, dangling raster blocks may exist in the raster data table. You can handle dangling raster blocks, as explained in Section 3.14.

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

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

Examples

The following example mosaics all GeoRaster objects in the GROBJ column of the table named GRTAB, and inserts the resulting mosaicked GeoRaster object into the LANDSAT table.

```
DECLARE
  gr sdo_georaster;
BEGIN
  gr := sdo_geor.init('tm_02');
  sdo_geor.mosaic('grtab', 'grobj', gr, 'blocksize=(512,512,1)');
  INSERT INTO landsat (id, tmimage) VALUES (12, gr);
  COMMIT;
END;
/
```

SDO_GEOR.scale

Format

```
SDO_GEOR.scale(
   inGeoraster       IN OUT SDO_GEORASTER,
   scaleParam        IN VARCHAR2,
   resampleParam  IN VARCHAR2,
   storageParam     IN VARCHAR2);
```

Description

Scales (enlarges or reduces) a GeoRaster object.

Parameters

inGeoraster
The SDO_GEORASTER object on which the scaling operation is to be performed.

scaleParam
A string specifying a scaling parameter keyword and its associated value. The keyword must be one of the following:

- `scaleFactor`, to reduce or enlarge as a multiple of the original size. This keyword must have a numeric value greater than 0 (zero) (for example, `'scaleFactor=0.75'`). A value of 1.0 will not change the current size; a value less than 1 will reduce the image; a value greater than 1 will enlarge the image. The number of cells along each dimension is the original number multiplied by `scaleFactor`. For example, if the `scaleFactor` value is 2 and the GeoRaster object has X and Y dimensions, the number of cells along each dimension is doubled.
- `maxDimSize`, to specify a size in terms of the maximum number of cells for each dimension. This keyword must have a numeric value for each dimension (for example, `'maxDimSize=(512,512)'`). The aspect ratio is not changed.

resampleParam
A string specifying a resampling parameter (for example, `'resampling=NN'`). For the current release, the only supported keyword is `resampling`, and its value must be one of the following:

- `NN`: value of the nearest neighbor cell in the original GeoRaster object
- `BILINEAR`: distance-weighted average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE4`: simple average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE16`: simple average of the 16 nearest cells in the original GeoRaster object
- `CUBIC`: cubic convolution of the 16 nearest cells in the original GeoRaster object

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

Usage Notes

Use this procedure to change a GeoRaster object to reflect the specified scaling. To create a new GeoRaster object reflecting a specified scaling (without changing the original object), use the SDO_GEOR.scaleCopy procedure.

This procedure does not scale along the band dimension.

Any pyramid data in the GeoRaster object is deleted as a result of the scaling operation.

After the operation, the row and column ULT coordinates are always set to 0 (zero), even if no scaling is performed (that is, even if `scaleFactor=1`).

If the metadata contains spatial reference information and if the GeoRaster object is georeferenced with a valid affine transformation, the spatial reference information is updated accordingly; otherwise, the spatial reference information is removed.

Examples

The following example reduces an image to three-fourths (0.75) size, specifies bilinear resampling, and specifies a block size of 64 for each dimension in the storage parameters.

```
DECLARE
  gr1 sdo_georaster;
BEGIN
```

```
  SELECT tmimage INTO gr1 FROM landsat WHERE id=21;
  sdo_geor.scale(gr1, 'scaleFactor=0.75', 'resampling=BILINEAR',
                   'blocksize=(64,64)');
  UPDATE landsat SET tmimage=gr1 WHERE id=21;
  COMMIT;
END;
/
```

SDO_GEOR.scaleCopy

Format

```
SDO_GEOR.scaleCopy(
   inGeoraster       IN SDO_GEORASTER,
   scaleParam        IN VARCHAR2,
   resampleParam  IN VARCHAR2,
   storageParam     IN VARCHAR2,
   outGeoraster      IN OUT SDO_GEORASTER);
```

Description

Scales (enlarges or reduces) a GeoRaster object and puts the result into a new object that reflects the scaling.

Parameters

inGeoraster
The SDO_GEORASTER object on which the scaling operation is to be performed to create the new object (`outGeoraster`).

scaleParam
A string specifying a scaling parameter keyword and its associated value. The keyword must be one of the following:

- `scaleFactor`, to reduce or enlarge as a multiple of the original size. This keyword must have a numeric value greater than 0 (zero) (for example, `'scaleFactor=0.75'`). A value of 1.0 will not change the current size; a value less than 1 will reduce the image; a value greater than 1 will enlarge the image. The number of cells along each dimension is the original number multiplied by `scaleFactor`. For example, if the `scaleFactor` value is 2 and the GeoRaster object has X and Y dimensions, the number of cells along each dimension is doubled.
- `maxDimSize`, to specify a size in terms of the maximum number of cells for each dimension. This keyword must have a numeric value for each dimension (for example, `'maxDimSize=(512,512)'`). The aspect ratio is not changed.

resampleParam
A string specifying a resampling parameter (for example, `'resampling=NN'`). For the current release, the only supported keyword is `resampling`, and its value must be one of the following:

- `NN`: value of the nearest neighbor cell in the original GeoRaster object
- `BILINEAR`: distance-weighted average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE4`: simple average of the 4 nearest cells in the original GeoRaster object
- `AVERAGE16`: simple average of the 16 nearest cells in the original GeoRaster object
- `CUBIC`: cubic convolution of the 16 nearest cells in the original GeoRaster object

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

outGeoraster
The new SDO_GEORASTER object that reflects the results of the scaling operation.

Usage Notes

Use this procedure to create a new GeoRaster object reflecting the specified scaling, without changing the original object. To change the original GeoRaster object to reflect a specified scaling, use the SDO_GEOR.scale procedure.

This procedure does not scale along the band dimension.

Any pyramid data in the input GeoRaster object is not included in the output GeoRaster object.

After the operation, the row and column ULT coordinates are always set to 0 (zero), even if no scaling is performed (that is, even if `scaleFactor=1`).

If the metadata contains spatial reference information and if the GeoRaster object is georeferenced with a valid affine transformation, the spatial reference and georeferencing information is updated accordingly; otherwise, the spatial reference information is removed.

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

- `inGeoraster` is invalid.
- `outGeoraster` has not been initialized.

- A raster data table for `outGeoraster` does not exist and `outGeoraster` is not a blank GeoRaster object.

Examples

The following example reduces an image to three-fourths (0.75) size, specifies `AVERAGE4` resampling, and specifies a block size of 32 for each dimension in the storage parameters.

```
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO landsat VALUES (21, '21', sdo_geor.init('T21'), null)
    returning tmimage INTO gr2;

  SELECT tmimage INTO gr1 FROM landsat WHERE id=2;

  sdo_geor.scaleCopy(gr1, 'scaleFactor=0.75', resampling=AVERAGE4',
                     'blocksize=(32,32)', gr2);
  UPDATE landsat SET tmimage=gr2 WHERE id=21;
  COMMIT;
END;
/
```

SDO_GEOR.schemaValidate

Format

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

Description

Validates a GeoRaster object's metadata against the GeoRaster XML schema.

Parameters

georaster
GeoRaster object.

Usage Notes

This function returns `TRUE` if the metadata is valid, `NULL` if the GeoRaster object or its metadata is null, or one or more Oracle error codes indicating why the metadata is not valid and the exact location of the errors.

Use this function with the SDO_GEOR.validateGeoraster function. If the SDO_GEOR.validateGeoraster function identifies a GeoRaster object as invalid with an error code of 13454, the object's metadata is not valid according to the GeoRaster XML schema. If this happens, call the SDO_GEOR.schemaValidate function to get specific information, including the location in the metadata, about the errors.

Examples

The following example validates a GeoRaster object's metadata.

```
SELECT t.georid,
       sdo_geor.schemavalidate(t.georaster)
  from georaster_table t
  where t.georid = 1;
```

SDO_GEOR.setBeginDateTime

Format

```
SDO_GEOR.setBeginDateTime(
   georaster   IN OUT SDO_GEORASTER,
   beginTime  TIMESTAMP WITH TIME ZONE);
```

Description

Sets the beginning date and time for raster data collection in the metadata for a GeoRaster object, or deletes the existing value if you specify a null `beginTime` parameter.

Parameters

georaster
GeoRaster object.

beginTime
Time specification.

Usage Notes

To see the current beginning date and time (if any) in the metadata for the GeoRaster object, use the SDO_GEOR.getBeginDateTime function.

An exception is raised if `beginTime` is later than the ending date and time specified in the metadata for the GeoRaster object (see the SDO_GEOR.setEndDateTime procedure).

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the beginning and ending dates and times for raster data collection in the metadata for a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=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 landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setBinTable

Format

```
SDO_GEOR.setBinTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   tableName      IN VARCHAR2);
```

Description

Sets the name of the bin table associated with a layer, or deletes the existing value if you specify a null `tableName` parameter.

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.

Examples

The following example sets BINT1 as the name of the bin table for layer number 3 of a specified GeoRaster object in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setBinTable(grobj, 3, 'BINT1');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setBlankCellValue

Format

```
SDO_GEOR.setBlankCellValue(
   georaster  IN OUT SDO_GEORASTER,
   value      IN NUMBER);
```

Description

Sets (modifies) the cell value to be used for all cells if a specified GeoRaster object is a blank GeoRaster object, or deletes the existing value if you specify a null `value` parameter.

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.

Examples

The following example specifies a value of 255 to be used for all cells in the GeoRaster object column (TMIMAGE) in the LANDSAT table for the row with an ID column value of 1.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=1 FOR UPDATE;
```

```
  sdo_geor.setBlankCellValue(grobj, 255);
  UPDATE landsat SET tmimage = grobj WHERE id=1;
  COMMIT;
END;
/
```

SDO_GEOR.setColorMap

Format

```
SDO_GEOR.setColorMap
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   colorMap       IN MYSYS.SDO_GEOR_COLORMAP);
```

Description

Sets the colormap for a layer in a GeoRaster object, or deletes the existing value if you specify a null `colorMap` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

colorMap
Colormap object of type SDO_GEOR_COLORMAP, which is described in Section 2.3.2.

Usage Notes

The following must be true of the specified colormap object:

- The `cellValue` values are consistent with and in the value range for the `cellDepth` value of the GeoRaster object.
- The red, green, blue, and alpha values are integers from 0 to 255.
- The `cellValue` array contains no duplicate entries.
- The entries in the `cellValue` array are in ascending order.

The GeoRaster object is automatically validated after the operation completes.

You can create a colormap or retrieve a colormap from an existing GeoRaster object for use. To return the colormap for a layer in a GeoRaster object, use the SDO_GEOR.getColorMap function.

Examples

The following example sets the colormap for layer 2 of the GeoRaster object (IMAGE column) in the row with the ID column value of 4 in the IMAGES table. It assumes that the GeoRaster object is a bitmap.

```
DECLARE
  grobj sdo_georaster;
  cmobj sdo_colormap;
BEGIN
  cmobj := sdo_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 image INTO grobj FROM images WHERE id=4 FOR UPDATE;
  sdo_geor.setColorMap(grobj, 2, cmobj);
  UPDATE images SET image = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setColorMapTable

Format

```
SDO_GEOR.setColorMapTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   tableName      IN VARCHAR2);
```

Description

Sets the colormap table for a layer in a GeoRaster object, or deletes the existing value if you specify a null `tableName` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

tableName
Name of the user-defined colormap table. Section 2.3.2 describes colormaps.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To return the colormap table for a layer in a GeoRaster object, use the SDO_GEOR.getColorMapTable function.

Examples

The following example sets the colormap table to be null for layer 2 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
```

```
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setColorMapTable(grobj, 2, null);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setDefaultBlue

Format

```
SDO_GEOR.setDefaultBlue(
   georaster    IN OUT SDO_GEORASTER,
   defaultBlue  NUMBER);
```

Description

Sets the number of the layer to be used for the blue color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultBlue` parameter.

Parameters

georaster
GeoRaster object.

defaultBlue
Number of the layer to be used for the blue color component (in the RGB color space) for displaying the specified GeoRaster object. Must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

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

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GROBJ column) in the LANDSAT table, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects.

```
DECLARE
  grobj sdo_georaster;
```

```
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 5);
  sdo_geor.setDefaultGreen(grobj, 4);
  sdo_geor.setDefaultBlue(grobj, 3);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(tmimage) FROM landsat WHERE id=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(TMIMAGE)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(5, 4, 3)

1 row selected.
```

SDO_GEOR.setDefaultColorLayer

Format

```
SDO_GEOR.setDefaultColorLayer(
   georaster     IN OUT SDO_GEORASTER,
   defaultRGB  SDO_NUMBER_ARRAY);
```

Description

Sets the default numbers of the layers to be used for the red, green, and blue color components, respectively, for displaying a GeoRaster object, or deletes the existing values if you specify a null `defaultRGB` parameter.

Parameters

georaster
GeoRaster object.

defaultRGB
Array of three numbers identifying the red, green, and blue color components, respectively, for displaying the specified GeoRaster object. Each number must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

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

The GeoRaster object is automatically validated after the operation completes.

You can set the layer number for each color component (RGB) by using the SDO_GEOR.setDefaultRed, SDO_GEOR.setDefaultGreen, and SDO_GEOR.setDefaultBlue procedures.

Examples

The following example specifies that layer number 1 is to be used for the red, green, and blue color components for displaying the GeoRaster object (TMIMAGE column) in the row with an ID column value of 2 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=2 FOR UPDATE;
  sdo_geor.setDefaultColorLayer(grobj, sdo_number_array(1,1,1));
  UPDATE landsat SET tmimage = grobj WHERE id=2;
  COMMIT;
END;
/
```

SDO_GEOR.setDefaultGreen

Format

```
SDO_GEOR.setDefaultGreen(
   georaster      IN OUT SDO_GEORASTER,
   defaultGreen  NUMBER);
```

Description

Sets the number of the layer to be used for the green color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultGreen` parameter.

Parameters

georaster
GeoRaster object.

defaultGreen
Number of the layer to be used for the green color component (in the RGB color space) for displaying the specified GeoRaster object. Must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

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

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GROBJ column) in the LANDSAT table, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects.

```
DECLARE
  grobj sdo_georaster;
```

```
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 5);
  sdo_geor.setDefaultGreen(grobj, 4);
  sdo_geor.setDefaultBlue(grobj, 3);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(tmimage) FROM landsat WHERE id=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(TMIMAGE)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(5, 4, 3)

1 row selected.
```

SDO_GEOR.setDefaultRed

Format

```
SDO_GEOR.setDefaultRed(
   georaster    IN OUT SDO_GEORASTER,
   defaultRed  IN NUMBER);
```

Description

Sets the number of the layer to be used for the red color component (in the RGB color space) for displaying a GeoRaster object, or deletes the existing value if you specify a null `defaultRed` parameter.

Parameters

georaster
GeoRaster object.

defaultRed
Number of the layer to be used for the red color component (in the RGB color space) for displaying the specified GeoRaster object. Must be greater than 0 (zero) and less than or equal to the highest layer number in the GeoRaster object.

Usage Notes

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

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the default red, green, and blue color layers for the GeoRaster objects (GROBJ column) in the LANDSAT table, and it returns an array with the layer numbers for the red, green, and blue color components for displaying these GeoRaster objects.

```
DECLARE
  grobj sdo_georaster;
```

```
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setDefaultRed(grobj, 5);
  sdo_geor.setDefaultGreen(grobj, 4);
  sdo_geor.setDefaultBlue(grobj, 3);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/

SELECT sdo_geor.getDefaultColorLayer(tmimage) FROM landsat WHERE id=4;

SDO_GEOR.GETDEFAULTCOLORLAYER(TMIMAGE)
--------------------------------------------------------------------------------
SDO_NUMBER_ARRAY(5, 4, 3)

1 row selected.
```

SDO_GEOR.setEndDateTime

Format

```
SDO_GEOR.setEndDateTime(
   georaster  IN OUT SDO_GEORASTER,
   endTime    TIMESTAMP WITH TIME ZONE);
```

Description

Sets the ending date and time for raster data collection in the metadata for a GeoRaster object, or deletes the existing value if you specify a null `endTime` parameter.

Parameters

georaster
GeoRaster object.

endTime
Time specification.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To see the current ending date and time (if any) in the metadata for the GeoRaster object, use the SDO_GEOR.getEndDateTime function.

An exception is raised if `endTime` is earlier than the beginning date and time specified in the metadata for the GeoRaster object (see the SDO_GEOR.setBeginDateTime procedure).

Examples

The following example sets the beginning and ending dates and times for raster data collection in the metadata for a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=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 landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setGrayScale

Format

```
SDO_GEOR.setGrayScale(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   grayScale      IN SDO_GEOR_GRAYSCALE);
```

Description

Sets the grayscale mappings for a layer in a GeoRaster object, or deletes the existing values if you specify a null `grayScale` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the grayscale mappings. A value of 0 (zero) indicates the object layer.

grayScale
An object of type SDO_GEOR_GRAYSCALE, which is described in Section 2.3.3.

Usage Notes

The following must be true of the specified colormap 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.

Examples

The following example sets the grayscale mappings for layer 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
  gsobj sdo_grayscale;
BEGIN
  gsobj := sdo_grayscale(sdo_number_array(1, 10, 20, 30, 255),
                         sdo_number_array(0, 180, 210, 230, 250));

  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setGrayScale(grobj, 3, gsobj);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setGrayScaleTable

Format

```
SDO_GEOR.setGrayScaleTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER,
   tableName      IN VARCHAR2);
```

Description

Sets the grayscale mapping table for a layer in a GeoRaster object, or deletes the existing value if you specify a null `tableName` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to set the grayscale mapping table. A value of 0 (zero) indicates the object layer.

tableName
Name of the grayscale mapping table for a layer in the specified GeoRaster object.

Usage Notes

Section 2.3.3 describes grayscale display.

The GeoRaster object is automatically validated after the operation completes.

To return the grayscale mapping table for a layer in a GeoRaster object, use the SDO_GEOR.getGrayScaleTable function.

Examples

The following example sets `GST1` as the grayscale mapping table for layer 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setGrayScaleTable(grobj, 3, 'GST1');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setHistogramTable

Format

```
SDO_GEOR.setHistogramTable(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber    IN NUMBER
   tableName      IN VARCHAR2);
```

Description

Sets the histogram table for a layer in a GeoRaster object.

Parameters

georaster
GeoRaster object.

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

tableName
Name of the histogram table. If this parameter is null, the metadata information for any existing histogram table (but not the actual table) is deleted. If there is no statistics information for the layer, this parameter must be null. The parameter value cannot be an empty string (that is, it cannot be `''`).

Usage Notes

This procedure specifies a user-defined histogram table. Section 2.3.1 briefly discusses histograms.

To return the name of the histogram table for a layer, use the SDO_GEOR.getHistogramTable function.

Examples

The following example sets HIST1 as the histogram table for layer 3 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setHistogramTable(grobj, 3, 'HIST1');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setID

Format

```
SDO_GEOR.setID(
   georaster  IN OUT SDO_GEORASTER,
   id         IN VARCHAR2);
```

Description

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

Parameters

georaster
GeoRaster object.

id
ID value to be associated with the GeoRaster object.

Usage Notes

This procedure is useful for assigning unique meaningful alphanumeric identifiers to GeoRaster objects, so that users and applications can easily identify the objects.

The GeoRaster object is automatically validated after the operation completes.

To return the user-defined identifier value for a GeoRaster object, use the SDO_GEOR.getID function.

Examples

The following example sets `newid` as the user-defined identifier value of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 2 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=2 FOR UPDATE;
  sdo_geor.setID(grobj, 'newid');
```

```
  UPDATE landsat SET tmimage = grobj WHERE id=2;
  COMMIT;
END;
/
```

SDO_GEOR.setLayerID

Format

```
SDO_GEOR.setLayerID(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   id                 IN VARCHAR2);
```

Description

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

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

id
ID value to be associated with the specified layer in the GeoRaster object.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To return the user-defined identifier value for a layer in a GeoRaster object, use the SDO_GEOR.getLayerID function.

Examples

The following example sets `TM_Band_2` as the user-defined identifier value of layer 2 in the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
```

```
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setLayerID(grobj, 2, 'TM_Band_2');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setLayerOrdinate

Format

```
SDO_GEOR.setLayerOrdinate(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   ordinate        IN NUMBER);
```

Description

Sets the band ordinate value for a specified layer in a GeoRaster object, or deletes the existing value if you specify a null `ordinate` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

ordinate
Band ordinate value of the layer along the band dimension.

Usage Notes

The band ordinate of the layer refers to the physical band that a layer (`layerNumber` parameter value) is associated with. For the current release, the associations must be as shown in Figure 1–4 in Section 1.5: layer 1 is band 0, layer 2 is band 1, and so on.

The band ordinate for the object layer is ignored by GeoRaster.

The GeoRaster object is automatically validated after the operation completes.

To return the band ordinate value for a layer, use the SDO_GEOR.getLayerOrdinate function.

Examples

The following example sets the band ordinate value for layer 1 to be 0 (zero) in the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setLayerOrdinate(grobj, 1, 0);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setModelSRID

Format

```
SDO_GEOR.setModelSRID(
   georaster  IN OUT SDO_GEORASTER,
   srid       IN NUMBER);
```

Description

Sets the coordinate system (SDO_SRID value) for the model (ground) space for a GeoRaster object, or deletes the existing value if you specify a null `srid` parameter.

Parameters

georaster
GeoRaster object.

srid
Coordinate system. Must be either a value from the SRID column of the MDSYS.CS_SRS table, or null to have no coordinate system associated with the model space.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

If the original GeoRaster object had a different model space SRID value, this procedure does not change the raster data itself. In other words, this procedure does not cause any reprojection or resampling on the cell data of the GeoRaster object.

To return the coordinate system (SDO_SRID value) associated with the model space for a GeoRaster object, use the SDO_GEOR.getModelSRID function.

Examples

The following example changes the coordinate system for a GeoRaster object to *Longitude / Latitude (WGS 66)*, which is the coordinate system associated with SRID value 8305 in the MDSYS.CS_SRS system table.

```
DECLARE
  grobj sdo_georaster;
```

```
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setModelSRID(grobj, 8305);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setOrthoRectified

Format

```
SDO_GEOR.setOrthoRectified(
   georaster   IN OUT SDO_GEORASTER,
   isOrthoRectified  IN VARCHAR2);
```

Description

Specifies whether or not a GeoRaster object is orthorectified, or deletes the existing value if you specify a null `isOrthoRectified` parameter.

Parameters

georaster
GeoRaster object.

isOrthoRectified
Specify `TRUE` to specify that the GeoRaster object is orthorectified, `FALSE` to specify that the GeoRaster object is not orthorectified, or null if it is not certain whether or not the GeoRaster object is orthorectified.

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 (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table as orthorectified.

```
DECLARE
  grobj sdo_georaster;
BEGIN
```

```
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setOrthoRectified(grobj, 'TRUE');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setRasterType

Format

```
SDO_GEOR.setRasterType(
   georaster    IN OUT SDO_GEORASTER,
   rasterType  IN NUMBER);
```

Description

Sets the raster type of a GeoRaster object.

Parameters

georaster
GeoRaster object.

rasterType
Numeric value to be set as the rasterType attribute of the GeoRaster object. Must be a valid 5-digit numeric value, in the format described in Section 2.1.1.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

Examples

The following example sets the `rasterType` attribute value of a GeoRaster object to `20001`.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=2 FOR UPDATE;
  sdo_geor.setRasterType(grobj, 20001);
  UPDATE landsat SET tmimage = grobj WHERE id=2;
  COMMIT;
END;
/
```

SDO_GEOR.setRectified

Format

```
SDO_GEOR.setRectified(
   georaster   IN OUT SDO_GEORASTER,
   isRectified  IN VARCHAR2);
```

Description

Specifies whether or not a GeoRaster object is rectified, or deletes the existing value if you specify a null `isRectified` parameter.

Parameters

georaster
GeoRaster object.

isRectified
Specify `TRUE` to specify that the GeoRaster object is rectified, `FALSE` to specify that the GeoRaster object is not rectified, or null if it is not certain whether or not the GeoRaster object is rectified.

Usage Notes

This procedure modifies the GeoRaster metadata for the object. It does not actually rectify the object. Users are responsible for ensuring that rectification is performed.

The GeoRaster object is automatically validated after the operation completes.

A GeoRaster object must be spatially referenced if you want to set `isRectified` to `TRUE` (see the SDO_GEOR.setSpatialReferenced procedure).

Examples

The following example identifies the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table as not rectified.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
```

```
  sdo_geor.setRectified(grobj, 'false');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setScaling

Format

```
SDO_GEOR.setScaling(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   scalingFunc    IN SDO_NUMBER_ARRAY);
```

Description

Sets the scaling function associated with a layer, or deletes the existing value if you specify a null `scalingFunc` parameter.

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.

Examples

The following example sets the coefficients of the scaling function for layer 2 of a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
```

```
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setScaling(grobj, 2, sdo_number_array(1, 0.5, 1, 0));
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSpatialReferenced

Format

```
SDO_GEOR.setSpatialReferenced(
   georaster      IN OUT SDO_GEORASTER,
   isReferenced   IN VARCHAR2);
```

Description

Specifies whether or not a GeoRaster object is spatially referenced, or deletes the existing value if you specify a null `isReferenced` parameter.

Parameters

georaster
GeoRaster object.

isReferenced
Specify `TRUE` to specify that the GeoRaster object is spatially referenced, or `FALSE` to specify that the GeoRaster object is not spatially referenced.

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 (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table as not spatially referenced.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setSpatialReferenced(grobj, 'FALSE');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
```

/

SDO_GEOR.setSpatialResolutions

Format

```
SDO_GEOR.setSpatialResolutions(
   georaster    IN OUT SDO_GEORASTER,
   resolutions  IN SDO_NUMBER_ARRAY);
```

Description

Sets the spatial resolution value along each spatial dimension of a GeoRaster object, or deletes the existing values if you specify a null `resolutions` parameter.

Parameters

georaster
GeoRaster object.

resolutions
An array of numeric values, one for each spatial dimension. Each value indicates the number of units of measurement associated with the data area represented by that spatial dimension of a pixel. For example, if the spatial resolution values are (10,10) and the unit of measurement for the ground data is meters, each pixel represents an area of 10 meters by 10 meters.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

See also the Usage Notes for the SDO_GEOR.getSpatialResolutions function.

Examples

The following example sets the spatial resolution values along the row and column dimensions of a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setSpatialResolutions(grobj, sdo_number_array(28.5,28.5));
  UPDATE landsat SET tmimage = grobj WHERE id=4;
```

```
  COMMIT;
END;
/
```

SDO_GEOR.setSpectralResolution

Format

```
SDO_GEOR.setSpectralResolution(
   georaster  IN OUT SDO_GEORASTER,
   resolution  IN NUMBER);
```

Description

Sets the spectral resolution of a GeoRaster object if it is a hyperspectral or multiband image, or deletes the existing value if you specify a null `resolution` parameter.

Parameters

georaster
GeoRaster object.

resolution
Spectral resolution value.

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 (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
```

```
  sdo_geor.setSpectralResolution(grobj, 0.5);
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSpectralUnit

Format

```
SDO_GEOR.setSpectralUnit(
   georaster  IN OUT SDO_GEORASTER,
   unit       IN VARCHAR2);
```

Description

Sets the unit of measurement for identifying the wavelength interval for a band, or deletes the existing value if you specify a null `unit` parameter.

Parameters

georaster
GeoRaster object.

unit
Spectral unit. Must be one of the following values: `METER`, `MILLIMETER`, `MICROMETER`, `NANOMETER`.

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 (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
```

```
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setSpectralUnit(grobj, 'micrometer');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setSRS

Format

```
SDO_GEOR.setSRS(
   georaster  IN OUT SDO_GEORASTER,
   srs        IN SDO_GEOR_SRS);
```

Description

Sets the spatial reference information of a GeoRaster object, or deletes the existing information if you specify a null `srs` parameter.

Parameters

georaster
GeoRaster object.

srs
An object of type SDO_GEOR_SRS, which is described in Section 2.3.5.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

To return the SDO_GEOR_SRS information for a GeoRaster object, use the SDO_GEOR.getSRS function.

Examples

The following example specifies changes to several spatial reference attributes of a GeoRaster object, calls the setSRS procedure to update the spatial reference information, and then updates the GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
  srs   sdo_geor_srs;

BEGIN

SELECT tmimage INTO grobj FROM landsat WHERE id=4;
srs := sdo_geor.getSRS(grobj);
```

```
srs.isReferenced := 'TRUE';
srs.isRectified := 'TRUE';
srs.isOrthoRectified := null;
srs.srid := 82262;
srs.spatialResolution := sdo_number_array(28.5, 28.5);
srs.rowOff := 0;
srs.columnOff := 0;
srs.xOff := 0;
srs.yOff := 0;
srs.zOff := 0;
srs.rowScale := 1;
srs.columnScale := 1;
srs.xScale := 1;
srs.yScale := 1;
srs.zScale := 1;
srs.rowNumerator := SDO_NUMBER_ARRAY(1, 2, 1, 3, 32631.5614, 0, -.03508772);
srs.rowDenominator := SDO_NUMBER_ARRAY(1, 0, 0, 1, 1);
srs.columnNumerator := SDO_NUMBER_ARRAY(1, 2, 1, 3, -7894.7544, .035087719, 0);
srs.columnDenominator := SDO_NUMBER_ARRAY(1, 0, 0, 1, 1);

sdo_geor.setSRS(grobj, srs);

UPDATE landsat SET tmimage = grobj WHERE id=4;
COMMIT;
END;
/
```

SDO_GEOR.setStatistics

Format

```
SDO_GEOR.setStatistics(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   statistics        IN SDO_NUMBER_ARRAY);
```

Description

Sets statistical data associated with a layer.

Parameters

georaster
GeoRaster object.

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

statistics
An array with the following numeric values: `MIN`, `MAX`, `MEAN`, `MEDIAN`, `MODEVALUE`, `STD`. You must specify non-null values for all values in the array.

If this parameter is null, all statistical information associated with the layer is deleted.

Usage Notes

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

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

Examples

The following example sets the statistical data for layer 0 of the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setStatistics(grobj, 0, SDO_NUMBER_ARRAY(0, 255, 100, 127, 95, 25));
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setULTCoordinate

Format

```
SDO_GEOR.setULTCoordinate(
   georaster  IN OUT SDO_GEORASTER,
   ultCoord    IN SDO_NUMBER_ARRAY);
```

Description

Sets the cell coordinate values of the upper-left corner of a GeoRaster object, or deletes the existing values if you specify a null `ultCoord` parameter.

Parameters

georaster
GeoRaster object.

ultCoord
An array of two numbers (row and column ordinates) or three numbers (row, column, and band ordinates).

Usage Notes

If the metadata contains spatial reference information and the GeoRaster object is georeferenced, the spatial reference information is checked for validity. If it is valid, the spatial reference information including the georeferencing information is updated and adjusted according to the new ULT coordinates; otherwise, an exception is raised.

To return the upper-left coordinate values for a GeoRaster object, use the SDO_GEOR.getULTCoordinate function.

Examples

The following example sets the row and column ordinates of the upper-left corner of a GeoRaster object.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=1 FOR UPDATE;
```

```
  sdo_geor.setULTCoordinate(grobj, sdo_number_array(100, 100));
  UPDATE landsat SET tmimage = grobj WHERE id=1;
  COMMIT;
END;
/
```

SDO_GEOR.setVAT

Format

```
SDO_GEOR.setVAT(
   georaster      IN OUT SDO_GEORASTER,
   layerNumber  IN NUMBER,
   vatName        IN VARCHAR2);
```

Description

Sets the name of the value attribute table (VAT) associated with a layer of a GeoRaster object, or deletes the existing value if you specify a null `vatName` parameter.

Parameters

georaster
GeoRaster object.

layerNumber
Number of the layer for which to perform the operation.

vatName
Name of the value attribute table.

Usage Notes

The GeoRaster object is automatically validated after the operation completes.

For more information about value attribute tables, see Section 1.2.3.

To return the name of the value attribute table associated with a layer of a GeoRaster object, use the SDO_GEOR.getVAT function.

Examples

The following example specifies `VATT1` as the value attribute table to be associated with layer 3 of the GeoRaster object (GROBJ column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setVAT(grobj, 3, 'VATT1');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.setVersion

Format

```
SDO_GEOR.setVersion(
   georaster      IN OUT SDO_GEORASTER,
   majorVersion   IN VARCHAR2,
   minorVersion   IN VARCHAR2);
```

Description

Sets the user-specified version of a GeoRaster object.

Parameters

georaster
GeoRaster object.

majorVersion
String representing the major version of the GeoRaster object. For example, if the complete version string is `15a.beta1`, specify the `majorVersion` value as `15a`.

If the parameter value is null, any existing `majorVersion` value in the GeoRaster object is deleted.

minorVersion
String representing the minor version of the GeoRaster object. For example, if the complete version string is `15a.beta1`, specify the `minorVersion` value as `beta1`.

If the parameter value is null, any existing `minorVersion` value in the GeoRaster object is deleted.

Usage Notes

The major and minor version strings can reflect any versioning scheme that you choose. The `majorVersion` and `minorVersion` values can be any string, except that neither can be an empty string (that is, neither can be `''`).

To retrieve the version string for a GeoRaster object, use the SDO_GEOR.getVersion function, which returns the version in the format *major-version.minor-version*.

Examples

The following example sets `15a.beta1` as the version for the GeoRaster object (TMIMAGE column) in the row with the ID column value of 4 in the LANDSAT table.

```
DECLARE
  grobj sdo_georaster;
BEGIN
  SELECT tmimage INTO grobj FROM landsat WHERE id=4 FOR UPDATE;
  sdo_geor.setVersion(grobj, '15a', 'beta1');
  UPDATE landsat SET tmimage = grobj WHERE id=4;
  COMMIT;
END;
/
```

SDO_GEOR.subset

Format

```
SDO_GEOR.subset(
    inGeoraster      IN SDO_GEORASTER,
    cropArea         IN SDO_GEOMETRY,
    layerNumbers     IN VARCHAR2,
    storageParam     IN VARCHAR2,
    outGeoraster     IN OUT SDO_GEORASTER);
```

or

```
SDO_GEOR.subset(
    inGeoraster      IN SDO_GEORASTER,
    cropArea         IN SDO_NUMBER_ARRAY,
    bandNumbers      IN VARCHAR2,
    storageParam     IN VARCHAR2,
    outGeoraster     IN OUT SDO_GEORASTER);
```

Description

Performs either or both of the following operations: (1) spatial crop, cut, or clip, or (2) layer or band subset or duplicate.

Parameters

inGeoraster
The SDO_GEORASTER object on which the operation or operations are to be performed.

cropArea
Crop area definition. If the data type is SDO_NUMBER_ARRAY, the parameter identifies the upper-left (row, column) and lower-right (row, column) coordinates of a rectangular window, and raster space is assumed. If the data type is SDO_

GEOMETRY, the minimum bounding rectangle (MBR) of the geometry object is used as the crop area; see also the Usage Notes for SDO_SRID requirements.

If `cropArea` is of type SDO_GEOMETRY, use the `layerNumbers` parameter to specify one or more layer numbers; if `cropArea` is of type SDO_NUMBER_ARRAY, use the `bandNumbers` parameter to specify one or more band numbers.

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

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

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

outGeoraster
The new SDO_GEORASTER object.

Usage Notes

This procedure has a variety of possible uses. For example, you can call it to crop a small area or obtain a subset of a few layers of a GeoRaster object, you can duplicate layers, and you can specify storage parameters such as blocking and interleaving for the resulting object.

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

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

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

Any pyramid data in the input GeoRaster object is not copied to the output GeoRaster object.

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

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

Examples

The following example creates a GeoRaster object that contains only specified bands from a specified window from the original object.

```
-- Subset: Band change
DECLARE
  gr1 sdo_georaster;
  gr2 sdo_georaster;
BEGIN
  INSERT INTO landsat VALUES (41, '41', sdo_geor.init('T41'), null)
    returning tmimage INTO gr2;

  SELECT tmimage INTO gr1 FROM landsat WHERE id=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 landsat SET tmimage=gr2 WHERE id=41;
  COMMIT;
END;
/
```

SDO_GEOR.validateGeoraster

Format

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

Description

Validates a GeoRaster object, checking its raster data and metadata.

Parameters

georaster
GeoRaster object to be checked for validity.

Usage Notes

This function returns `TRUE` if the GeoRaster object is valid, `NULL` if the GeoRaster object is null, an Oracle error code if the error is known, or `FALSE` for an unknown error.

You should use this function after you create, load, or modify a GeoRaster object, to ensure that it is valid before you process it further.

If this function identifies a GeoRaster object as invalid with an error code of 13454, this means that the object's metadata is not valid according to the GeoRaster XML schema. If this happens, call the SDO_GEOR.schemaValidate function to find specific locations and other information about the errors.

This function not only validates GeoRaster metadata against the GeoRaster XML schema, but it also enforces restrictions and requirements in the current release that are not described in the XML schema. The following are some of the restrictions and requirements enforced by this function:

- Layer numbers must be from 1 to *n* where *n* is the total number of layers.
- The `cellRepresentationType` value must be `UNDEFINED`.
- If `totalBandBlocks` or `bandBlockSize` is specified in the metadata, both must be specified. If there is only one band, no band blocking is allowed.

- The total number of blocks times the blocking size along a dimension must match the dimension size plus padding size, and the size of each cell data BLOB object must match the metadata description in terms of blocking or nonblocking.
- The size and number of GeoRaster data blocks stored in the raster data table must be consistent with the metadata description.
- The only pyramid types supported are `NONE` (no pyramids) and `DECREASE`. (For more information about pyramids, see Section 1.7.)
- Compression is not supported.
- Only one type of polynomial model is supported, as described in Section 1.6.1. The offsets, scales, and RMS values for the supported polynomial model are fixed. The pType, nVars, and number of coefficients are fixed for each polynomial, except for the values of the coefficients a, b, c, d, e, and f. The fixed values are described in Table 2–4 in Section 2.3.5. The SDO_GEOR_SRS data type is a precise map of the GeoRaster SRS polynomial model in the XML metadata document.
- The `RigorousModel` and `StoredFunction` georeferencing models are not supported, although you can store GCP (ground control point) data in an Oracle table and register the table name in the GeoRaster SRS metadata.
- Spatial resolutions can be inconsistent with the affine transformation scales if the GeoRaster object is georeferenced.
- GeoRaster temporal referencing and band referencing are not supported, although in the temporal reference system (TRS) and band reference system (BRS) you can store the beginning and ending date and time, the spectral resolution, the spectral unit, and related descriptive information.
- Only one `layerInfo` element is supported. A layer can be defined only along one dimension, and this dimension must be `BAND`. However, within the `layerInfo` element, the number of `subLayer` elements is limited only by the total number of layers. The layer number for the `objectLayer` elements is 0, and the layer numbers for `subLayer` elements are 1 to *n* where *n* is the total number of layers.
- The scaling function, BIN function, and statistical data or histogram can be stored in the GeoRaster metadata and must be valid against the XML schema, but the value ranges for these items are not restricted. GeoRaster interfaces that use this metadata are limited. Applications should validate this optional metadata before using it.

- The numbers of colormap values and grayscale mapping values are not restricted, but there must be no duplicate colormap or grayscale values, and the values in each array must be consistent with the `cellDepth` value of the GeoRaster object and must be in ascending order. The value range of the red, green, blue, alpha, and gray components must be integers from 0 to 255.

Examples

The following example validates the GeoRaster objects in a table.

```
SELECT t.georid,
       sdo_geor.validategeoraster(t.georaster) isvalid
  from georaster_table t order by georid;

    GEORID ISVALID
---------- ----------
         3 TRUE
         4 TRUE
```

5

SDO_GEOR_UTL Package Reference

The MDSYS.SDO_GEOR_UTL package contains subprograms (functions and procedures) for utility operations related to GeoRaster. (For the current release, the package contains only one trigger-related subprogram.) This chapter presents reference information, with one or more examples, for each subprogram.

SDO_GEOR_UTL.createDMLTrigger

Format

```
SDO_GEOR_UTL.createDMLTrigger(
   tableName     VARCHAR2,
   columnName  VARCHAR2);
```

Description

Creates the required standard GeoRaster data manipulation language (DML) trigger on a GeoRaster column in a GeoRaster table, so that the appropriate operations are performed when its associated trigger is fired.

Parameters

tableName
Name of a GeoRaster table (the table containing rows with at least one GeoRaster object column).

columnName
Name of a column of type SDO_GEORASTER in the GeoRaster table.

Usage Notes

To maintain the referential integrity of GeoRaster data structures, you should always use this procedure to create the standard GeoRaster DML trigger on each GeoRaster table and GeoRaster column combination before you perform any DML operations on the table. For example, if the GeoRaster table contains two GeoRaster columns, call this procedure twice, specifying the appropriate table name and column name for each call.

For more information about the standard GeoRaster DML trigger, including the operations that it performs, see Section 3.1.

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');
```

A

GeoRaster Metadata XML Schema

This appendix provides the XML schema definition that is used for GeoRaster metadata. The following is the definition of the GeoRaster metadata XML schema. (You can also see this definition by querying the SDO_GEOR_XMLSCHEMA_TABLE table, which is described in Section 2.5.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Oracle GeoRaster Metadata Schema -->
<xsd:schema targetNamespace="http://xmlns.oracle.com/spatial/georaster"
   xmlns="http://xmlns.oracle.com/spatial/georaster"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" version="0.0">
<xsd:annotation>
<xsd:documentation>=======================================
   This is the XML schema defining the metadata of Oracle GeoRaster object type.
   It consists of two parts: data type definitions and its element content.
    Part 1: Data Types
       Part 1.1: Data Types for Object Info
       Part 1.2: Data Types for Raster Info
       Part 1.3: Data Types for Spatial-Temporal-Band Reference Systems
       Part 1.3.1: Data Types for GeoRaster Spatial Reference System
       Part 1.3.2: Data Types for GeoRaster Temporal Reference System
       Part 1.3.3: Data Types for GeoRaster Band Reference System
       Part 1.4: Data Types for Layer Metadata
    Part 2: Metadata Elements / Content Structure of Oracle GeoRaster Object
                    ===================================
</xsd:documentation>
</xsd:annotation>
<xsd:annotation>
<xsd:documentation> ========================
                          Part 1:      Data Types
         ====================================
 </xsd:documentation>
```

```
</xsd:annotation>
<xsd:annotation>
<xsd:documentation> ==============================
         Part 1.1: Data Types for Object Info
         ==============================
</xsd:documentation>
</xsd:annotation>
<xsd:complexType name="objectDescriptionType">
  <xsd:sequence>
     <xsd:element name="rasterType" type="xsd:integer"/>
     <xsd:element name="ID" type="xsd:string" minOccurs="0"/>
     <xsd:element name="description" type="xsd:string" minOccurs="0"
           maxOccurs="unbounded"/>
     <xsd:element name="majorVersion" type="xsd:string" minOccurs="0"/>
     <xsd:element name="minorVersion" type="xsd:string" minOccurs="0"/>
     <xsd:element name="isBlank" type="xsd:boolean" default="false"/>
     <xsd:element name="blankCellValue" type="xsd:double" minOccurs="0"/>
     <xsd:element name="defaultRed" type="xsd:positiveInteger" minOccurs="0"/>
     <xsd:element name="defaultGreen" type="xsd:positiveInteger"
            minOccurs="0"/>
     <xsd:element name="defaultBlue" type="xsd:positiveInteger" minOccurs="0"/>
     <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>
  <xsd:annotation>
   <xsd:documentation> ==============================
           Part 1.2: Data Types for Raster Info
           ==============================
   </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType name="cellRepresentationType">
     <xsd:restriction base="xsd:string">
        <xsd:enumeration value="POINT"/>
        <xsd:enumeration value="SEGMENT"/>
        <xsd:enumeration value="TRIANGLE"/>
        <xsd:enumeration value="SQUARE"/>
        <xsd:enumeration value="RECTANGLE"/>
        <xsd:enumeration value="CUBE"/>
        <xsd:enumeration value="TETRAHEDRON"/>
        <xsd:enumeration value="HEXAHEDRON"/>
        <xsd:enumeration value="UNDEFINED"/>
     </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="cellDepthType">
     <xsd:restriction base="xsd:string">
```

```
      <xsd:enumeration value="1BIT"/>
      <xsd:enumeration value="2BIT"/>
      <xsd:enumeration value="4BIT"/>
      <xsd:enumeration value="8BIT_U"/>
      <xsd:enumeration value="8BIT_S"/>
      <xsd:enumeration value="16BIT_U"/>
      <xsd:enumeration value="16BIT_S"/>
      <xsd:enumeration value="32BIT_U"/>
      <xsd:enumeration value="32BIT_S"/>
      <xsd:enumeration value="32BIT_REAL"/>
      <xsd:enumeration value="64BIT_REAL"/>
      <xsd:enumeration value="64BIT_COMPLEX"/>
      <xsd:enumeration value="128BIT_COMPLEX"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="supportedDimensionNumber">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="3"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="cellDimensionType">
   <xsd:annotation>
      <xsd:documentation>
      The "Band" dimension can be treated as any other semantic dimension
      or any "Layer" if not remote sensing imagery or photographs
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ROW"/>
      <xsd:enumeration value="COLUMN"/>
      <xsd:enumeration value="VERTICAL"/>
      <xsd:enumeration value="BAND"/>
      <xsd:enumeration value="TEMPORAL"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="cellDimensionSizeType">
   <xsd:sequence>
      <xsd:element name="size" type="xsd:positiveInteger"
          default="1"/>
   </xsd:sequence>
   <xsd:attribute name="type" type="cellDimensionType"
          use="required"/>
</xsd:complexType>
<xsd:complexType name="cellCoordinateType">
```

```
   <xsd:sequence>
      <xsd:element name="row" type="xsd:integer" default="0"/>
      <xsd:element name="column" type="xsd:integer" default="0"/>
      <xsd:element name="vertical" type="xsd:integer"
               minOccurs="0"/>
      <xsd:element name="band" type="xsd:integer"
                minOccurs="0"/>
      <xsd:element name="temporal" type="xsd:integer"
               minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="compressionType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NONE"/>
      <xsd:enumeration value="RLE"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="compressionDescriptionType">
   <xsd:sequence>
      <xsd:element name="type" type="compressionType"
              default="NONE"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="blockingType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NONE"/>
      <xsd:enumeration value="REGULAR"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="blockingDescriptionType">
   <xsd:sequence>
      <xsd:element name="type" type="blockingType"
               default="NONE"/>
      <xsd:element name="totalRowBlocks" type="xsd:positiveInteger"
               default="1"/>
      <xsd:element name="totalColumnBlocks" type="xsd:positiveInteger"
               default="1"/>
      <xsd:element name="totalBandBlocks" type="xsd:positiveInteger"
               default="1" minOccurs="0"/>
      <xsd:element name="rowBlockSize" type="xsd:positiveInteger"/>
      <xsd:element name="columnBlockSize" type="xsd:positiveInteger"/>
      <xsd:element name="bandBlockSize" type="xsd:positiveInteger"
                minOccurs="0"/>
```

```
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="cellInterleavingType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="BSQ"/>
      <xsd:enumeration value="BIL"/>
      <xsd:enumeration value="BIP"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="pyramidType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NONE"/>
      <xsd:enumeration value="DECREASE"/>
      <xsd:enumeration value="INCREASE"/>
      <xsd:enumeration value="BIDIRECTION"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="resamplingType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NN"/>
      <xsd:enumeration value="BILINEAR"/>
      <xsd:enumeration value="CUBIC"/>
      <xsd:enumeration value="AVERAGE4"/>
      <xsd:enumeration value="AVERAGE16"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="pyramidDescriptionType">
   <xsd:sequence>
      <xsd:element name="type" type="pyramidType"
                default="NONE"/>
      <xsd:element name="resampling" type="resamplingType"
               default="NN" minOccurs="0"/>
      <xsd:element name="maxLevel" type="xsd:nonNegativeInteger"
               default="0" minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="rasterDescriptionType">
   <xsd:sequence>
      <xsd:element name="cellRepresentation"
               type="cellRepresentationType" default="UNDEFINED"/>
      <xsd:element name="cellDepth" type="cellDepthType"
               default="8BIT_U"/>
      <xsd:element name="NODATA" type="xsd:double"
```

```
                minOccurs="0"/>
      <xsd:element name="totalDimensions"
               type="supportedDimensionNumber" default="2"/>
      <xsd:element name="dimensionSize"
               type="cellDimensionSizeType" maxOccurs="5"/>
      <xsd:element name="ULTCoordinate" type="cellCoordinateType"/>
      <xsd:element name="blocking" type="blockingDescriptionType"/>
      <xsd:element name="interleaving" type="cellInterleavingType"
               default="BSQ"/>
      <xsd:element name="pyramid" type="pyramidDescriptionType"/>
      <xsd:element name="compression" type="compressionDescriptionType"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:annotation>
   <xsd:documentation>====================
  Part 1.3: Data Types for Spatial-Temporal-Band Reference Systems
    ===============================
         </xsd:documentation>
</xsd:annotation>
<xsd:annotation>
   <xsd:documentation>====================
  Part 1.3.1: Data Types for GeoRaster Spatial Reference System

  Spatial extent (footprint) is recorded as an attribute of the GroRaster
  object. Its type is SDO_GEOMETRY, so it is not included in the metadata.
  The cell space coordinates are named as (row, column, vertical).
  The model space coordinates are named as (x, y, z).
  Spatial unit information is stored in the WKT of the specified SRID.
    ===============================
         </xsd:documentation>
</xsd:annotation>
<xsd:simpleType name="modelDimensionType">
   <xsd:annotation>
      <xsd:documentation>
The following "S" means "Spectral" for remote sensing imagery.
Any of X, Y and Z can be horizontal or vertical or any other spatial direction
depending on the user interpretation in "modelDimensionDescription".
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="X"/>
      <xsd:enumeration value="Y"/>
      <xsd:enumeration value="Z"/>
      <xsd:enumeration value="T"/>
```

```
      <xsd:enumeration value="S"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="resolutionType">
   <xsd:sequence>
      <xsd:element name="resolution" type="xsd:double"
              default="1"/>
   </xsd:sequence>
   <xsd:attribute name="dimensionType"
            type="modelDimensionType" use="required"/>
</xsd:complexType>
<xsd:simpleType name="doubleNumberListType">
   <xsd:list itemType="xsd:double"/>
</xsd:simpleType>
<xsd:complexType name="polynomialType">
   <xsd:sequence>
      <xsd:element name="polynomialCoefficients"
               type="doubleNumberListType"/>
   </xsd:sequence>
   <xsd:attribute name="pType" type="xsd:nonNegativeInteger"
            use="optional" default="1"/>
   <xsd:attribute name="nVars" type="xsd:nonNegativeInteger"
            use="required"/>
   <xsd:attribute name="order" type="xsd:nonNegativeInteger"
            use="required"/>
   <xsd:attribute name="nCoefficients" type="xsd:nonNegativeInteger"
            use="required"/>
   <xsd:anyAttribute/>
</xsd:complexType>
<xsd:complexType name="rationalPolynomialType">
   <xsd:annotation>
      <xsd:documentation>
      row       =   pPolynomial(x, y, z) / qPolynomial(x, y, z)
      column =   rPolynomial(x, y, z) / sPolynomial(x, y, z)
            </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="pPolynomial" type="polynomialType"/>
      <xsd:element name="qPolynomial" type="polynomialType"/>
      <xsd:element name="rPolynomial" type="polynomialType"/>
      <xsd:element name="sPolynomial" type="polynomialType"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="rowOff" type="xsd:double" use="required"/>
   <xsd:attribute name="columnOff" type="xsd:double" use="required"/>
```

```
      <xsd:attribute name="xOff" type="xsd:double" use="required"/>
      <xsd:attribute name="yOff" type="xsd:double" use="required"/>
      <xsd:attribute name="zOff" type="xsd:double" use="required"/>
      <xsd:attribute name="rowScale" type="xsd:double" use="required"/>
      <xsd:attribute name="columnScale" type="xsd:double" use="required"/>
      <xsd:attribute name="xScale" type="xsd:double" use="required"/>
      <xsd:attribute name="yScale" type="xsd:double" use="required"/>
      <xsd:attribute name="zScale" type="xsd:double" use="required"/>
      <xsd:attribute name="rowRMS" type="xsd:double" use="optional"/>
      <xsd:attribute name="columnRMS" type="xsd:double" use="optional"/>
      <xsd:attribute name="totalRMS" type="xsd:double" use="optional"/>
      <xsd:anyAttribute/>
   </xsd:complexType>
   <xsd:simpleType name="rasterSpatialReferenceModelType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="RigorousModel"/>
         <xsd:enumeration value="StoredFunction"/>
         <xsd:enumeration value="FunctionalFitting"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="rasterSpatialReferenceSystemType">
      <xsd:sequence>
         <xsd:element name="isReferenced" type="xsd:boolean" default="false"/>
         <xsd:element name="isRectified" type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="isOrthoRectified" type="xsd:boolean" minOccurs="0"/>
         <xsd:element name="description" type="xsd:string" minOccurs="0"/>
         <xsd:element name="SRID" type="xsd:nonNegativeInteger" default="0"/>
         <xsd:element name="verticalSRID" type="xsd:integer" minOccurs="0"/>
         <xsd:element name="modelDimensionDescription" type="xsd:string"
minOccurs="0"/>
         <xsd:element name="spatialResolution" type="resolutionType"
minOccurs="0" maxOccurs="3"/>
         <xsd:element name="spatialTolerance" type="xsd:double" minOccurs="0"/>
         <xsd:element name="modelCoordinateLocation" minOccurs="0">
            <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="CENTER"/>
                  <xsd:enumeration value="UPPERLEFT"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="modelType" type="rasterSpatialReferenceModelType"
minOccurs="0" maxOccurs="3"/>
         <xsd:element name="polynomialModel" type="rationalPolynomialType"
minOccurs="0"/>
```

```
      <xsd:element name="gcpTableName" type="xsd:string" minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:annotation>
   <xsd:documentation> ==========
   Part 1.3.2: Data Types for GeoRaster Temporal Reference System

   The TRS will be modeled by formulas in the future.
                                      ======================
   </xsd:documentation>
</xsd:annotation>
<xsd:complexType name="rasterTemporalReferenceSystemType">
   <xsd:sequence>
      <xsd:element name="isReferenced" type="xsd:boolean" default="false"/>
      <xsd:element name="description" type="xsd:string" minOccurs="0"/>
      <xsd:element name="beginDateTime" type="xsd:dateTime" minOccurs="0"/>
      <xsd:element name="endDateTime" type="xsd:dateTime" minOccurs="0"/>
      <xsd:element name="temporalResolutionDescription" type="xsd:string"
              minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:annotation>
   <xsd:documentation> ==================
    Part 1.3.3: Data Types for GeoRaster Band Reference System

    For multispectral remote sensing images, each band is optionally
    described in the layerDescriptionType.
    The BRS is modeled by formulas for hyperspectral imagery
    (based on number of spectral segments, min and max wavelength,
    and number of bands for each segment).
    Detailed radiometric 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 sublayer the layerNumber is a positive integer; that is, layers are
logically numbered from 1 to n if the size of the specified layerDimension is n.
The layerDimensionOrdinate of each sublayer must be in the range of the
dimension and must be in the order of band ordinates.
For objectLayer, the layerNumber should be 0 but its layerDimensionOrdinate
is not used.
=======================
    </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType name="scalingFunctionType">
      <xsd:annotation>
         <xsd:documentation>
          value = (a0 + a1 * cellValue)  /  (b0 + b1 * cellValue)
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="a0" type="xsd:double" default="1"/>
         <xsd:element name="a1" type="xsd:double" default="0"/>
         <xsd:element name="b0" type="xsd:double" default="1"/>
         <xsd:element name="b1" type="xsd:double" default="0"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="binType">
      <xsd:annotation>
      <xsd:documentation>
LINEAR bin function:
binNumber = numbins * (cellValue - min) / (max - min) + firstBinNumber
if (binNumber less than 0) binNumber =  firstBinNumber
if (binNumber greater than or equal to numbins) binNumber = numbins +
     firstBinNumber - 1
LOGARITHM bin function:
binNumber = numbins * (ln (1.0 + ((cellValue - min)/(max - min)))/ ln (2.0))
     + firstBinNumber
if (binNumber less than 0) binNumber =  firstBinNumber
if (binNumber greater than or equal to numbins) binNumber = numbins +
      firstBinNumber - 1
EXPLICIT bin function means explicit (or direct) value (or value range)
for each bin, and it will be stored in a table.
      </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="LINEAR"/>
```

```
         <xsd:enumeration value="LOGARITHM"/>
         <xsd:enumeration value="EXPLICIT"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="binFunctionType">
      <xsd:annotation>
         <xsd:documentation>
The MAX and MIN in the statistics data set will be used if they are not provided
here. binTableName is used by EXPLICIT type only.
          </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:choice>
            <xsd:element name="binFunctionData" type="segmentationDataType"/>
            <xsd:element name="binTableName" type="xsd:string"/>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
         </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="type" type="binType" use="required"/>
   </xsd:complexType>
   <xsd:complexType name="rectangularWindowType">
      <xsd:sequence>
         <xsd:element name="origin" type="cellCoordinateType"/>
         <xsd:element name="rowHeight" type="xsd:positiveInteger"/>
         <xsd:element name="columnWidth" type="xsd:positiveInteger"/>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="cellCountType">
      <xsd:attribute name="value" type="xsd:double" use="required"/>
      <xsd:attribute name="count" type="xsd:nonNegativeInteger"
                use="required"/>
      <xsd:anyAttribute/>
   </xsd:complexType>
   <xsd:complexType name="rasterCountType">
      <xsd:sequence>
         <xsd:element name="cell" type="cellCountType"
                   maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="histogramType">
      <xsd:sequence>
         <xsd:choice>
            <xsd:element name="counts" type="rasterCountType"/>
            <xsd:element name="tableName" type="xsd:string"/>
```

```
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="statisticDatasetType">
   <xsd:sequence>
      <xsd:element name="samplingFactor" type="xsd:positiveInteger"
              default="1"/>
      <xsd:element name="samplingWindow"
               type="rectangularWindowType" minOccurs="0"/>
      <xsd:element name="MIN" type="xsd:double"/>
      <xsd:element name="MAX" type="xsd:double"/>
      <xsd:element name="MEAN" type="xsd:double"/>
      <xsd:element name="MEDIAN" type="xsd:double"/>
      <xsd:element name="MODEVALUE" type="xsd:double"/>
      <xsd:element name="STD" type="xsd:double"/>
      <xsd:element name="histogram" type="histogramType"
               minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="cellGrayType">
   <xsd:attribute name="value" type="xsd:double" use="required"/>
   <xsd:attribute name="gray" type="xsd:integer" use="required"/>
   <xsd:anyAttribute/>
</xsd:complexType>
<xsd:complexType name="rasterGrayType">
   <xsd:sequence>
      <xsd:element name="cell" type="cellGrayType"
               maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="grayScaleType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="grays" type="rasterGrayType"/>
         <xsd:element name="tableName" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="cellPseudoColorType">
   <xsd:attribute name="value" type="xsd:double" use="required"/>
   <xsd:attribute name="red" type="xsd:integer" use="required"/>
   <xsd:attribute name="green" type="xsd:integer" use="required"/>
   <xsd:attribute name="blue" type="xsd:integer" use="required"/>
   <xsd:attribute name="alpha" type="xsd:double" use="optional"/>
```

```
   <xsd:anyAttribute/>
</xsd:complexType>
<xsd:complexType name="rasterPseudoColorType">
   <xsd:sequence>
      <xsd:element name="cell" type="cellPseudoColorType"
               maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="colorMapType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="colors" type="rasterPseudoColorType"/>
         <xsd:element name="tableName" type="xsd:string"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="layerType">
   <xsd:sequence>
      <xsd:element name="layerNumber" type="xsd:nonNegativeInteger"/>
      <xsd:element name="layerDimensionOrdinate" type="xsd:integer"/>
      <xsd:element name="layerID" type="xsd:string"/>
      <xsd:element name="description" type="xsd:string" minOccurs="0"
              maxOccurs="unbounded"/>
      <xsd:element name="scalingFunction" type="scalingFunctionType"
              minOccurs="0"/>
      <xsd:element name="binFunction" type="binFunctionType"
               minOccurs="0"/>
      <xsd:element name="statisticDataset"
               type="statisticDatasetType" minOccurs="0"/>
      <xsd:element name="grayScale" type="grayScaleType" minOccurs="0"/>
      <xsd:element name="colorMap" type="colorMapType" minOccurs="0"/>
      <xsd:element name="vatTableName" type="xsd:string" minOccurs="0"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="layerDescriptionType">
   <xsd:sequence>
      <xsd:element name="layerDimension"
               type="cellDimensionType" default="BAND"/>
      <xsd:element name="objectLayer" type="layerType" minOccurs="0"/>
      <xsd:element name="subLayer" type="layerType"
               minOccurs="0" maxOccurs="unbounded"/>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

```
   <xsd:annotation>
      <xsd:documentation> =================
Part 2: Metadata Elements / Content Structure of Oracle GeoRaster Object
==============================
    </xsd:documentation>
   </xsd:annotation>
   <xsd:element name="georasterMetadata">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="objectInfo" type="objectDescriptionType"/>
            <xsd:element name="rasterInfo" type="rasterDescriptionType"/>
            <xsd:element name="spatialReferenceInfo"
                  type="rasterSpatialReferenceSystemType" minOccurs="0"/>
            <xsd:element name="temporalReferenceInfo"
                  type="rasterTemporalReferenceSystemType" minOccurs="0"/>
            <xsd:element name="bandReferenceInfo"
                  type="rasterBandReferenceSystemType" minOccurs="0"/>
            <xsd:element name="layerInfo" type="layerDescriptionType"
                  maxOccurs="unbounded"/>
            <xsd:element name="sourceInfo" type="xsd:string"
                  minOccurs="0" maxOccurs="unbounded"/>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Index

A

affine transformation
  support by GeoRaster, 1-20
ALL_SDO_GEOR_SYSDATA view, 2-11
alpha (opacity) value, 2-6
AVERAGE16 resampling method, 4-19
AVERAGE4 resampling method, 4-19

B

band numbers, 1-18
  bandBlockNumber attribute, 2-5
band reference system (BRS)
  description, 1-10
bandBlockNumber attribute of SDO_RASTER, 2-5
bands
  description, 1-18
BILINEAR resampling method, 4-19
blank GeoRaster objects, 1-17
BLOB data
  raster block data, 2-5
blocking keyword
  for importFrom storageParam parameter, 4-100
  for storageParam parameter, 1-14
blockMBR attribute of SDO_RASTER, 2-5
blockSize keyword for storageParam, 1-15
BMP image format
  support by GeoRaster, 1-36
BRS (band reference system)
  description, 1-10

C

cartography
  description, 1-4
cell coordinate system, 1-7
  relationship to model coordinate system, 1-9
cell data
  querying and updating, 3-7
cellDepth keyword for storageParam, 1-15
changeCellValue procedure, 4-2
changeFormat procedure, 4-5
changeFormatCopy procedure, 4-7
colormap
  alpha (opacity) value, 2-6
  getting, 4-41
  getting table, 4-44
  pseudocolor information, 4-97
  SDO_GEOR_COLORMAP object type, 2-6
COLUMN_NAME column (in USER_SDO_GEOR_SYSDATA view), 2-11
columnBlockNumber attribute of SDO_RASTER, 2-5
contrast table
  grayscale table, 2-7
copy procedure, 4-9
createBlank function, 4-11
createDMLTrigger procedure, 5-2
CUBIC resampling method, 4-20

D

dangling raster blocks, 3-10, 4-111
data model
  GeoRaster, 1-5

deletePyramid procedure, 4-13
demo files for GeoRaster
  PL/SQL, 1-36
digital image processing
  description, 1-5
DML trigger
  creating, 3-2, 5-2

E

empty GeoRaster objects, 1-17
ESRI world files
  loading, 4-101
  support by GeoRaster, 1-36
exporter tool for GeoRaster, 1-35
exporting
  GeoRaster objects, 3-8
exportTo procedure, 4-14

F

footprint, 2-2
formats
  image (supported by GeoRaster), 1-36
  vector and raster, 1-2

G

generatePyramid procedure, 4-19
generateSpatialExtent function, 4-21
geochemistry
  raster data used by, 1-5
geographic information systems (GIS)
  raster-based, 1-4
geology
  raster data used by, 1-5
geometadata, 2-3
geophysics
  raster data used by, 1-5
GeoRaster data model, 1-5
GeoRaster objects
  blank, 1-17
  changing physical storage, 3-6
  copying, 4-9
  copying and changing format, 4-7
  creating, 3-3
  creating blank, 4-11
  empty, 1-17
  exporting, 1-35, 3-8
  georeferencing, 3-4
  indexing spatial extent geometry, 3-5
  initializing, 4-103
  loading, 1-35, 3-3
  physical storage, 1-10
    changing, 3-6
  possible data problems, 3-10
  processing, 3-7
  pyramids
    definition, 1-22
  querying and updating cell data, 3-7
  querying and updating metadata, 3-6
  scaling, 4-112
  scaling and copying, 4-115
  subprograms to create, load, and export, 1-25
  subprograms to get and set metadata and data, 1-27
  subprograms to validate and process, 1-26
  transforming coordinate information, 3-5
  validating, 3-4
  viewing, 1-35, 3-7
GeoRaster tables
  column with GeoRaster object, 2-11
  definition, 1-11
  metadata column, 2-11
  name, 2-11
  other related tables, 2-12
  raster data table, 2-12
  raster ID, 2-12
GeoRaster tools, 1-35
GeoRaster XML schema table, 2-12
georeference procedure, 4-23
georeferencing
  description, 1-20
  details and formulas, 1-21
  methods for performing, 3-4
GeoTIFF image format
  support by GeoRaster, 1-36
getBandDimSize function, 4-26
getBeginDateTime function, 4-27
getBinTable function, 4-28

getBinType function, 4-30
getBlankCellValue function, 4-32
getBlockingType function, 4-34
getBlockSize function, 4-35
getCellCoordinate function, 4-36
getCellDepth function, 4-38
getCellValue function, 4-39
getColorMap function, 4-41
getColorMapTable function, 4-44
getCompressionType function, 4-46
getDefaultBlue function, 4-47
getDefaultColorLayer function, 4-48
getDefaultGreen function, 4-50
getDefaultRed function, 4-51
getEndDateTime function, 4-52
getGrayScale function, 4-53
getGrayScaleTable function, 4-55
getHistogram function, 4-57
getHistogramTable function, 4-58
getID function, 4-60
getInterleavingType function, 4-61
getLayerDimension function, 4-62
getLayerID function, 4-63
getLayerOrdinate function, 4-65
getModelCoordinate function, 4-67
getModelSRID function, 4-69
getNODATA function, 4-70
getPyramidMaxLevel function, 4-71
getPyramidType function, 4-72
getRasterBlocks function, 4-73
getRasterData procedure, 4-75
getRasterSubset procedure, 4-77
getScaling function, 4-80
getSpatialDimNumber function, 4-81
getSpatialDimSizes function, 4-83
getSpatialResolutions function, 4-84
getSpectralResolution function, 4-85
getSpectralUnit function, 4-86
getSRS function, 4-87
getStatistics function, 4-88
getTotalLayerNumber function, 4-90
getULTCoordinate function, 4-91
getVAT function, 4-92
getVersion function, 4-94
GIF image format
  support by GeoRaster, 1-36
grayscale
  checking for, 4-95
  returning mapping table, 4-55
  returning mappings, 4-53
  SDO_GEOR_GRAYSCALE object type, 2-7
  setting mapping table, 4-141
  setting mappings for a layer, 4-139
grayscale table, 2-7
gridded data, 1-2
ground coordinate system, 1-7

H

hasGrayScale function, 4-95
hasPseudoColor function, 4-97
histogram table
  getting, 4-58
  setting, 4-143
histograms
  getting, 4-57
  SDO_GEOR_HISTOGRAM object type, 2-5

I

image formats
  supported by GeoRaster, 1-36
importFrom procedure, 4-99
indexing
  GeoRaster data, 3-5
init function, 4-103
initializing
  GeoRaster objects, 4-103
interleaving, 1-20
  getting type, 4-61
  keyword for storageParam, 1-15
isBlank function, 4-105
isOrthoRectified function, 4-106
isRectified function, 4-108
isSpatialReferenced function, 4-109

J

JPEG image format
  loading (GeoRaster loader tool only), 4-101

support by GeoRaster, 1-36

L

layer numbers, 1-18
layers
description, 1-18
dimension, 4-62
ID, 4-63
ordinate, 4-65
loader tool for GeoRaster, 1-35
loading
GeoRaster data, 3-3
lookup table
grayscale table, 2-7

M

maps
managing with GeoRaster, 1-4
MBR (minimum bounding rectangle)
blockMBR attribute, 2-5
metadata
GeoRaster, 2-3
XML schema, A-1
metadata attribute of SDO_GEORASTER, 2-3
METADATA_COLUMN_NAME column (in USER_SDO_GEOR_SYSDATA view), 2-11
minimum bounding rectangle (MBR)
blockMBR attribute, 2-5
model coordinate system, 1-7
relationship to cell coordinate system, 1-9
model space, 1-7
mosaic procedure, 4-110

N

Nearest Neighbor (NN) resampling method, 4-19
NN (Nearest Neighbor) resampling method, 4-19
nodata cells
getting value for, 4-70

O

opacity
alpha value, 2-6
orthorectification, 1-20
checking for, 4-106
setting, 4-153
See also rectification
OTHER_TABLE_NAMES column (in USER_SDO_GEOR_SYSDATA view), 2-12

P

padding, 1-10
palette table, 2-6
photogrammetry
description, 1-3
physical storage
GeoRaster objects, 1-10
PL/SQL demo files for GeoRaster, 1-36
PNG image format
support by GeoRaster, 1-36
pseudocolor
checking for, 4-97
pseudocolor table, 2-6
pyramid keyword for storageParam, 1-15
pyramid levels
definition, 1-22
pyramidLevel attribute of SDO_GEORASTER, 2-4
pyramid type, 1-22
pyramidLevel attribute of SDO_RASTER, 2-4
pyramidParams parameter, 4-19
pyramids, 1-22
deleting data for, 4-13
formulas for determining, 1-23
generating data for, 4-19
illustration of, 1-23
pyramid parameters, 4-19
returning level number of top pyramid, 4-71

R

raster block data, 2-5
raster data
introduction, 1-2
raster data table (RDT)
definition, 1-11

object table of type SDO_RASTER, 2-4
rasterDataTable attribute, 2-3
RDT_TABLE_NAME column, 2-12
raster ID, 2-3, 2-4
raster space, 1-7
raster type, 2-2
RASTER_ID column (in USER_SDO_GEOR_SYSDATA view), 2-12
rasterBlock attribute of SDO_RASTER, 2-5
rasterDataTable attribute of SDO_GEORASTER, 2-3
rasterID attribute of SDO_GEORASTER, 2-3
rasterID attribute of SDO_RASTER, 2-4
rasterType attribute of SDO_GEORASTER, 2-2
RDT_TABLE_NAME column (in USER_SDO_GEOR_SYSDATA view), 2-12
README file
for GeoRaster demo files, 1-36
for GeoRaster tools, 1-35
rectification, 1-20
checking for, 4-108
setting, 4-156
See also orthorectification
remote sensing
description, 1-3
resampling method, 4-19
resolution
spectral, 4-164
rLevel keyword, 4-19
rowBlockNumber attribute of SDO_RASTER, 2-5

S

scale procedure, 4-112
scaleCopy procedure, 4-115
schemaValidate function, 4-118
SDO_GEOR package
changeCellValue, 4-2
changeFormat, 4-5
changeFormatCopy, 4-7
copy, 4-9
createBlank, 4-11
deletePyramid, 4-13
exportTo, 4-14
generatePyramid, 4-19
generateSpatialExtent, 4-21
georeference, 4-23
getBandDimSize, 4-26
getBeginDateTime, 4-27
getBinTable, 4-28
getBinType, 4-30
getBlankCellValue, 4-32
getBlockingType, 4-34
getBlockSize, 4-35
getCellCoordinate, 4-36
getCellDepth, 4-38
getCellValue, 4-39
getColorMap, 4-41
getColorMapTable, 4-44
getCompressionType, 4-46
getDefaultBlue, 4-47
getDefaultColorLayer, 4-48
getDefaultGreen, 4-50
getDefaultRed, 4-51
getEndDateTime, 4-52
getGrayScale, 4-53
getGrayScaleTable, 4-55
getHistogram, 4-57
getHistogramTable, 4-58
getID, 4-60
getInterleavingType, 4-61
getLayerDimension, 4-62
getLayerID, 4-63
getLayerOrdinate, 4-65
getModelCoordinate, 4-67
getModelSRID, 4-69
getNODATA, 4-70
getPyramidMaxLevel, 4-71
getPyramidType, 4-72
getRasterBlocks, 4-73
getRasterData, 4-75
getRasterSubset, 4-77
getScaling, 4-80
getSpatialDimNumber, 4-81
getSpatialDimSizes, 4-83
getSpatialResolutions, 4-84
getSpectralResolution, 4-85
getSpectralUnit, 4-86
getSRS, 4-87
getStatistics, 4-88

getTotalLayerNumber, 4-90
getULTCoordinate, 4-91
getVAT, 4-92
getVersion, 4-94
hasGrayScale, 4-95
hasPseudoColor, 4-97
importFrom, 4-99
init, 4-103
isBlank, 4-105
isOrthoRectified, 4-106
isRectified, 4-108
isSpatialReferenced, 4-109
mosaic, 4-110
reference information, 4-1
scale, 4-112
scaleCopy, 4-115
schemaValidate, 4-118
setBeginDateTime, 4-119
setBinTable, 4-121
setBlankCellValue, 4-123
setColorMap, 4-125
setColorMapTable, 4-127
setDefaultBlue, 4-129
setDefaultColorLayer, 4-131
setDefaultGreen, 4-133
setDefaultRed, 4-135
setEndDateTime, 4-137
setGrayScale, 4-139
setGrayScaleTable, 4-141
setHistogramTable, 4-143
setID, 4-145
setLayerID, 4-147
setLayerOrdinate, 4-149
setModelSRID, 4-151
setOrthoRectified, 4-153
setRasterType, 4-155
setRectified, 4-156
setScaling, 4-158
setSpatialReferenced, 4-160
setSpatialResolutions, 4-162
setSpectralResolution, 4-164
setSpectralUnit, 4-166
setSRS, 4-168
setStatistics, 4-170
setULTCoordinate, 4-172
setVAT, 4-174
setVersion, 4-176
subset, 4-178
validateGeoraster, 4-181
SDO_GEOR_COLORMAP object type, 2-6
SDO_GEOR_GRAYSCALE object type, 2-7
SDO_GEOR_HISTOGRAM object type, 2-5
SDO_GEOR_SRS object type, 2-8
SDO_GEOR_UTL package
createDMLTrigger, 5-2
reference information, 5-1
SDO_GEOR_XMLSCHEMA_TABLE table, 2-12
SDO_GEORASTER object type, 2-1
metadata attribute, 2-3
rasterDataTable attribute, 2-3
rasterID attribute, 2-3
rasterType attribute, 2-2
spatialExtent attribute, 2-2
SDO_RASTER object type, 2-4
bandBlockNumber attribute, 2-5
blockMBR attribute, 2-5
columnBlockNumber attribute, 2-5
pyramidLevel attribute, 2-4
rasterBlock attribute, 2-5
rasterID attribute, 2-4
rowBlockNumber attribute, 2-5
SDO_RASTERSET collection type, 2-8
setBeginDateTime procedure, 4-119
setBinTable procedure, 4-121
setBlankCellValue procedure, 4-123
setColorMap procedure, 4-125
setColorMapTable procedure, 4-127
setDefaultBlue procedure, 4-129
setDefaultColorLayer procedure, 4-131
setDefaultGreen procedure, 4-133
setDefaultRed procedure, 4-135
setEndDateTime procedure, 4-137
setGrayScale procedure, 4-139
setGrayScaleTable procedure, 4-141
setHistogramTable procedure, 4-143
setID procedure, 4-145
setLayerID procedure, 4-147
setLayerOrdinate procedure, 4-149
setModelSRID procedure, 4-151
setOrthoRectified procedure, 4-153

setRasterType procedure, 4-155
setRectified procedure, 4-156
setScaling procedure, 4-158
setSpatialReferenced procedure, 4-160
setSpatialResolutions procedure, 4-162
setSpectralResolution procedure, 4-164
setSpectralUnit procedure, 4-166
setSRS procedure, 4-168
setStatistics procedure, 4-170
setULTCoordinate procedure, 4-172
setVAT procedure, 4-174
setVersion procedure, 4-176
spatial extent, 2-2
spatial reference system (SRS)
  description, 1-9
spatial resolution values
  getting, 4-84
  setting, 4-162
spatialExtent attribute of SDO_GEORASTER, 2-2
spectral resolution
  getting, 4-85
  setting, 4-164
spectral unit
  getting, 4-86
  setting, 4-166
sRGB ColorSpace, 2-6
SRS (spatial reference system)
  description, 1-9
storage parameters, 1-14
storageParam parameter, 1-14
subset procedure, 4-178

T

TABLE_NAME column (in USER_SDO_GEOR_SYSDATA view), 2-11
temporal reference system (TRS)
  description, 1-10
themes
  raster layers, 1-18
TIF image format
  support by GeoRaster, 1-36
transforming
  GeoRaster coordinate information, 3-5
triggers
  creating GeoRaster DML trigger, 3-2, 5-2
troubleshooting, 3-10
TRS (temporal reference system)
  description, 1-10

U

ULTCoordinate
  definition, 1-9
USER_SDO_GEOR_SYSDATA view, 2-10
utility subprograms
  GeoRaster, 5-1

V

validateGeoraster function, 4-181
validating
  GeoRaster objects, 3-4
value attribute table (VAT)
  description, 1-4
  getting name of, 4-92
  setting name of, 4-174
vector data
  description, 1-2
viewer tool for GeoRaster, 1-35
views
  ALL_SDO_GEOR_SYSDATA, 2-10
  USER_SDO_GEOR_SYSDATA, 2-10

W

world files (ESRI)
  loading, 4-101
  support by GeoRaster, 1-36

X

XML schema for GeoRaster metadata, A-1
XML schema table for GeoRaster, 2-12