



Image: Courtesy of IBM

RASTER IN THE DATABASE

JON SKIFFINGTON AND KIRK MCKELVEY
DELVE INTO THE TOPIC OF RASTER IMAGE
STORAGE INSIDE THE DATABASE AS THE
NEED TO MAINTAIN AND PROCESS
IMAGERY MORE EFFICIENTLY GROWS.

Many GIS analysts, administrators, and programmers are familiar with the benefits of storing geospatial information within a relational database. Databases have become indispensable for the management of geospatial feature data such as roads, bodies of water, topologies, and other additional types of spatial data. However, very few users store raster imagery in a database. While nearly every other form of data can be stored and managed in a database relatively easily, raster imagery has presented difficulties that have for the most part relegated it to file servers, portable hard drives, and the DVD format.

The primary reason why databases have not been used to store raster imagery is its size. Raster images, particularly those used for geospatial purposes, can often be large in file size. While feature data for a county might be measured in megabytes, raster data for the same area could easily be measured in tens of gigabytes. Additionally, raster imagery continues to be acquired at higher and higher resolution, resulting in even greater file sizes. Consequently, raster imagery is also being acquired more often, which means that managing and locating stored files is also becoming more difficult. A database would be the ideal solution if the size issue for raster images were addressed.

New technology solutions are becoming available. One solution, storage within the database, is becoming a viable alternative. In this article, we'll review the historical situation and take a look at a new possibility for storing raster imagery in a database and the benefits of doing so.

The Way it Used to Work

The database has been an indispensable component of Geographic

Information Systems for quite some time. Its role has primarily been to manage "feature data," which describes the physical location and dimensions of roads, rivers, political borders, and physical objects, among others. These types of information are relatively easy to store in a database because they can be represented as a simple sequence of coordinates and metadata. This makes operations such as "find all manhole covers within ten miles of a center point" or "find all rivers that touch this border" possible without too much work.

By contrast, images are much more difficult to work with. Take, for example, an image of King County in Washington State from 2005, which is approximately 250MB in compressed form. If a user wants to extract a ten-square-mile image based on a center point, it can be both a processor- and time-intensive activity to extract that one 16,000 x 16,000 pixel image of an image that's roughly 77,000 pixels square. This became a problem with older database methods, as previously, databases have stored images as a single Binary Large Object (BLOB), which must first be decompressed into memory before the extraction process can even take place. This resulting in making database operations take double or triple the time necessary to extract the same imagery off of a file system.

Traditionally a compromise has been reached by storing information about the raster imagery such as its location, coordinate system, and so forth in a database while storing the actual image on a file system. This allow users to easily query and search for the imagery they need along with all of the organization's other geospatial data, but take advantage of the fact that it's faster and easier to use raster imagery located on a file system

rather than in the database itself.

This compromise has its own problems, however. For instance, situations easily arise where the imagery is no longer located in the place where the database thinks it is. This can happen due to unforeseen circumstances such as a server crash, or due to more malicious activities. In order to prevent situations like this from happening, organizations must devote more resources to network administration, independent archive and security policies, and access control. Furthermore, organizations that frequently update their imagery (a scenario becoming more common as the costs of obtaining high-quality imagery continues to decrease) run into problems associated with keeping the database and disparate file systems synchronized.

Why the Database is Viable Now

Recently, vendors have begun developing solutions for storing raster imagery in a database, providing many benefits for both administrators and users, while decreasing the time and effort traditionally required. Using this approach, access control becomes unified under one system and users can more easily find the imagery they need. The also allows for useful database features such as support for transactions and rollbacks can be utilized.

How is this achieved? Different vendors offer different solutions to optimize database image performance. Most of the speed improvements are gained by simply storing the imagery in a way that is more appropriate for the way it will be accessed. Rather than storing the image as one record in the database, the image is decoded from its source format and then interpolated to many different sizes (often referred to as "multiple

resolutions"). With this work done up front, requests for imagery at a later date can be fulfilled quickly using these image "pyramids." These pyramid levels can then be further broken apart into individual tiles, allowing just the data the user wants to be retrieved quickly.

Pyramids are not without their own pitfalls, however. To begin with, pyramids increase the size of the image, typically by about thirty percent. Additionally, the initial creation of the pyramid can take a significant amount of time. Users who are working with hundreds of gigabytes of imagery can typically expect the import process to take months. When image datasets are being updated with regularity, this can become problematic. Finally, breaking the image into tiles can result in artifacts due to the way the images will line up when reassembled for viewing.

Going Even Further

Greater gains in speed, space-savings, and image quality can be achieved by using wavelet image formats such as MrSID (LizardTech's proprietary wavelet format) and JPEG 2000 (the ISO standard wavelet format) when combined with LizardTech's Spatial Express product. Instead of compressing each tile individually as a self-contained image, the entire image is encoded into a wavelet format. Not only does this improve image quality by eliminating flaws at tile boundaries, it also eliminates the storage space requirement for multi-resolution pyramids since wavelet images inherently contain multiple resolutions.

Because there is no need to pyramid the imagery, the time necessary to import the image into the database is greatly reduced. Further, since MrSID and JPEG 2000 images can be compressed to very small file sizes without visual loss in quality, it's possible to use imagery that is two percent of the size of a comparable pyramided raw file, while still retaining all of the necessary visual information.

Regardless of the method you choose, the benefits of storing geospatial raster imagery in a database are clear. A common repository for all of an organization's geospatial data makes it easier for users to find the data they need and for administrators to organize, back up, and restore, all of which results in imagery that is easy to find and use when critical business and government decisions need to be made.

Jon Skiffington, Product Manager, Geospatial Imaging and **Kirk McKelvey**, Senior Software Engineer at LizardTech. More information: www.lizardtech.com

