ORACLE

May 21, 2014
Walter E. Washington Convention Center
Washington, DC USA

Effectively Utilize Raster Data In Your Business Processes

Albert Godfrind
Spatial Solutions Architect, Oracle

Program Agenda

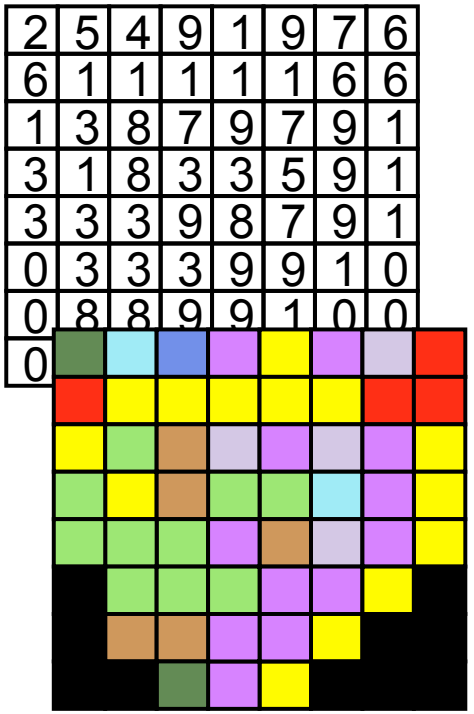
- Georaster Concepts
- Creating, loading and viewing
- Raster Algebra and Analytics
- Image Processing and Virtual Mosaic
- Performance and Parallelism

Program Agenda

- Georaster Concepts
- Creating, loading and viewing
- Raster Algebra and Analytics
- Image Processing and Virtual Mosaic
- Performance and Parallelism

What is a Raster ?

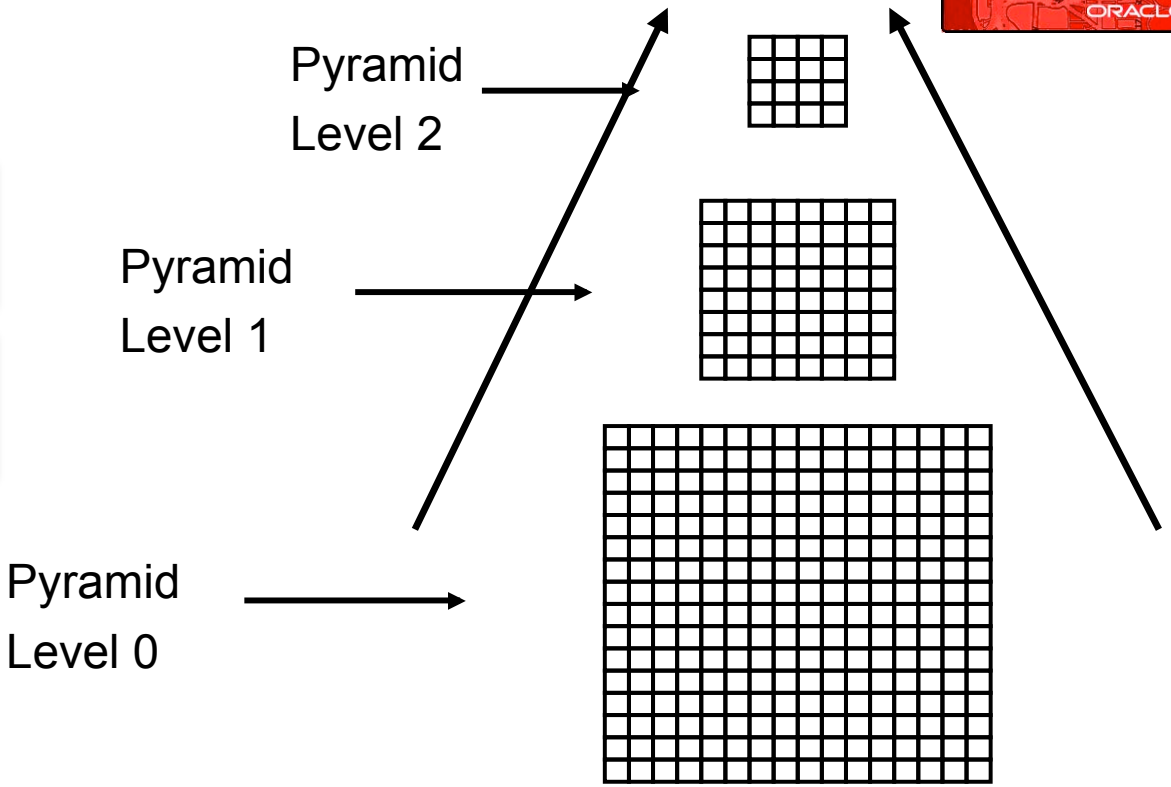
- Two dimensional array of regularly spaced elements (*pixels* or *cells*)
  - Orthophotos
  - Remote Sensing
  - Gridded data (raster GIS)
- Image data is collected by a variety of technologies
  - Satellite remote sensing
  - Airborne photogrammetry,
  - …
- Digital images can be composed of one or more bands
  - Bands often represent an interval of wavelengths along the electromagnetic spectrum
  - Band data can be simultaneously recorded



Resolution Pyramid

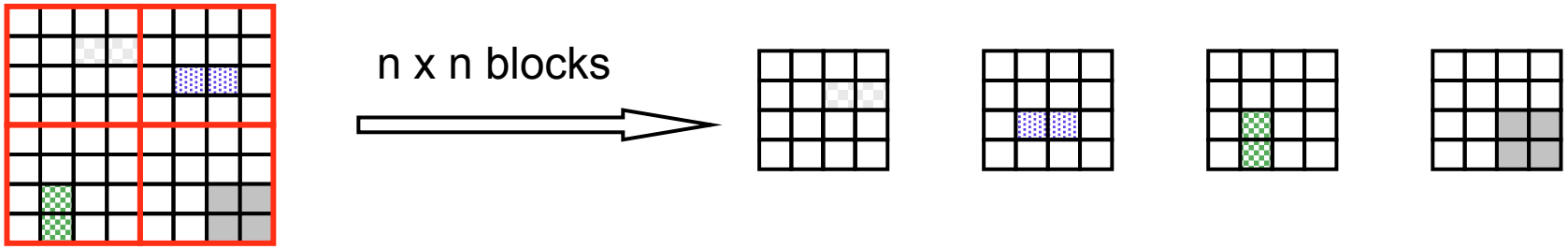
Gradually simplified copies of the raster

Step down the pyramid when zooming in



Blocking

- A GeoRaster can be composed of an extremely large number of cells
- It is more efficient in terms of storage and retrieval to break large images into smaller blocks
- In GeoRaster, users/applications can determine how data is blocked
  - Specify rows, columns, and optionally bands



Cell Depth

- Specifies the size of the cells or pixels
- Integers can be 8bits, 16bits or 32bits, signed or unsigned
- Default for imagery is 8BIT_U

1BIT	16BIT_U
2BIT	16BIT_S
4BIT	32BIT_U
8BIT_U	32BIT_S
8BIT_S	32BIT_REAL
	64BIT_REAL

Compression

JPEG Compression

- Lossy compression
- Use for **imagery**
- For rasters with cellDepth=8BIT_U and no more than 4 bands per block
- JPEG-B or JPEG-F mode
- Control the compression level using the quality parameter
  - 0 (max compression) to 100 (no compression)

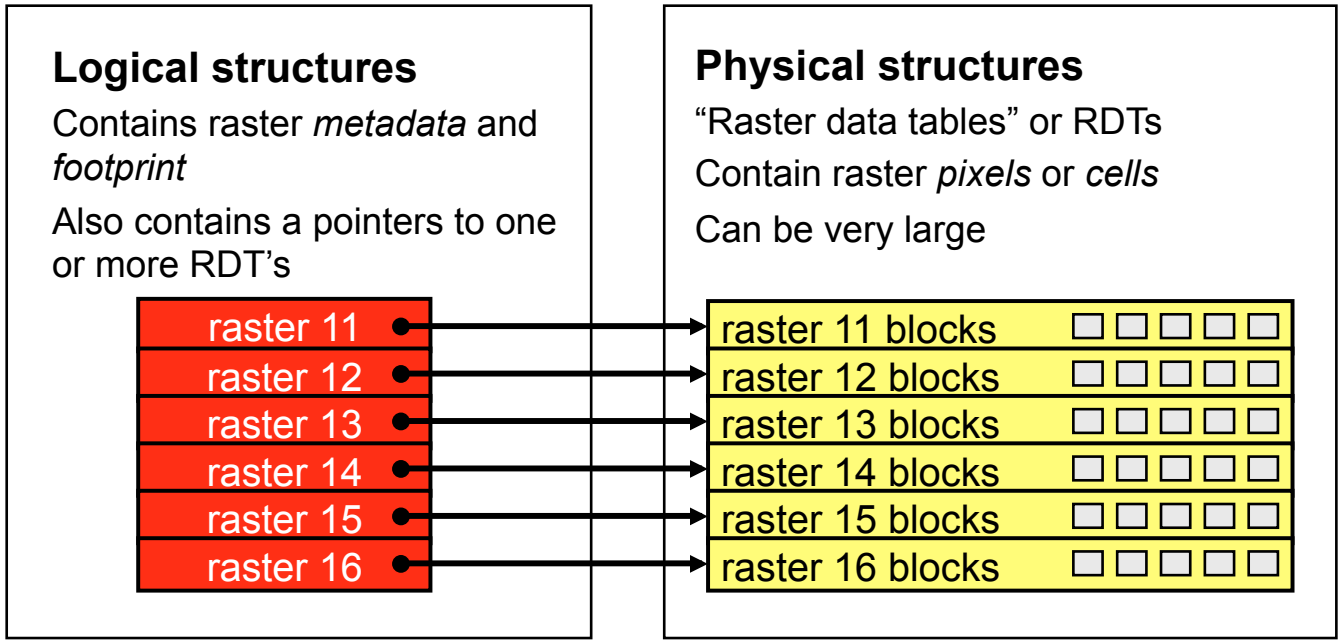
DEFLATE Compression

- Lossless compression
- Use for **grids**, terrain models, etc
- Uses the ZLIB format

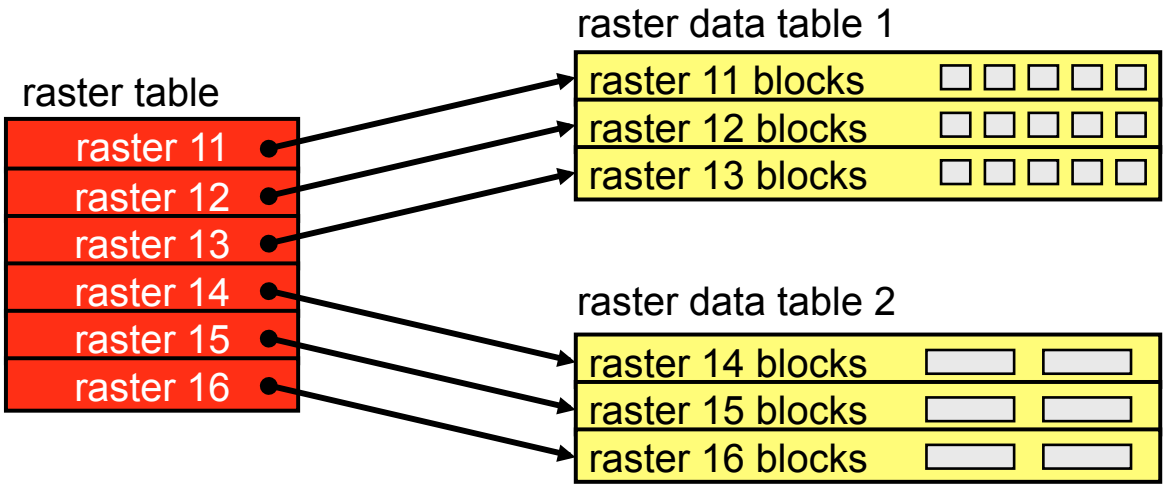
You can also use securefiles (blob) compression. This is part of the Advanced Compression option.

Storage Model

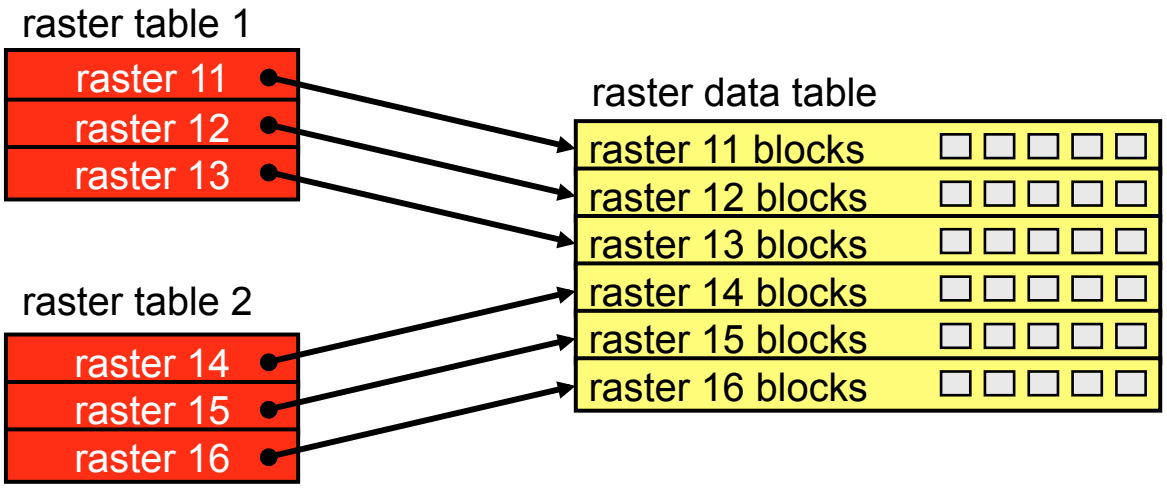
Separates Logical from Physical structures



Storage Model



Storage Model



Program Agenda

- Georaster Concepts
- Creating, loading and viewing
- Raster Algebra and Analytics
- Image Processing and Virtual Mosaic
- Performance and Parallelism

Creating Raster Tables


```
CREATE TABLE US_RASTERS(
  GEORID        NUMBER PRIMARY KEY,
  SOURCE_FILE   VARCHAR2(80),
  DESCRIPTION   VARCHAR2(32),
  GEORASTER     SDO_GEORASTER
);
```

- Use the SDO_GEORASTER type
- Combine with any other attributes

```
RASTERTYPE          NUMBER
SPATIALEXTENT       SDO_GEOMETRY
RASTERDATATABLE     VARCHAR2(32)
RASTERID            NUMBER
METADATA            SYS.XMLTYPE
```

Creating Raster Data Tables

- Use the SDO_RASTER type – describes one raster block
- Control the storage of LOBs
- Use SECUREFILE lobs

```
CREATE TABLE RDT_01 OF SDO_RASTER (
  PRIMARY KEY (
     RASTERID, PYRAMIDLEVEL, BANDBLOCKNUMBER,
     ROWBLOCKNUMBER, COLUMNBLOCKNUMBER
  )
)
LOB(RASTERBLOCK) STORE AS SECUREFILE (
  STORAGE (INITIAL 1024G)
  NOCACHE NOLOGGING
);
```

```
RASTERID              NUMBER
PYRAMIDLEVEL          NUMBER
BANDBLOCKNUMBER       NUMBER
ROWBLOCKNUMBER        NUMBER
COLUMNBLOCKNUMBER     NUMBER
BLOCKMBR              SDO_GEOMETRY
RASTERBLOCK           BLOB
```

Initializing Rasters

- Specify the storage location for the blocks
  - Name of **raster data table**
  - **Unique identifier** in that table
  - If no id provided, one will be assigned **automatically**

This is generally done by the loading tool

```
INSERT INTO US_RASTERS (GEORID, GEORASTER)
  VALUES (1, SDO_GEOR.INIT('RDT_01',1));
INSERT INTO US_RASTERS (GEORID, GEORASTER)
  VALUES (2, SDO_GEOR.INIT('RDT_01',2));
INSERT INTO US_RASTERS (GEORID, GEORASTER)
  VALUES (3, SDO_GEOR.INIT('RDT_02',3));
INSERT INTO US_RASTERS (GEORID, GEORASTER)
  VALUES (4, SDO_GEOR.INIT('RDT_02',4));
```

Raster data table

Raster id

ORACLE

Loading Rasters

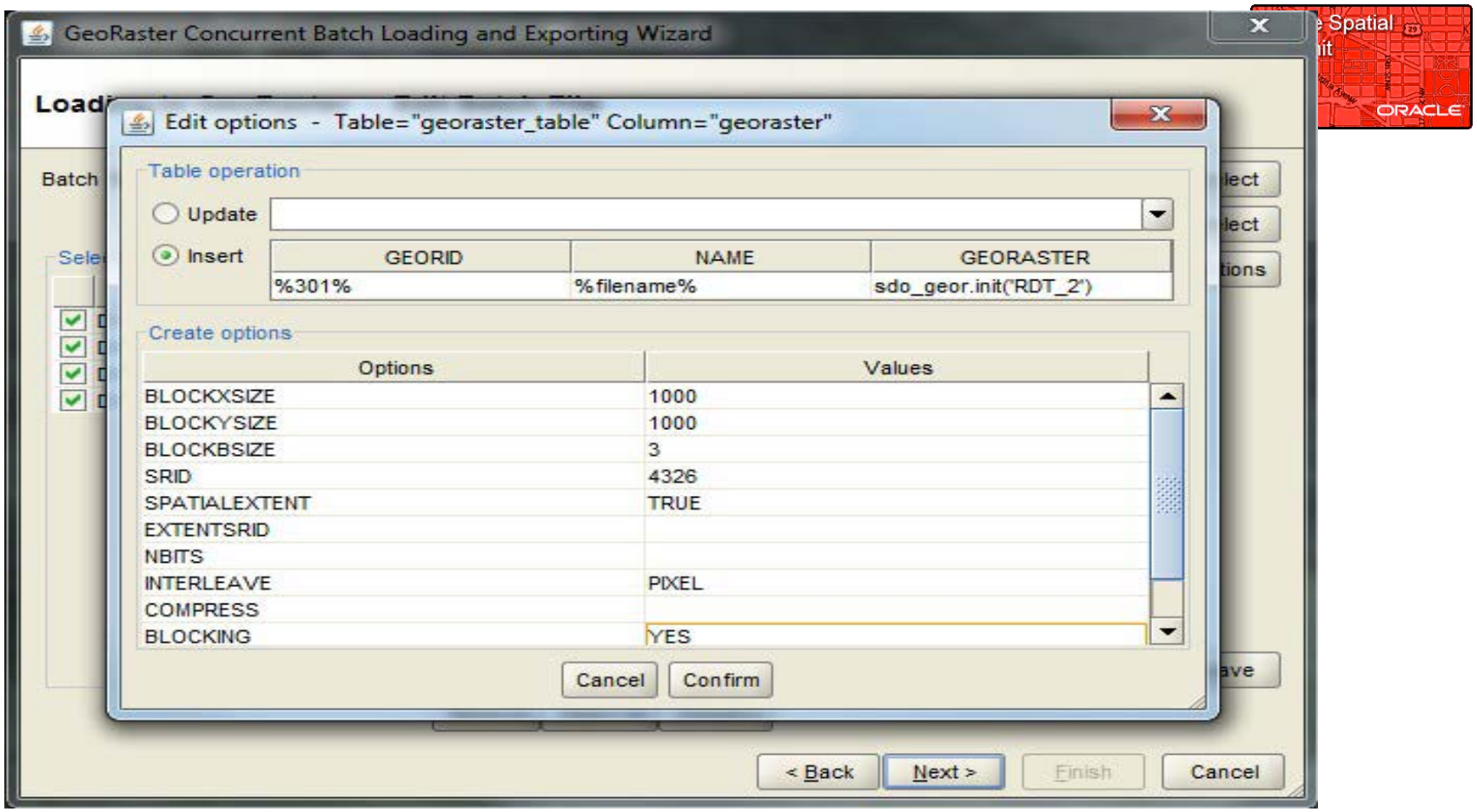
- Oracle tools
  - Java command line
  - Raster viewer/importer/exporter
  - Mapbuilder
  - Java API
- Open Source
  - GDAL
- Commercial tools
  - FME

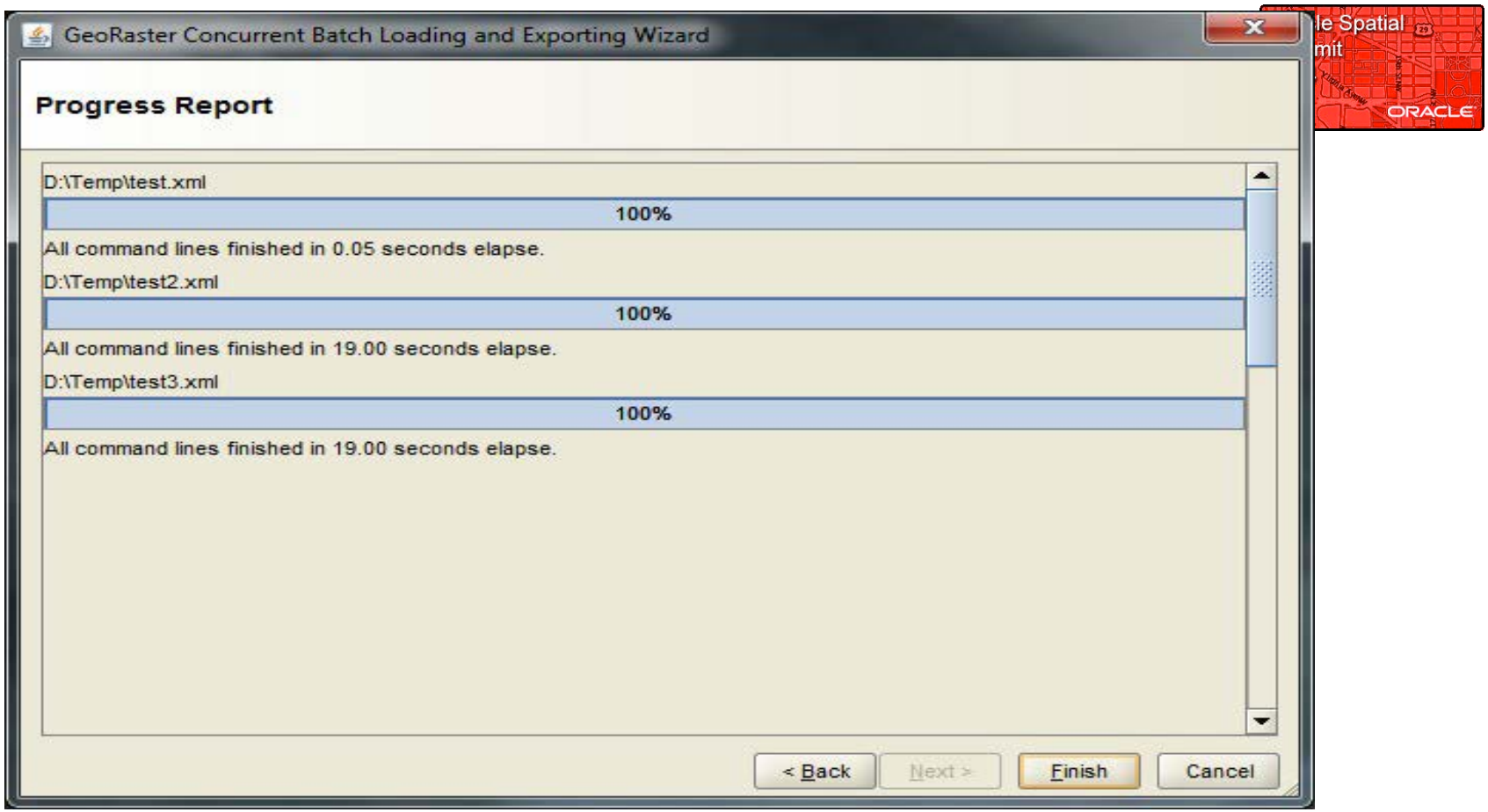
- Database-Driven
  - PL/SQL
  - SDO_GEOR.ImportFrom()

- Georaster ETL
  - Choose files to load
  - Set parameters
  - Uses GDAL for the actual loading
  - Parallel loading

GeoRaster ETL

- **Simplifies** the process of loading large number of rasters
  - GUI to select files to load and choose loading options
  - Block size, compression, pyramiding, …
  - Configures the load process in an XML description file
- Uses **GDAL** to perform the actual loading
  - Includes an Oracle-built distribution of GDAL for popular platforms
  - Windows 32 and 64-bit, Linux 64-bit
  - Loads from all GDAL supported file formats
- Drives the loading in **parallel**
  - Multiple concurrent GDAL jobs





GeoRaster ETL: Where is it ?

- In your Oracle 12c installation:
  - $ORACLE_HOME/md/demo/georaster/etl
  - Need to install the examples kit (*Oracle Database 12c Release 1 Examples*)
- On OTN
  - http://www.oracle.com/technetwork/indexes/samplecode/spatial-1433316.html

Importing using GDAL


```
gdal_translate -of georaster sf1.tif
  georaster:scott/tiger@orcl121,us_rasters,georaster
  -co blockxsize=512 -co blockysize=512 -co blockbsize=3
  -co interleave=bip -co srid=26943
  -co "insert=(georid, source_file, georaster) values (1, 'sf1.tif',
      sdo_geor.init('us_rasters_rdt_01'))"
```

`-of georaster`	Use Oracle Georaster as output
`sf1.tif`	Name of input TIFF file
`georaster:scott/tiger@orcl121`	Database connection
`us_rasters,georaster`	Destination table and column
`-co block%size`	Block size in x, y and b (band)
`-co SRID=26943`	Coordinate system of the raster
`-co interleave`	Interleaving (default is band sequential)
`-co insert`	The parameters of an insert statement to insert a row in the raster table

Generating the resolution pyramid


```
declare
  geor sdo_georaster;
begin
  select georaster into geor from us_rasters
  where georid = 1 for update;

  sdo_geor.generatePyramid(geor, 'rlevel=4');

  update us_rasters
     set georaster = geor
   where georid = 1;
end;
```

Read Raster for update

Generate 4-level pyramid

Update Raster

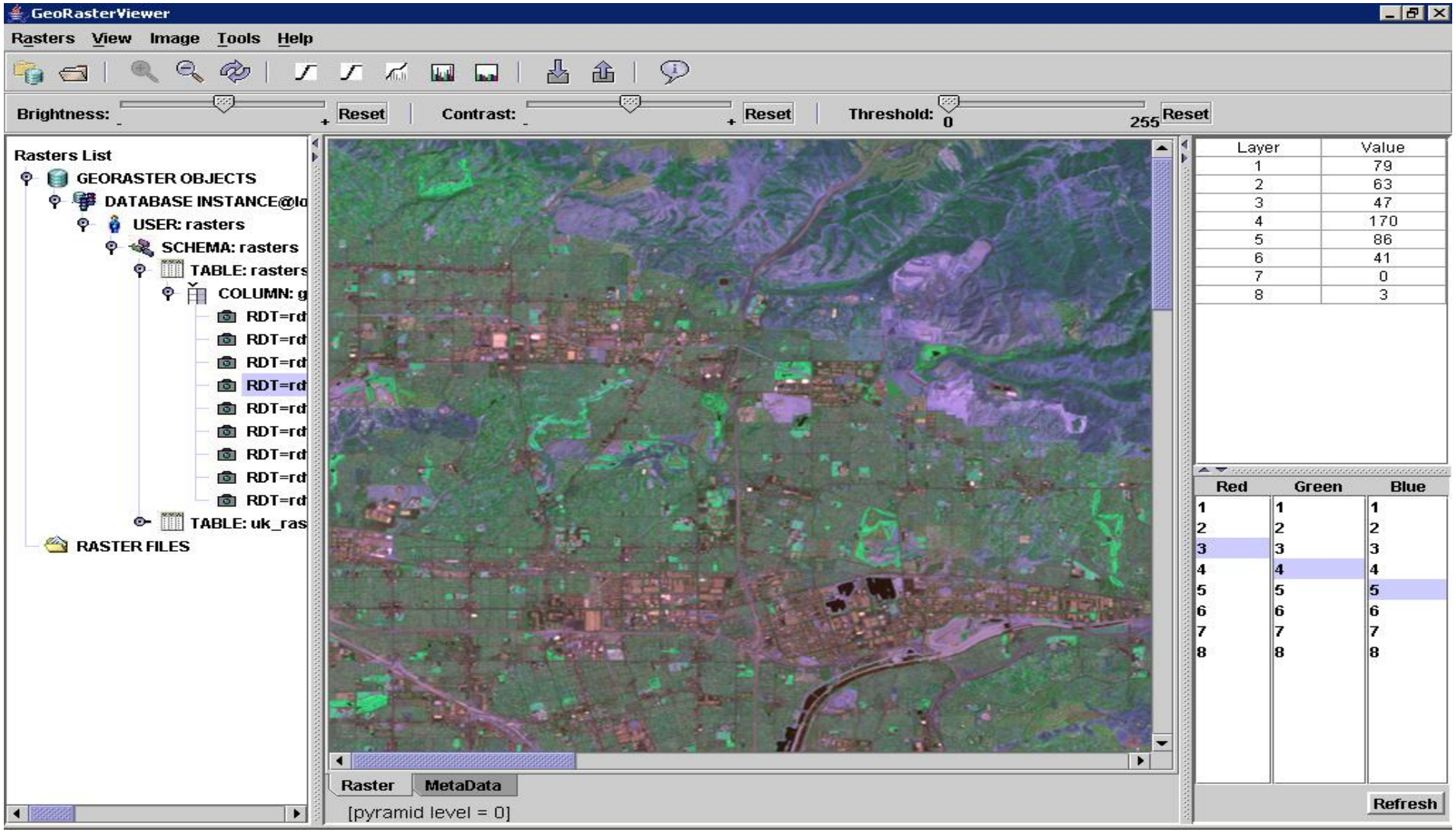
Pyramiding parameters

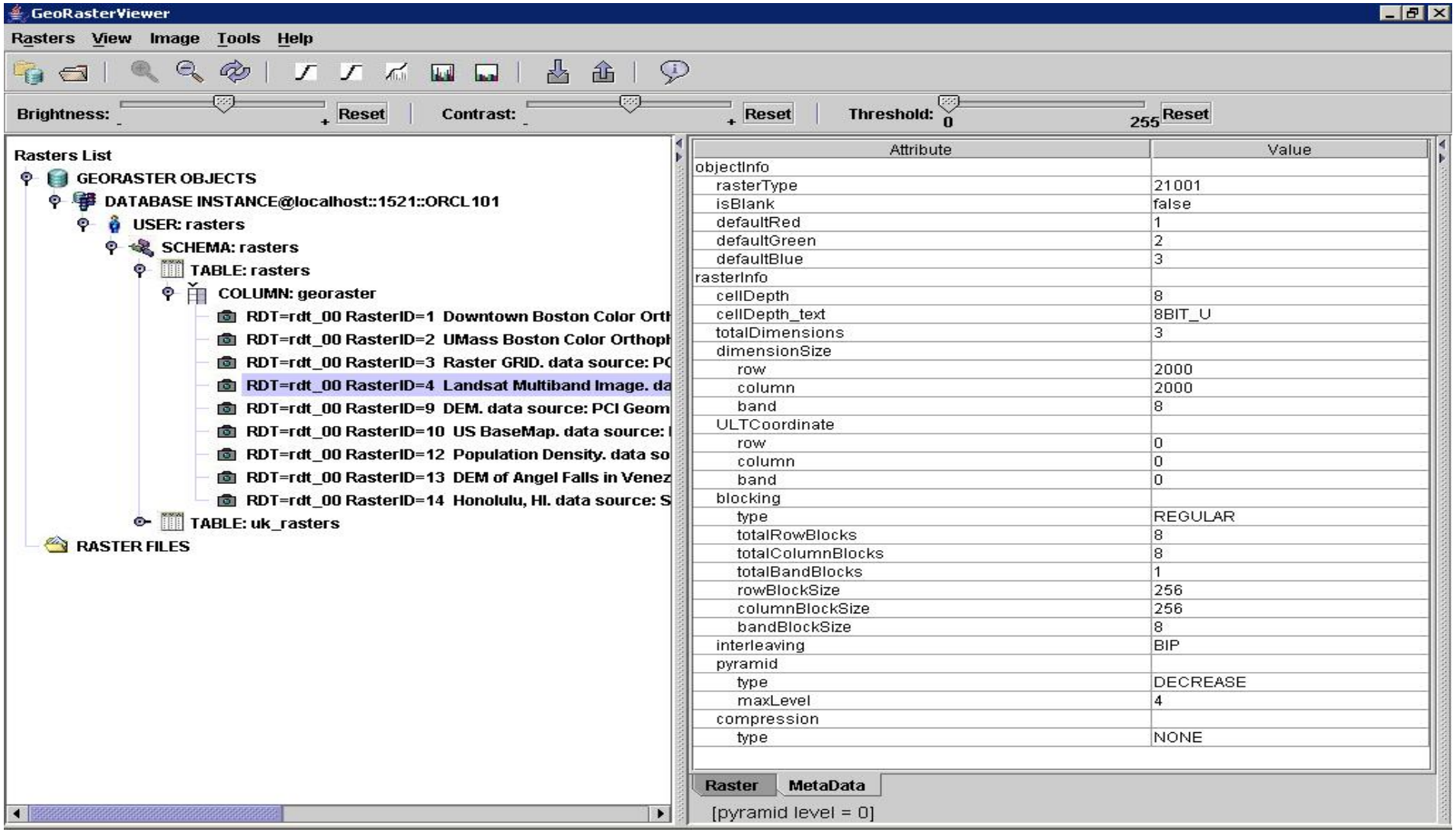
- **Resampling Modes**
  - **NN**: Value of the nearest neighbor cell (used by default)
  - **BILINEAR**: Distance-weighted average of the 4 nearest cells
  - **AVERAGE4**: Simple average of the 4 nearest cells
  - **AVERAGE16**: Simple average of the 16 nearest cells
  - **CUBIC**: Cubic convolution of the 16 nearest cells
- **Number of levels**
  - If not specified, pyramid levels are generated until the smaller of the number of rows or columns is between 64 and 128.

Creating the Spatial Index


```
insert into user_sdo_geom_metadata values (
  'US_RASTERS',
  'GEORASTER.SPATIALEXTENT',
   sdo_dim_array(
     sdo_dim_element('Easting',  1000000, 2000000, 1),
     sdo_dim_element('Northing',  500000,  800000, 1)
   ),
   40978);
```

```
create index us_rasters_sx
  on us_rasters (georaster.spatialextent)
  indextype is mdsys.spatial_index;
```


GeoRasterViewer
Rasters View Image Tools Help
Brightness: Reset
Contrast: Reset
Threshold: 0 255 Reset
Rasters List
GEORASTER OBJECTS
DATABASE INSTANCE@lo
USER: rasters
SCHEMA: rasters
TABLE: rasters
COLUMN: g
RDT=rd
RDT=rd
RDT=rd
RDT=rd
RDT=rd
RDT=rd
RDT=rd
RDT=rd
RDT=rd
TABLE: uk_ras
RASTER FILES
Raster
MetaData
[pyramid level = 0]
Layer
Value
1 79
2 63
3 47
4 170
5 86
6 41
7 0
8 3
Red
Green
Blue
Refresh


GeoRasterViewer
Rasters View Image Tools Help
Brightness: Reset
Contrast: Reset
Threshold: 0 255 Reset
Rasters List
GEORASTER OBJECTS
DATABASE INSTANCE@localhost::1521::ORCL101
USER: rasters
SCHEMA: rasters
TABLE: rasters
COLUMN: georaster
RDT=rdt_00 RasterID=1 Downtown Boston Color Orth
RDT=rdt_00 RasterID=2 UMass Boston Color Orthopl
RDT=rdt_00 RasterID=3 Raster GRID. data source: PC
RDT=rdt_00 RasterID=4 Landsat Multiband Image. da
RDT=rdt_00 RasterID=9 DEM. data source: PCI Geom
RDT=rdt_00 RasterID=10 US BaseMap. data source:
RDT=rdt_00 RasterID=12 Population Density. data so
RDT=rdt_00 RasterID=13 DEM of Angel Falls in Venez
RDT=rdt_00 RasterID=14 Honolulu, HI. data source: S
TABLE: uk_rasters
RASTER FILES
Attribute
Value
objectInfo
rasterType 21001
isBlank false
defaultRed 1
defaultGreen 2
defaultBlue 3
rasterInfo
cellDepth 8
cellDepth_text 8BIT_U
totalDimensions 3
dimensionSize
row 2000
column 2000
band 8
ULTCoordinate
row 0
column 0
band 0
blocking
type REGULAR
totalRowBlocks 8
totalColumnBlocks 8
totalBandBlocks 1
rowBlockSize 256
columnBlockSize 256
bandBlockSize 8
interleaving BIP
pyramid
type DECREASE
maxLevel 4
compression
type NONE
Raster
MetaData
[pyramid level = 0]

Program Agenda

- Georaster Concepts
- Creating, loading and viewing
- Raster Algebra and Analytics
- Image Processing and Virtual Mosaic
- Performance and Parallelism

Key GeoRaster 12c Features

- Advanced Queries and Manipulations
  - From general spatial queries to content and context based raster analytical queries and image aggregations
- Massive In-Database Processing
  - From a focus of database management capabilities to in-database image processing and raster analysis engines
- High Performance Computing
  - From serial processing to parallel processing

What is Raster Algebra

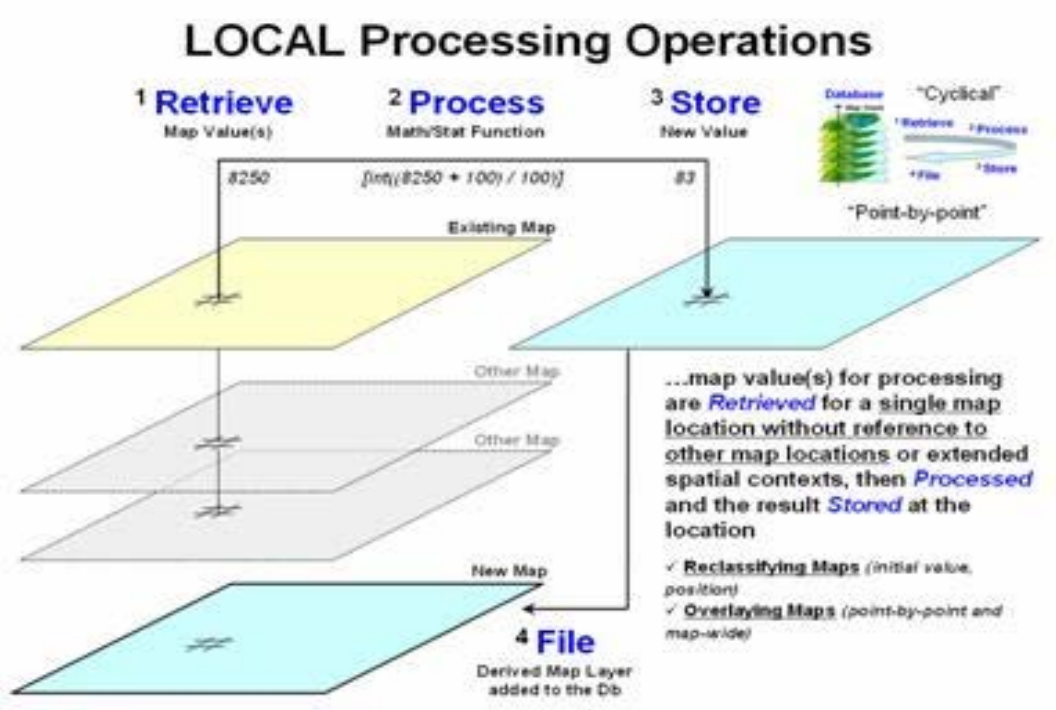
- **Raster Algebra (Map Algebra)** is a set-based algebra for manipulating geographic raster data, proposed by Dr. Dana Tomlin in the early 1980s. It's widely used for cartographic modeling and spatial analysis.
- It includes a set of primitive algebraic operators applied on one or more raster layers of similar dimensions to produce one or more new raster layers or other values.
- There are 4 types of operations: **local**, **focal**, **zonal** and **global**.
- It provides a procedural or scripting language enabling very complex operations
- However, all implementations are different.

Local Raster Algebra Operations

Local raster algebra applies the same operation point-by-point for each cell location of the input raster layers.

$Z = (X+500)/(Y+100)$

where X, Y are input raster layers. Z is the output layer



(Local operations use point-by-point processing of map values that occur at each map location. *Source:* **Joseph K. Berry**, *Beyond Mapping III)*

Raster Algebra and Analytics Engine

- The Raster Algebra Language is provided as an extension to PL/SQL
- Supports **local** raster algebra only in this release
- It includes an expression language for raster algebra operators
  - general arithmetic, casting, logical and relational operators
- It includes 4 raster algebra functions
  - cell value based conditional queries; cell value based conditional updates; arithmetic operations; raster classification/segmentation
- Polygon clipping based statistics generation functions to support interactive analysis on-the-fly, which are considered as focal/zonal/global functions.

Raster Algebra and Analytics Engine

- Raster Algebra Language includes an **expression** language and four **functions** of Map Algebra:
  - Cell-value based conditional **queries**: searches/masks cells based on boolean expression
  - Cell-value based **updates**: update cells of a raster-based on boolean expression
  - **Arithmetic operations** on cell values: ADD, DIVIDE, LOG, etc.
  - **Classification** for raster segmentation: applies arithmetic expression to cells and then segments the raster
- Polygon clipping based statistics generation functions, to support interactive analysis on-the-fly

The Raster Algebra Language

An Extension to the PL/SQL Language

- **PL/SQL provides**
  - ✓ declaration of variables and constants
  - ✓ general expressions and functions
  - ✓ statements
  - ✓ Programs

- **GeoRaster provides**
  - ✓ raster algebra operators/expressions
  - ✓ raster algebra functions

Searching For Cells

A Sample Application of Raster Algebra

Select all cells for which the value of band 0 is between 10 and 50, band 1 is between 100 and 150 and band 2 is between 200 and 245

```
declare
  geor1   SDO_GEORASTER; -- source 3-band image
  geor2   SDO_GEORASTER; -- result 3-band image
begin
  select georaster into geor1 from georaster_table where georid = 1;
  select georaster into geor2 from georaster_table where georid = 2;
  sdo_geor_ra.findcells (
    geor1,
    '({0}>=10&{0}<50)&({1}>=100&{1}<150)&({2}>200&{2}<245)',
    null, geor2, null, 'false','parallel=4'
  );
  update georaster_table set georaster = geor2 where georid = 2;
end;
```

The Raster Algebra Expressions

- Elements are variables, constants, operators (boolean, comparison or arithmetic), and functions
- **Variables** refer to the values of cells in the raster(s) to process
  - Form is {r,b} or {b}
  - {2,0} refers to band 0 of the third input raster (for functions that operate on multiple rasters)
  - {1} refers to band 1 in the only input raster
- **Constants** are any number (integer or decimal)
  - 0, 255, 0.44556,-273.15 are all valid numbers

The Raster Algebra Expressions

- **Arithmetic** Operators are what you expect:
  - +, -, *, /
- **Comparison** Operators:
  - >, <, =, !=, <=, >=
- **Boolean** Operators:
  - &, |, !
- Usual precedence rules for operators apply
  - Use parentheses to force the order
- Boolean expressions return 0 or 1

The Raster Algebra Expressions

- **Functions**:
  - *abs, sqrt, exp, log, ln, sin, cos, tan, sinh, cosh, tanh, arcsin, arccos, arctan, ceil, floor*
  - Apply to any arithmetic expression
- **Casting** a cell value to a specific data type:
  - Useful when converting between different cell depths
  - For example from 32BIT_FLOAT to 8BIT_UNSIGNED
  - *castint, castonebit, casttwobit, castfourbit, casteightbit*
  - Apply to any arithmetic expression

The Raster Algebra Expressions

- An expression applies to all cells in the input rasters
- Input rasters must be coherent:
  - Same row and column dimensions,
  - Same geographical area.
- Can have different cell depth and different number of bands

The Raster Algebra Expressions

- For example: ({0}+{1}+{2})/3 applies to all cells in the input raster.
  - It computes the average of the values in bands 0, 1 and 2 for each cell.

- Same for boolean expressions: {0,1}>{1,1}*1.10 applies the comparison to all cells of the two input rasters.
  - It returns “true” when the value in band 1 in the first raster is more than 10% larger than the same value in the second raster
  - (actually returns 1 for “true” and 0 for “false”)

The Raster Algebra Expressions

arithmeticExpr:
unaryArithmeticExpr
| binaryArithmeticExpr
| functionArithmeticExpr
| booleanExpr
| castingExpr
| constantNumber
| identifier
| (arithmeticExpr)

booleanExpr:
unaryBooleanExpr
| binaryBooleanExpr
| arithmeticExpr comparisonOp
arithmeticExpr
| (booleanExpr)

unaryArithmeticExpr:
(arithmeticUnaryOp arithmeticExpr)

binaryArithmeticExpr:
arithmeticExpr arithmeticBinaryOp
arithmeticExpr)

functionArithmeticExpr:
numericFunction (arithmeticExpr)

castingExpr:
rangeType (arithmeticExpr)

unaryBooleanExpr:
booleanUnaryOp booleanExpr

binaryBooleanExpr:
booleanExpr booleanBinaryOp
booleanExpr

arithmeticBinaryOp:
+
| -
| *
| /

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

The Raster Algebra Expressions


```
arithmeticUnaryOp:
          +
        | -

booleanBinaryOp:
          &
        | |

booleanUnaryOp:
        !

rangeType:
          castint
        | castonebit
        | casttwobit
        | castfourbit
        | casteightbit

numericFunction:
          abs
        | sqrt
        | exp
        | log
        | ln
```

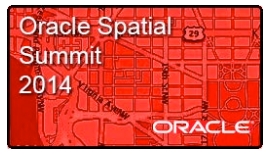
```
        | sin
        | cos
        | tan
        | sinh
        | cosh
        | tanh
        | arcsin
        | arccos
        | arctan
        | ceil
        | floor

constantNumber:
        double number

identifier:
          {ID,band}
        | {band}

ID:
        integer number

band:
        integer number
```

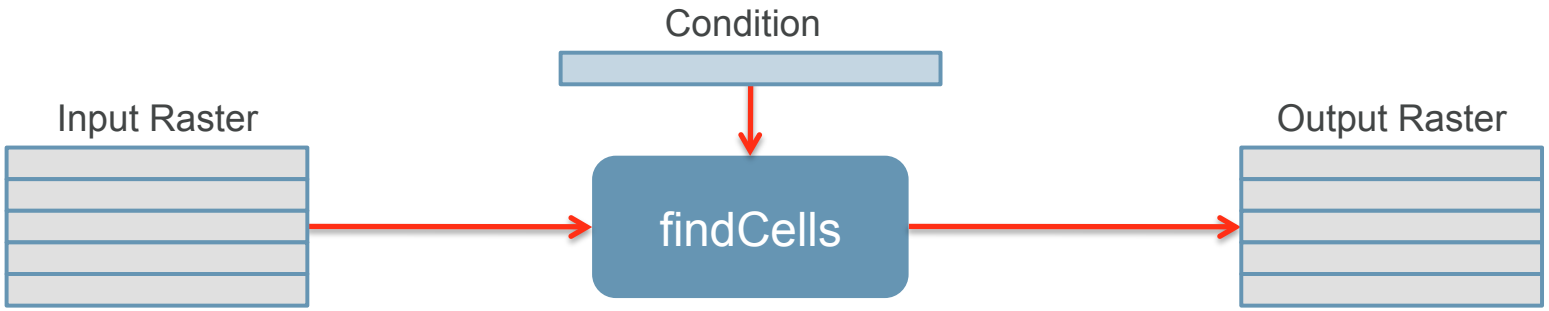
The Raster Algebra Functions

sdo_geor_ra.findCells	Selects cells based on a boolean expression
sdo_geor_ra.rasterUpdate	Updates cells of a raster based on a boolean expression
sdo_geor_ra.rasterMathOp	Performs arithmetic operations
sdo_geor_ra.classify	Applies arithmetic operations to cells and then segments the raster

Cell Value-Based Conditional Queries

sdo_geor_ra.findCells





- Select all cells from the input raster that match the boolean condition
- Copy them into the output raster
- Other cells in the output raster are set to a chosen fixed value
- The output raster has the same structure as the input raster (same bands, same cell depth)

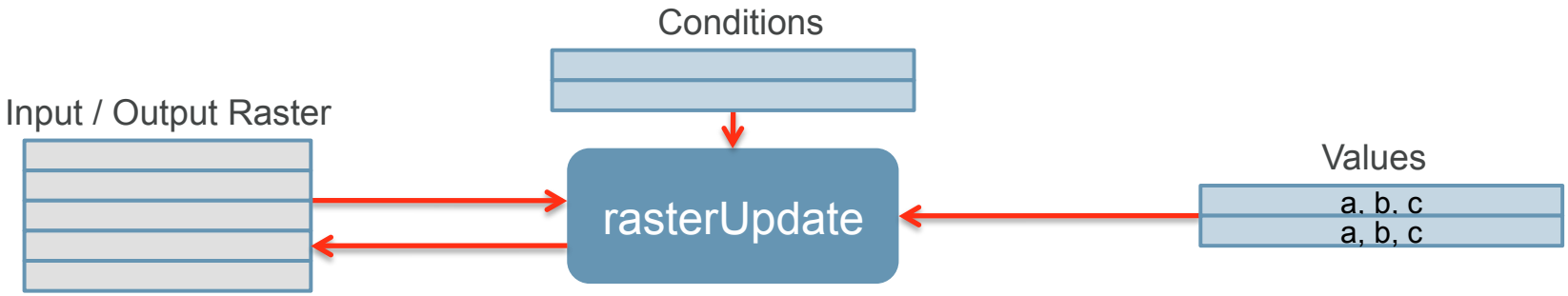
findCells example

- Input is a 3-band RGB color image
- Find all cells from the input raster where green dominates
  - Value of band 1 is greater than value of band 0 and value of band 2

```
sdo_geor_ra.findCells (
  inGeoraster   => gr1,
  condition     => '{1}>={0}&{1}>={2}',
  storageParam  => null,
  outGeoraster  => gr2,
  bgValues      => sdo_number_array (255,255,255)
);
```

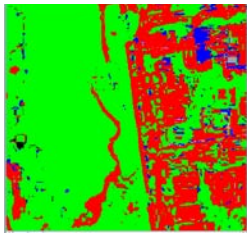
Cell Value-Based Conditional Updates (Edits)

sdo_geor_ra.rasterUpdate



- There can be one or more input conditions
- Each condition is associated with a set of numeric values
- Select the cells from the input raster that match each of the boolean conditions
- When matched, replace its value(s) from the corresponding value(s) provided

rasterUpdate example

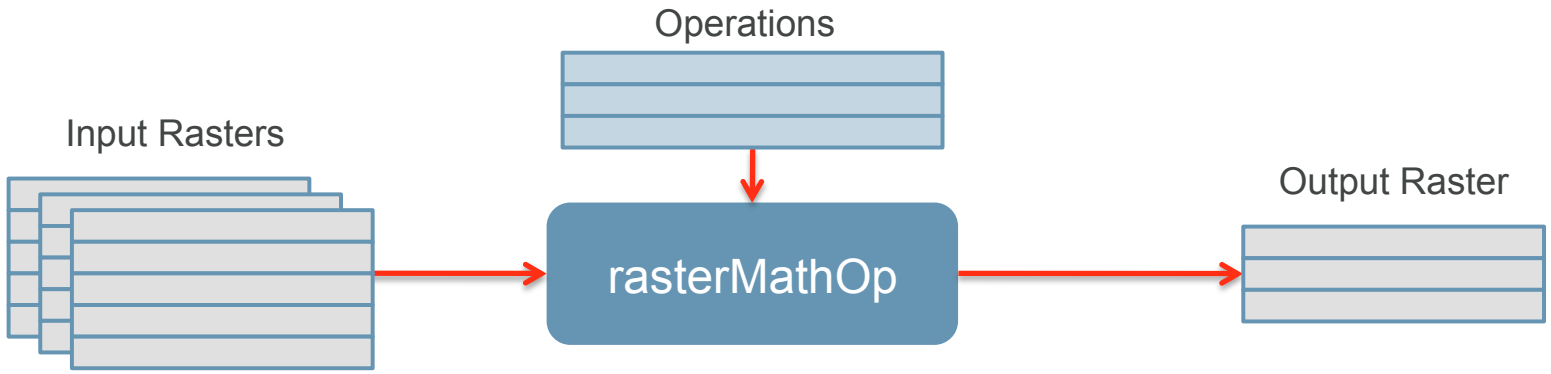


- Input is a 3-band RGB color image
- Replace cells with their dominant color

```
sdo_geor_ra.rasterUpdate (
  georaster     =>  gr1,
  pyramidLevel  =>  null,
  conditions    =>  sdo_string2_array(
                      '({0}>{1})&({0}>{2})',     -- Red dominates
                      '({1}>{0})&({1}>{2})',     -- Green dominates
                      '({2}>{0})&({2}>{1})'      -- Blue dominates
                    ),
  vals          =>  sdo_string2_arrayset(
                      sdo_string2_array('255','0','0'),
                      sdo_string2_array('0','255','0'),
                      sdo_string2_array('0','0','255')
                    )
);
```

Mathematical Operations

sdo_geor_ra.rasterMathOp



- There are one or several input operations (arithmetic expressions)
- There are one or several input rasters and one output raster
- Apply each operation to the input raster(s)
- Write the result into one of the bands of the output raster
- The output raster has one band per operation

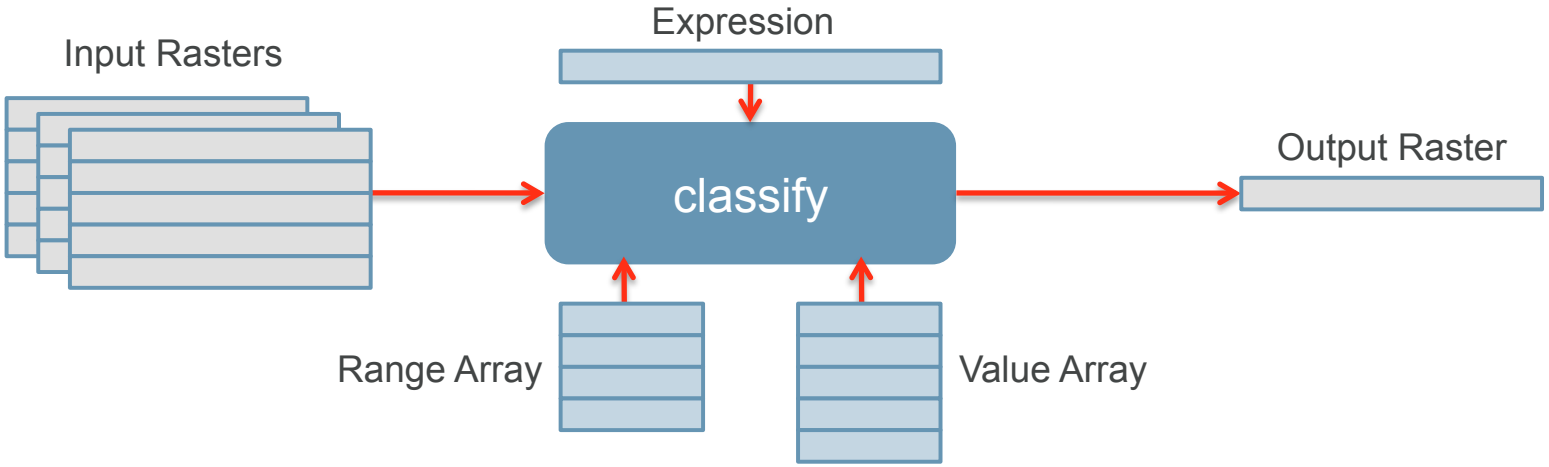
rasterMathOp example

- Input is a 3-band RGB color image
- Create new single band raster in grayscale

```
sdo_geor_ra.rasterMathOp (
  inGeoraster   => gr1,
  operation     => sdo_string2_array('({0}+{1}+{2})/3'),
  outGeoraster  => gr2,
  storageParam  => null
);
```

Classification Operations

sdo_geor_ra.classify



- Apply the expression on the input rasters
- Lookup the result in the Range Array
- Write the corresponding value from the Value Array in the output raster.

classify example

```
33333333000012332231 0
33333333221123322212 2
33333112211233222121 2
33331120110011222123 0
33300033221123323331 2
33331123112111122212 3
33333330221123322002 2
33333300221123322212 1
22212210022112332221 2
```

- Input is a 3-band RGB color image
- Classify cells in four groups based on the average of the red, green and blue pixels
- Output is a single-band raster with a cell depth of 4 bits.

From	To	Group
0	63	0
64	127	1
128	191	2
192	255	3

```
sdo_geor_ra.classify (
  inGeoraster     => gr1,
  expression      => '({0}+{1}+{2})/3',
  rangeArray      => SDO_NUMBER_ARRAY(63,127,191),
  valueArray      => SDO_NUMBER_ARRAY(0,1,2,3),
  outGeoraster    => gr2,
  storageParam    => 'cellDepth=4BIT'
);
```

Historical Temperature Analysis

A more complex example

- A collection of 33 years of global temperature data for each month. In total, 396 layers stored in one georaster object:

1. The values are in Kelvin, which is converted to Fahrenheit
2. Generate the Average (mean) temperature for each month
3. Compute the Mean Absolute Deviation per Month (the mean difference from the average across 33 years for each month).

- use PL/SQL program to generate all expressions on the left and execute the rasterMathOp operations.

```
-- converting Kelvin to Fahrenheit
'({0}-273.15)*9/5+32'
```

```
-- compute average temperature for a month (January)
'({0}+{12}+{24}+{36}+{48}+{60}+{72}+{84}+{96}+{108}+{120}+{132}
+{144}+{156}+{168}+{180}+{192}+{204}+{216}+{228}+{240}+{252}
+{264}+{276}+{288}+{300}+{312}+{324}+{336}+{348}+{360}+{372}
+{384})/33'
```

```
-- compute mean absolute deviation for a month (January)
'(abs({1,0}-{0,0})+abs({1,0}-{0,12})+abs({1,0}-{0,24})
  +abs({1,0}-{0,36})+abs({1,0}-{0,48})+abs({1,0}-{0,60})
  +abs({1,0}-{0,72})+abs({1,0}-{0,84})+abs({1,0}-{0,96})
  +abs({1,0}-{0,108})+abs({1,0}-{0,120})+abs({1,0}-{0,132})
  +abs({1,0}-{0,144})+abs({1,0}-{0,156})+abs({1,0}-{0,168})
  +abs({1,0}-{0,180})+abs({1,0}-{0,192})+abs({1,0}-{0,204})
  +abs({1,0}-{0,216})+abs({1,0}-{0,228})+abs({1,0}-{0,240})
  +abs({1,0}-{0,252})+abs({1,0}-{0,264})+abs({1,0}-{0,276})
  +abs({1,0}-{0,288})+abs({1,0}-{0,300})+abs({1,0}-{0,312})
  +abs({1,0}-{0,324})+abs({1,0}-{0,336})+abs({1,0}-{0,348})
  +abs({1,0}-{0,360})+abs({1,0}-{0,372})+abs({1,0}-{0,384}))/33'
```

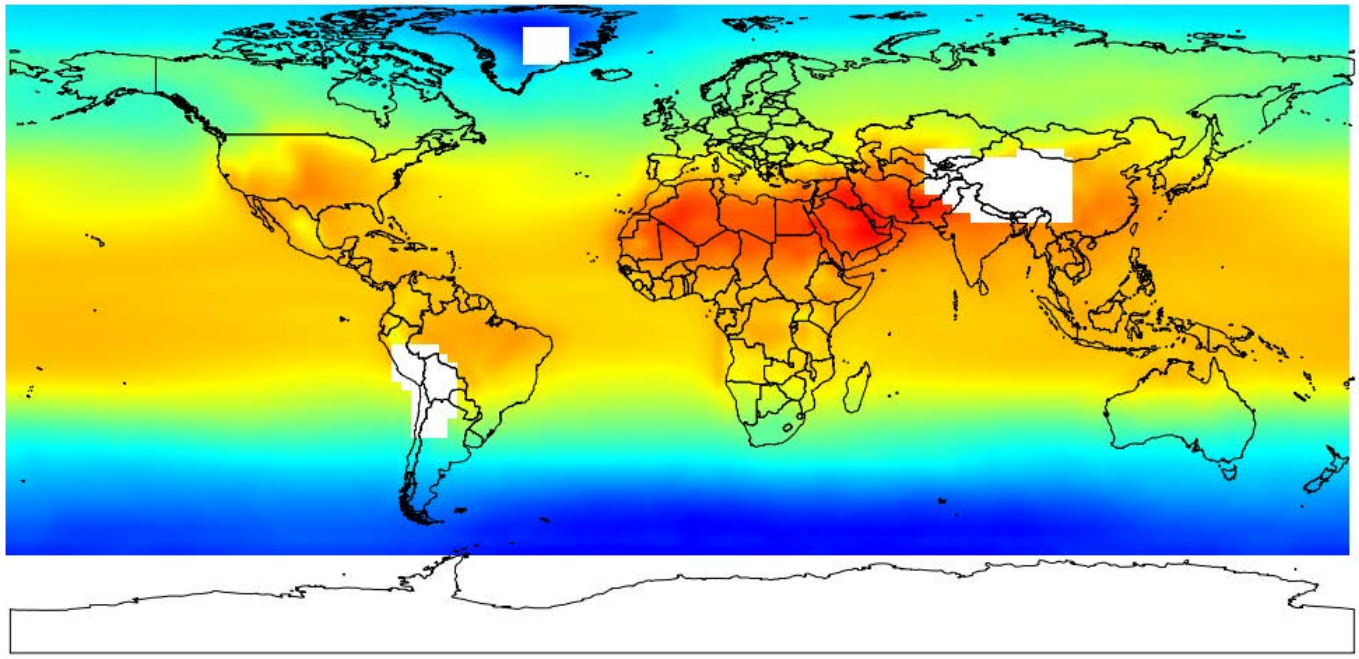
Use PL/SQL to generate expressions


```
operation sdo_string2_array := sdo_string2_array();
```

```
-- Generate raster algebra operations to convert all bands from Kelvin to Fahrenheit
-- One operation per band b
-- Each operation looks like this:
--   ({b}-273.15)*9/5+32
-- where b is the band number, from 0 to 395
operation.Extend(numBands);
for b in 1 .. numBands loop
  operation(i) := '({' || ( b - 1 ) || '}-273.15)*9/5+32';
end loop;
```

```
SDO_GEOR_RA.rasterMathOp(
  inGeoRaster   => gr5,
  operation     => operation,
  storageParam  => null,
  outGeoraster  => gr6,
  nodata        => 'TRUE',
  nodataValue   => -9999
);
```

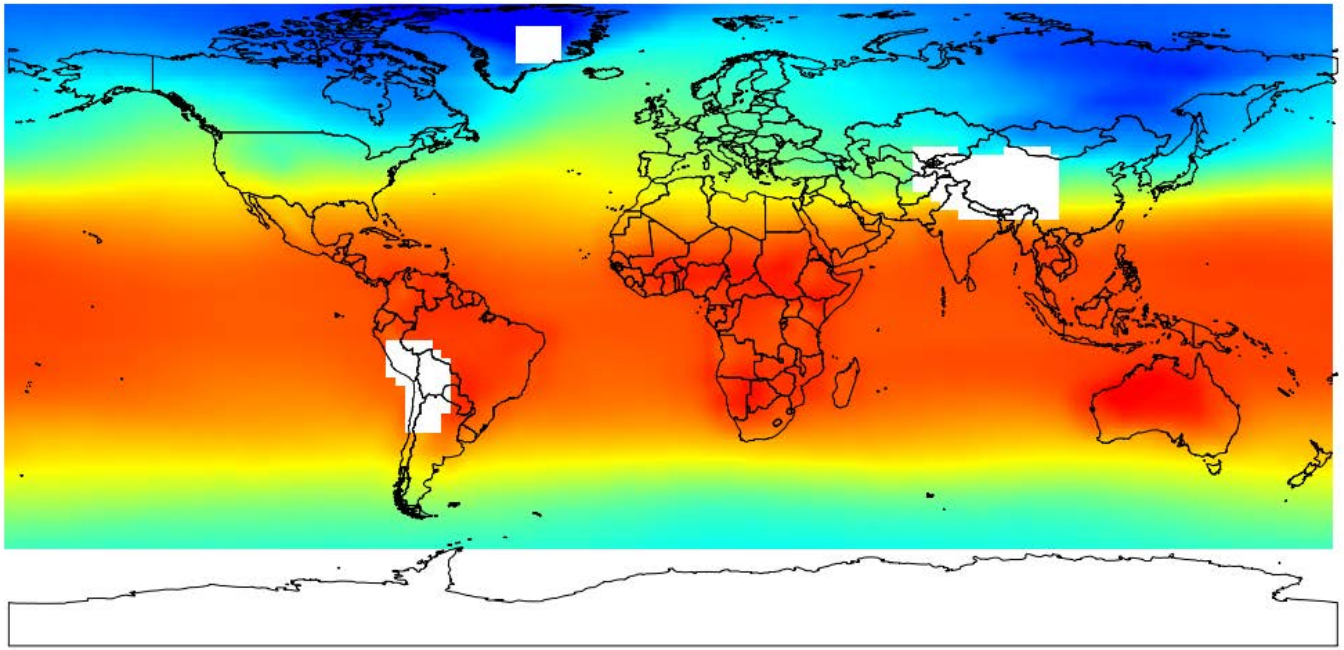
Historical Temperature Analysis: Average



June Average Temperature over 33 Years

(Source Temperature Data Courtesy of Remote Sensing Systems - http://www.remss.com/)

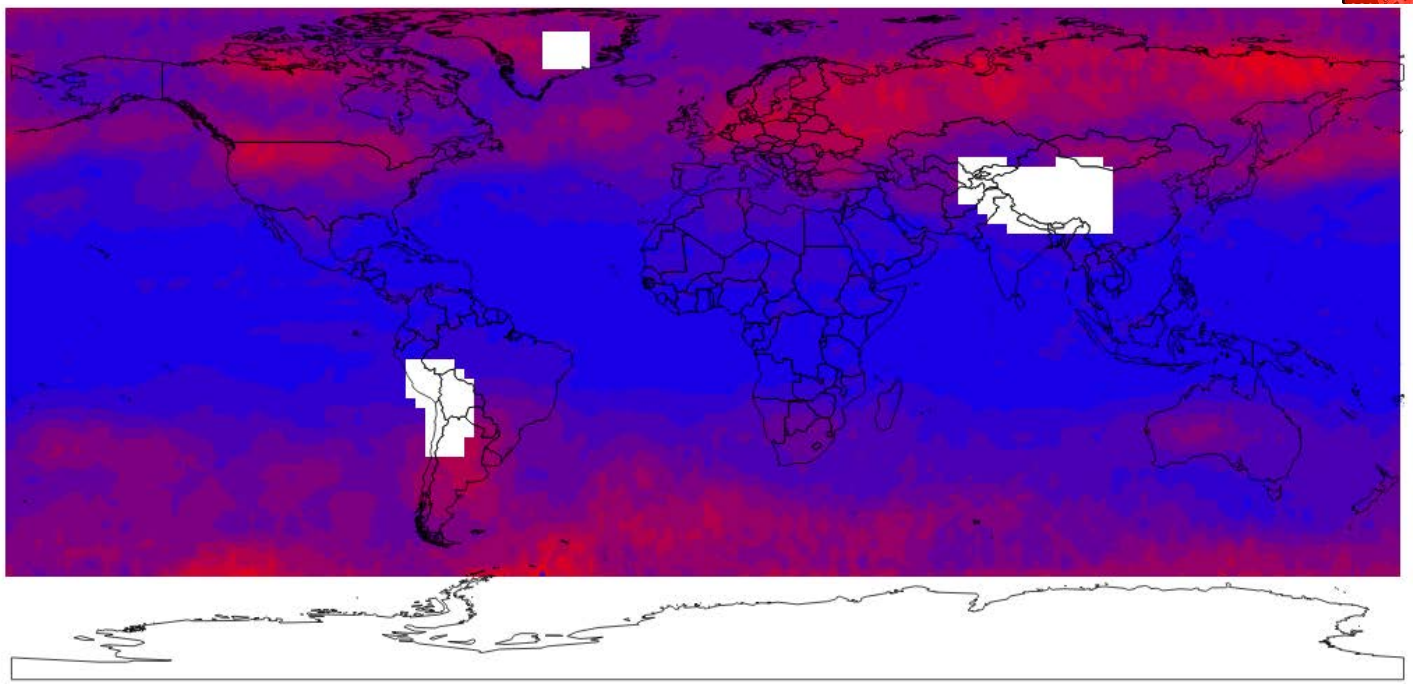
Historical Temperature Analysis: Average



December Average Temperature over 33 Years

(Source Temperature Data Courtesy of Remote Sensing Systems - http://www.remss.com/)

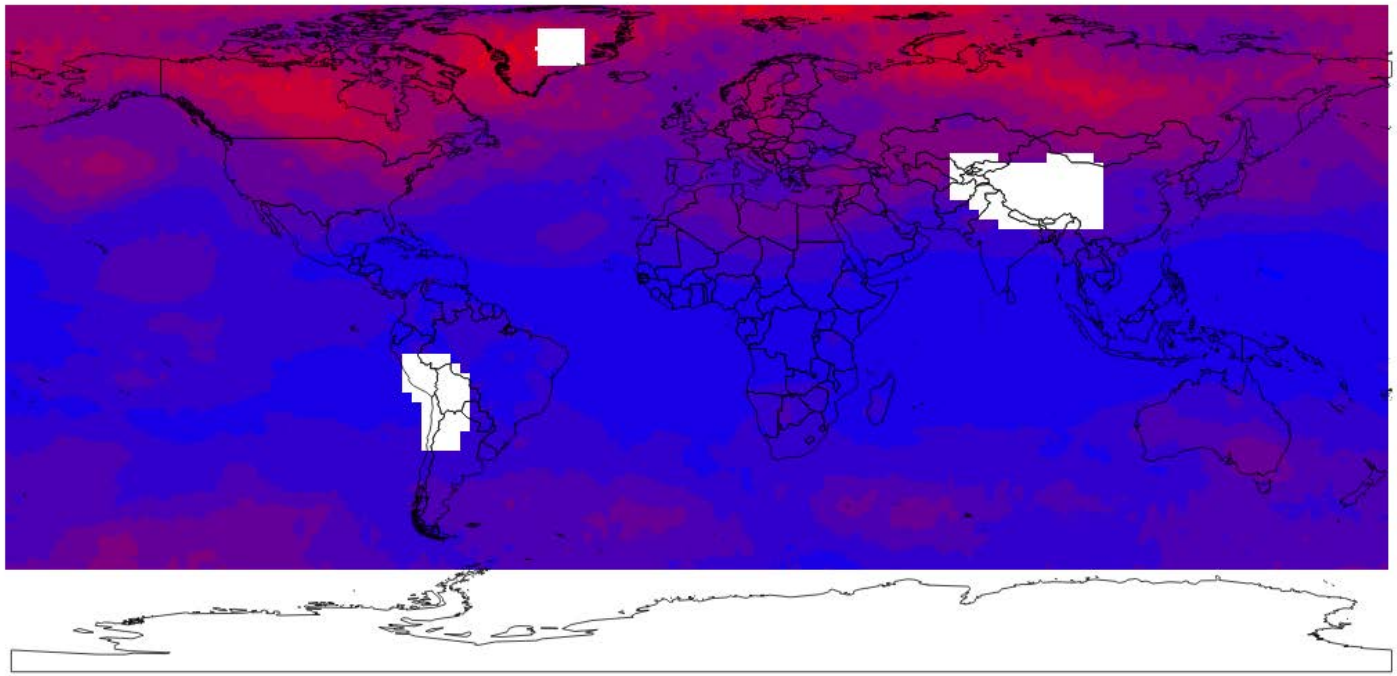
Historical Temperature Analysis: Anomaly



June Temperature Mean Absolute Deviation Analysis

(Source Temperature Data Courtesy of Remote Sensing Systems - http://www.remss.com/)

Historical Temperature Analysis: Anomaly



December Temperature Mean Absolute Deviation Analysis

(Source Temperature Data Courtesy of Remote Sensing Systems - http://www.remss.com/)

NDVI Computation

Normalized Difference Vegetation Index

- A widely used vegetation index, enabling users to quickly identify vegetated areas and monitor the growth and "condition" of plants.
- Using Landsat TM imagery, the standard NDVI computation formula is: (TM4 - TM3) / (TM4 + TM3).

```
sdo_geor_ra.rasterMathOp (
  inGeoraster    => gr1,
  operation      =>
   sdo_string2_array('({3}-{2})/({3}+{2})'),
  outGeoraster   => gr2,
  storageParam   => 'celldepth=32bit_real'
);
```

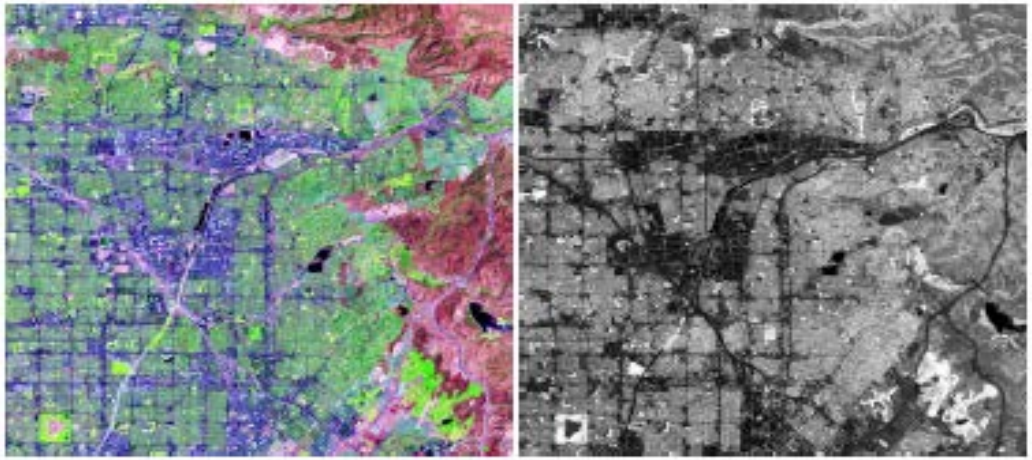
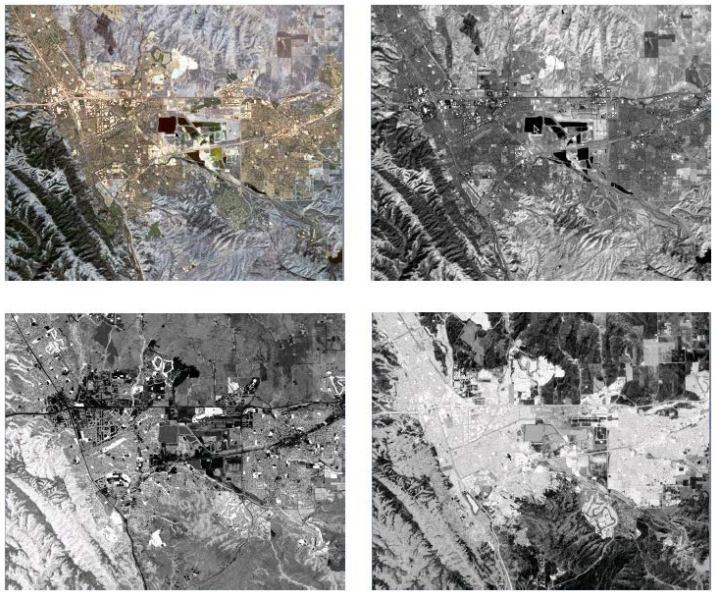


Fig. 1, ETM+ 543 color image (left) and NDVI image (right). Image Courtesy of PCI Geomatics.

Tasseled Cap Transformation (TCT)

- Bands of an image converted into a new set of bands with defined interpretations that are useful for vegetation mapping.
- Each tasseled-cap band is created by the sum of image band 1 * a coefficient + image band 2 * a coefficient etc…
- Coefficients are derived statistically from images and empirical observations.



Source TM 123 Color Image (upper left), Brightness (upper right), Greenness (lower left) and Wetness (lower right). Landsat 5 TM Image Courtesy of the U.S. Geological Survey

Tasseled Cap Transformation

Example


```
sdo_geor_ra.rasterMathOp (
  inGeoraster    => gr1,
  operation      => sdo_string2_array(
   '0.3561*{0}+0.3972*{1}+0.3904*{2}+0.6966*{3}+0.2286*{4}+0.1596*{6}',
   '(-0.3344)*{0}-0.3544*{1}-0.4556*{2}+0.6966*{3}-0.0242*{4}-0.2630*{6}',
   '0.2626*{0}+0.2141*{1}+0.0926*{2}+0.0656*{3}-0.7629*{4}-0.5388*{6}',
   '0.0805*{0}-0.0498*{1}+0.1950*{2}-0.1327*{3}+0.5752*{4}-0.7775*{6}',
   '(-0.7252)*{0}-0.0202*{1}+0.6683*{2}+0.0631*{3}-0.1494*{4}-0.0274*{6}',
   '0.4000*{0}-0.8172*{1}+0.3832*{2}+0.0602*{3}-0.1095*{4}+0.0985*{6}'
  ),
  outGeoraster   => gr2,
  storageParam   => 'celldepth=32bit_real'
);
```

Cartographic Modeling

A Hypothetical Analytical Model

- Assume that a hypothetical cartographic model involves 7 different raster layers (stored in 7 georasters) and has an expression as follows:

```
if (100 < layer1 <= 500) &
   (layer2 == 3 or layer2 == 10) &
   ((((layer3+layer4) * log(layer5) /
sqrt(layer5)) >= layer6) or (layer7 != 1) )
then output = 1
else output = 0
```

```
expr := '((100<{0,0})&({0,0}
  <=500))&(({1,0}=3)|({1,0}
  =10))&((((({2,0}+{3,0})*log({4,0})/
  sqrt({4,0}))>={5,0})|({6,0}!=1))';

sdo_geor_ra.rasterMathOp (
  inGeoRasters  => my_cursor,
  operation     =>
    sdo_string2_array(expr),
  outGeoraster  => gr2,
  storageParam  => 'celldepth=1bit'
  nodata        => 'TRUE',
  nodataValue   => 0,
  parallelParam => 'parallel=4'
);
```

Program Agenda

- Georaster Concepts
- Creating, loading and viewing
- Raster Algebra and Analytics
- Image Processing and Virtual Mosaic
- Performance and Parallelism

Image Processing and Virtual Mosaic

- Advanced Large-Scale Image **Mosaicking**
- Large-Scale Image **Appending**
- **Virtual Mosaic**
- **Rectification** and **Orthorectification**
- Image **Masking**
- Image **Stretching**
- Image **Segmentation**

Advanced Large-Scale Mosaicking

sdo_geor_aggr.mosaicSubset

- ✓A complete new implementation, parallelized and scalable
- ✓Source images can be tables, views or a SQL statement
- ✓Source images can be rectified or unrectified or in different CS
- ✓Support internal resampling, reprojection or rectification
- ✓Supports gaps, no data, and overlapping regions
- ✓Support 8 common point rules (max, min, avg, LATEST, OLDEST ...)
- ✓User defined priority for overlapping regions (Date or SQL ORDER BY)
- ✓Simple color balancing (linear stretch and normalization)

An Example – Mosaic of Landsat Images

Source images:

✓ 31 Landsat 5 Level 1T TM images, from *the U.S. Geological Survey*.

✓ The total size of the image set is 11.9 GB.

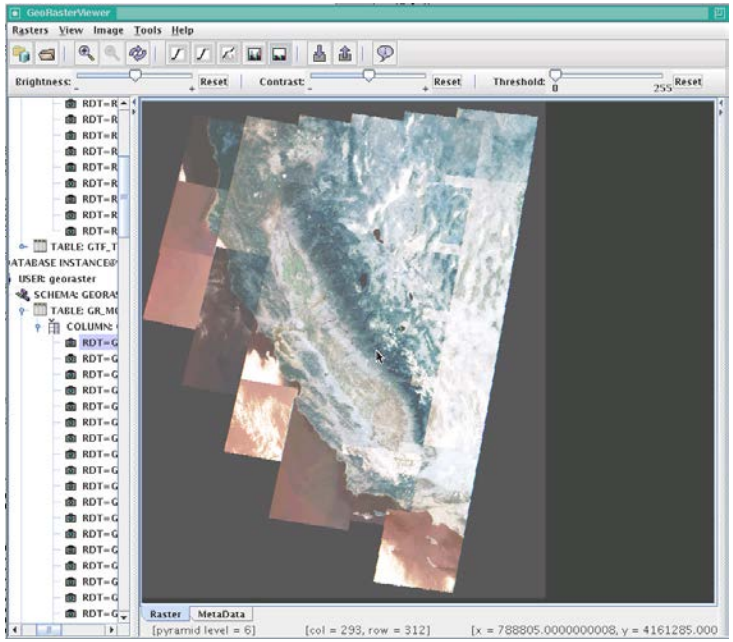
✓ 15 images are in UTM zone 11N

✓ 16 images are in UTM zone 10N projection.

✓They overlap each other.

Output mosaic:

about 7.7GB in size



Large-Scale Appending

sdo_geor_aggr.append to update existing image with new ones

- Used to append or update a large physical mosaic using newly acquired smaller images
- The source image can be on ANY SIDE of the target image!
- Same functionality as mosaicking:
  - Source images can be rectified or unrectified or in different CS
  - Support internal resampling, reprojection or rectification
  - Supports gaps, no data, and overlapping regions
  - Support 8 common point rules (max, min, avg, LATEST, OLDEST, etc...)
  - Simple color balancing (linear stretch and normalization)

Virtual Mosaic

An Advanced Image Query and Image Serving Engine

- A **virtual mosaic** is defined as any large collection of georeferenced GeoRaster objects (images) that is treated as if it is a single GeoRaster object (**physical mosaic**).
- Three ways to define a virtual mosaic:
  - a list of GeoRaster tables
  - a database view with a GeoRaster column
  - a SQL query statement (i.e., a CURSOR)
- A virtual mosaic can contain unlimited number of images of any size
- There is no need to define a description file for the virtual mosaic

Virtual Mosaic

- To query or process the virtual mosaic, use:
  - **sdo_geor_aggr.getMosaicSubset**

    to perform on-the-fly cropping queries over the virtual mosaic
  - **sdo_geor_aggr.mosaicSubset**

    to perform a query and store the mosaicked subset as a GeoRaster object

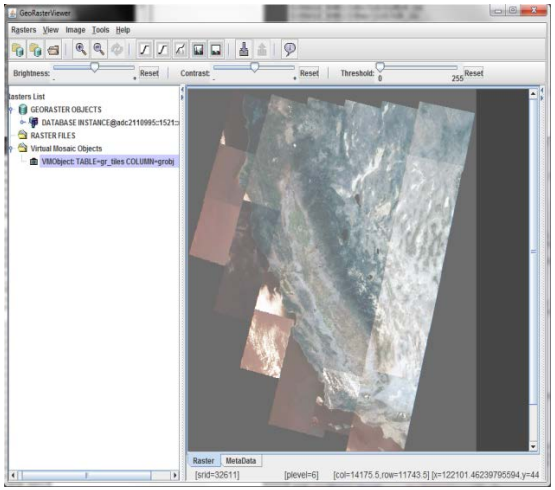
- To facilitate application development, call:
  - sdo_geor_aggr.validateForMosaicSubset
  - sdo_geor_aggr.getMosaicExtent
  - sdo_geor_aggr.getMosaicResolution

Virtual Mosaic

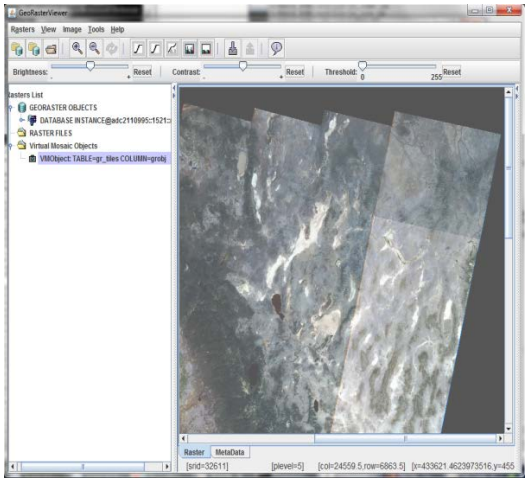
Same functionality as large-scale mosaicking

- Source images can be rectified or unrectified or in different CS
- Support internal resampling, reprojection or rectification
- Supports gaps, no data, and overlapping regions
- Support 8 common point rules (max, min, avg, LATEST, OLDEST, etc...)
- User defined priority for overlapping regions (Date or SQL ORDER BY)
- Simple color balancing (linear stretch and normalization)

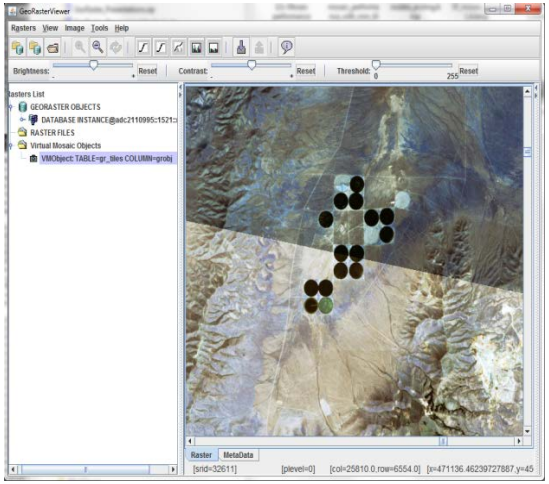
An Example - Virtual Mosaic



Overview



Zoom In



Zoom In Further

(Image Courtesy of the U.S. Geological Survey)

Some Virtual Mosaic Use Cases

- ✓DOQs can be stored as is - but used as seamless mosaic (as fast as physical mosaics)
- ✓Users may not want to mosaic DEM's
- ✓Store large volume of imagery without making too many extra copies (save disk spaces)
- ✓New images coming in and immediately displayed on the mosaic (dynamic updates)
- ✓The same images can be displayed or removed in different virtual mosaics (flexible model)

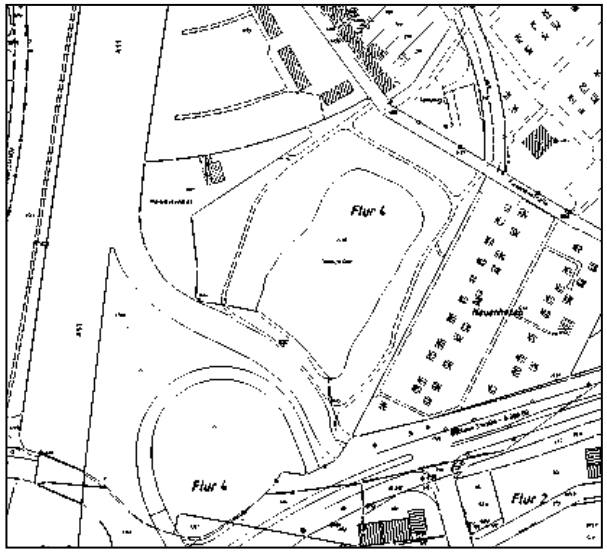
Image Rectification and Orthorectification

- **Rectification** of georeferenced raw images
- **Orthorectification** of georeferenced raw image using DEM
- Both support on-the-fly cropping queries or for persistent storage

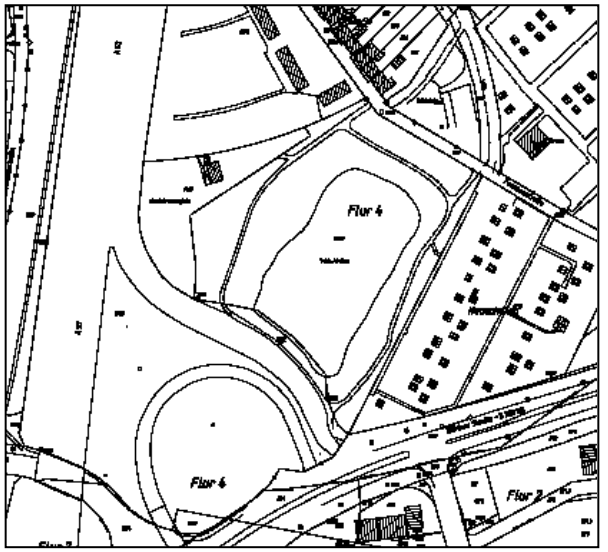
(Image Courtesy of Digital Globe)

New Bitmap Pyramiding

- A new algorithm to pyramid bit raster maps
- Preserve point or line features and avoid dash lines
- Useful for rasterized point, line or polyline features



Old Pyramid
(with missing and dash lines)



New Pyramid
(no dash lines)

(Map Data Courtesy of G.ON)

More Image Processing

- Image **Stretching**
- Image **Segmentation**
- Image **Masking**

Program Agenda

- Georaster Concepts
- Creating, loading and viewing
- Raster Algebra and Analytics
- Image Processing and Virtual Mosaic
- Performance and Parallelism

GeoRaster Performance Improvement

- **New Algorithms**
  - Improved Resampling Algorithms, particularly Cubic Convolution
  - More Efficient Memory Management

- **Parallel Processing**
  - Parallel Execution of SQL Statements
  - Parallelized GeoRaster Procedures

New Algorithms Improve Performance

Comparing with 11g and without parallelism

Geo Raster Performance Improvements

- generatePyramid Cubic resampling: 3x
- scaleCopy with Cubic resampling: 2.5x
- reproject with Cubic resampling: 2.5x
- mosaicSubset: 1.15x to several times (*)

(*) depending on image overlapping

Parallelized GeoRaster Operations

Most raster functions can parallelize

SERIAL Operations perform up to 3x faster

Scale to over 100 times faster on highly parallel systems

Mosaicking Performance

Comparing with File-System Based GDAL Mosaicking

Test Data: 31 Landsat TM images. Total size 11.9GB

Test Machine: x4170 M2 Sun Server with 24 CPUs and 8 SAS disks.

GeoRaster is much faster:
- ✓ serial: 2.3x faster
- ✓ parallel: 11.3x faster

GDAL vs. GeoRaster Image Mosaicking Times in Minutes

	Serial	Parallel
GDAL	43.12	42.1
GeoRaster	18.65	3.73

Parallelized Mosaicking Performance

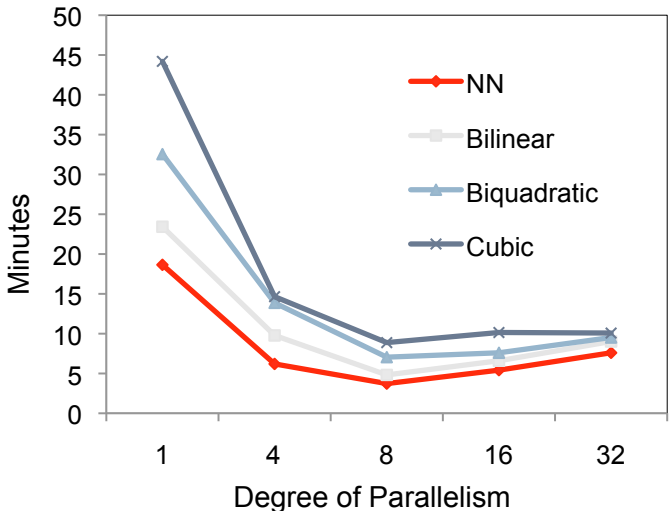
Test Data: 31 Landsat TM images. Total size 11.9GB

Test Machine: x4170 M2 Sun Server with 24 CPUs and 8 SAS disks.

Speed up: up to 5x faster than serial
- ✓ 18.65m down to 3.75m (NN)
- ✓ 44.2m down to 8.88m (Cubic)
- ✓ many more times faster

(if more overlapping)

Parallelized Mosaicking with Different Degrees of Parallelism



Parallelized Pyramiding Performance

Test Data: 31 Landsat TM images. Total size 11.9GB

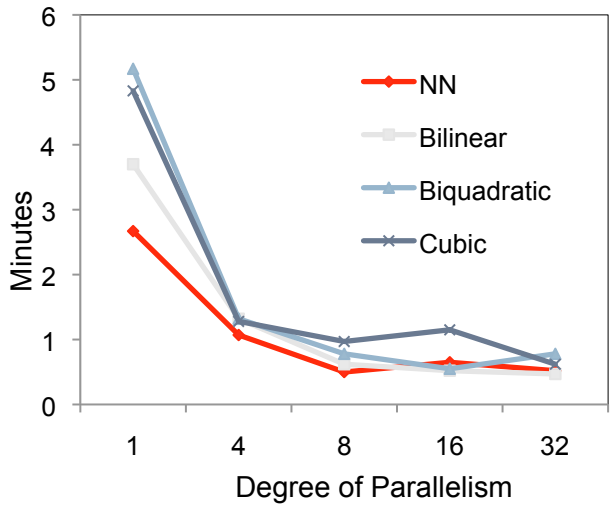
Test Machine: x4170 M2 Sun Server with 24 CPUs and 8 SAS disks.

Speed up:

Up to 10x faster than serial takes only 0.47m to 0.62m

about 30x faster than 11g

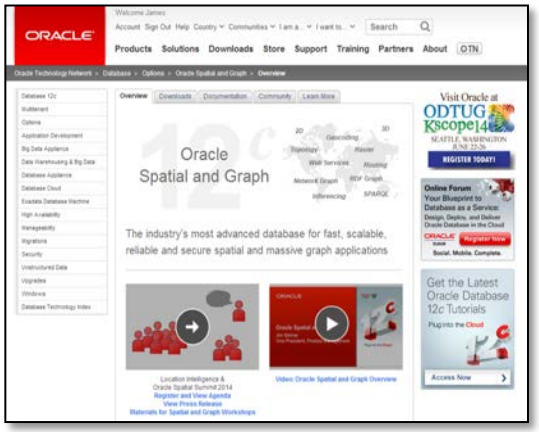
Parallelized Pyramiding with Different Degrees of Parallelism

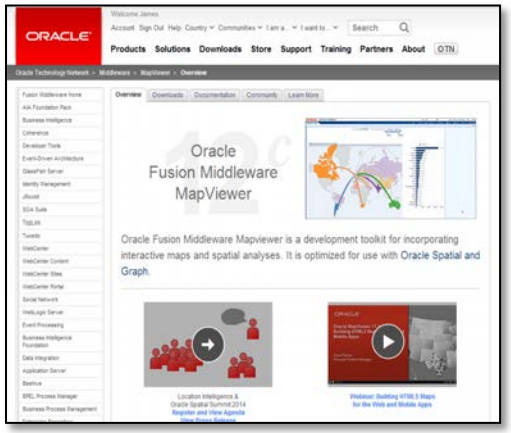


Resources

Resources

Oracle Technology Network

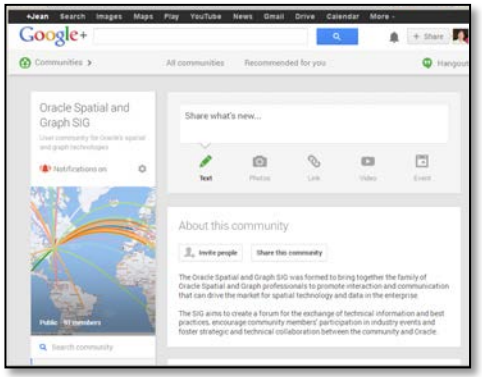




- www.oracle.com/technetwork/database/options/spatialand graph
- www.oracle.com/technetwork/middleware/mapviewer
- blogs.oracle.com ➔ oraclespatial ➔ oracle_maps_blog

Oracle Spatial & Graph Special Interest Group

Connect and exchange knowledge with the community of Spatial & Graph users



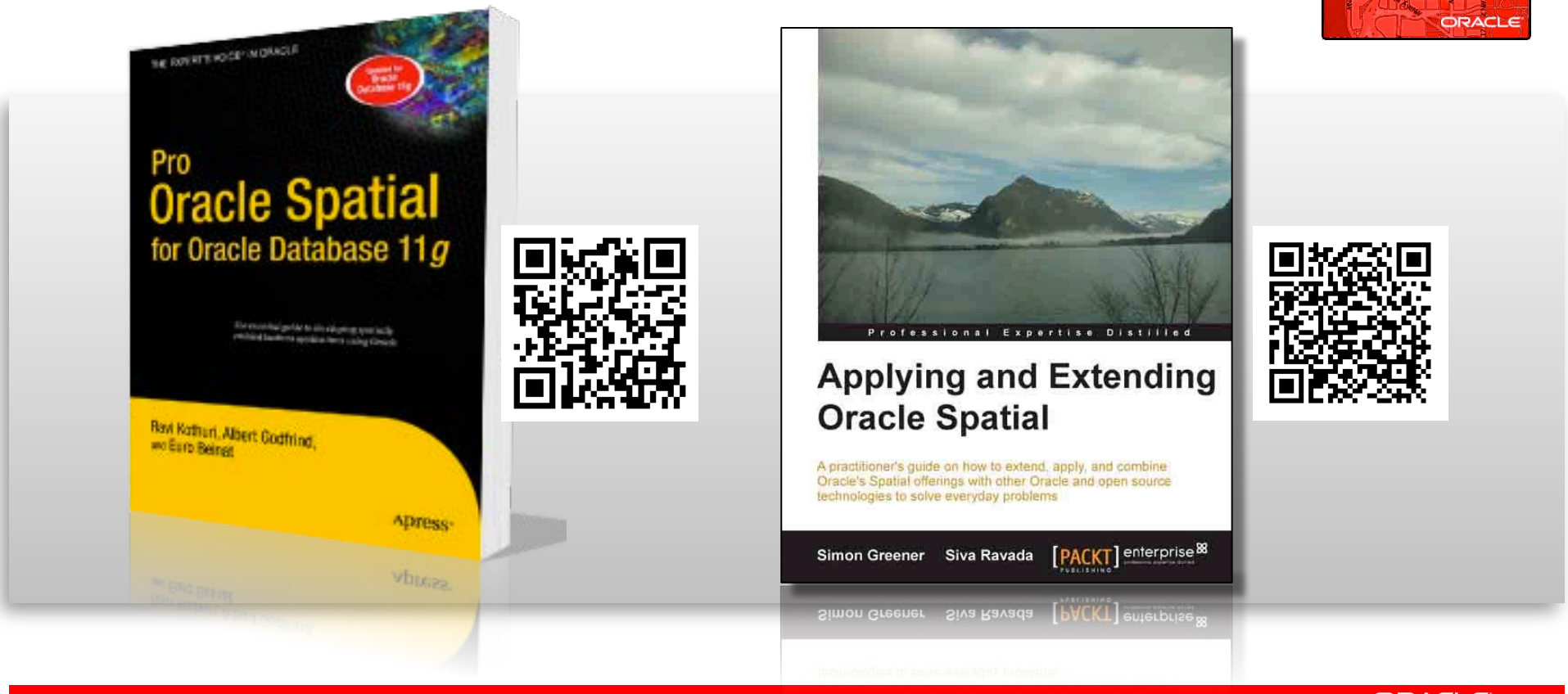
- **Talk with the Board this week**
  - Wednesday lunch – SIG Board presentation (150AB)
  - Stop by the SIG User Group roundtable at Meet the Experts, 4:30pm Wednesday in150AB
  - Visit Oracle's exhibitor table at breaks & sign up
- **Join us**
  - Online communities: LinkedIn , Google+ , IOUG SIG (free membership)
  - Visit OTN Spatial Community page www.oracle.com/technetwork/database/options/spatialandgraph/community (or search online for "Oracle Spatial and Graph Community")
  - Email oraclespatialsig@gmail.com

Spatial Certification & Partner Specialization

Get valuable credentials – differentiate your skills

- **Learn more at the Summit**
  - Wed, Track C 3:30 – Exam preparation session
  - Talk to us at Oracle's exhibitor table & "Meet the Experts" Certification table (Wed 4:30-5:00)
- **Take the next steps**
  - Schedule an exam, access topic lists / online training, learn about Partner Specialization requirements www.oracle.com/technetwork/database/options/spatialandgraph/learnmore/spatial-specialization-1835642.html
  - Online training materials for Certified Implementation Specialist exam https://competencycenter.oracle.com/oncc/full_glp.cc?group_id=22003

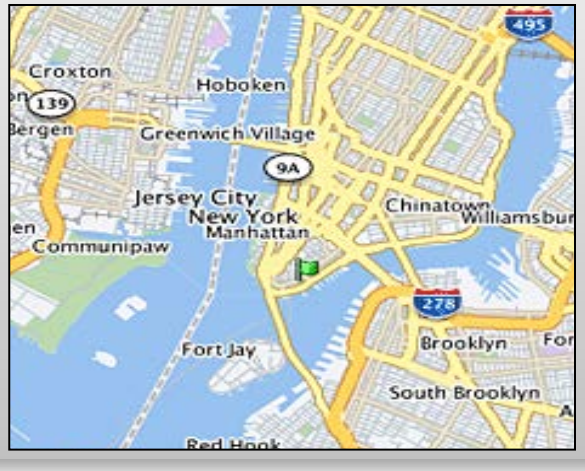
More Resources



MapViewer in Action

Oracle eLocation Services

http://maps.oracle.com/elocation



ORACLE

May 21, 2014
Walter E. Washington Convention Center
Washington, DC USA