Next generation of raster support for the GeoTools-GeoServer stack

Dott. Ing Simone Giannecchini (GeoSolutions),
Ing. Alessio Fabiani (GeoSolutions),
Dott. Ing. Daniele Romagnoli (GeoSolutions),
PhD, Elena Camossi (University College of Dublin)

FOSS4G07, Victoria
September 26th, 2007

Outline

- Past achievements
- GDAL-ImageIO integration through ImageIO-ext
- RasterSymbolizer
- PostGISRaster
- Multidimensional spatiotemporal coverage support

Past Achievements

- *GeoTools* improved coverage I/O (through JAI-Image I/O)
  - Tiling
  - Overviews
  - Decimation on reading
  - Mosaicking
  - Compression
- GeoServer coverage support for WMS
- Brand-new GeoServer WCS 1.0 service (OGC Compliant)
- GeoServer WMS raster output formats (through JAI-Image I/O)
- GeoServer WMS on the fly palette generation and color inversion
- Base statistics operations for GeoTools
  - Extrema (suppor for spatial ROI)
  - Histogram (suppor for spatial ROI)

GDAL – JAI Image I/O Integration

- **SUN JAI** (High performance image processing library):
  - Deferred Execution Model + Operation chaining (Load Data only when need it)
  - Image Tiling + Tile Caching + Tile Scheduling
  - Many ready to use high performance operations
  - Native accelerated operations available for various platforms
- **SUN Image I/O:**
  - Pluggable and extensible architecture
  - Reader/writer plugins
  - Metadata management
- **SUN JAI – Image I/O**
  - ImageRead operation ties ImageReaders to JAI processing
  - ImageWrite operation ties ImageWriters to JAI processing
- **GDAL**
  - Do I need to talk about it?

Image I/O - Architecture

- **Pluggable Architecture: SPIs and plugins**

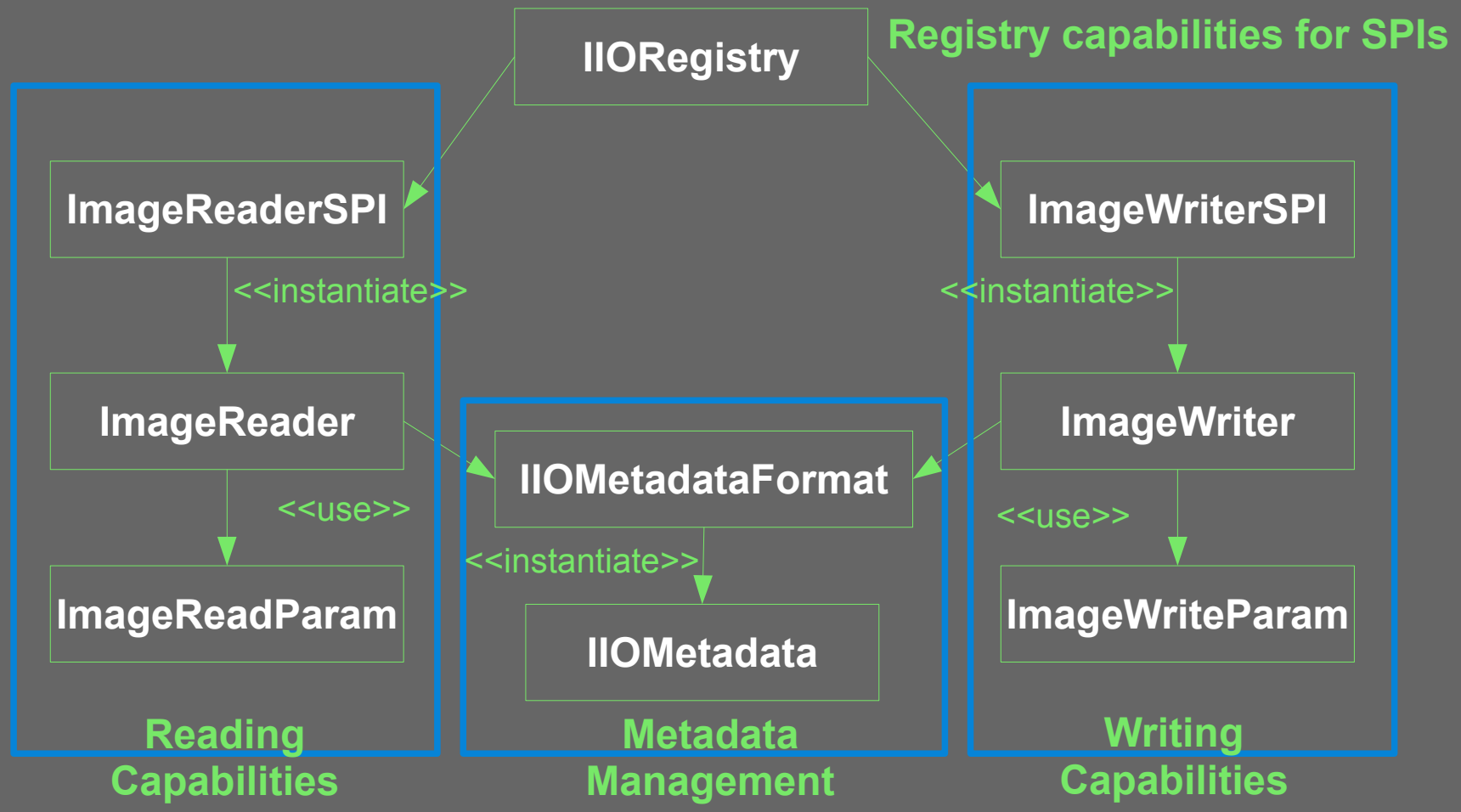


Image I/O – Read/Write Concepts

- Subsampling (decimation on read-write)
- Tiling
- MultiImage file access (i.e. Tiff)
- SourceRegion/DestinationRegion selection
- SubBand selection
- Compression/Decompression
- Progress Listener capabilities

Image I/O - Metadata

- Image Metadata & Stream Metadata
  - IOMetadata vs IIOMetadataFormat
  - Multiple sets supported
  - XML DOM vs Java classes
- Transcoding

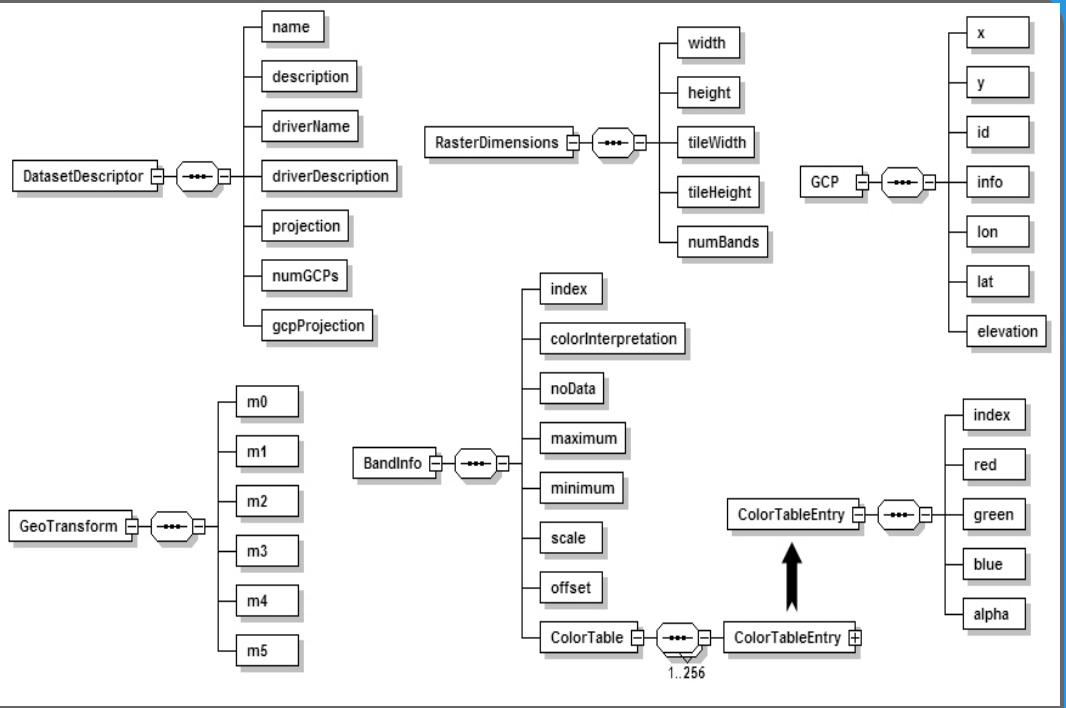


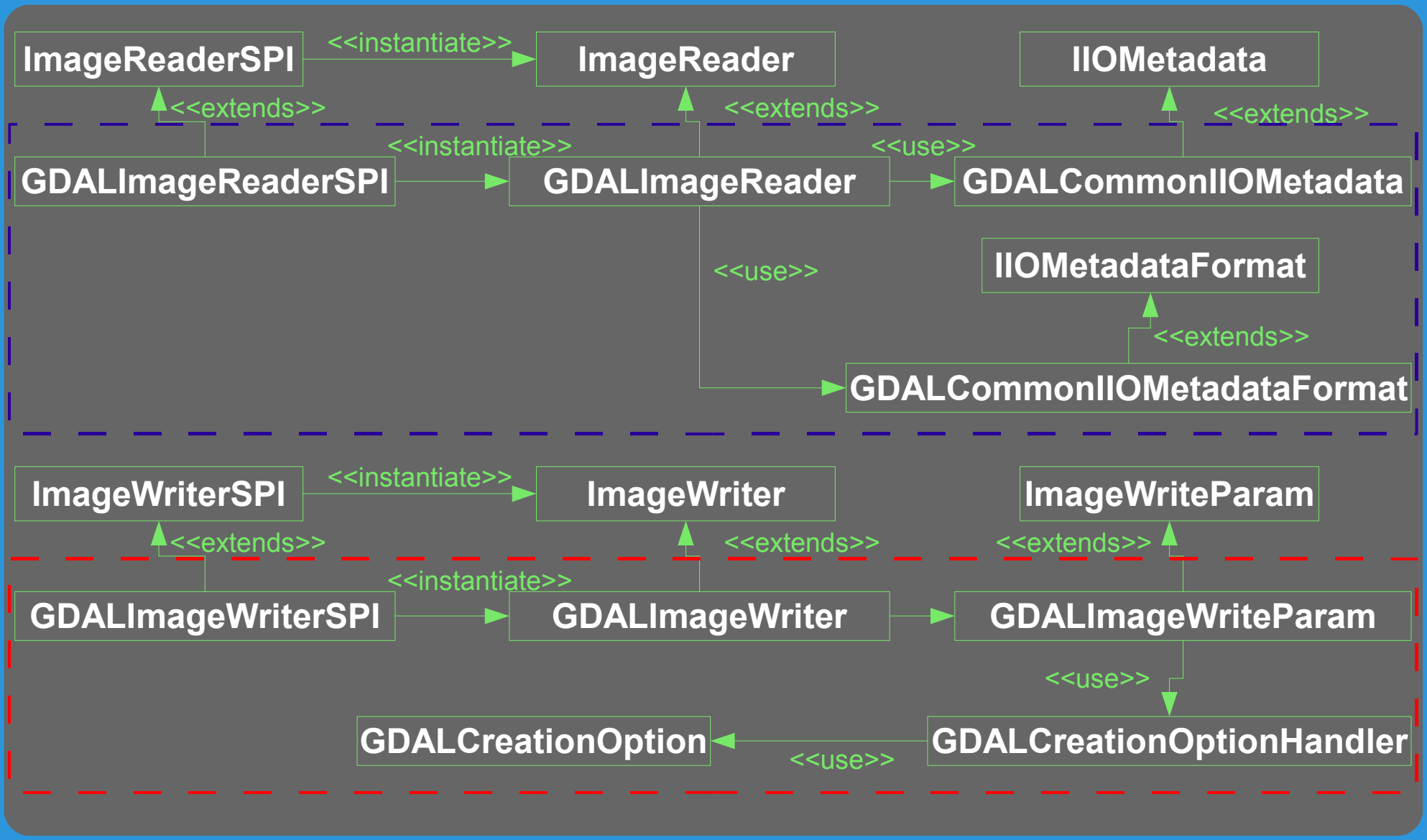
Image I/O-ext

- **Extends SUN's Image I/O architecture**
- **Core libraries**
  - Gdalframework
  - Customstreams
  - Imagereadmt
  - Jhdf4
  - Grib1 (spike)
- **Plugins**
  - Jpeg (gdal), geotiff(gdal), jp2kakadu (gdal), directkakadu, ecw (gdal), mrsid (gdal), arcgrid (gdal), javaarcgrid (arcgrid), hdf4 (gdal), swan
- https://imageio-ext.dev.java.net/

GDALframework

- Started as a GSOC 06 project by Daniele Romagnoli
- Adapt GDAL SWIG Java bindings model to the SUN's Image I/O architecture to
  - GDAL driver ⟷ Image I/O Reader/Writer
  - Translates Image I/O semantics into GDAL semantics
  - Acts as a base framework for GDAL oriented plugins
- Easily create new ImageReader/ImageWriter from existing GDAL driver
- Few improvements to the GDAL bindings
  - Multiband reads (Dataset read vs RasterBand read)
  - PixelInterleaved compliant reads (rendering)
  - Available as a patch

GDALframework - Architecture

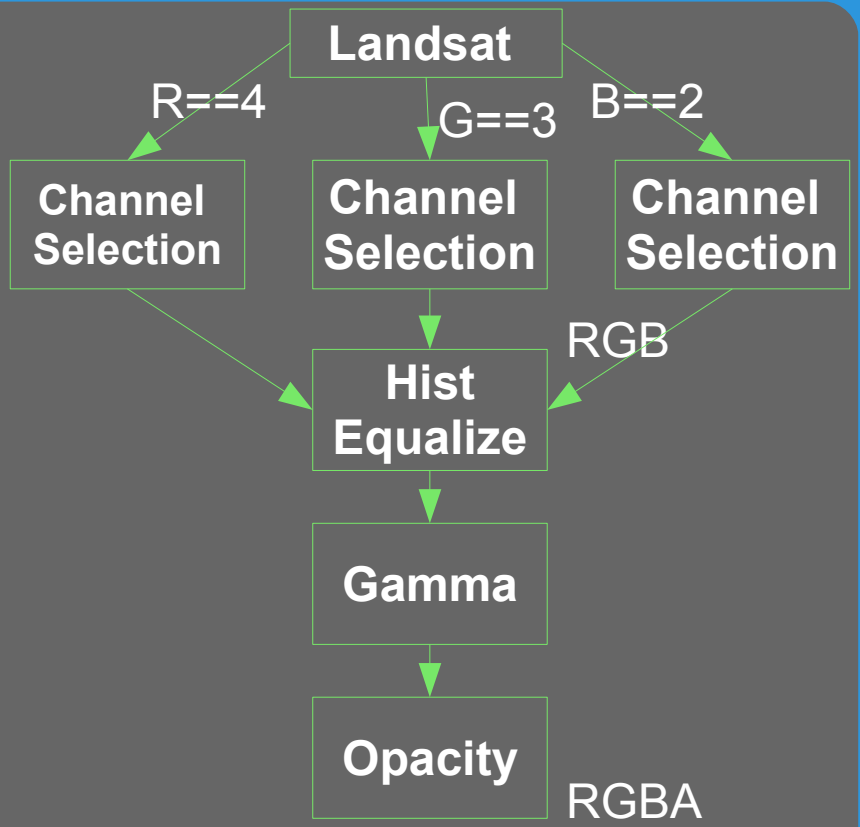


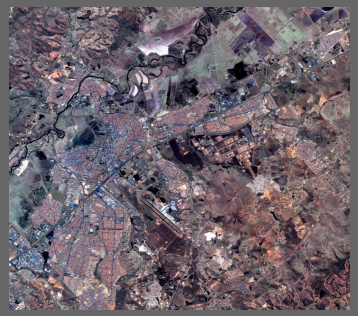
RasterSymbolizer

- SLD 1.0
- Goals
  - Render raw data
    - DEM, scientific models
    - Multispectral rasters
  - Enhance contrast
  - Sub-band selection
  - Opacity management
  - Mosaicking order management
- Graph (directed acyclic graph) of operations
- http://svn.geotools.org/geotools/branches/2.4.x_rs/

RasterSymbolizer

- Geometry
- Opacity
- ChannelSelection
  - (from whatever to RGB or Gray)
- OverlapBehavior
- ColorMap
- ContrastEnhancement
  - (Histogram Enhancement, Gamma correction)
- ShadedRelief
- ImageOutline



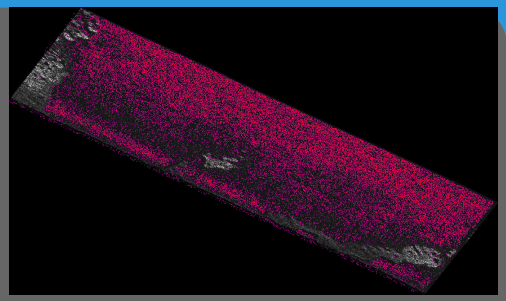


RasterSymbolizer - ColorMap

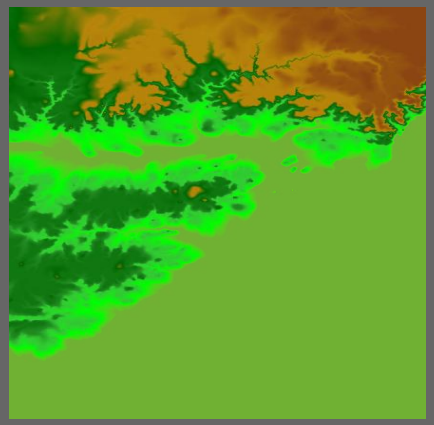
- Apply color maps (palette) to single band raw data throw intervals specification (*categories*)
- Honour input NoData
- Be tolerant with gaps in input *categories*
- Be tolerant with NoData overlapping “real” *categories*
- Support 255 or 65535 colors
- Support opacity on single color map entries
- Allow users to map:
  - Intervals to color ramps
  - Intervals to single color (EXTENSION)
  - Single values to single colors (EXTENSION)

RasterSymbolizer - ColorMap

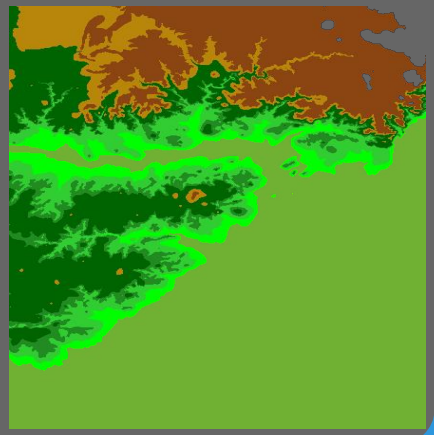
```
<ChannelSelection>
        <Gray><SourceChannelName>1</SourceChannelName></Gray>
</ChannelSelection>
<ColorMap type="values" extended="false">
        <ColorMapEntry  color="#ff0000" quantity="20" opacity="1.0"/>
</ColorMap>
```



```
<ChannelSelection>
        <Gray><SourceChannelName>1</SourceChannelName></Gray>
</ChannelSelection>
<ColorMap type="ramps" extended="false">
        <ColorMapEntry color="#7cfc00" quantity="50" opacity="1.0"/>
        ....
        <ColorMapEntry color="#8b4513" quantity="1000" opacity="1.0"/>
</ColorMap>
```

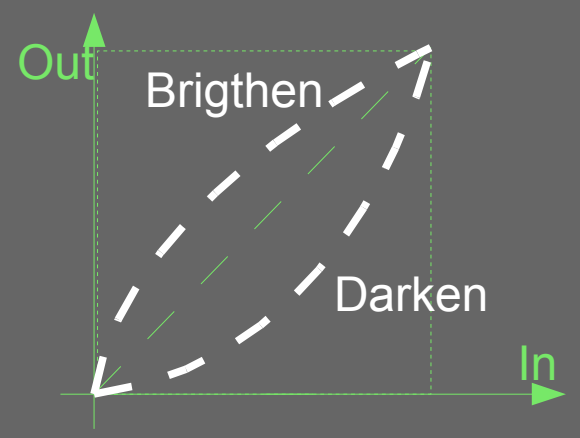


```
<ColorMap type="intervals" extended="false">
        <ColorMapEntry color="#7cfc00" quantity="50" opacity="1.0"/>
        .....
        <ColorMapEntry color="#8b4513" quantity="1000" opacity="1.0"/>
</ColorMap>
```



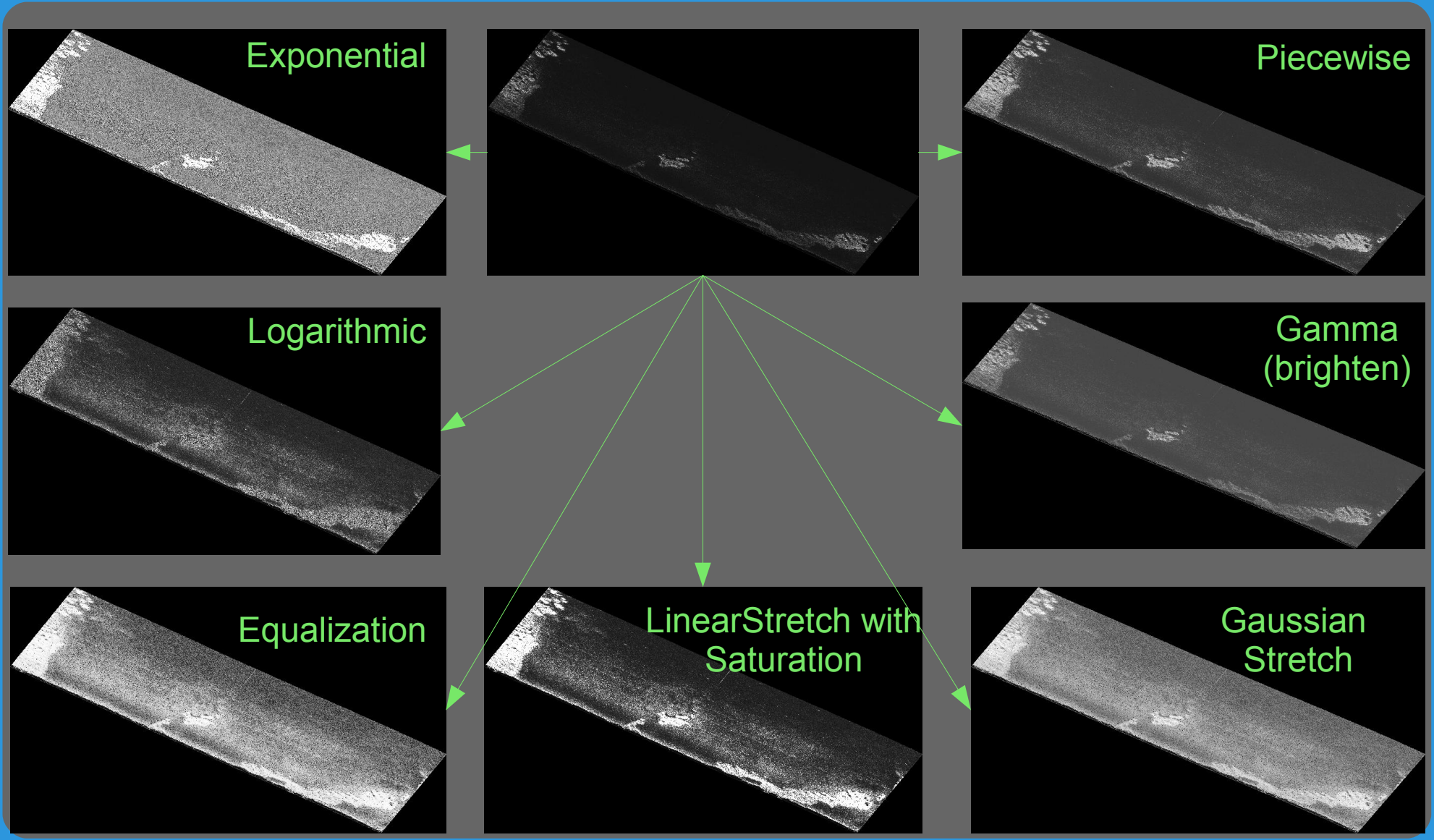
RasterSymbolizer – Contrast Enh.

- Enhance contrast to:
  - Highlight hidden ***features***
  - Improve visual experience
- Histogram enhancement, various algorithms:
  - Histogram equalization
  - Histogram normalization (*contrast stretch*)
  - Many others...
- Gamma Correction



*Limitation,*we work only with 8 bits data

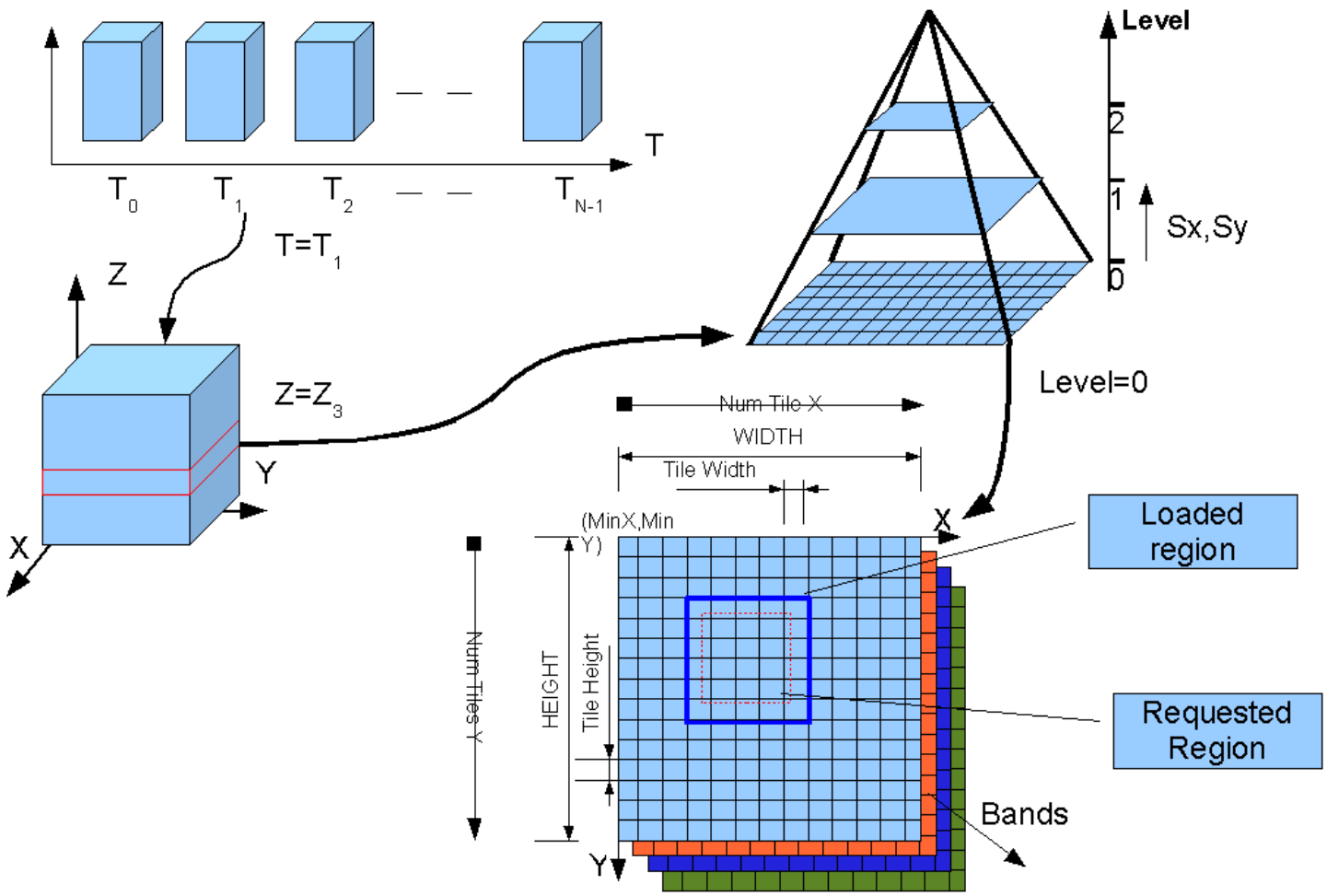
RasterSymbolizer – Contrast Enh.



PostGISRaster

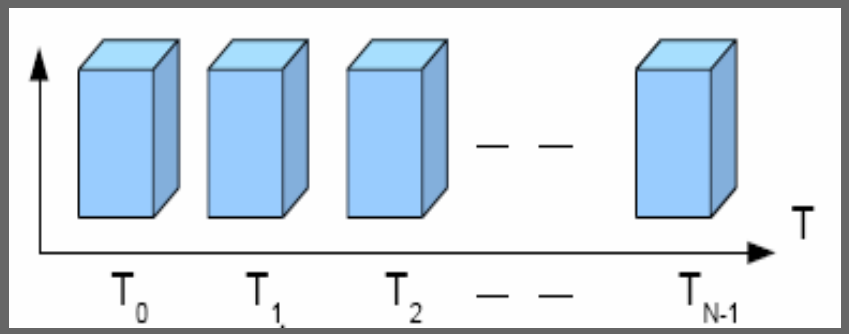
- Support for high resolution raster with overviews
- Support for complex spatiotemporal raster (nD data)
- Why using a DBMS:
  - Exploiting existing support for metadata (ISO 19115, 19139)
  - Exploiting existing spatial and temporal indexing
  - Investigate data ingestion option (open issue)
- Producing schema and I/O classes
- Producing (robust) tools to preprocess data
- Integration with PostGRID project (http://seagis.sourceforge.net)
- ~~Why PostGIS?~~

PostGISRaster

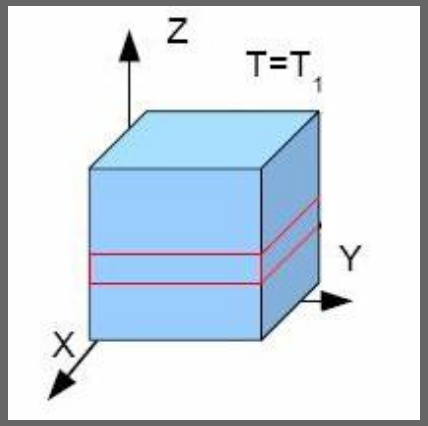


PostGISRaster – T and Z

- **Time Management**
  - Sequence of 2.5D snapshots
  - Temporal ranges [T1, T2] with Nearest Neighbor interpolation
  - B-Trees (logarithmic access) to improve performances

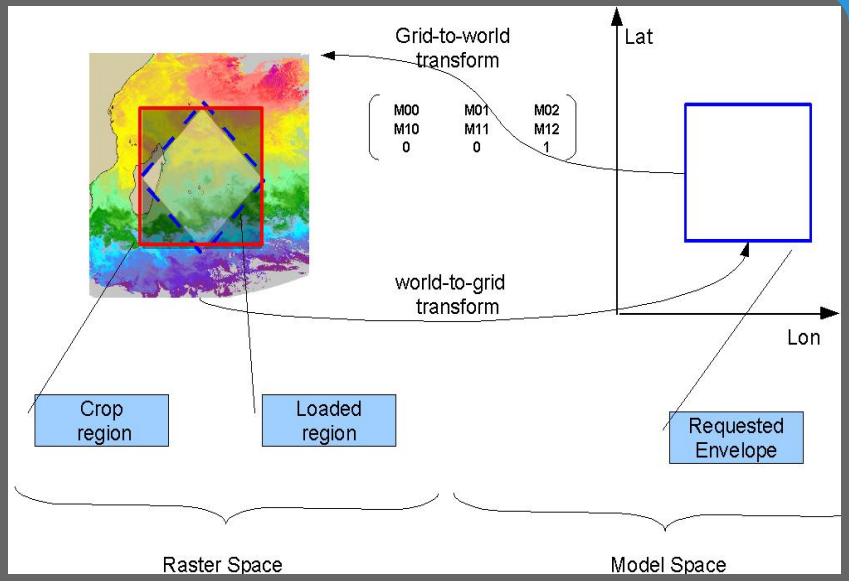


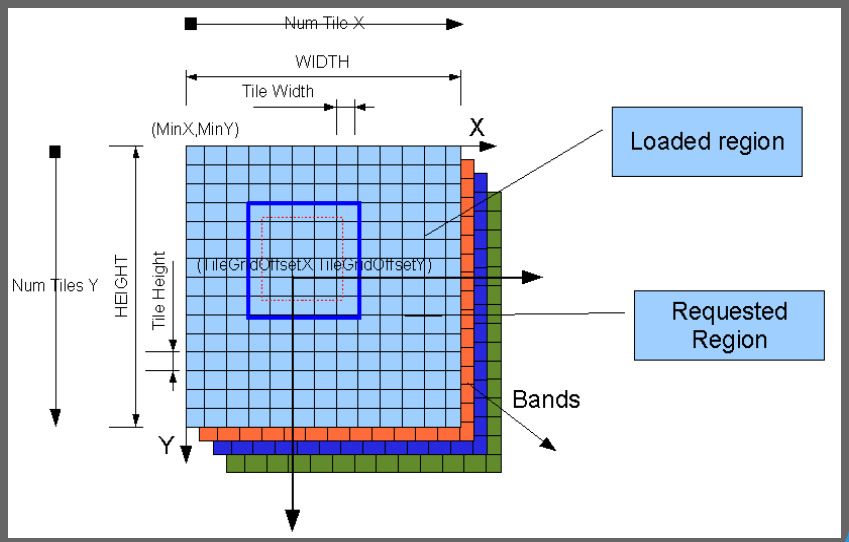
- **Elevation Management**
  - Sequence of 2D layers
  - Positions values with Nearest Neighbor interpolation
  - B-Trees (logarithmic access) to improve performances



PostGISRaster – 2D slice

- Create a regular tessellation of original data
  - Work in raster space
  - Save georeferencing
  - Save original translation factors
- Tile integer indexing
  - Optimize most frequent case (TMS like approach)
  - Simplified tile lookup (no DBMS query)
  - Integer arithmetic + Recomposition by simple translation ⟺ <u>High Perf</u>
  - Tile Caching higly simplified
- Multiband Tiles
- Consider MetaTiling
  - Reduce file access

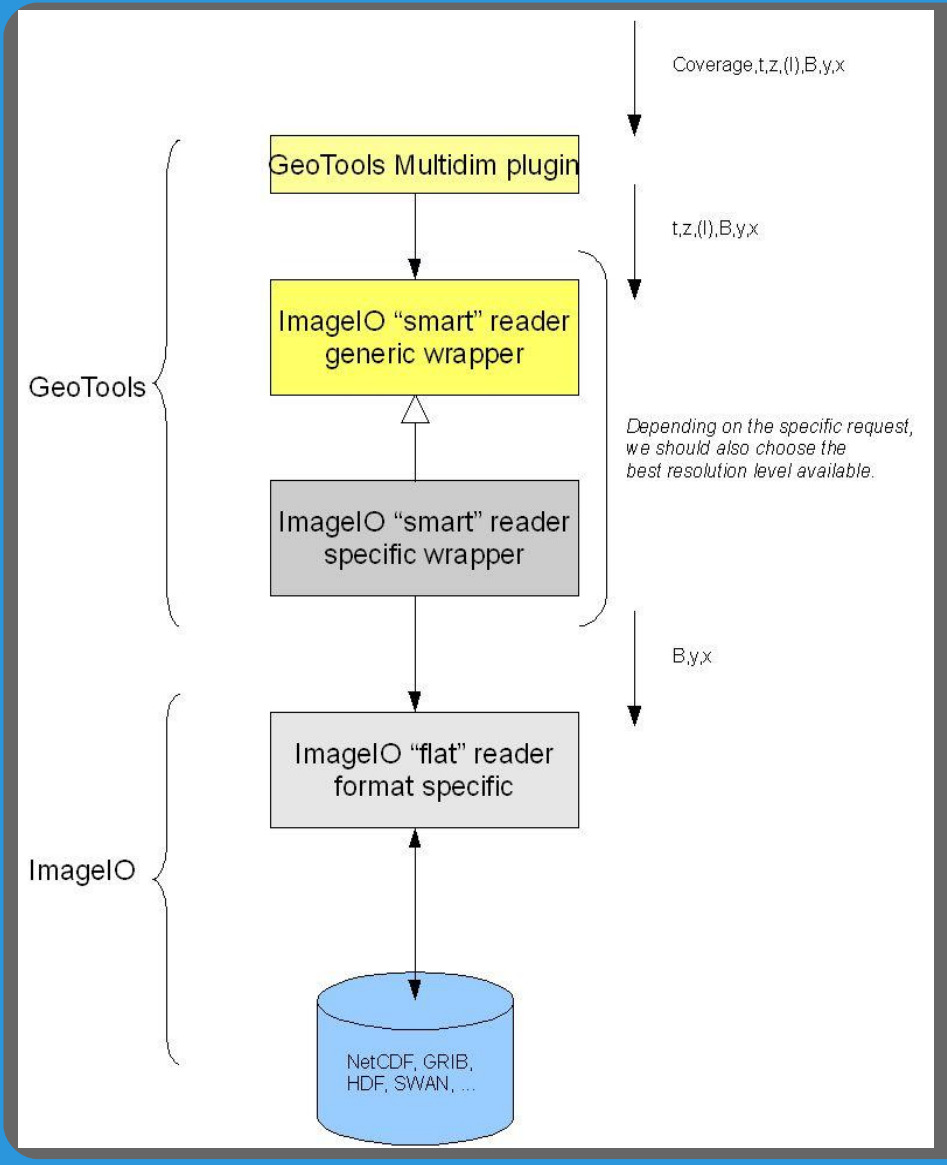


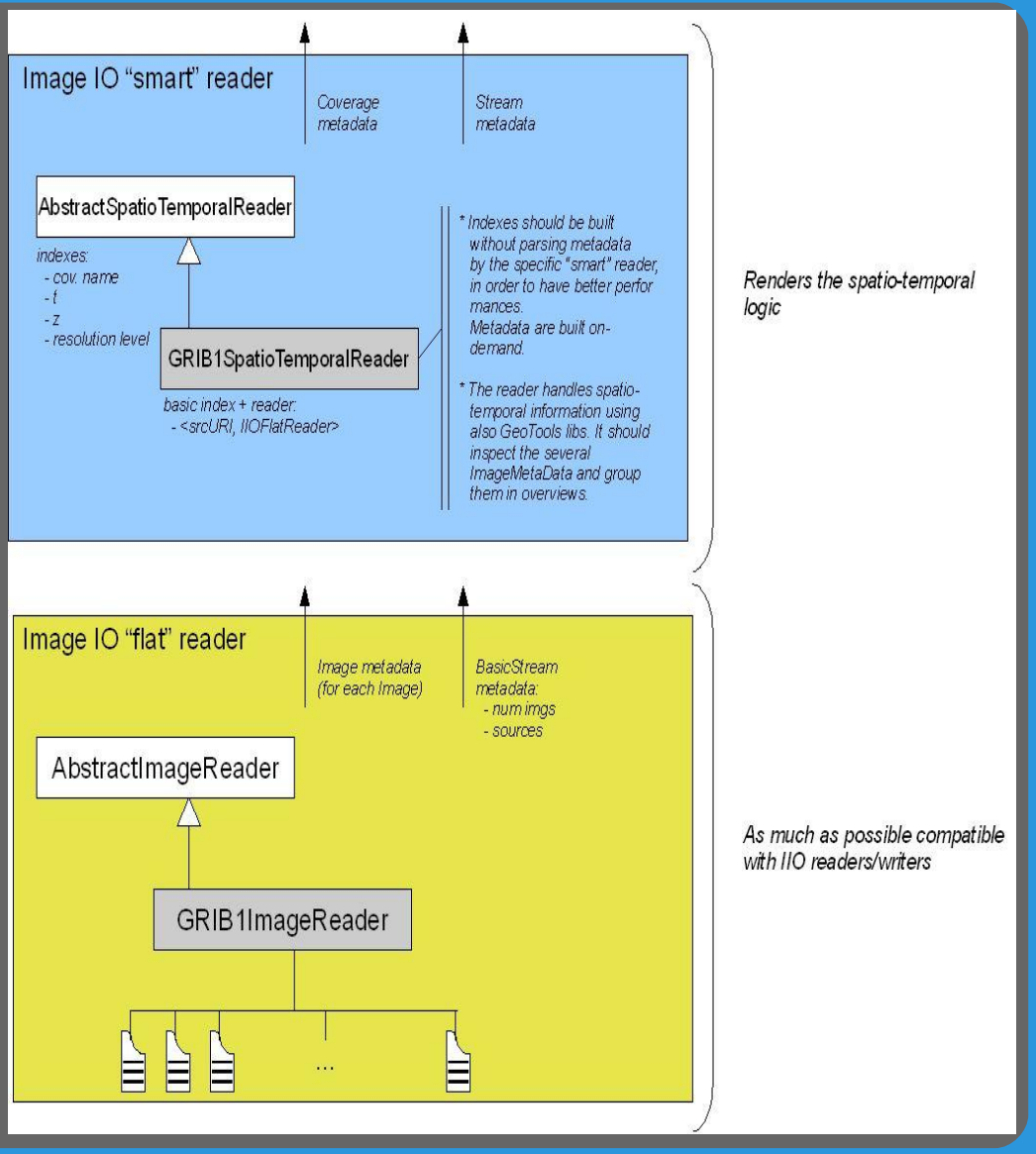


nD Coverage Support

- Supporting coverage with complex domain (t,z,l,b,y,x) and complex codomain (e.g. Multispectral data).
- 2 levels of work
  - Well-known formats
    - Hdf4, HDF, Grib1, SWAN, COAMPS (GeoSolutions)
    - NetCDF, HDF (Geomatys)
  - Metadata and indexing (common middleware)
    - Cooperation GeoSolutions – GeoMatys
- Goals:
  - GeoServer WCS 1.1 with full spatiotemporal support
  - GeoServer WMS 1.1.1 with full temporal and elevation support

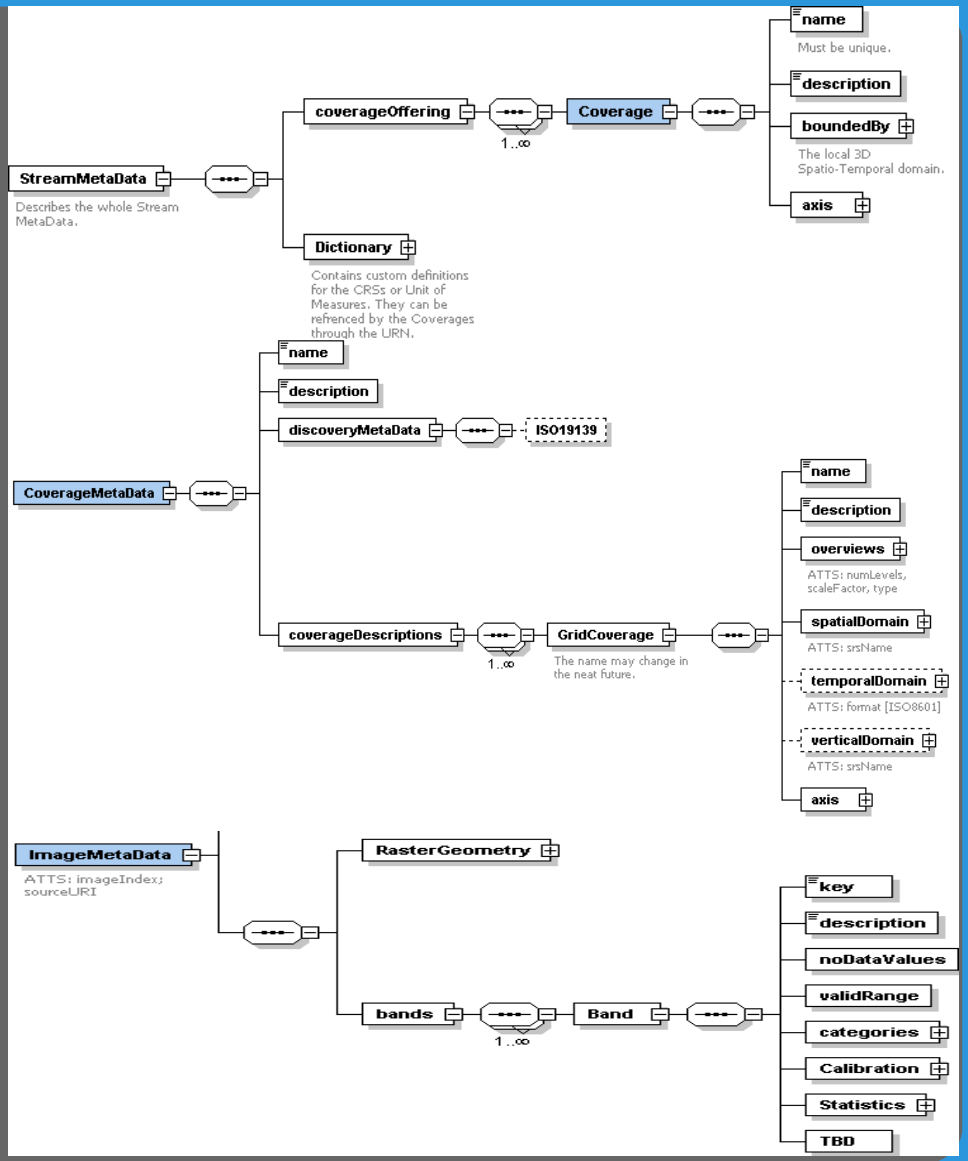
nD Coverage Support





nD Coverage Support

- **Stream Metadata**
  - Brief description of a source's content
  - Similar to WCS GetCapabilities
  - Brief description each coverage
- **Coverage Metadata**
  - Detailed description of a coverage
    - Domain, Range, Overviews...
  - May point to external set of metadata (ISO 19115, 19139)
- **Image Metadata**
  - Detailed description of a 2D slice
    - Raster geometry, band structure
  - Description of exploitation metadata
    - No data values, calibration offset and calibration scale, raster statistics



Next generation of raster support for the GeoTools-GeoServer stack

Dott. Ing. Simone Giannecchini (GeoSolutions)

simone.giannecchini@geo-solutions.it

FOSS4G07, Victoria

September 26th, 2007

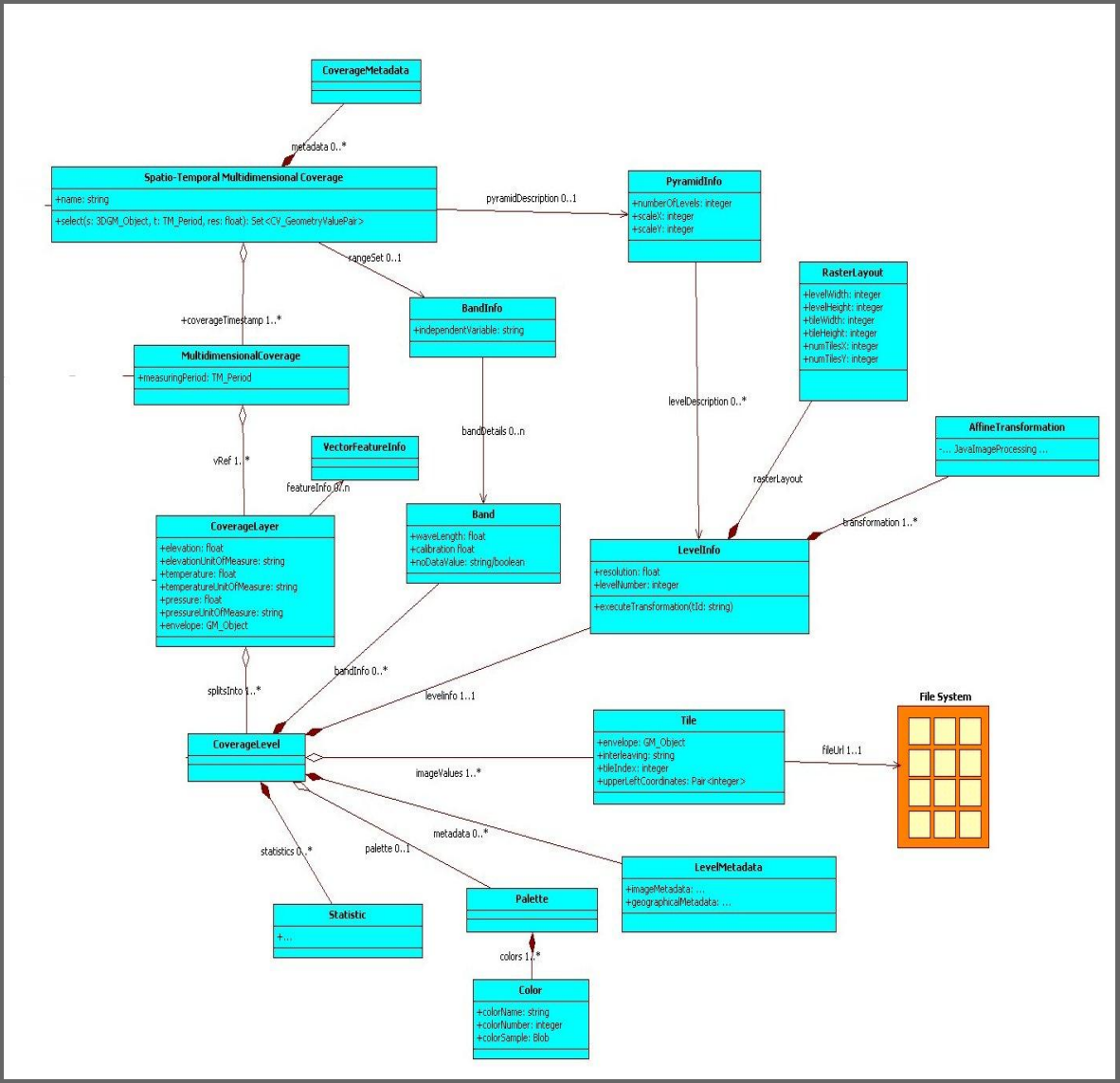
Image I/O-ext

```
// ////////////////////////////////////////////////////////////////
// preparing to read
// ////////////////////////////////////////////////////////////////
final ParameterBlockJAI pbjImageRead;
final ImageReadParam irp = new ImageReadParam();

//subsample by 8 on both dimensions
final  int xSubSampling = 8;
final  int ySubSampling =8;
final  int xSubSamplingOffset =0;
final  int ySubSamplingOffset = 0;
irp.setSourceSubsampling(xSubSampling, ySubSampling
, xSubSamplingOffset,
ySubSamplingOffset);

//re-tile on the fly to 512x512
final ImageLayout l = new ImageLayout();
l.setTileGridXOffset(0).setTileGridYOffset(0).setTileHeight(512)
.setTileWidth(512);

pbjImageRead = new ParameterBlockJAI("ImageRead");
pbjImageRead.setParameter("Input", file);
pbjImageRead.setParameter("readParam", irp);

//get a RenderedImage
RenderedOp image = JAI.create("ImageRead", pbjImageRead,
new RenderingHints(JAI.KEY_IMAGE_LAYOUT, l));
```

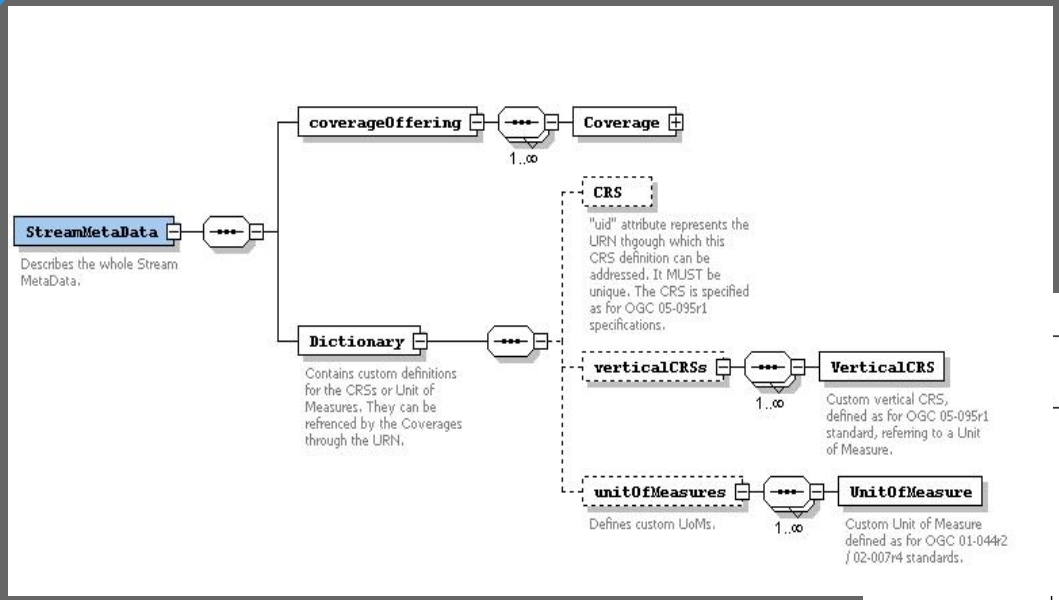
RasterSymbolizer

```
<xsd:complexType>
    <xsd:complexContent>
        <xsd:extension base="sld:SymbolizerType">
            <xsd:sequence>
                <xsd:element ref="sld:Geometry" minOccurs="0"/>
                <xsd:element ref="sld:Opacity" minOccurs="0"/>
                <xsd:element ref="sld:ChannelSelection" minOccurs="0"/>
                <xsd:element ref="sld:OverlapBehavior" minOccurs="0"/>
                <xsd:element ref="sld:ColorMap" minOccurs="0"/>
                <xsd:element ref="sld:ContrastEnhancement" minOccurs="0"/>
                <xsd:element ref="sld:ShadedRelief" minOccurs="0"/>
                <xsd:element ref="sld:ImageOutline" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

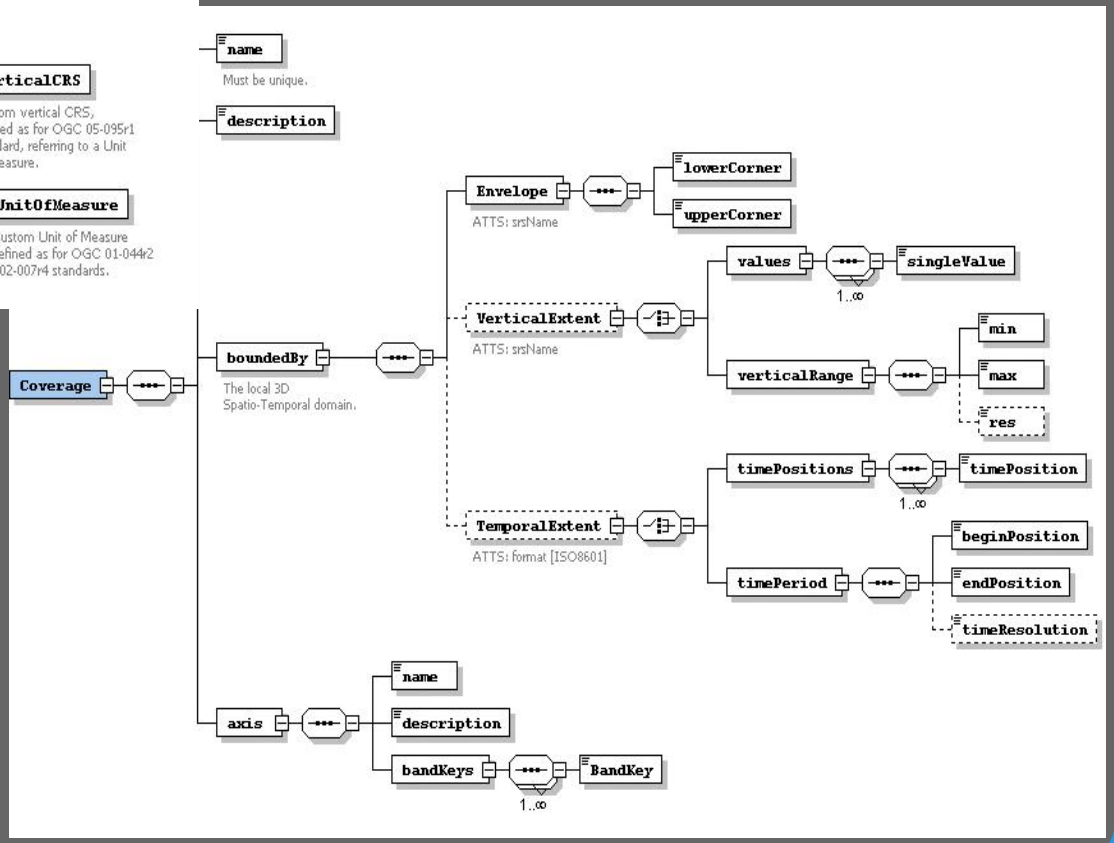
PostGISRaster – Schema



nD Coverage Support



StreamMetadata



nD Coverage Support

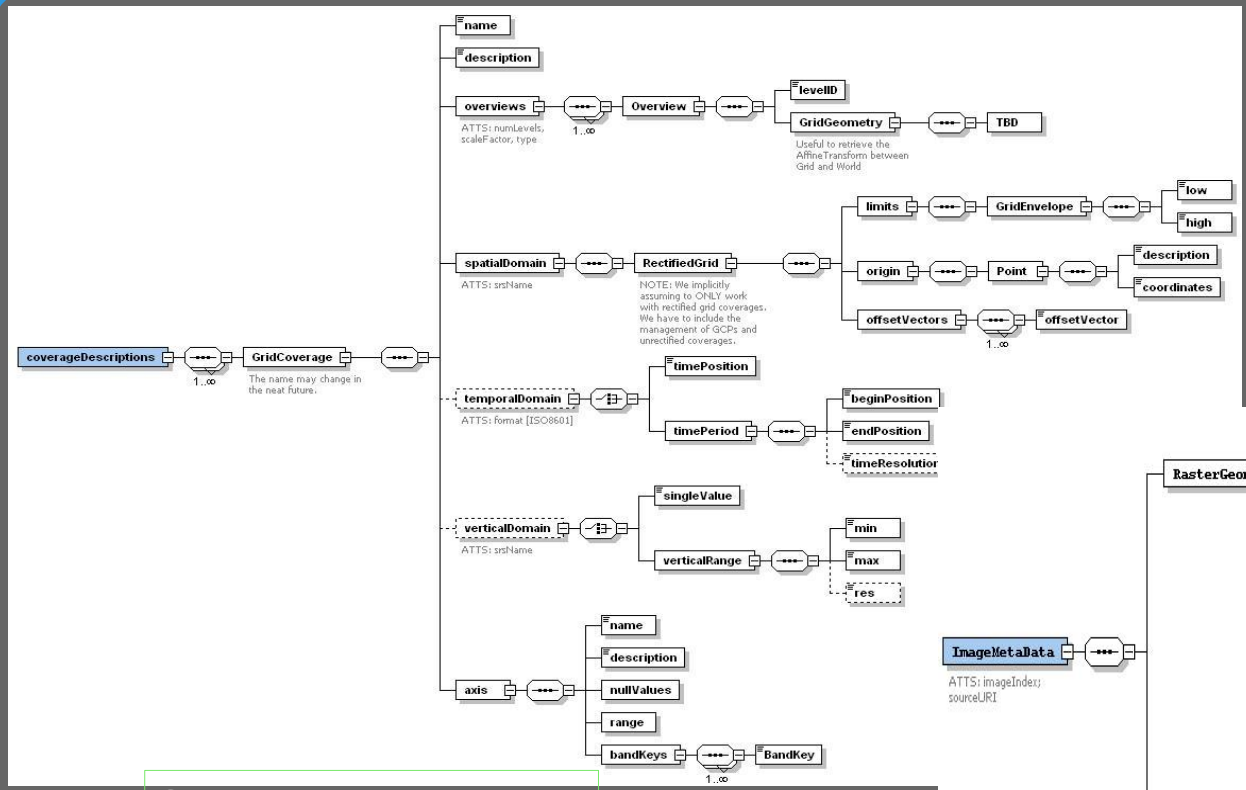
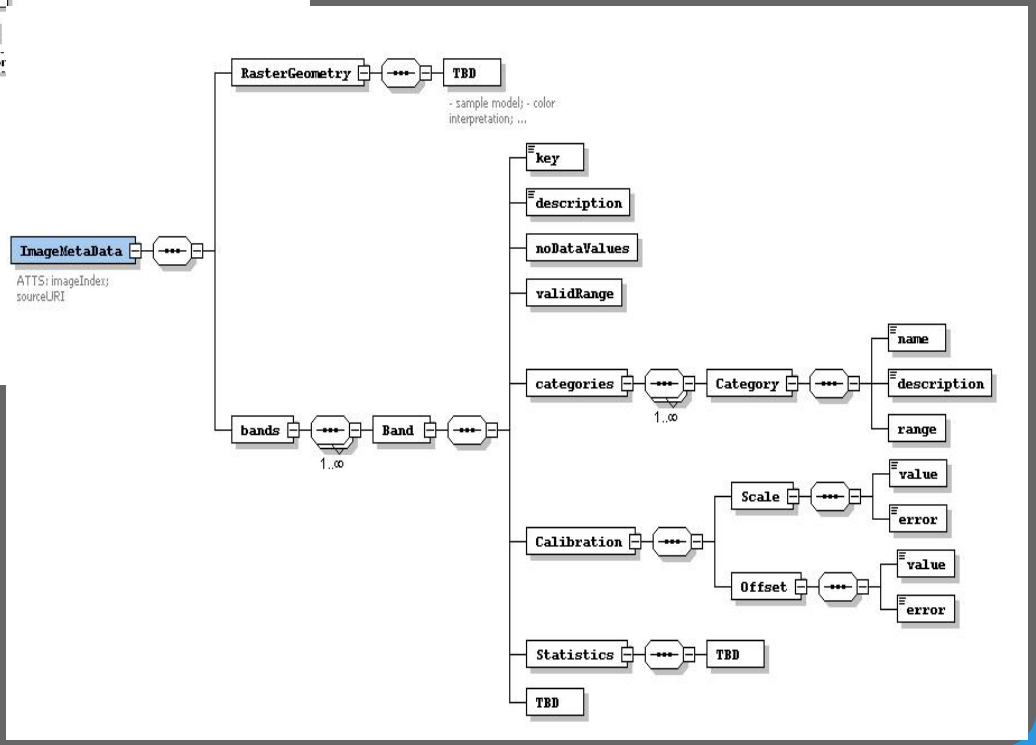


Image Metadata



Coverage Metadata