A Database Approach for Raster Data Management inGeographic Information System

Zhiguang Han Qian Liu
Environmental Systems Research Institute, Inc.
380 New York Street, Redlands, California 92373, USA
{zhiguang_han, qliu}@esri.com

ABSTRACT

The efficient raster data management on very large raster or raster mosaics is becoming more and more important in spatial data sharing in a distributed GIS environment. The success of such large raster data repository or data warehouse will often demand much greater system scalability and more advanced query ability than the traditional file-based approach. In this paper, we'll present a DBMS-independent spatial raster management solution that combines the spatial data management with the conventional corporate data in a common relational database. In particular, we'll take ESRI's ArcSDE[1] product as an example, following its design to implementation to give a complete view on how this can be accomplished.

1 INTRODUCTION

The storage and retrieval of large raster data sets such as high-resolution aerial, satellite imageries play an increasingly important role in a modern geographic information system (GIS). Especially with the popularity of the Internet and the recent trend of web-based services, more and more companies and organizations are collaborating with each other in an effort to build the infrastructure for sharing the vast amount of spatial data they've collected individually over the Internet and make them more accessible to GIS professionals and ordinary users as well.

A key element in spatial data sharing is how we're going to manage the gigabytes or terabytes of raster data efficiently in a heterogeneous, multi-user environment. The file-based approach that has been working quite well in the past in a disconnected, closed environment apparently has lost its charm in this connected, open environment. It has considerable limitations in terms of size and functionality, especially when it comes to the query ability, scalability, concurrency and access control. These features are often offered by a database system and are fundamental in building web-based GIS services that may serve hundreds of thousands of users concurrently.

However, most of the database systems available today are not very GIS friendly out of the box. The traditional relational database has its strength in managing business oriented tabular data, but it usually lacks of support for most of the GIS oriented data. For object relational database, some recent efforts have been made to incorporate vector-based spatial data types through various spatial extenders. Certain database vendors also provide raster-based spatial extender but with limited functionality. In general, such spatial extenders or extensions are usually tied closely to a particular DBMS, which may prevent efficient data sharing among GIS systems built on different DBMS technologies.

To address this problem, we'll need to define a common spatial data access model which is independent of a particular database and in the meantime is flexible enough to incorporate the latest advancement that different DBMS can offer. GIS systems based on such model will most likely achieve a greater extent of interoperability which becomes more and more important in this Internet era.

In this paper, we'll present a DBMS-independent approach for efficient raster data management in a GIS, especially for large scale raster data sets. This approach is also incorporated into ESRI's Spatial Database Engine (ArcSDE) as part of the ArcGIS[1] software products for providing enterprise-level spatial data access.

2 DBMS-INDEPENDENT SPATIAL RASTER MANAGEMENT

Before getting into the detail of raster data modeling in a DBMS, let's first examine a few basic concepts, i.e. what's a raster, what kinds of raster data we're dealing with in a common GIS environment. The term raster[2] is usually used to represent certain geographic phenomena as locations on a grid with values. It's essentially a rectangular array of equally spaced cells, which not only applies to GIS oriented thematic and spectral data but also to regular pictures as well, such as a photo of a house.

The common forms of raster data include various remote sensing images, such as satellite and aerial imageries, scanned maps, etc. Among them, high resolution orthoimages and orthoimage mosaics often impose a greater challenge in data storage and retrieval due to the sheer size of such data sets, such as DOQs (Digital Orthophoto Quadrangles) and DEMs (Digital Elevation Models). A complete seamless hillshade relief image covering the continental US generated from USGS National Elevation Dataset can easily take well over 10 GB storage space. It's hard to imagine a file-based approach will work well with size of this scale, let alone the system scalability in a distributed multi-user environment. It becomes clear that the route to the integration of raster data management with DBMS will eventually lead us to a more robust, production quality system that scales to large numbers of users and large volumes of data.

Most of the database products available on the market today offer support for Large Objects (LOBs). LOB can be used to store either binary data, such as a video clip, or text information, such as word-processing documents or hyper-linked web pages. In particular, Binary Large Objects (BLOBs) can be used as a generic representation for raster data as variable-length byte strings. Abstract Data Types (ADTs) offered by object relational database can also be used to represent raster in a DBMS environment. ADTs can usually provide a higher level of abstraction, which allows more advanced raster operations to be performed within the scope of a DBMS, while BLOBs only offer a lower level of abstraction that supports a limited set of operations, such as insert, update, delete, select of the entire BLOB or just part of it.

Due to the limited support of ADT-based spatial raster type at DBMS level and the desire for a DBMS independent spatial raster management solution, we'll choose a more generic BLOB-based approach. Though the kind of operations offered by BLOB is not as diverse as ADT, but it's still flexible and sufficient enough for more advanced raster operations to be implemented on top of DBMS. In addition, it's already been widely supported by all major DBMS vendors. With the standardization on ADT-based spatial raster type, hopefully we'll be able to offer a choice between them to our users very soon.

In the next several sections, we'll discuss in detail from the design perspective a DBMS independent spatial raster management solution as implemented in ESRI's ArcSDE.

2.1 ArcSDE Overview

ArcSDE is a GIS gateway that facilitates spatial data management in a database system. It has a cooperative client/server architecture and can be configured either as a server-side application server in

a 3-tier system architecture or as an application-side software component in a 2-tier system architecture. It plays an important role in bridging the gap between the fast paced GIS and database fields, changing the way how GIS users managing their spatial data in a much more connected world. It supports all major functions and capabilities of a "spatially enabled" DBMS like Oracle Spatial, IBM's DB2 Spatial Extender, and Informix's Spatial DataBlade.

ArcSDE provides an open data access interface through a set of well-defined Application Programming Interfaces (APIs) at low level in C/C++ or Java, and at component level through COM-based ArcObjects [1]. By defining a single logical model for spatial data access implemented on top of a particular physical database, applications developed with ArcSDE's API will run with little or no changes at all regardless of the underlying DBMS.

2.2 ArcSDE Raster Data Model

ArcSDE employs the following data model for raster. An ArcSDE raster is a collection of raster band that are generally referenced together. A raster band is a single channel of a rectangular pixel array, which often represents a segment of the electromagnetic spectrum collected by a sensor. Unlike most of file-based image representation, each raster band in an ArcSDE raster might not share certain common properties, such as dimension, pixel depth, etc. The reason for this is to be able to more closely model multi-spectral images as in remote sensing field where individual band may be obtained from different sensors and as a result may have different dimension or pixel depth. By associating raster metadata or attributes with raster band, we'll be able to apply this raster model to a much greater variety of raster data sources.

A raster layer in ArcSDE refers to a collection of ArcSDE raster that are grouped together by certain properties, such as spatial reference information. And it corresponds to the raster type column in a DBMS table. Currently this raster type column is supported in ArcSDE by spatially extending relational databases that do not have raster type support at DBMS level.

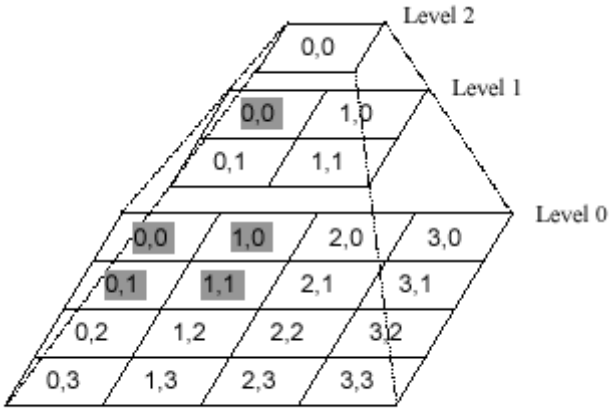

Figure 1: *Tiles in Image Pyramid.*

One of the important design goals of ArcSDE raster model is to support very large "seamless" raster mosaic. Such raster mosaic is typically geo-referenced and appears to the application as a single raster from which any desired rectangular portion can be extracted and displayed. In order to achieve better performance and make it more manageable, we adopt a tile-based storage schema with multi-resolution image pyramids. The raster data in each band is spatially partitioned into rectangular pixel blocks or tiles. Tiles within each band will have the same dimension, however it can be changed for other bands even within the same ArcSDE raster. The basic I/O units in data loading and queries are raster

tiles, which are stored in DBMS table as BLOBs.

From application point of view, it's not always necessary to access the full resolution data, especially for browsing purpose. It's desirable to generate down-sampled images at certain interval to reduce the amount of data involved and speed up certain queries.

The most commonly used multi-resolution techniques include image pyramids, spatial multi-resolution encoding, such as wavelet encoding, or maybe just as simple as thumbnail images. Among them, wavelet-based approach is particularly popular in recent years, such as in the areas of digital image processing and data compression. For such encoding schemes, the data itself already contains the multi-resolution information, thus eliminates the need for separate image pyramids or thumbnails. This may work well on individual raster with limited size, but it's not easily applicable for much larger seamless raster mosaic, nor for the incremental construction of such image mosaic within a DBMS.

In addition, to avoid unnecessary complexity, tile size for the full resolution data and pyramid data will be kept the same. As a result, tiles from different levels in the pyramid do not represent the same area spatially as in the traditional image pyramids, but they rather share the same dimension in pixel space. This actually will fit in well for scenarios when performing zoom in or out with a fixed viewing window.

2.3 ArcSDE Raster Table Schema

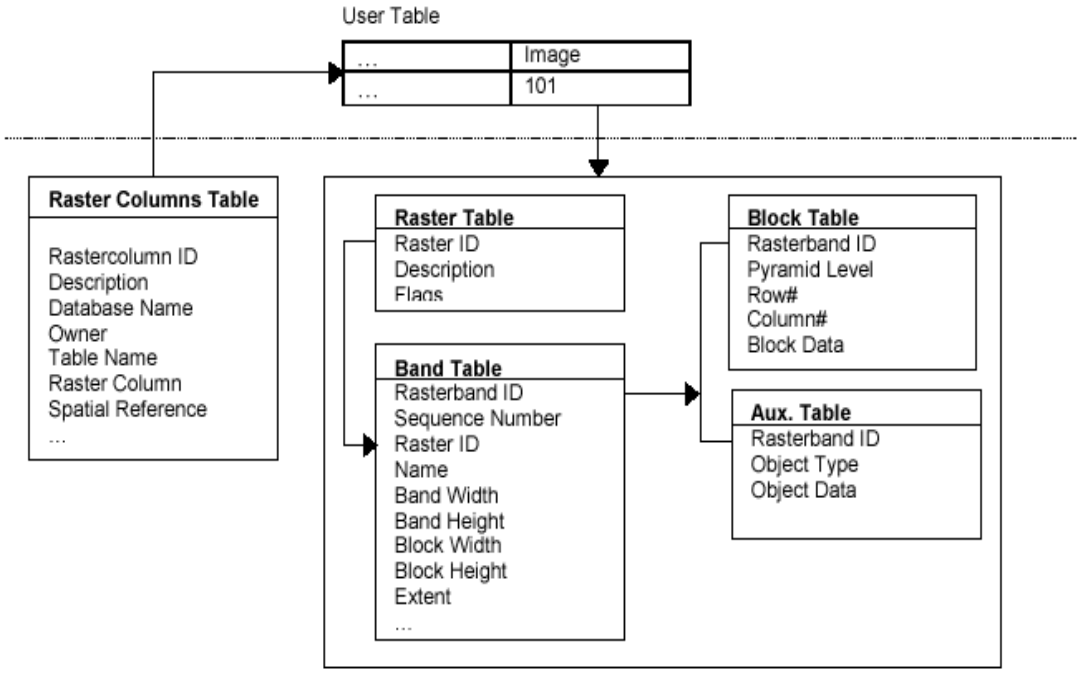

Figure 2: *ArcSDE Raster Table Schema.*

Currently ArcSDE raster model is implemented as a set of metadata tables inside a DBMS. For each ArcSDE maintained raster type column, there will be four associated metadata tables. A raster value in a raster type column is a foreign key reference to the raster represented by the metadata tables. They are: raster table, raster band table, auxiliary band table and block table. The first two metadata tables are used to store information about raster and raster band, such as image dimension, pixel depth, etc. The auxiliary table is used to save additional band properties, such as optional color lookup table, statistics and histogram. The last table is where the actual pixel blocks/tiles are stored. This includes

tiles from image pyramids as well.

In addition, there's an ArcSDE system table used to keep track of all "registered" raster type columns. Each raster column can have an optional spatial reference information. This information is shared among all spatial raster and vector data in ArcSDE. ArcSDE uses POSC/EPSG (Petrotechnical Open Software Corporation/European Petroleum Survey Group) georeference model, which is later adopted as a standard by the Open GIS Consortium [3].

Tiles in a block table can be saved in a compressed format to save disk space and reduce network traffic. In addition, each tile will be attached with an optional bitmap used to mask out no-data pixels. Regular images rarely contain any no-data pixels, but it's not that unusual for remote sensing images. Tiles can be merged on the fly at the server side in case of mosaic, where tiles from different images may happen to be located at the same geo-location. To resolve the conflict, the merge policy will determine how the new pixel values in the merged tile will be calculated. It can be as simple as replacing the previous value, or using a interpolated value instead.

2.4 Raster Data Retrieval

Queries on raster data in a DBMS can be categorized into three different types. The first is metadata-based raster attribute queries, where no pixel data is involved. The second is the queries directly on pixel data. Such queries are usually used to retrieve the entire or portion of a raster. The query window can be given either in geographic coordinates for images with spatial reference information or in pixel coordinates for all images regardless of spatial reference information. The most common type of queries is actually a combination of the first two, where individual images are selected based on certain image properties and then the pixel data from the matching image or a portion of the image will be returned. In addition, vector data and raster data can be associated spatially, either one of them can be treated as an attribute of the other to construct more complex and meaningful queries.

Given the size of an average image, it's usually very expensive to send back the complete image data in a result set. A more appropriate way is to return an image reference instead. The actual pixel data will be retrieved and sent back on demand through this image reference. This way, we'll be able to maintain the logical concept of raster-as-a-value and in the meantime without sacrificing any performance degradations.

Currently, queries on pixel data are tile-based, for a given query window, the matching tiles fetched from the block table will usually go through additional processing. For example, if the client expects a rectangular pixel array as the result, we'll need to merge individual tiles back together and clip off extra pixels that fall out of the query boundary. In addition, if the requested scaling level is different from the data in the block table, tiles from the next closest pyramid level with a higher resolution will be fetched and down-sampled to the correct scaling level. It's also possible that the original raster is in a different projection and needs to be re-projected. These operations can be done either on the client side or on the application server side. It really depends on the nature of the front-end applications and how the data get used.

For example, in desktop ArcGIS, its raster rendering pipeline at data I/O level is actually tile-based too. In this case, it makes more sense to defer the post-processing to ArcGIS and return raster tiles directly. One may argue that without clipping or other additional processing, we may end up shipping more data than we need across the network. This is true, but in general, such "extra" data will only be a small percentage of the actual data. By choosing the proper tile size or applying data compression will help to reduce such overhead. In addition, the client-side tile caching mechanism in ArcGIS will further reduce

the network traffic, since the “extra” data in one query will very likely be requested in subsequent query especially for browsing operations such as pan.

But this does impose a problem when a much slower network connection is involved, which is quite possible for clients accessing raster repositories remotely such as from a web browser. In such scenarios, client applications often have a very limited image processing ability and it’s desirable to shift such processing tasks back to the server. In addition, not doing so will seriously affect the performance since the network just couldn’t handle the amount of data involved with tile transfer in a timely fashion.

This tile-based spatial partitioning will also help to simplify the indexing mechanism used in a raster query. Each tile in the block table will be assigned to a (row#, column#) pair based on its location and a scaling factor. Tiles from the full resolution image will have a scaling factor of 0. In combination, it can uniquely identify a specific tile within a raster band. Such indexing can be easily implemented in a relational database. The tile-based spatial queries will be eventually translated into normal database range queries, which don’t require any special purpose spatial indexing techniques, such as quadtree or R-tree, and can be effectively optimized in a conventional relational database.

Some previous researches have suggested that by arranging physical tiles in certain order, such as Morton coding, it may help to further improve the query performance. But in case of image update or mosaic, where tiles are being deleted or added on the fly, it's not very practical to maintain such physical tile ordering. Even if we’ve managed to do so, the performance gain from this will be limited, and does not necessarily offset the overhead in maintaining such physical tile ordering.

2.5 Issues with Data Loading

ArcSDE provides a mosaic option for loading large seamless image mosaics piece by piece, eliminates the need to pre-mosaic them in advance before the actual data loading. This feature is crucial in loading very large image mosaics. Similarly, images already in the database can be updated in this fashion too, i.e. piece by piece, which makes it possible to update only a portion of an existing image without reloading the entire image. As a result of this “piecewise” approach, image pyramids or certain image attribute data such as statistics or histograms will have to be computed and maintained on the server side in order to correctly reflect the changes of the underlying data.

Another data loading issue is related to image pre-processing, such as georeferencing, rectification, etc. Currently, ArcSDE does not provide direct support for these operations. Usually they can be achieved with a variety of existing software products, such as ArcGIS. It’s under the assumption that any source images that come with a spatial reference information should be georeferenced and rectified.

3 PERFORMANCES

We've been doing some preliminary tests to evaluate the system performance, in particular on its ability to handle very large continuous raster mosaics. The initial test results are all positive. ArcSDE server scales quite well on large raster data sets in terms of data loading and retrieval. In one of our test cases, we're able to load a fairly large continuous raster that consists of about 100 individual pieces with a total size close to 20GB. The final mosaicked image in the database has a dimension of 212400 by 93600 pixels. And in term of query performance, there's no apparent degradation on such large images. For a single band 8-bit raster, the full screen redraw time with a screen resolution set at 1280x1024 is usually under 1 second using the tile-based data I/O mode with both the client and the server on the same local network.

4 APPLICATIONS

ArcSDE provides a set of published APIs to support application development. Its ability in managing both vector and raster data along with the conventional business data can be easily integrated with a wide range of applications. It provides a complete data access solution for GIS and related systems. The available 2-tier or 3-tier architecture will also help its deployment to suit particular needs.

As a gateway to relational database, ArcSDE is part of the ArcGIS software products. It provides a unified interface for managing spatial data in a distributed, heterogeneous network. The spatial raster data can be accessed efficiently in an enterprise environment regardless of its location. Advanced query ability is also provided for searching and retrieving specific raster from very large raster repositories.

ArcSDE can be also deployed to provide web-based data services, such as building very large raster data repositories and serving them over enterprise wide intranet or the world wide web.

5 CONCLUSIONS

Database-independent raster data management provides some key advantages over the conventional file-based approach, especially in a distributed, heterogeneous network environment. It helps to improve system's scalability and provides a much more efficient way to share or publish very large raster data sets over the Internet.

With the convergence from stand-alone GIS to web-based GIS services, we're going to expect more and more web-based raster data repositories being built based on such ideas. Also, with the advancement in database field, we'll be able to expect a closer collaboration between GIS software vendors and database vendors to combine their strength in providing a more seamless spatial data management solution to the GIS community.

ACKNOWLEDGEMENTS

We'd like to thank everyone involved with this project at ESRI, especially to Keith Ryden, who heads the spatial database development team, and also to Steve Kopp who has provided invaluable feedback and suggestions throughout the whole development process.

REFERENCES

[1] ESRI GIS Software. http://www.esri.com/software/.

[2] Zeiler M., *Modeling Our World: The ESRI Guide to Geodatabase Desig*n, ESRI, 1999

[3] Open GIS Consortium. http://www.opengis.org.