

Using LizardTech Wavelet Compression with Oracle Spatial GeoRaster

Oracle Spatial GeoRaster 10g Release 2 provides a plugin architecture to support wavelet compression schemes such as those from LizardTech. Users who install the LizardTech plugin may compress GeoRaster objects using any of three wavelet compression types: MrSID Generation 2, MrSID Generation 3 and LizardTech JPEG 2000.

This document describes the use of the LizardTech plugin. It first briefly describes how to register the plugin with Oracle Spatial GeoRaster. Next, it describes the additional metadata that may exist in the GeoRaster object, and finally it outlines the typical use of the LizardTech plugin with the GeoRaster PL/SQL API, including some restrictions.

Important note: Any errors/issues related to the wavelet compression and decompression should be addressed to the supplier of the compression library. Oracle is not responsible for any errors/issues related to use of the LizardTech plugin.

This document is intended to complement the Oracle Spatial GeoRaster user guide and is intended for information purposes only. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle. See the end of this document for other important legal notices.

1. Installing and registering the plugin

A full description of the installation process for the plugin is provided by LizardTech, and is available at:

<http://www.lizardtech.com/products/geo/oracle.php>

Once the plugin is installed, verify that the plugin is registered with Oracle Spatial GeoRaster. You can check this using the following statement:

```
select plugin_name, plugin
       from MDSYS.SDO_GEOG_PLUGIN_REGISTRY
       where plugin_type = 'COMPRESSION';
```

For LizardTech wavelet compressions, the "plugin_name" must be LT-MG2, LT-MG3 or LT-JP2 and the "plugin" must be TRUE in the SDO_GEOG_PLUGIN_REGISTRY table.

2. Wavelet Compressed GeoRaster Objects

In order to support the wavelet compressions, three new values are listed in the compressionType of the GeoRaster XML schema:

```
<xsd:simpleType name="compressionType">
  <xsd:restriction base="xsd:string">
```

```

... ..
<xsd:enumeration value="LT-MG2"/>
<xsd:enumeration value="LT-MG3"/>
<xsd:enumeration value="LT-JP2"/>
</xsd:restriction>
</xsd:simpleType>

```

These three new compression types (“LT-MG2”, “LT-MG3” and “LT-JP2”), are also used as the keyword values for the “compression” parameter in various GeoRaster functions or procedures.

When a GeoRaster object is compressed using one of the wavelet compression types, the `compressionType` is set to the corresponding value. The pyramid type is set to `DECREASE`, the pyramid resampling is absent, and the pyramid `maxLevel` is set to an appropriate value as determined by the LizardTech compression plugin. All other metadata and attributes of the GeoRaster object are not changed. The raster, or cell, data is stored in raster data tables (RDTs) using the provider-specific binary structure.

The cell data is still stored in the raster data table as blocks and identified by the `rasterID`. However, the physical organization of raster blocks and the binary storage format of the wavelet coefficients (i.e., cell data) are specific to, and hence only decipherable by, the LizardTech plugin. The blocks of the wavelet coefficients are not the same as the blocks of the uncompressed raster blocks. Moreover, pyramids are a built-in feature of the wavelet compressions themselves, so there is no need for an extra step to build the pyramids. In summary, this means the plugin must always be involved when accessing the raster data. So once the GeoRaster object is compressed using one of the LizardTech formats, you must use only the GeoRaster APIs to access these objects and never attempt to decode or encode the content of raster data tables directly.

While the metadata is managed by GeoRaster and the cell data is stored in raster data tables using the same architecture, the wavelet transformed cell data is reorganized and managed by the plugin. So, once a GeoRaster object is compressed into one of the above three wavelet compressions, the plugin is responsible for correct decompression and related operations.

3. Wavelet Compression and Decompression of GeoRaster Objects

There is no new API added for wavelet compression and decompression. You only need to use the existing GeoRaster API, such as the `SDO_GEOR.changeFormatCopy` procedure. There is no direct access to the plugin and users should not attempt that.

To compress and decompress GeoRaster objects, simply call `SDO_GEOR.changeFormatCopy` or other procedures and specify one of “LT-MG2”, “LT-MG3” or “LT-JP2” as the **compression** keyword in the `storageParam` parameter. Section 1.4.1 in the Oracle Spatial GeoRaster manual has further details on this

parameter. Check the LizardTech documentation for any generic or version-specific restrictions on supported compression types. To decompress GeoRaster objects, you must specify “NONE” as the **compression** keyword. To change the compression from one wavelet type to another, such as from LT-MG2 to LT-JP2, specify the new compression type.

Two additional new storageParam parameter keywords, “ratio” and “colorspace”, have been added and are valid only for these wavelet compression types. They are ignored if the resulting compression type is not a wavelet compression.

* **ratio**: Specifies the compression ratio, a floating point value between 0 and MAX_FLOAT. The default value is 0 (lossless). For a given uncompressed GeoRaster object, the greater the compression ratio, the smaller size of the compressed GeoRaster object. For example, with a compression ratio of 20 (that is, 20:1), the data is compressed to 5 percent of its original size.

* **colorspace**: The value must be either "RGB" or "MULTI". The default is "MULTI". "RGB" can only be specified when the resulting GeoRaster object has exactly three bands. If “RGB” is specified, color space conversion will be internally done by the plugin to achieve better compression and ensure better quality. If “MULTI” is specified, the resulting GeoRaster object can be single layer, multilayer, or even hyperspectral.

These keywords, and keyword values, may be used in the storageParam parameter of several GeoRaster procedures, including SDO_GEOR.changeFormatCopy, SDO_GEOR.subset and SDO_GEOR.mosaic. These procedures automatically set the resulting object's compression and pyramid metadata to appropriate values. There is no need to call SDO_GEOR.generatePyramid to build the pyramids for a GeoRaster object compressed using a wavelet compression. Calling SDO_GEOR.deletePyramid on such objects will not delete the pyramids. When such a GeoRaster object is decompressed using SDO_GEOR.changeFormatCopy, you can specify whether to keep the pyramids by setting the “pyramid” keyword (the default value is TRUE) in the storageParam parameter.

The following example illustrates the compression of a GeoRaster object using MrSID Generation 3 (LT-MG3):

```
DECLARE
  gr1 SDO_GEOASTER;
  gr2 SDO_GEOASTER;
BEGIN
  SELECT georaster INTO gr1 FROM georaster_table WHERE georid=1;
  INSERT INTO georaster_table (georid, georaster)
    VALUES (2, SDO_GEOR.init('RDT_1'))
    RETURNING georaster INTO gr2;
  SDO_GEOR.changeFormatCopy(
    gr1,
```

```

                'compression=LT-MG3 ratio=30 colorspace=RGB',
                gr2);
    UPDATE georaster_table SET georaster=gr2 WHERE georid=2;
    COMMIT;
END;
/

```

The next example shows how to decompress the above LT-MG3 compressed GeoRaster object and generate the pyramids for the uncompressed object:

```

DECLARE
    gr2 SDO_GEORASTER;
    gr3 SDO_GEORASTER;
BEGIN
    SELECT georaster INTO gr1 FROM georaster_table WHERE georid=2;
    INSERT INTO georaster_table (georid, georaster)
        VALUES (3, SDO_GEOOR.init('RDT_1'))
        RETURNING georaster INTO gr3;
    SDO_GEOOR.changeFormatCopy(
        gr2,
        'compression=NONE pyramid=TRUE',
        gr3);
    UPDATE georaster_table SET georaster=gr3 WHERE georid=3;
    COMMIT;
END;
/

```

4. Restrictions when working with Wavelet Compressed GeoRaster Objects

Most of the existing GeoRaster procedures and functions work directly on the wavelet compressed GeoRaster objects. In other words, users don't need to explicitly decompress or compress them and then apply those operations. GeoRaster will internally compress or decompress the objects whenever necessary and appropriate.

In the current release, however, there are a few restrictions when using LizardTech wavelet compressions.

(1) Supported cell depths: For LT-MG2 compression type use one of 8BIT_U, 16BIT_U, or 32BIT_REAL. For LT-MG3 compression type use 8BIT_U or 16BIT_U. For LT-JP2 compression type use one of 8BIT_U, 8BIT_S, 16BIT_U, or 16BIT_S. When wavelet compression or decompression is involved, no cell depth change can be specified in the storageParam parameter of the GeoRaster procedures.

(2) Conversions among the LT-MG2, LT-MG3 and LT-JP2 compression types are supported through the LizardTech plugin. However, conversions between these wavelet compression types and the other compression types supported by GeoRaster (i.e.,

DEFLATE, JPEG-B and JPEG-F) are not permitted with changeFormatCopy and subset. Users must decompress and recompress the GeoRaster objects in separate steps.

(3) GeoRaster objects that have colormap or grayscale metadata information cannot be compressed with the LT-MG2, LT-MG3 or LT-JP2 type. Remove the colormap or grayscale metadata before performing the compression operation on such GeoRaster objects. You cannot set colormap or grayscale metadata for a GeoRaster object compressed with the wavelet compression types.

(4) The following procedures are disabled on GeoRaster objects compressed with the wavelet compression types:

SDO_GEOR.changeCellValue,
SDO_GEOR.generatePyramid,
SDO_GEOR.deletePyramid,
SDO_GEOR.exportTo,
SDO_GEOR.setColorMap,
SDO_GEOR.setColorMapTable,
SDO_GEOR.setGrayScale, and
SDO_GEOR.setGrayScaleTable.

(5) SDO_GEOR.scaleCopy is disabled if either the source or the resulting GeoRaster object is compressed, or is to be compressed using one of the wavelet compression types. If you want to use this procedure, you must decompress and recompress the GeoRaster objects in separate steps.

(6) The SDO_GEOR.importFrom doesn't support any of the wavelet compression types.

(7) The result of the SDO_GEOR.getRasterSubset, SDO_GEOR.getRasterData and SDO_GEOR.mosaic function cannot be compressed with any of the wavelet compression types. However, the output of these functions can be decompressed or compressed into DEFLATE or JPEG. So, when these functions are called on LizardTech compressed GeoRaster objects, you must specify keyword 'compression' with one of the four values: NONE, DEFLATE, JPEG-B or JPEG-F. If the source GeoRaster object is compressed with one of the wavelet compression types, the cell depth of the result cannot be changed via the celldepth keyword in the storageParam parameter.

(8) Other restrictions may apply. Please check the LizardTech documentation.

All error messages related to the LizardTech plugin will be raised through Oracle error code 13498, followed by detailed descriptions.

Product availability, installation guide, and other information about the plugin will be available at: (<http://www.lizardtech.com/products/geo/oracle.php>). This site also provides information on utilities for loading MrSID or JPEG2000 files into Oracle Spatial GeoRaster.

Copyright © 2006, Oracle. All rights reserved.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Any errors/issues related to the wavelet compression and decompression should be addressed to the supplier of the compression library. Oracle is not responsible for any errors/issues related to use of the LizardTech plugin.

This document is intended to complement the Oracle Spatial GeoRaster user guide and is intended for information purposes only. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

This document provides links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.