Integrating Imagery Data into Spatial Databases

Qian Liu Jason Willison
Environmental System Research Institute, Inc
380 New York St. Redlands, CA 92373, U.S.A.
qliu@esri.com jwillison@esri.com

Abstract: The advances in sensors development and data acquisition techniques increasingly contribute to the quality and quantity of imagery data. The challenges that remote sensing communities are facing today are no longer the lack of imagery data, on the contrary, we are overwhelmed by the vast amount of data collected from various sources. To maximize the usage and distribution of the data, an efficient management system for storing and managing the data is critical. Spatial database has been developed primarily for managing vector based data in the early days, however the integration of remote sensing and GIS disciplines over the past decade has brought an insight into the possibility of managing imagery data along with vector data in one spatially enabled database. The new developments in database technology are also in favor of dealing with large imagery data in the database. The release of ESRI® ArcSDE® product [1] with raster storage support a few years ago marked the new generation of spatial database that is enabled for managing image data in addition to vector and tabular data. There have been some new developments since then for better integration and management of imagery in spatial databases. This paper is intended to outline the requirements for integrating imagery data into a spatial database in different aspects including image data model, database schema, storage consideration, data access, update, and retrieval.

Keywords: Spatial database, Imagery data storage.

1. Introduction

The ever-increasing volumes of imagery data bring great opportunities for us to better understand and monitor our environment. In the meantime, efficient managing and timely accessing these valuable assets also impose a great challenge. Collected in different resolutions from various sources such as satellites imageries, radar images, aerial photos, scanned maps, imagery data tend to be huge in size. A complete USGS DEM (Digital Elevation Model) covering continental United States at 30 meter resolution may take up 70 gigabytes of file space, if the resolution goes higher the size will grow exponentially. Current available sensors are capable of producing images down to centimeter resolution. The volume adds up easily. In modern enterprise environments, more and more companies and organizations stay connected both within the community and across. With the popularities of internet and web services, more than ever, professionals and ordinary users are relying heavily on centralized data storage system to obtain appropriate data and information and utilize them on regular basis. Duplicating of such large amount of imagery data on every single client machine is costly, inconvenient, and unnecessary. .A robust and flexible centralized image data management system is more appealing.

The challenge for efficient management of imageries has brought attentions to both remote sensing and GIS communities. GIS professionals have been gradually introducing imagery data to the GIS realm and treating it as another spatial data type. Dated back to the 70's, grid was first introduced and implemented in ArcInfo workstation. Even though it was more vector-oriented implementation and geared mainly for spatial analysis, it demonstrated great a prototype and great potentials for integrating imagery with other spatial data in a common GIS framework. In recent years, even more GIS companies have put substantial efforts to provide functionalities to access, manage, visualize, and analyze imagery data in their GIS systems. On the other hand, remote sensing communities have been seeking tools from GIS packages to manage imagery data so that they are ready to be used for processing and analysis. The tighter integration of remote sensing and GIS in recent trends has brought new insights on addressing imagery related issues using experience gained on vector data in certain aspect, such as management of large quantity of data.

Traditionally, imagery data were kept in file systems, either as disk files or on tapes. In earlier days, the file-based approach used to work well in a disconnected environment. However, it has considerable limitations in terms of concurrency, data transfer, distribution, and scalability when it comes to an enterprise environment [3]. These functions are often offered by a database system as fundamental features, however spatial data support are not directly available in most commercial database systems. It is worth mentioning that most of commercial database management systems offer support for large objects which can be used to store binary data. This feature makes it possible to store image data in a relational database and eventually leads to the development of database based image management system.

Spatial database is simply a database that is capable of handling spatial data. Built upon traditional relational database management systems, spatial database takes full advantages of what a RDMS system has to offer and provides a generic framework for spatial information and business information. A major benefit of enterprise spatial databases is the powerful combination of manageability, size and scalability. With traditional GIS software, it was essentially impossible to deal with multiple users and large datasets. Only with enterprise systems, multiple terabytes or pedabytes of data are possible [4]. Spatial database was initiated primarily for managing large volume of GIS or vector data. Over the past few years, new technologies and improvements in RDMS systems have enabled spatial information to become an integrated part of mainstream enterprise information system. Spatial databases have proven to be reliable and flexible for managing large spatial data repositories, especially for vector data. There are a few spatial database systems currently on the market. These include ESRI ArcSDE, MapInfo SpatialWare [7], and Oracle Spatial. Recent storage technology makes it feasible to build sizeable image repository in a database system [2], furthermore, researches in fast access of imagery data in a database system also contributes to new generation of spatial databases with the capability of handling imagery data effectively. ESRI ArcSDE solution first brought this vision into reality with full support for imagery storage, management and fast access of imagery data.

In this paper, we will propose a list of requirements for integrating imagery data into spatial databases and discuss implementation approaches when applicable.

2. Spatial Database Architecture

Client/server architectures are the major trend in enterprise systems, especially thin client/fat server architectures. While server is responsible for business logic, data processing and data storage, client simply acts as a light-weighted browser. To bring spatial database engine into the picture, a middle tier makes sense. Now client, spatial database engine and server form a three-tier enterprise spatial information system (Fig.1). At a more detailed level, in some cases, the engine can also be configured as an application-side software component and stays in the client tier in a two-tier system architecture. As a gateway between client and server, spatial database engine sits on top of the server where the database resides. It plays a key role in filling the gap between the client and database in the server. Built with rich spatial-aware components, the engine interfaces to the database for spatial data management and operations.

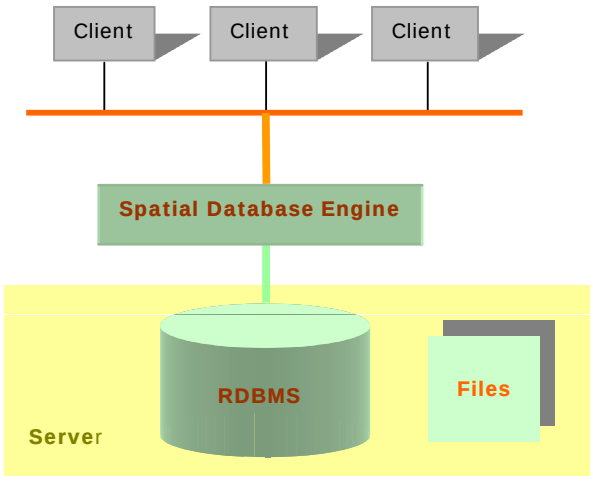

Figure 1 Three tier client/server architecture with spatial database engine

The general workflow for simple data retrieval through spatial database engine includes:

- client sends request to spatial database engine to attain desired spatial data in certain spatial location in certain resolution
- spatial database engine processes the request and translates the request into database languages, e.g. SQL
- engine sends the SQL query to the database

- database parse the query and fetches the proper data
- returned rows and objects are sent back to the engine
- engine compiles the rows and objects into images and applies associated spatial information to the image
- engine sends the image back for the client to consume

A key element in integrating imagery data into spatial database is to store the data in such a manner that it is fast and easy to access and update. However, most of spatial database engines available are tailored for GIS or vector data. In order to fulfill this basic requirement for image data management, the spatial database engine has to be extended so that it supports proper database schema and retrieval mechanism and instructs underlying databases to behave accordingly.

3. Requirements for Integrating Imagery Data

As mentioned earlier, efficient management and timely access of large imagery data are the fundamental requirements for a successful image database. Managing imagery data are far more challenging compared with vector data simply because of the sheer volume of imager data. A solid image data model is a starting point for efficient imagery storage and management. Such model has to be robust and flexible to satisfy all sorts of requirements from imagery data vendors to consumers. The implementation of image data model requires a cooperative database schema where table structures and relationships are optimized for storing and retrieving image data. Once image data are uploaded into the spatial database, it should be ready to be consumed with little interaction.

1) Image Data Model and Storage

Imagery data are different from vector data which are represented by points, lines and polygons. Before we can conceptualize the image data model, we need to examine the characteristics of image data in detail. Image is basically a rectangular array of pixels, and each pixel represents some characteristics of an area on earth, size of the area is defined by the image resolution. An image may have one or multiple layers or bands and each band represents a specific measurement, for example, a LandSat dataset has a total of 7 bands, the pixel values on each band reflect responses of the ground subject in certain wavelength.

An ever-increasing number of sensors and platforms are continuously generating imagery data. These data often come in various formats, resolutions, and types. In addition to wide-used LandSat images, many new comers have joined the crowd. The more recent satellites collect image data in much higher resolution or a wider geographic span. SPOT 5 obtains 5 meter (2.5 meter in supermode) resolution in panchromatic mode and 10 meter (5 meter in supermode) resolution in multispectral mode; IKONOS data records 4 channels of multispectral data at 4 meter resolution and one panchromatic channel with 1 meter resolution; QuickBird acquires 61 centimeter (2-foot) resolution panchromatic and 2.44 meter (8-foot) multispectral imagery. To make the matrix more complicated, more radar data are becoming available, such as Radarsat data, ENVISAT data, and ERS data [9]. These raw data are the bases for generating more derived image data by image processing and analysis.

Usage pattern of imagery data also plays an important role in data model design. Some consumers may use a huge mosaic of images covering a large extent as background for map generations, some may extract a small portion of an image and perform analysis on the data, some data vendors may wish to be able to grab a set of images that ordered by their customer and send them directly to the consumer with little process. A versatile image data model needs to take all the combinations into consideration so that all possible situations are covered.

Image data can be conceptually modeled into 3 levels (Fig.2), the top level is a collection of raster layer that all layers in the collection are bound with common properties, such as time series and history data collection that contain a number of raster layers covering the same geographic area but at different time. Raster layer is at the next level, a raster layer is comprised of one or many raster bands, all the raster bands share common properties and they are normally obtained from the same satellite for the same location at the same time, for example multispectral data. Raster band comes next in the picture. A raster band is a rectangular array of pixels. Metadata is associated with the object at all levels.

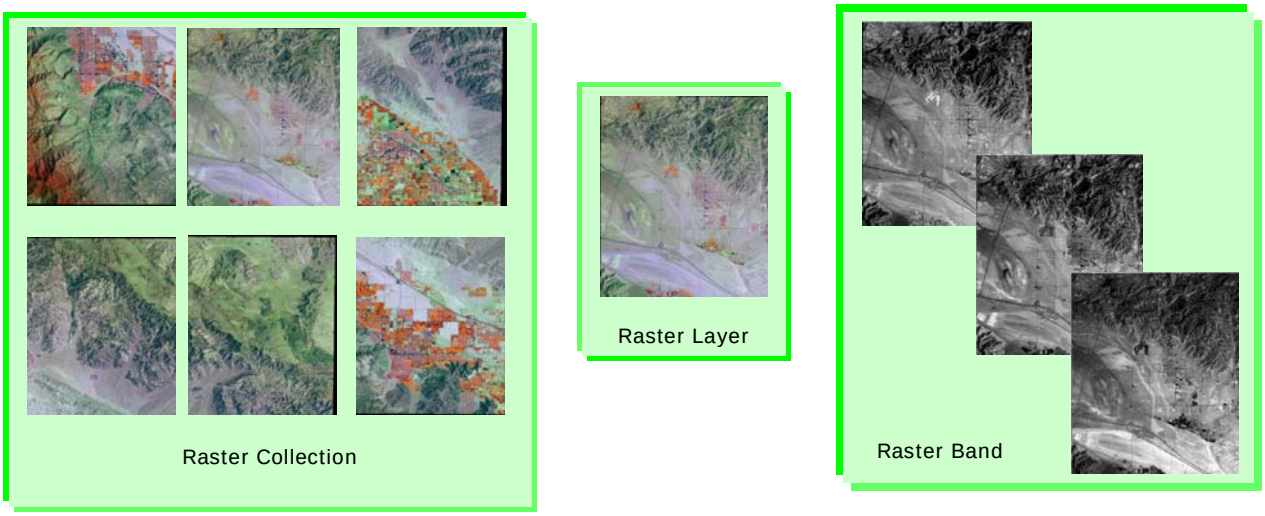

Figure 2 Image data model

Metadata is as important as pixel values for image data, without metadata, the pixel data are useless. Metadata can contain a lot of useful information about the pixel data, such as the time of acquisition, sensor information, process procedures, spatial properties, attributes, etc. Storing metadata effectively along with the pixel data facilitates data query and data processing tremendously.

One of the goals in designing image data model is to support large seamless image in the database. In most cases, such seamless image dataset is generated from mosaicking or appending a number of spatially continuous image data with same characteristics. This seamless dataset satisfies those data consumers who desire to extract a portion or display it as a whole in their applications. However, if stored as one entity or object in the database, the object of this magnitude in the database would be unmanageable and performance in turn would be very poor. Most of spatial database systems that support large image data management (e.g. ESRI ArcSDE, TerreServer [10], Oracle GeoRaster [8], ImageSTOR [6]) have adopted tile-based solution for storing large image dataset in which each tile is stored as an object.

2) Image Data Storage

Tile-based storage mechanism for image data is efficient for almost all cases. A raster band is divided into many rectangular tiles. As a basic unit, all spatial operations are performed on the tiles. When tiles are stored, its location in the raster band is recorded for accessing. The tile dimension for all tiles in a raster band are normally homogeneous [5], however some spatial database systems do support user defined tile partitioning [1].

Even though raster band is pieced into small tiles, the application may still not need to access the data at full resolution. If the application requests to display the whole world image of high resolution on the computer screen, the database is going to retrieve all the tiles in the world image and send it back to the application. In fact it is not possible to display all the pixels in that high resolution on the screen and resampling will take place to downsample the data to screen resolution. Transferring such large amount of data from database to the client over the network and resampling are very slow and unnecessary. It would be beneficial to generate images in lower resolutions and store them alongside with the full resolution data. When queried, only images in appropriate resolution are retrieved.

Multiple resolution layers are commonly generated through building reduced resolution layers by resampling full resolution image. Each raster band has its own set of multiple resolution layers. Commonly used resampling techniques include nearest neighbor, bilinear, bicubic, and majority. Depending on the properties of the raster band, one resampling

method is applied, for discrete data such as land use / land cover data, nearest neighbor resampling is more appropriate, on the other hand for continuous data such as DEMs or multispectral data, bilinear or bicubic are desirable. Aside from creating reduced resolution layers, a number of other techniques are available as well. Wavelet encoding mechanism is getting more attention lately and it looks promising as well. Instead of generating separate multiple raster layers for multiple resolution, wavelet algorithm allows all multiple resolution layers stay in the same layer which reduces joining tables when queries take place. Another benefit of wavelet encoding is to reduce the storage size. However, this technique has not been employed in any spatial database systems yet. Another option for multiple resolution layers is custom layers, it is the user or application's choice to associate any layer and assign it as one reduced resolution layer. It is very likely that the layer is not related to the full resolution layer at all. An example could be that a state holds images for the state in different resolutions, for instance low resolution DEM and high resolution orthophotos, and an application needs to display certain resolution image when zooming to a certain scale. Instead of storing multiple raster layers with each layer creating its own multiple resolution subsets, only one master raster layer with highest resolution is necessary. According to the image resolutions, each image layer is designated as a reduced resolution layer to the base layer for its resolution. Though this approach is simply conceptually, there are some barriers in implementation. One of the difficulties is how to keep coherent relations between base layer and these custom multiple resolution layers in terms of dimension, tiling, and pixel alignment.

Compression is another factor to consider when storing large amount of image data. The main advantage of compression is to reduce storage space and facilitate data block transfer over the network. A few compression algorithms are currently available and free to use for lossy and lossless compression, for example LZ77, JPEG, and JPEG2000. Compression can be performed before tiling takes place or after. If performed before tiling, it might be troublesome when creating seamless raster layers. Every time new data is appended, the compression process will kick in and brings the whole raster layer for compression. Compression on tiles is more reasonable since tile is small and easy to manage.

3) Database Schema

In general, the database table schema cooperates with a data model and storage requirements. It requires an optimized table schema to support objects of all three levels in the image data model, namely, raster collection, raster layer and raster band. Optimized tile storage schema is critical as well.

A raster collection consists of one or more raster layers, and a raster layer is then a special case of a raster collection in which there is only one element. A raster layer relates to raster band in a one-to-many relationship. A raster band contains many tiles from the band and its multiple resolution layers, it relates to tiles in a one-to-many relationship (Fig.3).

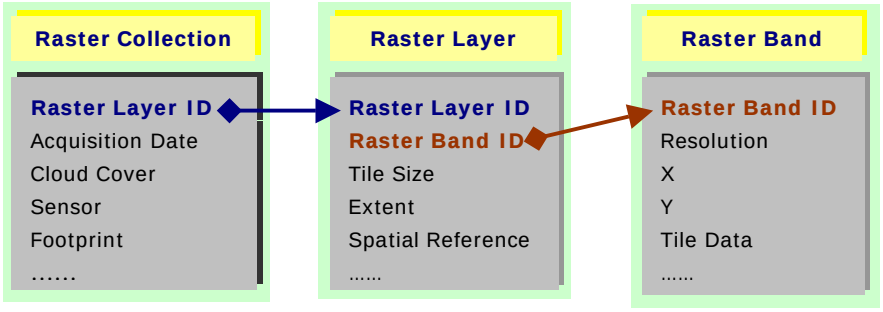

Figure 3 Image database schema

3) Spatial Indexing and Image Data Retrieval

Queries on image data in a database are basically of one of three types, they are spatial query, metadata query, and content-based query. Spatial query is relying on desired spatial location, it could be a geographic extent or dimension in image space, this type of query often occurs when displaying and zooming/panning around a raster layer or data extraction. Metadata query applies conditions on metadata of the raster layers, for example, a query that finds all the

raster layers that have cloud coverage less than 10%, this type of query is not related to pixels or their locations. This is more common in querying a raster collection when a special set of image layers are requested. Content-based query or feature extraction extracts maximum information from the available data using spectral and spatial measurements such as object color, size, shape, texture, or connectivity. A good example of feature extraction is a query that finds all the airports in a series of image layers in the database. A number of techniques are available for feature extraction using pregenerated sample features in a library. Sample features generation can be supervised and unsupervised, just as regular classification in image processing. More complicated query on image data may combine any of the three types which is very often used in web services applications.

No matter what type of query is performed, whether it is pixel related or not, it will always lead to the spatial query on basic operation unit, the tiles. Eventually the query returns tiles to the application. Spatial indexing records relative location of rows in a table for fast retrieval, different index schema may create different ordered list of locators for fast search of a specific row. Indexing on a table eliminates the long process of full table scan when querying a table. Given the size of image data, the table that stores all tiles could contain millions of rows and be huge, without indexing it would be a nightmare to find any tiles in such big table.

Tile-based storage mechanism itself implies an index schema. Each tile is coupled with its relative position (x and y dimension) in the raster band and spatial resolution that are also stored in the same table in different columns. A composite index on these three fields is sufficient for uniquely identifying a tile in a raster band that satisfies certain spatial conditions. Such indexing can also be easily implemented in a relational database system. The spatial queries on the tiles are eventually translated into queries in database languages (Fig.4).

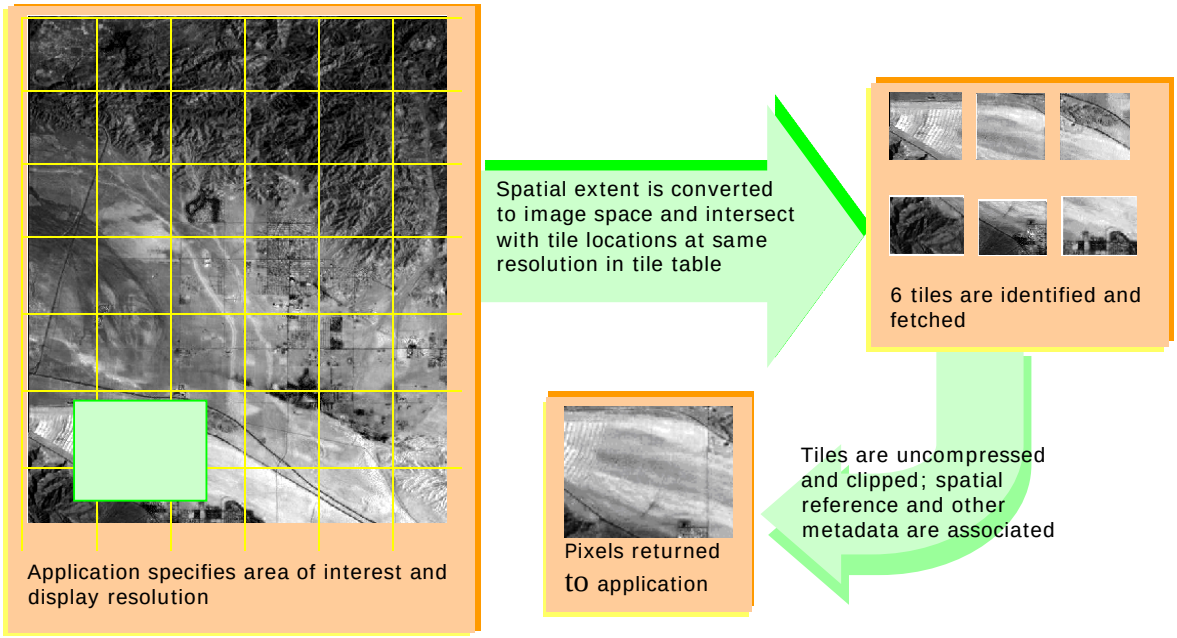

Figure 4 Tile-based spatial query

Queries at raster layer collection level add another layer on top of tile queries. Before it goes down to the tile search, the raster layers have to be identified. The raster layer collection table records a pointer to each raster layer and the layer's metadata, and attribute or metadata query is fairly straightforward. However, since the table doesn't store the actual tiles and their locations as spatial objects no spatial query can be performed. If a spatial query is requested on the collection table, it will go through its metadata where spatial extent information is extracted to identify appropriate raster layers. This process often takes longer than using spatial operations for spatial query. To improve spatial query speed, adding a spatial column to the collection table is recommended. In the spatial column, the extent of footprint of each raster layer is stored and indexed for fast searching. Instead of going through the metadata, once spatial query is

submitted, without probing into the actual tiles spatial operations on the footprint column are performed with the help of its spatial index and the raster layers within the spatial search range are identified. The spatial query then relay down to the raster band level which will lead to the actual queries on the tiles. For each raster band in the selected raster layers, a number of tiles may be fetched based on the resolution and spatial location. All the tiles will be compiled into one seamless layer and sent back to the application along with metadata such as georeferencing information. If the resolution of the tiles is different from the need of the application, resampling is performed. This operation could be implemented at the application server side or client side.

4) Image Data Update

Relatively speaking, image data are static, and this is especially true for satellite data. There isn't any available model that fully supports image updates on the market yet. Image data update can be categorized into two major types. The first type is editing of pixels in a raster band. This is more often seen for scanned maps editing when an old scanned map is updated with new information such as adding new features from recent urban developments. A couple of options exist for this type of update, one is replacement model and the other is version model. The replacement model implies that each pixel can have only one value associated with it, once the value is changed, the original value is lost and no way to bring it back. This model is easy to implement on the tile-based storage structure in the database. Every time a tile is modified, the original one is deleted and replaced with the updated tile. On the other hand, the version model keeps all versions of edits on pixel values, and at any time certain version of the tiles can be retrieved. Versioning model is mostly adopted for vector data editing where each modified feature (point, line or polygon) is stored separately in the database with proper version information. Versioning tiles of pixels means duplication of tiles which will double the space occupied for the same tile, if more versions are involved, the storage size can easily mount up. This brings another argument, is it really necessary to have all the changes on pixels versioned? To most consumers, the answer is no, but to some data vendors it might be a nice feature to have, so that they can provide users with different versions based on different levels of editing at different price. If storage space is not an issue, versioning pixels at tile level is easily achievable. The second type for image update is incremental update. In the case of building a seamless raster layer, each piece is appended to the master layer incrementally. This type of update is in favor of replacement model.

4. Conclusions

Through adequate data modeling and spatial indexing, storing and rapid retrieving imagery data in spatial database is clearly the future for managing large quantities of image data. The integration of image data and spatial database provides key advantages over traditional file-based approaches, especially in an enterprise environment. Such system scales well in a more complicated enterprise environment, provides multiple user access, keeps data integrity in check, and even more, makes it possible in sharing all sorts of spatial data and non-spatial data from a centralized repository. A number of successful image database systems have proven the value of such approach. Even though not all the issues have been addressed in the currently available systems and a lot more work need to be done, it is by any means a great start.

Widespread popularity of web services will push even more researches on solutions for more efficient and robust image management systems. With the advancement in database technology and remote sensing science and the closer collaboration between remote sensing and GIS communities, we can expect a more seamless image data management solution in the near future.

Acknowledgement

We'd like to acknowledge the raster team at ESRI for its continuous leadership in the field of developing image data management solutions. We are very proud of being members of this diverse and dynamic group.

References

[1] ArcSDE, Advanced Spatial Data Server, http://www.esri.com/software/arcgis/arcsde/index.html
[2] Baumann, P., 2001. Web-enabled Raster GIS Services for Large Image and Map Databases
[3] Caganoff, S., 1999, Enterprise Spatial Information Systems, URL: http://www.spatialinfo.com/enterpriseesis.htm
[4] Fredrik W.L., 2001, From the Back Room to the Glass Room – Spatial Databases Break Computing Barriers Enterprisewide, GEOWorld, August 2001.

[5] Han, Z. and Q. Liu, 2001. A Database Approach for Raster Data Management in Geographic Information System, Proc. ICC2001, China, pp.1436 – 1442.
[6] ImageSTOR, a system for managing spatial raster databases, http://www.lorijen.com/ImageSTOR.
[7] MapInfo, Enterprise Mapping Deployments, Managing Spatial Data in a Relational Database Management System, A MapInfo Whitepaper, http://extranet.mapinfo.com/common/library/enterprise_whitepaper.pdf.
[8] Oracle Database 10g, Managing Geographic Raster Data Using GeoRaster, an Oracle Technical White Paper, December 2003.
[9] **URL**: http://www.infoterra-global.com
[10] **URL**: http://www.terraserver.com