Google *"Google Summer of Code 2007 Project"*

GeoRaster – A Coverage/Raster Model and Operations for PostGIS

Participant: Xing Lin (solo.lin@gmail.com)

Mentor: Timothy H. Keitt (tkeitt@gmail.com)

Last Update: 2007/06/25

1. Introduction to Coverage/Raster Model for PostGIS

1.1 About Vector and Raster Model in GIS

"Modern Geographic Information (GIS) data represents real world objects (roads, land use, elevation) with digital data. Real world objects can be divided into two abstractions: discrete objects (a house) and continuous fields (rain fall amount or elevation). There are two broad methods used to store data in a GIS for both abstractions: Raster and Vector." (Wikepedia: Data Representation of GIS)

"Vector data type uses geometries such as points, lines (series of point coordinates), or polygons, also called areas (shapes bounded by lines), to represent objects. Examples include property boundaries for a housing subdivision represented as polygons and well locations represented as points. Vector features can be made to respect spatial integrity through the application of topology rules such as 'polygons must not overlap'. Vector data can also be used to represent continuously varying phenomena." (Wikepedia: Data Representation of GIS)

"Raster data type consists of rows and columns of cells where in each cell is stored a single value. Most often, raster data are images (raster images), but besides just color, the value recorded for each cell may be a discrete value, such as land use, a continuous value, such as rainfall, or a null value if no data is available. While a raster cell stores a single value, it can be extended by using raster bands to represent RGB (red, green, blue) colors, colormaps (a mapping between a thematic code and RGB value), or an extended attribute table with one row for each unique cell value. The resolution of the raster data set is its cell width in ground units. " (Wikepedia: Data Representation of GIS)

Nowadays, both vector and raster data models are widely well-supported and used within various GIS application and platform system. With the development of remote sensing techniques and other raster data acquiring methods, the raster data model is becoming more and more important, which is one of the main motivation of me to carry out this project.

In order to understand the raster model for PostGIS proposed in this document, you might need to have the basic knowledge of GIS raster data model especially such terms as resolution, geo-referencing/spatial reference system (SRS), data type, value type, DEM/DTM and so on. Some other information regarding the basic of PostGIS might also be necessary before reading through this document.

For more information about the information about Raster data model and the discussion between vector and raster model in GIS, please look up the following items in *Wikipedia*: Geographic information system, DEM/DTM. The references of this documents might also give you some useful information on these topics.

1.2 Introduction to Coverage/Raster Model for PostGIS

The Coverage/Raster Model for PostGIS is the about the solution on how to store, query and process raster model data (such as DEM, satellite images and so on) in PostGIS/PostgreSQL ORDBMS. The following issues regarding how to implement a Coverage/Raster Model for PostGIS will be covered in this document:

- Abstract Raster Data Model (Class and Object Types Diagram)
- Physical Storage Method of Raster Data in ORDBMS like PostGIS/PostgreSQL
- Georeferencing/Spatial Referencing System of Raster Data
- Indexing Raster Data

- Compression and Decompression of Raster Data
- Metadata
- System Utilities and Useful Tools: Import, Export and Viewers
- Example and Demo

As one of the Google SoC 2007 projects, with the help of my mentor, I will only finish some fundamental part of this Raster data model for PostGIS. Some advanced features, like spatial analysis tools on this new data model will not be covered for this summer. After public release, you can contribute to this project if you like. Regarding the TODO list within this summer project, please refer to the section of "Task and Schedule".

2. Advantage and Challenge of Coverage/Raster Data Model in ORDBMS

Many people in the PostGIS community have already posted great a lot concerning the advantages and disadvantages of implementing a coverage/raster data model in such an ORDBMS like PostgreSQL. Although there are some challenges to carry out such a implementation work, I still believe the work stated here will improve the ability of PostGIS in the geographic data management and benefit quite a lot on the open source GIS community.

Before continuing with the design of Coverage/Raster model for PostGIS, I would like to declaim the advantages and challenges for the implementation of a coverage/raster data model in ORDBMS.

- *Multiuser Access* - When many users are accessing the same raster files simultaneously, better performance is possible from a properly tuned, centralized database than from a file-based system.
- *Integrated Data Management* - A database allows common data management and retrieval for all geospatial data including raster, vector, metadata, and tabular data. A database also provides access to extremely large images (many gigabytes to many terabytes) of continuous spatial data (e.g., 30-meter digital elevation model composite of North America).
- *Data Security* - A database has tools for multiple security levels to be established and enforced. Users can be given access to the imagery that is relevant to the job they are being asked to work on.
- *Data Query* - A database allows for a common query environment. Queries can be made to show all data related to an area during a particular time period or for a particular subject.
- *Better Network Accessibility* – Modern database system provide friendly network-enable access interface, such as JDBC, OLEDB and some other private interfaces. Raster data stored in such as DBMS could be better accessed from network comparing the traditional file system.
- Less Overhead When Dealing With the Interaction Between vector (SDB) and raster (file system) data.

There are some more other arguments on what's the point to implement such a raster data model with a ORDB. I also strongly agree that there is totally no point if we just "put the whole image in the database, then take the complete image back out". Under such circumstance, the flat file will definitely better than image database. I think the key question or challenge to such a question is the performance of image database comparing the flat file solution. I think the main challenge of a raster database lies in the following aspects:

- Free Extraction – There are various kinds of remote sensing image formats in used around the

world. If this image database could only be extracted to the form it has been imported from, there would not be much points to carry out the implementation. A generic coverage/raster model and storage schema might be a possible way, but there are still something to think about to improve the suitability and performance. Another aspect of free extraction lies in the extraction to an arbitrary subset. It will involve the interaction between raster and vector objects.

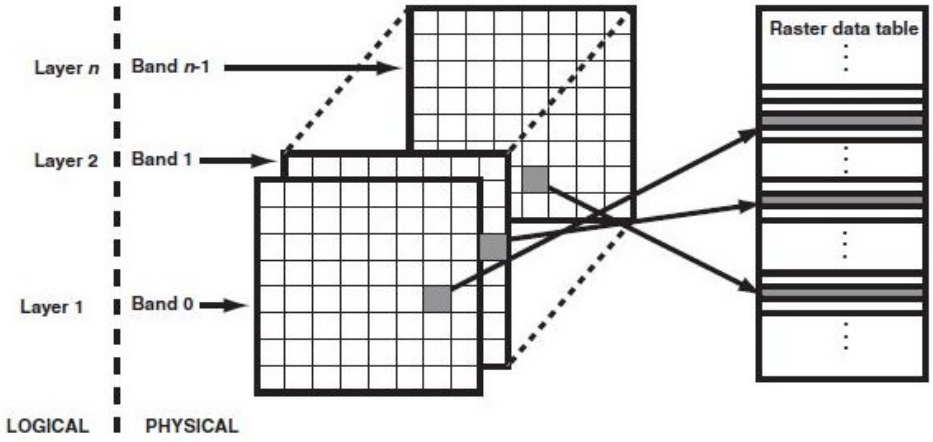
- Performance – Performance is always the main point of discussion in the PostGIS-Dev mail-list. Basically, the flat image files are faster than any image database under the environment of a single-user accessed via local file system. But image database could beat flat file system if the amount of access grows huge and most of them come from the network. Many steps could be taken to achieve such a goal, such as the tiling/blocking storage method, pyramid structure, image compression/decompression and so on.
- Optimization for Certain Application – Typical application could be divided into two groups. One is the data processing and analysis with high computational density from a single and direct connection (local or LAN). The other is the data browsing and query with high transactional density from the Internet. Based on such a special type of data modeling and storage, quite a lot of optimization could be carried out to improve the performance. For data processing and image analysis, special algorithm could be invented to make use of the tiling storage schema and multiple processors. For the Internet based image browsing, high-ratio and progressive compression techniques could be applied, such as JPEG2000 format.

(Note: Some discussion posted in the PostGIS-Dev mail-list could be found here.)

3. General Ideas of Design

As well as Oracle GeoRaster, PostGIS GeoRaster uses a generic raster data model that is component-based, logically layered, and multidimensional. A new user-define object type named GEORASTER will be defined as well as some other useful object types. A table with a column of type GEORASTER is called a GeoRaster table. Each row in the GeoRaster table denotes an geo-referenced images (satellite images, aerial photos...)or other raster coverage (DTM/DEM...). The other columns could be defined as user required. Quite a lot of functions will be defined and executed upon objects of type GEORASTER, such as the functions to create subset of raster coverage, the functions to interpolate a point within a certain raster coverage. All such new object types and functions are executable from SQL language.

The real raster (or image) data is stored in another toasted table that is connected to its corresponding GEORASTER object. Such tables are called Raster table, everyone of which contains special column of type RASTER to store raster data. Within a single RASTER object, there are a sequence of image pixels or coverage cells, each of which represent the smallest unit of information within a raster dataset.

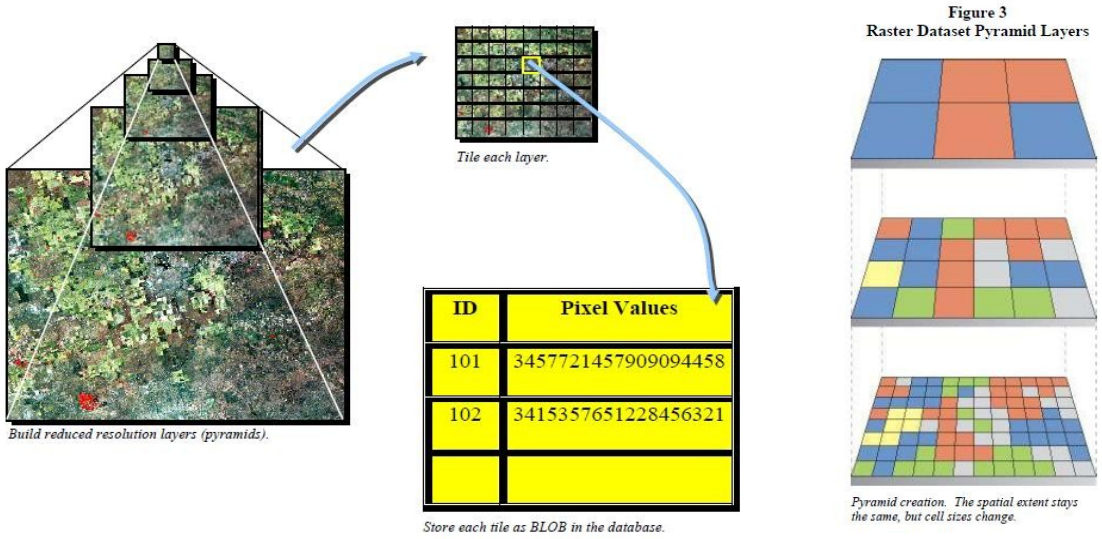


(GeoRaster, Bands, and Raster data table. Source: Oracel GeoRaster)

Images and raster dataset could be saved in tiles/blocks. It means the entire images or raster dataset will be divided into several smaller tiles of regular size before imported into image database. Each row in Raster table will only store one block/tile of raster data. Then there need to be some other columns together with the one of type RASTER within such as Raster table. These information are used to help record the location of each blocks, so that they could be rebuilt as a whole when necessary. For multidimensional satellites images as well as ordinary colorful images, the blocking/tiling techniques could also be applied to the band dimension. The parameters of blocking storage could be defined by users. By default, it will be used with an proper set of parameters advised by GeoRaster image database itself automatically.

In addition to reducing the data amount transferred over network, tiling/blocking could also be used to improve the performance of data visualization and analysis. The entire image data could be display onto the viewport of client window asynchronously tile by tile. It could save the time of response and improve user experience than show up a big final image after a long time waiting. Data analysis process could also be improved by optimizing certain algorithms to make use of the tiling storage system and multiple processors' environment.

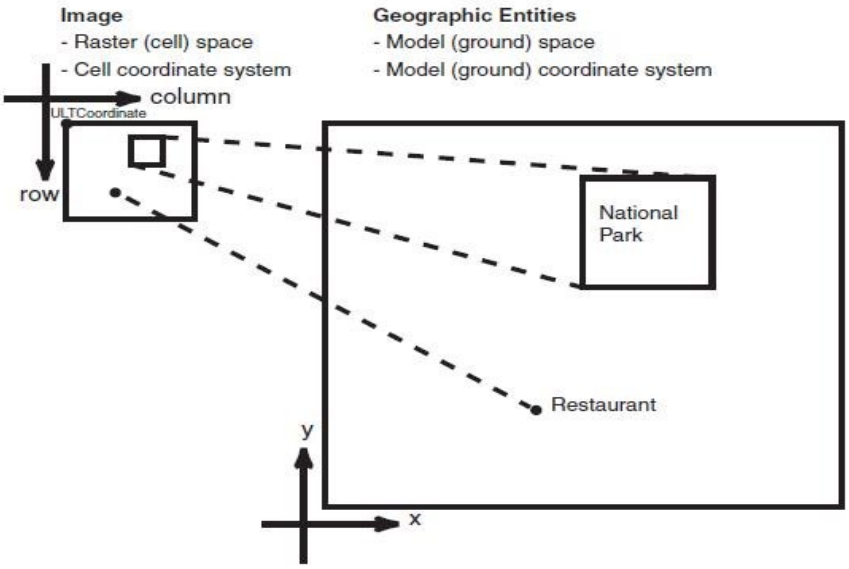
Another technique that is widely used to improve the performance of large image browsing is called pyramid structure. As you know from the name of Pyramid structure, you can build several “copy” datasets from the original one with coarser resolution using a certain kind of resampling algorithms. Such coarser copies of raster datasets will be much smaller in size than the original one but adequate for visualization at a lower scale (zoom ratio). Imaging you have a 400pixel*400pixel viewport in the client and a 800pixel*800pixel image in the database. You can just display a half-reduced version of the original image, which will appear nearly the same as you put all the pixels to a small viewport. But it will save 75% of data transfering from sever to client. When the user zoom in to a different scale, a special level of image in the pyramid structure with a proper accuracy will be packed, transferred to and rebuild at the client side. Pyramid structure could obviously improve the performance of the large image browsing via Internet application. The parameters that control the arrangement and storage of pyramid structures could be altered by user. By default, it will be used with a set of optimized parameters that are advised by GeoRaster image database itself automatically. Blocking/tiling techniques will also be apply to pyramid levels also.



(Pictures source from ESRI ArcSDE 9.1 Raster. Should I contact ESRI for permission?)

Geo-reference is the thing that differs GIS raster/coverage datasets from ordinary image files. It tells the location of raster image within a geographic coordinate system, projected coordinate system or local coordinate system. The relationships between cell coordinates and model coordinates are modeled by GeoRaster reference systems (mapping schemes). Similarly as Oracle GeoRaster, the following reference systems are defined in PostGIS GeoRaster:

- 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).



(Mapping between ULTCoordinates and Model Coordinates. Source: Oracle GeoRaster)

Initially, the GeoRaster SRS will be implemented by setting up a mapping from cell coordinates system (up-left coordinate, ULTCoordinate) to model coordinates (geographic or projected). GeoRaster currently supports six-parameter affine transformation that geo-references two-dimensional raster data. Such affline transformation will be recorded by the six-parameters as well as ground control point (GCP) in a special tables (in Oracle GeoRaster terms, such tables are called value attribute table, VAT). There are also some other aspects about georeferencing of raster dataset, such as rectification and orthorectification. A special object of type GEOSRS will be set up to record such information for a raster dataset and be stored as part of metadata for GEORASTER objects.

Data compression and decompression will be one of the key techniques to be used in PostGIS GeoRaster image database. Using compression before data storage could reduce storage space and amount of data to transfer via network. But it depends on the types of data to be compressed and kinds of application. Currently, GeoRaster will provide two native data compression algorithm to reduce data storage space: JPEG2000 (Lossy) and LZW(Lossless). Some raster data, such as DEM/DTM data, could be used lossless data compression algorithm, because it is always involved in a data analysis process required the high accuracy of data. But sometimes for satellite images and aerial photos, the lossy compression algorithms could be compelling which could have a acceptable visualization effect, but highly reduce the data amount. This could be especially true for pyramid data. Of course, you can choose not to do any compression or decompression staff towards the raster dataset. The compression information will be recorded as part of GEORASTER object.

Indexing raster image is based on the spatial footprint of each block of raster data which will be stored as a GEOMETRY object in cell coordinates space. Certain kinds of spatial index (GiST-R Tree) will be applied on such GEOMETRY column to indexing raster images. When certain subset of the entire image is needed or user is navigating throughout the whole dataset, involved blocks of raster data could be easily located and selected out with the help of spatial indexing tied to the blocks.

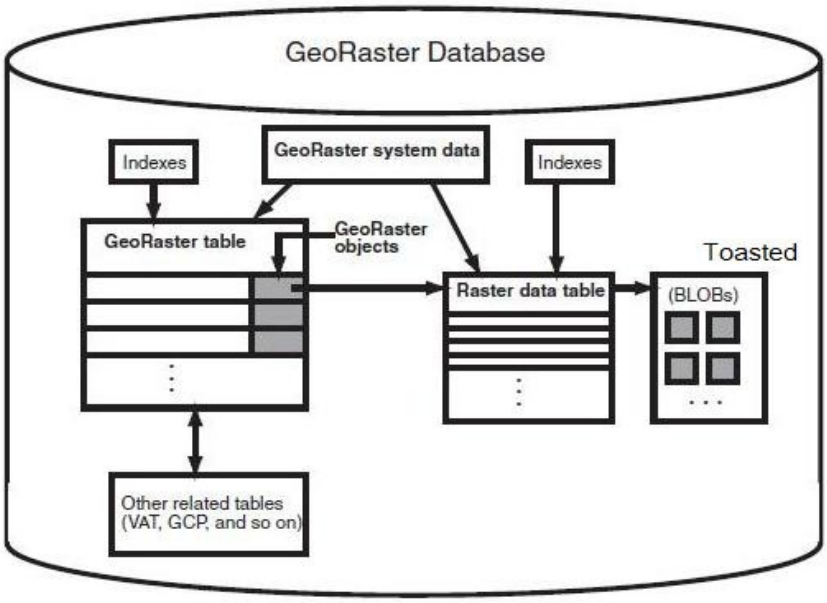
As required by PostgreSQL user-extended data type schema, certain types of input and output functions need to be provided together with the definition of new object types. Because the specialty of raster data model (RAW, cell cloud....), the similar way to WKT and WKB will be invented to help the manual input of raster data (sometimes, people need to create some constant, blank, template, or filter raster dataset.) Common raster data encoding techniques will be adopted here, such as RLE (Run-Length Encoding), QTE(Quad-Tree Encoding) and so on. Some import and export tools will also be

provided to handle the translation between PostGIS GeoRaster and the famous raster data format (ESRI GRD/ASCII, ERDAS IMAGE, GEOTIFF and so on).

As well PostGIS the spatial extension for PostgreSQL, PostGIS GeoRaster also need to have some system tables as data dictionary and place to save metadata. Certain system tables will be established after introduction new data types into GeoRaster. Trigger and stored procedures will be set up to maintain the data integrity when inserting, modifying or deleting data from GeoRaster table (you need to delete corresponding data records in Raster data table).

There are some other parts regarding the physical storage solution for raster data in ORDBMS, such as data type, value type, band interleaving (BSQ, BIP, or BIL), data padding, raster data encoding and so on. More detail information will be included in the following parts. For more informations, please refer to the corresponding section below.

Finally, the raster dataset will be storage and processed within PostgreSQL/PostGIS ORDBMS in such as schema.



(Source: Oracle GeoRaster. Modified to be proper within PostgreSQL environment)