Load Rasters in ArcSDE Using sderaster

By Michael Mannion, ESRI Education Services

Rasters have rapidly gained popularity among the ArcSDE user community due in part to the intuitive, graphic-based tools for creating and viewing ArcSDE rasters. In addition to the tools provided through the user interface, this release also added a command line raster management tool called sderaster. This article describes how sderaster can be used to create and manage rasters in ArcSDE.

Building a Raster Database

Because storing rasters in a single file becomes impractical with file sizes of several hundred megabytes and impossible with several gigabytes-size files, ArcSDE and its underlying DBMS are an attractive alternative to physically segmenting rasters into smaller files. In addition, DBMS caching mechanisms can provide more efficient access to shared data in a high-concurrency environment than the native file sharing capabilities of the host operating system.

ArcSDE administrators should note that altering even a single pixel can necessitate computationally and input/output (I/O) intensive operations such as rebuilding pyramids and recalculating raster statistics. It is important that data remains fairly static. The consequences of data changes are more detrimental as file sizes increase, especially when DBMS transactions are being logged for recovery purposes. As a result, most ArcSDE rasters are read-only or read-mostly.

ArcSDE administrators have two methods available for loading rasters in ArcSDE—the tools provided by ArcCatalog and ArcToolbox and sderaster, a command line tool. Sderaster offers administrators a fast, flexible method for loading tagged image file format (TIFF) and band sequential (BSQ) multiband image files into ArcSDE rasters. It also provides the added benefits of simple shell scripting and is supported on all ArcSDE platforms. A detailed comparison of ArcGIS tools and sderaster, as well as information on other raster concepts, can be found in another article in this issue, "Raster Data Management with ArcSDE."

Two Scenarios

This article discusses how to use sderaster and describes common operations associated with its use in loading two adjacent O-series United States Geological Survey (USGS) digital raster graphics (DRG) files, stored in TIFF format as ArcSDE rasters.

One option when loading these files is to create two, independent ArcSDE rasters. While this may be acceptable for a small number of inputs, when storing enough DRGs to cover a county or state, this strategy quickly becomes cumbersome for both users browsing the database and administrators managing the prolific supporting tables and indexes. Because the input rasters are logically related in a manner similar to a map series and share the same physical properties (e.g., projection, pixel size, pixel depth, colormap), a better alternative is to load both source files to a single ArcSDE raster.

To accomplish this, an ArcSDE administrator has two choices—store both files in a seamless raster mosaic or as individual members of an embedded raster catalog. The following scenarios illustrate each choice.

Creating a Mosaic

The DRG files reproduce the 1:24,000-scale USGS topographic quadrangle source maps and include marginalia that overlaps pixels in the bodies of adjacent maps. To support mosaicking, the images must be preprocessed to align pixels and remove the map collar so that only the body of the map remains. NODATA areas caused by the rotation of projected maps should be assigned a pixel value of 99.

Step 1: Import the first file

The easiest way to create an ArcSDE raster from a file is to use the sderaster command with the import operation (sderaster -o import). This will create the business table, add the raster column, and load the pixel data. Alternatively, but less efficiently, each step can be explicitly performed using a combination of sdetable and sderaster commands. See the accompanying article "ArcSDE Administration Commands" for information on the syntax and options used with ArcSDE administration commands. The syntax for importing the first raster file is shown in Figure 1. All examples use a forward slash (\) (the Bourne shell suppression character) to split the command onto multiple lines for readability and assume that the SDEUSER (-u), SDEPASSWORD (-p), SDESERVER (-s), SDEINSTANCE (-i), and SDEDATABASE (-D) environment variables have been set. For the Windows command shell, use the carat (^) to break lines.

The command specified in Figure 1 loads the o35077e6.tif TIFF file into a new ArcSDE raster called drg_mosaic. The -N, -G, and -L options direct sderaster not to load the colormap present in the source file, create spatial reference metadata for the new raster, and refrain from building pyramids.

To mosaic the second file, the ArcSDE raster must not contain a colormap and must have a defined spatial reference. If either of these conditions are not satisfied, the next step will fail and generate a SE_MOSAIC_NOT_ALLOWED error message. If source data lacks an ArcSDE-formatted projection file but the projection is known, use the -G parameter and copy the appropriate file from the ArcCatalog Coordinate Systems folder by choosing Tools > Options > General > Coordinate Systems to add the folder to the Catalog tree. Alternatively, specify a predefined coordinate system by its projection ID. A list of projection IDs can be found in the pedef.h header file of the ArcSDE C API. A copy of this file is available from the ArcSDE Developer Help (Developer Interfaces > C API > C API Function Reference > Include Files > pedef.h).

Figure 1: *Syntax for importing the first raster file*

```
sderaster -o import \
          -f o35077e6.tif \
          -l drg_mosaic,image \
          -g \
          -k large \
          -a 99 \
          -t 128,128 \
          -c lz77 \
          -I nearest \
          -N \
          -G file=utm17_nad27.prj \
          -L 0
```

Although building pyramids does not cause the mosaicking operation to fail, it wastes time and processing resources because ArcSDE scans the entire raster when building pyramids and pyramids become invalid when the underlying data changes. Building pyramids after loading each file in a mosaic would be equivalent to building coverage topology after editing each feature in ArcEdit or rebuilding a DBMS column index after inserting each row in a table. Additional information on pyramids is listed under Step 4 in this article.

Finally, loading TIFF files containing GeoTIFF tags will generate warnings. These warnings indicate that sderaster is ignoring the embedded GeoTIFF tags and instead relying on the associated world file and projection file to georeference the images. These warnings are normal and are not a problem.

Step 2: Mosaic the second file

To append additional raster files to the ArcSDE raster, use sderaster -o mosaic. Because the structure of the output raster has been previously

Continued on page 38

-<letter>	Specifies a command operation or option
< >	Required argument
|	Mutually exclusive arguments
{ }	Used with | to specify a list of choices for an argument
[ ]	Optional parameter

ArcSDE administration commands use UNIX command syntax and notation observing those conventions.

-a	Sets pixels with specified value as no data pixels.
-c	Compress a raster upon entry. LZ77 uses the lossless LZ77 compression algorithm. JPEG uses the lossy JPEG compression algorithm.
-C	rgb: expands teh color map into a true color image.
-d	Deletes color map.
-D	Database or data source name. Not supported by all DBMSs.
-e	Extraction window in world coordinates.
-f	The name of the image file or ArcSDE raster.
-g	Directs the import operation to register the raster layer with the geodatabase.
-G	Coordinate system specifier. <projection_id>: coordinate system ID (see pedef.h file for integer codes) file=<proj_file_name>: file containing coordinate system description string
-h	Prints usage and options
-i	ArcSDE service name or port number.
-I	The resampling technique used during the contruction of the pyramid. nearest: The neight method selects the closest pixel. bilinear: The bilinear method interpolates four adjacent pixels. bicubic: the bicubie mthod interpolates 16 adjacent pixels.
-I	(WITH -o export ONLY) Inverts bilevel images.
-k	Configuration keyword present in DBTUNE table. the storage parameters specific to the raster column will be found under the specific keyword.
-l	The raster layer's business table and raster column. If you are not the table owner, you must qualify the table name as "owner.table."
-L	The pyramid level. Set to a number greater than 0, ArcSDE creates the levels specified unless the apex is reached first. Set to -1, ArcSDE calculates the pyramid level. Set to 0, the pyramid is deleted.
-M	Minimum feature ID. New raster IDs are assigned the larger of the minimum ID or the maximum assigned ID plus one.
-n	Image name.
-N	Ignores color map in data source.
-p	ArcSDE user DBMS password.
-q	Compression quality for JPEG (range between 5 and 95).
-R	Removes pixels with background color in a rotated image.
-s	ArcSDE server host hame (default: localhost)
-S	The raster description (quoted string).
-t	The raster tile width and height measured in pixels. Each tile is stored as a separate raster block.
-t	(WITH -o drop ONLY) DBMS table name.
-u	ArcSDE user DBMS user name.
-v	The raster ID.
-V	Enable verbose mode to describe all properties.
-w	Extraction window in pixel coordinates.

ArcSDE Administration Commands

An operation does a specific task related to a command and is specified using the format –o <name_of_task>. Each operation has a set of options that are specified by -<letter>. Options are denoted by letters and the options that are standard across all commands are listed below.

Load Rasters in ArcSDE Using sderaster

Continued from page 36

defined by the import operation, this command requires fewer arguments as shown in Figure 2.

The example in Figure 2 adds pixels from

Figure 2: *Arguments for sderaster -o mosaic*

```
sderaster -o mosaic \
          -f o35077e7.tif \
          -l drg_mosaic,image \
          -v 1 \
          -N \
          -L 0 \
          -a 99
```

the o35077e7.tif file to the drg_mosaic ArcSDE raster. Specifically, the pixels are appended to raster ID 1 (-v) in the drg_mosaic business table. To determine this raster ID, execute the SQL statement.

```
SELECT * FROM DRG_MOSAIC,
```

or use the command

```
sdetable -o list \
-t drg_mosaic \
-c name \
-v 'ESRI_SDERASTERDATASET'.
```

Because the -g parameter was specified during the import, the name of the one raster in the drg_mosaic business table will be ESRI_SDERASTERDATASET. Also be aware of these three points.

1. In areas of overlap, pixels from the new file will overwrite existing pixels in the ArcSDE raster.
2. The -N parameter is required so that ArcSDE will ignore the colormap when mosaicking.
3. It is good practice to explicitly set the pyramid level to zero (using -L 0) to ensure pyramids will not be built during the mosaicking operation.

If the ArcSDE -303 error SE_MOSAIC_NOT_ALLOWED occurs at this point, the most likely cause is incorrect pixel alignment. This can result from the input and output rasters being different sizes or, more commonly, the origin of the input raster was not offset by an exact multiple of the pixel size from the origin of the output ArcSDE raster. View the world files for the two input rasters to determine if this is the case. If the pixel size (i.e., first and fourth parameters) is the same but the origins (i.e., fifth and sixth parameters) are not offset by an even multiple of that pixel size, the world file of the second image can be manually edited to place the origin at a correctly aligned location. This is the logical equivalent of performing a nearest-neighbor resampling without the overhead of actually processing the pixel data. If more than two input rasters are being mosaicked, execute the sderaster -o mosaic command once for each additional input file, altering only the source file name using the -f option.

Step 3: Add the colormap

Because rasters may not share the same colormap, sderaster will not allow rasters with colormaps to be mosaicked unless the sderaster -N parameter is set so the colormap is ignored. For example, one colormap could display the pixel value 1 in red while another displays pixel

Figure 3: *Loading a colormap*

```
sderaster -o colormap \
          -f o35077e6.tif \
          -l drg_mosaic,image \
          -v 1
```

value 1 as blue. Because an ArcSDE raster may only contain a single colormap, this example would require the creation of two different ArcSDE rasters from the input files.

When mosaicking the DRG files in our scenario, the input colormaps are the same for each file. Therefore, the colormap can be loaded from any of the input files as a property of the ArcSDE raster mosaic. Prior to loading the colormap, the ArcSDE raster will display the image in a grayscale image. After loading the colormap, it will be symbolized in the same manner as the TIFF files.

Step 4: Calculate pyramids, raster statistics, and DBMS statistics

Pyramids, raster statistics, and DBMS statistics can be calculated once all input files have been loaded. Strictly speaking, these three steps aren't required to use rasters in ArcSDE but the significant performance gains realized by building pyramids and statistics make these tasks required for practical purposes. When pixels in an ArcSDE raster change, pyramids and statistics must be recalculated. For large rasters, the time needed for this process can range from several hours to more than a day. Consequently, it is worthwhile to either plan a loading strategy that avoids extraneous operations, or use an embedded raster catalog instead of mosaicking rasters.

Figures 4, 5, and 6 list the commands required to calculate pyramids, raster statistics, and DBMS statistics. In Figure 4, the -L -1 parameter instructs ArcSDE to calculate the optimum number of pyramid levels, based on the width and height of the raster. Alternatively, one can use a positive integer value to define an explicit number of pyramid levels for ArcSDE to build. Creating statistics requires no customized user input because ArcSDE automatically calculates basic statistics (minimum, mean, maximum, and standard deviation) and a histogram. Calculating DBMS-level statistics, on the other hand, may require that the user supply a platform-specific string to control how the underlying database engine samples the raster tables. Refer to the ArcSDE Developer Help for more information on the -m parameter for a specific DBMS.

Figure 4: *Building pyramids*

```
sderaster -o pyramid \
          -l drg_mosaic,image \
          -v 1 \
          -L -1
```

Figure 5: *Building raster statistics*

```
sderaster -o stats \
          -l drg_mosaic,image \
          -v 1
```

Figure 6: *Building DBMS statistics*

```
sdetable -o update_dbms_stats \
         -t drg_mosaic \
         -m <DBMS-specific string>
```

Creating an Embedded Raster Catalog

Creating an embedded raster catalog is an alternative to storing multiple raster files in ArcSDE. In instances where the data consumers require access to the entire map sheet, including the supporting text and graphics surrounding the map body, a raster catalog lets the administrator preserve overlapping input pixels. With raster catalogs, users can query the catalog for individual maps and the source files do not require preprocessing to align the pixels.

Step 1: Import the first file

The process of loading the first file into an embedded raster catalog is essentially the same process used for creating a mosaic, with three exceptions—the -g option is changed to -n, the -N parameter is omitted, and -L -1 is specified telling ArcSDE to build pyramids.

The example in Figure 7 lists the command to load the first file in an embedded catalog. It uses the -n option in place of the -g option, which creates a user-defined name for the raster that will distinguish it from other rasters in the same catalog. If -g is used, the raster

Figure 7: *Loading the first file for an embedded raster catalog*

```
sderaster -o import \
          -f o35077e6.tif \
          -l drg_catalog,image \
          -n o35077e6 \
          -k catalog \
          -t 128,128 \
          -c lz77 \
          -I nearest \
          -G file=utm17_nad27.prj \
          -L -1
```

will be named ESRI_SDERASTERDATASET and, when creating a raster catalog, ArcGIS will detect the ESRI_SDERASTERDATASET name and will not interpret the table as a catalog. Optionally, the catalog can contain additional attributes for its members that can be created initially by building a nonspatial business table with the appropriate columns and adding the raster column using sderaster -o add, or to an existing catalog by adding columns using DBMS-level tools.

The -N parameter is also omitted so that sderaster can load the colormap from the source file. Although each raster in an embedded catalog may store its own colormap, ArcGIS clients use the first colormap accessed from the catalog to symbolize all catalog members.

Finally, -L -1 directs ArcSDE to build pyramids as part of the load. Although pixels for each catalog member are stored in the same table, ArcSDE tracks and manages them independently. Changing pixels in one member of the catalog does not impact the pyramids of other members, so loading a new raster into an embedded catalog does not require rebuilding pyramids for existing data. This is an important consideration for organizations that plan to update a raster database incrementally.

Step 2: Insert the second file

To add a member to the embedded raster catalog, use sderaster -o insert. The syntax for inserting the second raster, and all subsequent rasters, is almost the same as importing the first one and is shown in Figure 8.

Figure 8: *Adding a member to an embedded raster catalog*

```
sderaster -o insert \
          -f o35077e7.tif \
          -l drg_catalog,image \
          -n o35077e7 \
          -t 128,128 \
          -c lz77 \
          -I nearest \
          -G file=utm17_nad27.prj \
          -L -1
```

The absence of the -k parameter is noteworthy. Because all members of the embedded catalog are stored in the same DBMS tables, a dbtune keyword does not, and cannot, be specified. The -G parameter, which prevents loading files with different or missing projection information, is required because all members of the catalog must share the same projection.

If client applications require raster statistics, consider calculating them after importing or inserting each member of the catalog. Like colormaps, statistics are stored individually and can be built for one member without affecting other members of the catalog. Building raster statistics after loading each file, when the name of the most recently loaded member is readily available, allows for easy automation of the process using ArcSDE commands. For example, assume that a large number of files are being loaded using a loop. In that loop, the variable $name stores the name (-n) of the current raster being loaded. Using a combination of sdetable and built-in operating system commands, the raster ID value required by sderaster -o stats can be determined, stored in a variable called raster_id, and immediately used before loading the next file as shown in Figure 9. This technique offers a DBMS-independent method for determining raster IDs and uses those IDs to calculate raster statistics. If a Windows batch is used for this purpose, substitute "find" command for grep, and "for /f" for awk.

Figure 9: *Automating statistics building using sdetable and OS commands*

```
raster_id=`sdetable -o list \
                -t drg_catalog \
                -c name \
                -v $name | \
        grep IMAGE | \
        awk -F: '{ print $2 }'``

sderaster -o stats \
       -l drg_catalog,image \
          -v $raster_id
```

Step 3: Calculate DBMS statistics

Calculating DBMS statistics operates on the entire raster catalog so sdetable -o update_dbms_stats is run only once at the end of the loading process. The syntax used is the same as in Figure 6, part of the mosaicking example.

Conclusion

The sderaster command complements the graphic interface tools in ArcCatalog and ArcToolbox. Its multiplatform support, ease of use when scripting loads, and flexibility for managing pyramids and colormaps make it an attractive choice for some applications. The sample sderaster and sdetable commands shown in the figures can be used to load TIFF and BSQ raster files that meet requisite conditions for mosaicking into a seamless ArcSDE raster, or as individual entries in an embedded raster catalog.